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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему: «Проєкт системи виявлення аномалій трафіку корпоративної мережі на основі поведінкового аналізу»

Виконав: студент 4-го курсу групи КБ-22-1
спеціальності 125 «Кібербезпека»
_____ Алєкс КУХАРЧУК
Керівник: старший викладач кафедри КІ та КБ
_____ Ольга ДЯЧУК
Рецензент: професор кафедри КІ та КБ, д.т.н., доцент
_____ Володимир ВОРОТНІКОВ

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 випускну кваліфікаційну роботу бакалавра

Студент: Кухарчук Алєкс Андріанович

Тема роботи: «Проєкт системи виявлення аномалій трафіку корпоративної мережі на основі поведінкового аналізу»

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

Термін здачі студентом закінченої роботи 22 червня 2026 р.

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

Об'єкт дослідження: процеси виявлення аномалій мережевого трафіку в корпоративній мережі;

Предмет дослідження: методи та засоби виявлення аномалій трафіку на основі поведінкового аналізу із застосуванням Zeek і Wazuh у середовищі GNS3.

Вихідні дані для роботи: нормативно-правовий акт України «Про захист інформації в інформаційно-комунікаційних системах», нормативні документи та стандарти у сфері інформаційної безпеки (ISO/IEC 27001, NIST SP 800-94 «Guide to Intrusion Detection and Prevention Systems», NIST SP 800-137 «Information Security Continuous Monitoring»); технічна документація мережевого аналізатора Zeek (Book of Zeek) та SIEM-платформи Wazuh; науково-технічна література з

питань виявлення аномалій мережевого трафіку та поведінкового аналізу мережевої безпеки; база знань тактик і технік атак MITRE ATT&CK.

Обладнання та програмне забезпечення:

1. Ноутбук із підтримкою віртуалізації.
2. VMware Workstation та GNS3 для побудови симуляції корпоративної мережі.
3. Віртуальні машини на базі Ubuntu Linux та Kali Linux.
4. Zeek Network Security Monitor для пасивного аналізу мережевого трафіку.
5. Wazuh Manager, Wazuh Indexer, Wazuh Dashboard та Wazuh Agent для збору, обробки, кореляції та візуалізації подій безпеки.
6. Допоміжні мережеві інструменти: dnsmasq, DVWA/web-сервер, Nmap, dig, curl, netcat.

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Дячук О.Ю.	02.02.2026	02.02.2026
2	Дячук О.Ю.	30.03.2026	30.03.2026
3	Дячук О.Ю.	22.05.2026	22.05.2026

Студент _____ Алєкс КУХАРЧУК

Керівник _____ Ольга ДЯЧУК

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Обґрунтування вибору теми кваліфікаційної роботи.	02.02.2026	Виконано
2	Аналіз літературних джерел щодо поведінкового аналізу трафіку.	13.02.2026	Виконано
3	Дослідження можливостей Zeek для аналізу мережевого трафіку.	20.02.2026	Виконано
4	Дослідження можливостей Wazuh для обробки та кореляції подій.	25.03.2026	Виконано
5	Проєктування архітектури системи Zeek + Wazuh.	01.04.2026	Виконано
6	Розгортання симуляційного стенду та налаштування мережі	20.04.2026	Виконано
7	Реалізація baseline, Zeek-детекторів і Wazuh-правил.	02.05.2026	Виконано
8	Тестування системи, аналіз результатів.	20.05.2026	Виконано
9	Формування висновків та аналіз проведеної роботи	25.05.2026	Виконано
10	Оформлення пояснювальної записки	01.06.2026	Виконано
11	Захист кваліфікаційної роботи		

Студент _____ Алєкс КУХАРЧУК

Керівник _____ Ольга ДЯЧУК

АНОТАЦІЯ

Кваліфікаційна робота бакалавра присвячена розробці проєкту системи виявлення аномалій трафіку корпоративної мережі на основі поведінкового аналізу.

Пояснювальна записка містить 130 сторінок, 76 рисунків, 14 таблиць. Список використаних джерел містить 20 найменувань і розміщений на 3 сторінках.

Метою кваліфікаційної роботи є розробка проєкту корпоративної мережі з системою виявлення аномалій трафіку на основі поведінкового аналізу із застосуванням SIEM платформи Wazuh та мережевого аналізатору трафіку Zeek.

У роботі проаналізовано сучасні підходи до виявлення аномалій мережевого трафіку, розглянуто особливості сигнатурного та поведінкового аналізу, визначено роль baseline нормальної поведінки в мережі у процесі виявлення відхилень. Досліджено можливості Zeek як засобу пасивного аналізу мережевого трафіку та Wazuh як платформи для збору, обробки, кореляції та візуалізації подій безпеки мережі. Розроблено архітектуру мережі та системи виявлення аномалій трафіку корпоративної мережі на основі інтеграції Zeek та Wazuh. Виконано розділення мережі на функціональні сегменти, реалізовано збір поведінкових логів, сформовано baseline нормальної поведінки активності мережі та визначено критерії аномальності для мережевих подій. Реалізовано скрипти для збору baseline нормальної активності мережі та написано кастомні Zeek-детектори для виявлення аномальної поведінки трафіку всередині корпоративної мережі. Наведено результати експериментального тестування системи з імітацією типових сценаріїв аномальної активності.

Ключові слова: ПОВЕДІНКОВИЙ АНАЛІЗ, АНОМАЛІЇ МЕРЕЖЕВОГО ТРАФІКУ, BASELINE, ДЕТЕКТОРИ, СКРИПТИ, СЕГМЕНТАЦІЯ МЕРЕЖІ, ZEEK, WAZUH, КОРЕЛЯЦІЯ ПОДІЙ.

ANNOTATION

The bachelor's qualification work is devoted to the development of a project for a system for detecting traffic anomalies in a corporate network based on behavioral analysis.

The explanatory note contains 130 pages, 76 figures, and 14 tables. The list of references includes 20 sources and is presented on 3 pages.

The purpose of the qualification work is to develop a corporate network project with a traffic anomaly detection system based on behavioral analysis using the Wazuh SIEM platform and the Zeek network traffic analyzer.

The work analyzes modern approaches to detecting network traffic anomalies, considers the features of signature-based and behavioral analysis, and defines the role of the baseline of normal network behavior in the process of detecting deviations. The capabilities of Zeek as a tool for passive network traffic analysis and Wazuh as a platform for collecting, processing, correlating, and visualizing network security events are investigated. The architecture of the network and the traffic anomaly detection system for a corporate network based on the integration of Zeek and Wazuh is developed. The network is divided into functional segments, behavioral logs are collected, a baseline of normal network activity is formed, and anomaly criteria for network events are defined. Scripts for collecting the baseline of normal network activity are implemented, and custom Zeek detectors are developed to detect anomalous traffic behavior within the corporate network. The results of experimental testing of the system with the simulation of typical scenarios of anomalous activity are presented.

Keywords: BEHAVIORAL ANALYSIS, NETWORK TRAFFIC ANOMALIES, BASELINE, DETECTORS, SCRIPTS, NETWORK SEGMENTATION, ZEEK, WAZUH, EVENT CORRELATION.

ЗМІСТ

ВСТУП 10

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ МЕРЕЖЕВОГО ТРАФІКУ ТА ПОВЕДІНКОВОГО АНАЛІЗУ 12

1.1. Поняття та класифікація аномалій мережевого трафіку ... 12

1.2. Сигнатурний та поведінковий підходи до виявлення мережевих загроз .. 15

1.3. Методи поведінкового аналізу мережевого трафіку ... 19

1.4. Концепція baseline нормальної поведінки мережі ... 23

1.5. Network Security Monitoring та роль Zeek у поведінковому аналізі ... 25

1.6. Wazuh як платформа агрегації, кореляції та візуалізації подій безпеки ... 29

1.7. MITRE ATT&CK у контексті класифікації мережевих аномалій ... 31

РОЗДІЛ 2. ТЕОРЕТИЧНІ ЗАСАДИ ТА ПРОЄКТУВАННЯ СИСТЕМИ..... 34

2.1. Архітектура мережевого аналізатора Zeek ... 34

2.2. Можливості Zeek для поведінкового аналізу ... 36

2.3. Архітектура Wazuh та його роль у проєктованій системі ... 39

2.4. Інтеграція Zeek із Wazuh ... 42

2.5. Модель нормальної поведінки корпоративної мережі ... 44

2.6. Проєктування загальної архітектури системи виявлення аномалій трафіку ... 48

2.7. Проєктування кастомний Zeek детекторів для виявлення аномалій ... 52

2.8. Проєктування Wazuh-декодерів, правила та кореляцій подій ... 53

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ 58

3.1. Вибір операційних систем для розгортання потреб проєкту ... 58

3.2. Налаштування мережевої інфраструктури проєкту корпоративної мережі ... 63

3.3. Розгортання та налаштування Zeek-сенсора ... 69

3.4. Розгортання Wazuh: manager, indexer, dashboard ... 71

3.5. Інтеграція Zeek із Wazuh, пересилання журналів, базове декодування та тестування ... 75

3.6. Побудова Baseline нормальної поведінки корпоративної мережі 78
3.7. Реалізація кастомний Zeek-детекторів аномалій 81
3.8. Налаштування Wazuh декодерів, правила та кореляції подій 84
3.9. Сценарії експериментів і верифікації детекції 89
3.10. Оцінка результатів проведених експериментів 105
3.11. Аналіз False Positives і налаштування порогів 107
ВИСНОВКИ 111
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 114
ДОДАТКИ 117

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ACL – список контролю доступу

API – програмний інтерфейс застосунку

C2 – командно-контрольна інфраструктура

DGA – алгоритм генерації доменних імен

DHCP – протокол автоматичного призначення мережевих параметрів

DNS – система доменних імен

DVWA – вразливий веб-застосунок для тестування безпеки

HTTP – протокол передавання гіпертексту

HTTPS – захищений протокол передавання гіпертексту

IDS – система виявлення вторгнень

IPS – система запобігання вторгненням

IOC – індикатор компрометації

IP – інтернет-протокол

JSON – текстовий формат структурованого подання даних

MITRE ATT&CK – база знань тактик і технік атак

NAT – трансляція мережевих адрес

NSM – моніторинг мережевої безпеки

NXDOMAIN – відповідь DNS про відсутність доменного імені

SIEM – система збору, аналізу та кореляції подій безпеки

SOC – центр моніторингу та реагування на інциденти кібербезпеки

SPAN – дзеркалювання мережевого трафіку на порт моніторингу

SSH – захищений протокол віддаленого адміністрування

TCP – протокол керування передаванням даних

UDP – протокол передавання датаграм

VLAN – віртуальна локальна мережа

VM – віртуальна машина

Wazuh – платформа збору, обробки, кореляції та візуалізації подій безпеки

Zeek – мережевий аналізатор трафіку та система Network Security Monitoring

Baseline – базова модель нормальної поведінки мережі

ПЗ – програмне забезпечення

ОС – операційна система

ВСТУП

У сучасних умовах корпоративні мережі є одним із ключових об’єктів захисту інформаційної інфраструктури організацій. В корпоративних мережах здійснюється обмін службовими даними, надається доступ до серверів, проводиться робота користувачів із внутрішніми та зовнішніми ресурсами, передача конфіденційної інформації та адміністрування інформаційних систем. Із цим зростає кількість мережевих загроз, в тому числі атак, які можуть призводити до порушення конфіденційності, цілісності та доступності інформації.

Традиційні засоби виявлення атак значною мірою ґрунтуються на сигнатурному методі виявлення загроз. Такий підхід є ефективним виявленням для завчасно відомих загроз, для яких існують заздалегідь підготовлені правила або сигнатури. Сучасні атаки часто змінюють свою поведінку, використовують нестандартні сценарії, легітимні протоколи, приховану періодичну активність або поступове накопичення шкідливих дій. У таких випадках аналіз сигнатурним методом може не виявити загрозу або зафіксувати її на пізніх етапах. Саме тому актуальним є використання поведінкового аналізу мережевого трафіку. Поведінковий аналіз на відміну від сигнатурного підходу орієнтований не лише на пошук відомих ознак атаки, а й на виявлення відхилень від нормальної поведінки трафіку в мережі. Для цього формується базова модель нормальної поведінки, яка містить типові комунікаційні пари, характерні часові інтервали активності, типові напрямки з’єднань, допустимі обсяги трафіку, дозволені сервіси та відомі DNS-домени. Коли поточна активність відрізняється від такої моделі, то система може сформувати повідомлення про потенційну аномалію.

Метою кваліфікаційної роботи є розробка проєкту корпоративної мережі з системою виявлення аномалій трафіку на основі поведінкового аналізу із застосуванням SIEM платформи Wazuh та мережевого аналізатору трафіку Zeek.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати сучасні підходи до виявлення аномалій мережевого трафіку та визначити особливості поведінкового аналізу;
- дослідити принципи роботи Zeek як засобу пасивного аналізу мережевого трафіку;

- дослідити можливості Wazuh щодо збору, обробки, кореляції та візуалізації подій безпеки;
- розробити модель нормальної поведінки корпоративної мережі на основі baseline-підходу;
- спроєктувати архітектуру системи виявлення аномалій трафіку на базі інтеграції Zeek і Wazuh;
- реалізувати симуляційний стенд корпоративної мережі з розділенням на функціональні сегменти;
- налаштувати Zeek для збору поведінкових логів і формування notice-подій;
- розробити кастомні Zeek-детектори для виявлення мережевих аномалій;
- розробити Wazuh-декодери, правила виявлення та кореляційні правила для обробки Zeek-подій;
- налаштувати Wazuh Dashboard для візуалізації поведінкових алертів;
- провести експериментальне тестування системи та проаналізувати отримані результати.

Об’єктом дослідження є процеси виявлення аномалій мережевого трафіку в корпоративній мережі.

Предметом дослідження є методи, засоби та технології виявлення аномалій трафіку на основі поведінкового аналізу із застосуванням Zeek та Wazuh. Методи дослідження, використані в роботі, включають аналіз літературних джерел і технічних джерел, порівняльний аналіз підходів до виявлення мережевих загроз, моделювання корпоративної мережі, побудову baseline нормальної поведінки мережі, пасивний аналіз трафіку, розробку правил виявлення, експериментальне тестування та аналіз результатів роботи системи. Також застосовано методи статистичного аналізу подій, порогового виявлення, аналізу новизни подій, аналізу періодичності з’єднань і кореляції декількох подій безпеки.

У процесі виконання кваліфікаційної роботи було розроблено проєкт системи виявлення аномалій трафіку корпоративної мережі на основі поведінкового аналізу. Система поєднує пасивний аналіз мережевого трафіку за допомогою Zeek, побудову baseline нормальної активності, виявлення відхилень кастомними Zeek-скриптами та централізовану обробку, кореляцію й візуалізацію подій у Wazuh.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ МЕРЕЖЕВОГО ТРАФІКУ ТА ПОВЕДІНКОВОГО АНАЛІЗУ

Виявлення аномалій мережевого трафіку є одним із важливих напрямів забезпечення кібербезпеки корпоративних мереж. У сучасних умовах інформаційні системи організацій постійно взаємодіють із внутрішніми та зовнішніми користувачами, обробляють великі обсяги даних і забезпечують доступ користувачів до різних сервісів. Така активність створює складне мережеве середовище, у якому необхідно виявляти своєчасно відхилення від нормальної роботи, потенційні атаки, порушення політики доступу та інші небажані події. Засоби традиційного захисту часто орієнтовані на виявлення відомих загроз за допомогою сигнатур, правил або заздалегідь визначених індикаторів компрометації. Такий підхід є ефективним для типових і вже досліджених атак, однак він має обмеження у випадках, коли зловмисник змінює тактику та техніку атаки, використовує легітимні протоколи або маскує активність під звичайний мережевий трафік. У зв'язку з цим усе більшого значення набуває поведінковий аналіз, який базується на виявленні відхилень від нормальної активності мережі [1].

У цьому розділі розглядаються теоретичні основи виявлення аномалій мережевого трафіку, поняття мережевої аномалії, основні типи аномальної активності, а також підходи до виявлення загроз. Окрему увагу приділено сигнатурному та поведінковому аналізу, побудові та збору нормальної активності в мережі (baseline), засобам Network Security Monitoring і SIEM-платформам, які використовуються для збору, обробки кореляції та візуалізації подій безпеки

1.1. Поняття та класифікація аномалій мережевого трафіку

Корпоративна мережа є сукупністю взаємопов'язаних інформаційних ресурсів, користувацьких пристроїв, серверів, мережевого обладнання та сервісів, які забезпечують виконання бізнес-процесів організації. В нормальних умовах мережа має певні закономірності функціонування: користувачі звертаються до типових серверів, використовують визначені протоколи й порти, DNS-запити мають повторюваний характер, адміністративний доступ здійснюється з

обмеженого вузла та сегмента. Наявність цих закономірностей дає можливість аналізувати нормальну поведінку мережі та виявляти відхилення від неї. Аномалією мережевого трафіку можна вважати подію або сукупність подій, які відрізняються від базової, типової, дозволеної або очікуваної поведінки мережі. Відхиленнями можуть бути незвичні напрямки з'єднань, звернення до нетипових портів, різке збільшення кількості запитів, передача або отримання незвично великого обсягу даних, підозріла періодичність активності або поява нових комунікаційних пар [2].

Важливо також враховувати, що аномалія не завжди буде означати атаку, а насамперед, вона попереджує про відхилення від нормальної поведінки мережі. Наприклад новий сервіс може з'явитись внаслідок легітимного оновлення інфраструктури, збільшення трафіку може бути пов'язане з резервним копіюванням, а активність у нетиповий час – з запланованим адмініструванням, але саме аномальні події часто є першими ознаками сканування, спроби несанкціонованого доступу, командно-контрольної активності або витоку даних та завдяки цим ознакам їх своєчасне виявлення має важливе значення для захисту корпоративної мережі.

За характером прояву аномалії мережевого трафіку можна поділити на кілька основних груп:

- точкові аномалії;
- контекстні аномалії;
- колективні аномалії;
- об'ємні аномалії;
- протокольні аномалії;
- поведінкові аномалії.

Точкова аномалія – це окрема подія, яка відрізняється суттєво від нормальної активності. Прикладом цієї аномалії може бути спроба SSH доступу до серверного вузла з хоста, який не має повноважень до таких підключень та не належить до адміністративного сегменту. Саме така подія може свідчити про порушення політики доступу або про спробу несанкціонованого підключення.

Контекстні аномалії – це аномалія, яка залежить від умов, у яких відбувається подія. Приклад: звернення до користувача web-сервера в робочий час може бути нормальною подією, але така сама активність у неробочий час може розглядатись як відхилення від звичної поведінки. Для виявлення таких аномалій необхідно враховувати не лише сам факт події, а й додатковий контекст: час, джерело, призначення, роль хоста, сегмент мережі та попередню історію активності [3].

Колективні аномалії проявляються не в одній окремій події, а в сукупності взаємопов'язаних дій. Наприклад, одиничне звернення до нового порту не завжди є критичним, однак послідовне звернення до великої кількості портів одного вузла за короткий проміжок часу може означати про можливі ознаки сканування. Аналогічно багаторазові DNS-запити до одного сегменту, або велика кількість відповідей NXDOMAIN можуть свідчити про DNS-аномалію.

Об'ємні аномалії пов'язані зі значним збільшенням кількості трафіку в мережі або кількості однотипних подій. До них можна віднести передачу великого обсягу даних, різке зростання кількості DNS-запитів, масове встановлення з'єднань або надмірну кількість звернень до певного сервісу. Такі аномалії можуть бути пов'язані з витоком даних, автоматизованим скануванням, некоректною роботою пз або шкідливою активністю.

Протокольні аномалії виникають тоді, коли поведінка певного протоколу відрізняється від очікуваної. Наприклад, DNS зазвичай використовується для перетворення доменних імен в IP-адреси, однак надмірно довгі або псевдовипадкові доменні імена можуть бути ознакою DGA-подібної активності. Також підозрілими можуть бути повторювані DNS-запити, нетипова кількість NXDOMAIN-відповідей або використання DNS як каналу прихованої комунікації.

Поведінкові аномалії визначаються через відхилення від типової поведінки хоста, користувача або мережевого сегмента. Коли робоча станція зазвичай звертається лише до DNS або веб-сервера, але раптом починає ініціювати з'єднання до великої кількості портів або невідомих вузлів, така активність може бути визначена як аномальна. Саме поведінкові аномалії становлять основний інтерес у межах цієї кваліфікаційної роботи.

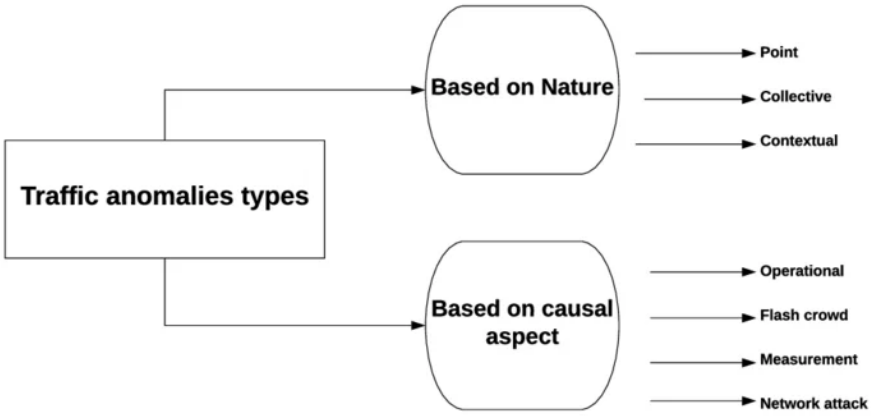


Рисунок 1.1 – Класифікація аномалій мережевого трафіку

В контексті корпоративної мережі особливе значення мають аномалії, які можуть свідчити про ранні етапи атаки. До таких атак належать: сканування портів, поява нових комунікаційних пар, спроби доступу до сервісів не з дозволених сегментів, підозрілі DNS-запити, періодичні з'єднання, передача великого обсягу даних та активність у нетиповий час. Такі події можуть бути неочевидними при ручному аналізу журналів, однак вони можуть бути виявлені за допомогою поведінкових правил, статистичних методів і кореляції подій.

Для виявлення аномалій в мережі потрібно мати уявлення про нормальну поведінку, без такої моделі складно визначити, чи є певна ситуація справді підозрілою.

Отже, аномалії мережевого трафіку є важливим джерелом інформації про потенційні загрози. Їх аналіз дозволяє виявляти не лише відомі атаки, а й нетипову поведінку, яка може передувати інціденту безпеки. Саме тому системи виявлення аномалій мають базуватись не лише на окремих правилах, а й на комплексному аналізі поведінки мережі [4].

1.2. Сигнатурний та поведінковий підходи до виявлення мережевих загроз

У системах кібербезпеки традиційно використовуються різні підходи для виявлення мережевих загроз, серед них є сигнатурний і поведінковий підходи. Вони мають різні принципи роботи, обмеження та переваги, але часто їх поєднання використовується для підвищення ефективності виявлення атак.

Сигнатурний підхід базується на використанні заздалегідь визначених правил, шаблонів або сигнатур, які описують відомі атаки чи небажану активність. Якщо мережева подія відповідає певній сигнатурі, система формує повідомлення про загрозу. Такий підхід використовується в багатьох IDS/IPS-системах, антивірусних рішеннях, засобах аналізу журналів і системах моніторингу безпеки.

Серед переваг сигнатурного підходу є висока точність при виявленні відомих атак, коли сигнатура якісно описує певний сценарій атаки, то відповідно система може швидко ідентифікувати його у трафіку. Сигнатурні правила також зазвичай є зрозумілими для аналітика, оскільки вони прямо вказують, яка саме ознака була виявлена. Однак сигнатурний підхід має низку обмежень. По-перше, він залежить від наявності відомих правил. Якщо атака є новою або суттєво зміненою, система може її не виявити. По друге, зловмисники можуть змінювати структуру запитів, використовувати шифрування, легітимні інструменти або нестандартні послідовності дій, щоб уникнути спрацювання сигнатур. Одна й та сама дія може бути нормальною для адміністратора, але підозрілою для звичайного користувача.

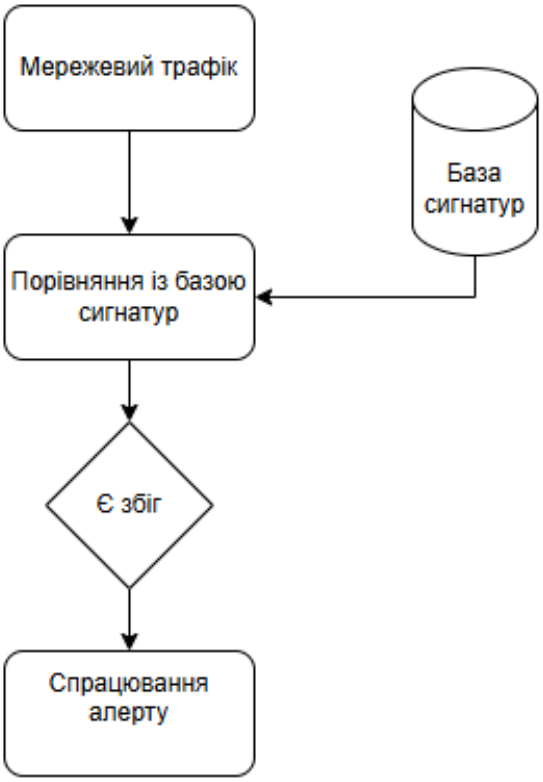


Рисунок 1.2 – Принцип роботи сигнатурного підходу до виявлення загроз

Сигнатурний підхід можна подати як послідовність перевірок, у межах яких мережева подія порівнюється із заздалегідь підготовленою базою правил або шаблонів. Якщо параметри події відповідають відомій сигнатурі, система формує сповіщення про загрозу. Такий підхід є зручним для виявлення типових атак, однак його ефективність залежить від актуальності бази сигнатур і наявності відомого шаблону атаки

Поведінковий підхід, на відміну від сигнатурного більше орієнтований на аналіз відхилень від нормальної активності. Основна ідея полягає в тому, що для мережі, користувача або хоста можна визначити типову поведінку, а потім порівнювати з нею поточні події. Якщо активність суттєво відрізняється від нормальної, вона може бути класифікована як аномальна. Поведінковий аналіз може враховувати різні параметри мережевої активності, такі як: IP-адреси джерела й призначення, порти, протоколи, DNS-запити, кількість з'єднань, обсяг переданих даних, часові інтервали, частоту повторень і послідовності подій. Завдяки цьому поведінковий підхід дозволяє виявляти не лише відомі атаки, а й незвичну активність, яка може свідчити про нові або модифіковані загрози.

Головна перевага поведінкового підходу – це те, що він може виявляти відхилення навіть тоді, коли для них немає готової сигнатури. Це особливо корисно для виявлення ранніх етапів атаки, підготовчих дій, беконінгу або спроби витоку даних. Але водночас такий підхід потребує правильної побудови baseline нормальної поведінки та налаштування порогів спрацьовування.

Основним недоліком поведінкового аналізу є ризик появи false positive спрацьовування, тобто помилкового. Якщо нормальна активність побудована неправильно або неточно, не враховує легітимні зміни в мережі, тоді система може класифікувати нормально події як аномальні. Наприклад, оновлення в системі на хостах операційної системи, встановлення нового сервісу або планове адміністрування може створювати активність, схожу на відхилення. Тому поведінкові системи потребують налаштування, фільтрації шуму та періодичного оновлення baseline для додавання легітимних процесів всередині мережі.

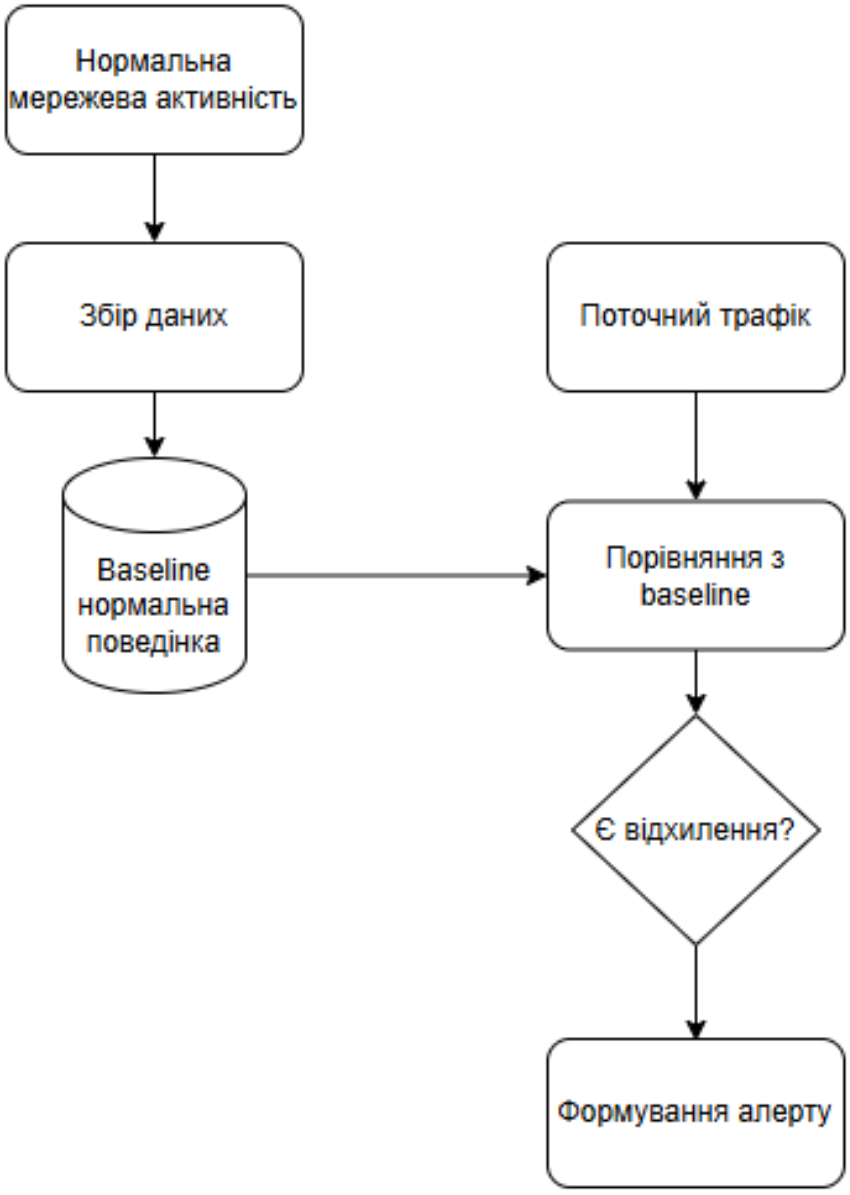


Рисунок 1.3 – Принцип роботи поведінкового підходу до виявлення загроз

Поведінковий підхід є більш гнучким, оскільки дозволяє виявляти не лише заздалегідь відомі атаки, а й нетипову активність, яка відрізняється від нормальної поведінки мережі. У такому випадку важливим етапом є побудова baseline, тобто еталонної моделі нормальної активності. Вона може містити інформацію про типові комунікаційні пари, дозволені сервіси, звичні DNS-запити, нормальні часові інтервали активності та характерні обсяги трафіку

Порівняння сигнатурного та поведінкового методу виявлення подій наведено в таблиці 1.1.

Таблиця 1.1

Порівняння методів виявлення подій

Критерій	Сигнатурний підхід	Поведінковий підхід
Основний принцип	Пошук відомих ознак атак	Виявлення відхилень від нормальної поведінки
Джерело правил	Готові сигнатури, IOC, шаблони	Baseline, статистика, поведінкові критерії
Виявлення відомих атак	Висока ефективність	Залежіть від моделі поведінки
Виявлення нових атак	Обмежене	Можливе через аналіз відхилень
Рівень хибних спрацювань	Зазвичай нижчий	Може бути вищим без налаштування
Необхідність оновлення	Оновлення сигнатур	Оновлення baseline і порогів
Врахування контексту	Обмежене	Високе
Приклад	Виявлення відомого exploit-запиту	Виявлення нового сервісу або нетипового DNS-запиту

У практичних системах в кібербезпеці доцільно поєднувати обидва підходи для кращої ефективності фільтрації та моніторингу підозрілої активності в мережі.

Сигнатурний аналіз дозволяє швидко виявляти відомі загрози, а поведінковий аналіз допомагає знаходити нетипову активність, яка не відповідає звичайній моделі роботи мережі. Таке поєднання підвищує повноту виявлення інцидентів і дає змогу краще аналізувати складні сценарії.

У межах цієї кваліфікаційної роботи основний акцент зроблено саме на поведінковому підході. Для цього використовується пасивний аналіз мережевого трафіку, побудова нормальної поведінки в мережі, розробка кастомний детекторів аномалій, а також подальша обробка й кореляція подій у SIEM системі. Такий підхід дозволяє виявляти не лише окремі події, а й закономірності, які можуть свідчити про потенційну загрозу [5].

1.3. Методи поведінкового аналізу мережевого трафіку

Поведінковий аналіз мережевого трафіку ґрунтується на виявленні відхилень від нормальної активності мережі, окремих хостів, користувачів або сервісів. На

відміну від сигнатурного підходу, який орієнтований на пошук відомих ознак атак, поведінкові методи аналізують закономірності мережевої взаємодії та визначають, чи відповідає поточна активність очікуваній моделі поведінки.

У межах корпоративної мережі поведінковий аналіз може враховувати різні параметри, такі як: IP-адреси джерела й призначення, порти, протоколи, HTTP/S та DNS-запити, кількість з'єднань, обсяг переданих даних, часові інтервали активності, повторюваність подій і появу нових мережевих взаємодій. Саме сукупність таких параметрів дозволяє визначити, чи є аномалія типовою для мережі, чи ні, та вона свідчить про потенційну аномалію.

До основних методів поведінкового аналізу, які доцільно застосовувати для виявлення аномалій мережевого трафіку, належать:

- порогові методи;
- статистична агрегація подій;
- аналіз новизни подій;
- аналіз DNS-запитів;
- entropy-based аналіз;
- аналіз періодичності з'єднань;
- кореляція подій безпеки.

Порогові методи є типовими та найлегшими для реалізації. Вони базуються на встановленні певних граничних значень для мережевих параметрів. Якщо кількість DNS-запитів від одного хоста за короткий проміжок часу перевищує допустиме значення, система може сформувати повідомлення про аномалію. Аналогічно визначаються port scan, велика передача даних або надмірна кількість однотипних подій.

Перевагою порогових методів є проста реалізація та зрозумілість результатів. Аналітик може чітко визначити, яка саме умова призвела до спрацювання правила. Водночас важливим недоліком такого методу є залежність від правильного вибору порогів. Якщо поріг спрацювання буде занадто великим або навпаки ж, занадто малим, це може викликати велику кількість хибних спрацювань, або взагалі їх відсутність.

Статистична агрегація подій дає змогу аналізувати не кожну подію окремо, а їх сукупність за певний проміжок часу. Такий підхід є особливо корисним у випадках, коли активність стає підозрілою не через один окремий факт, а через повторюваність або накопичення однотипних дій. Наприклад, одиничне звернення до нового порту не завжди свідчить про загрозу. Проте велика кількість звернень до різних портів одного хоста за короткий час може вказувати на спробу сканування мережі. Аналогічно, багаторазові DNS-запити до одного й того самого домену можуть бути ознакою DNS флуду.

Аналіз новизни подій один із найважливіших елементів baseline підходу. Його суть полягає у виявленні таких подій, який раніше не було у нормальній моделі поведінки мережі. До таких подій можна віднести появу нової комунікаційної пари, якої раніше не було спостережено в мережі, або ж нового сервісу для вже відомої пари хостів, або ж нового DNS домену. Якщо певна мережева взаємодія не була зафіксована у baseline, вона може бути визначена як потенційне відхилення.

Аналіз DNS запитів має особливе значення для виявлення мережевих аномалій, часто використовується не лише для легітимного перетворення доменних імен, а й у сценаріях прихованої комунікації, DGA-подібної активності, або підготовка до взаємодії з командним сервером. Підозрілими можуть бути надмірно довгі доменні імена, велика кількість запитів до одного домену, повторювані NXDOMAIN-відповіді або звернення до доменів, яких не було у baseline [6].

Аналіз ентропії використовується для оцінки випадковості або невпорядкованості певних параметрів трафіку. Найчастіше він застосовується до DNS-запитів, які можуть містити в собі підозрілі імена. Імена згенеровані автоматично, можуть мати псевдовипадково структуру, велику довжину та незвичний набір символів. У такому випадку підвищений рівень ентропії може бути однією з ознак підозрілого DNS-запиту [7].

Аналіз періодичності з’єднань використовується для виявлення беконінгу, тобто регулярних звернень до певного вузла або сервісу з приблизно однаковим часовим інтервалом. Така активність може бути характерною для командно-

контрольних комунікацій. Коли хост звертається до керуючого сервера. У межах поведінкового аналізу важливо не лише зафіксувати сам факт з'єднання, а й оцінити його повторюваність у часі.

Кореляція подій, один із найбільш значущих інцидентів. Він дозволяє об'єднувати декілька окремих аномалій у більш значущий інцидент. Однієї мережевої події не завжди достатньо для того, щоб вважати, що це є потенційною атакою. Один одиничний DNS-запит до нового домену може бути легітимним, однак якщо він спрацьовує періодично або з однаковим інтервалом, це може говорити про можливу кореляцію подій, яка значно підвищує потенційне виявлення атаки [8].

У таблиці 1.2 наведено узагальнення методів поведінкового аналізу, які доцільно використовувати для виявлення аномалій мережевого трафіку.

Таблиця 1.2

Методи поведінкового аналізу мережевого трафіку

Метод	Основна ідея	Приклад застосування
Пороговий аналіз	Порівняння параметра з визначеним порогом	Виявлення port scan, DNS flood, великого трансферу даних
Статистична агрегація	Аналіз кількості подій за певний проміжок часу	Підрахунок унікальних портів або повторюваних DNS-запитів
Аналіз новизни	Пошук подій, яких не було у baseline	Нова комунікаційна пара, новий сервіс, новий DNS домен
DNS-аналіз	Оцінка структури та частоти DNS -запитів	Підозріла назва DNS , NXDOMAIN
Ентропічний аналіз	Оцінка випадковості доменного імені або іншого параметра	Виявлення псевдовипадкових DNS запитів
Аналіз періодичності	Пошук регулярни з'єднань з однаковим інтервалом	Мережевий beaconing
Кореляція подій	Об'єднання кількох аномалій в один інцидент	Beaconing у поєднанні з великим трансфером даних

Методи поведінкового аналізу дозволяють виявляти аномалії, які не завжди можуть бути знайдені за допомогою сигнатурного підходу. Доцільним є використання гібридної системи в поєднанні сигнатурного та поведінкового підходів.

1.4. Концепція baseline нормальної поведінки мережі

Одним із ключових елементів поведінкового аналізу мережевого трафіку є baseline нормальної поведінки. Під baseline розуміють модель типової активності мережі, яка формується на основі спостереження за легітимним трафіком. Така модель дозволяє визначити, які з'єднання, сервіси, порти, DNS-запити, часові інтервали активності та напрямки взаємодії є звичайними для конкретної корпоративної мережі [9].

На відміну від сигнатурного підоду, який використовує заздалегідь відомі ознаки атак, baseline описує нормально поведінку конкретного середовища. Важливо, що кожна корпоративна мережа має власну структуру, набір серверів, користувацьких вузлів, адміністративних сегментів, дозволених сервісів і політик доступу. Наприклад, підключення по SSH з адміністративного сегмента може бути нормальним, а така сама активність з користувацького сегмента може бути потенційним порушенням політик безпеки. Основна ідея baseline-підходу полягає в тому, що система спочатку фіксує нормальну активність мережі, а потім порівнює з нею поточні події. Якщо нова подія не відповідає сформованій моделі, вона може бути визначена як потенційна аномалія. Наприклад, поява нової комунікаційної пари, нового сервісу для вже відомої пари хостів, DNS-запиту до нетипового домену або активності в незвичний час може свідчити про відхилення від нормальної поведінки.

Схематично процес побудови та використання baseline наведено на рис. 1.4

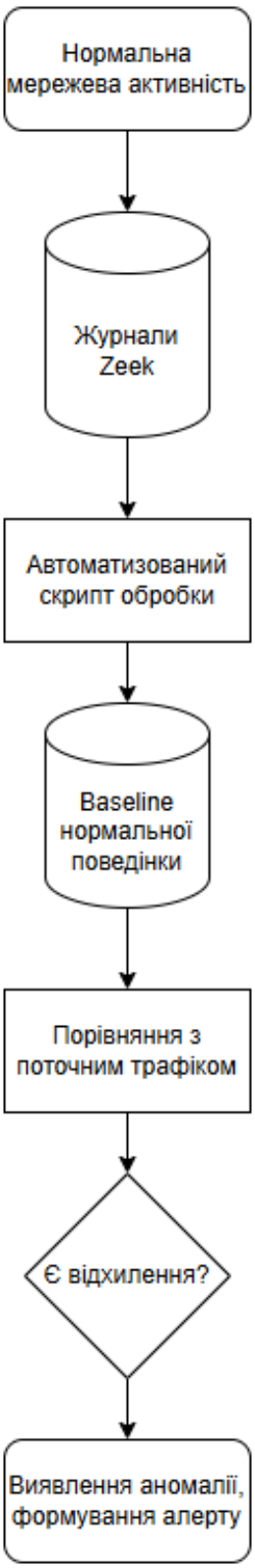


Рисунок 1.4 – Процес побудови та використання Zeek baseline нормальної поведінки мережі

У таблиці 1.3 наведено основні параметри baseline, які можуть використовуватися для виявлення аномалій мережевого трафіку.

Таблиця 1.3

Основні параметри baseline нормальної поведінки

Комунікаційні пари	Зміст параметра	Приклад використання
Комунікаційні пари	Типові взаємодії між хостами	Виявлення нового з’єднання між вузлами
Сервіси та порти	Дозволені або звичні порти для відомих пар	Виявлення нового сервісу для відомої пари
DNS-домени	Домени, до яких хости звертаються в нормальному режимі	Виявлення нового або підозрілого DNS-запиту
Час активності	Типові години роботи хостів або користувачів	Виявлення активності поза нормальним часом
Обсяг трафіку	Орієнтовані межі нормальної передачі даних	Виявлення великого передачі даних
Адміністративні напрямки	Дозволені джерела адміністративного доступу	Виявлення SSH-доступу не з admin сегмента

Важливою особливістю baseline є його залежність від конкретної мережі. Модель нормальної поведінки не може бути універсальною для всіх організацій, оскільки мережі відрізняються структурою, кількістю хостів, сервісами, політиками доступу та характером роботи користувачів. Тому baseline має формуватися з урахуванням фактичної топології та логіки роботи конкретної корпоративної мережі.

Також baseline потребує періодичного оновлення. У мережі можуть з’являтися нові сервіси, змінюватися дозволені напрямки доступу, додаватися нові сервери або змінюватися графік активності користувачів. Якщо baseline не оновлювати, система може сприймати легітимні зміни як аномалії. Водночас оновлення baseline має виконуватися контрольовано, щоб випадково не внести до нормальної моделі шкідливу або підозрілу активність.

1.5. Network Security Monitoring та роль Zeek у поведінковому аналізі

Network Security Monitoring це підхід до забезпечення кібербезпеки, який передбачає постійний збір, аналіз і оцінювання мережевої активності з метою виявлення підозрілих подій, аномалій та потенційних інцидентів безпеки. На

відміну від засобів, які працюють лише за принципом блокування або фільтрації трафіку, системи Network Security Monitoring орієнтовані на глибоке спостереження за тим, що відбувається в мережі, і надання аналітику детальної інформації для подальшого розслідування.

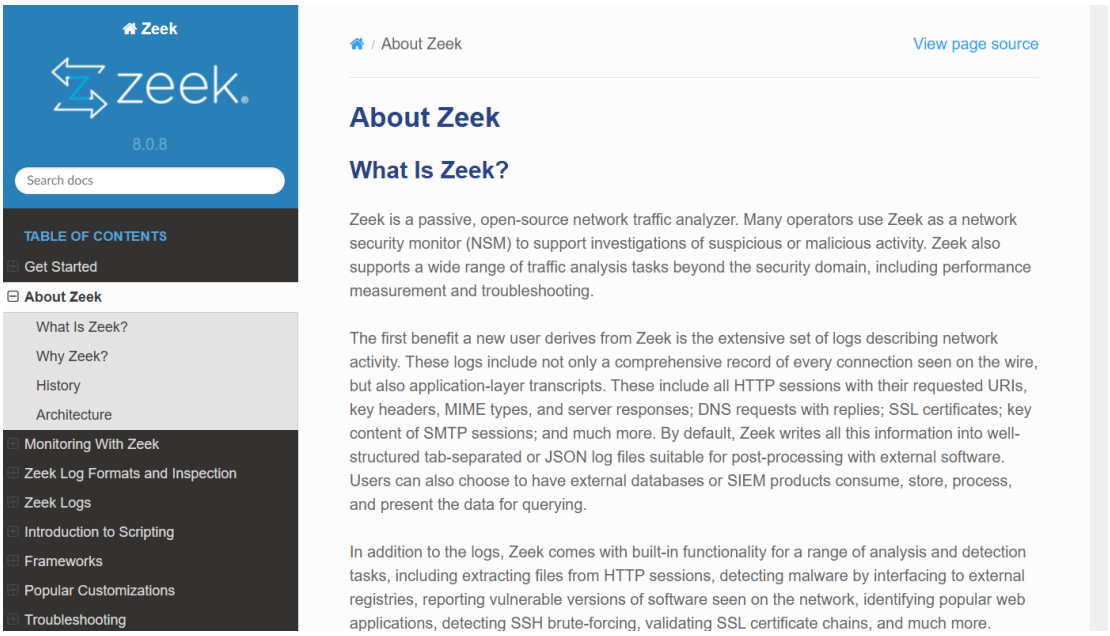


Рисунок 1.5 – Офіційна сторінка Zeek

У контексті поведінкового аналізу Network Security Monitoring має особливе значення, оскільки дозволяє збирати дані про реальну активність мережі. Саме на основі таких даних можна визначити типові комунікаційні пари, звичні сервіси, характерні DNS-запити, обсяги трафіку та часові закономірності роботи хостів. Без якісного збору мережевих подій побудова baseline нормальної поведінки та подальше виявлення відхилень були б неможливими.

Одним із поширених інструментів Network Security Monitoring є Zeek. Він виконує пасивний аналіз мережевого трафіку та формує структуровані журнали подій. На відміну від простого перехоплення пакетів, Zeek не лише фіксує мережевий трафік, а й перетворює його на зручні для аналізу записи, які містять інформацію про з'єднання, протоколи, DNS-запити, HTTP-звернення, SSL/TLS-сесії, файли та події безпеки [10].

Із основних журналів Zeek'а для поведінкового аналізу є такі:

- Conn.log - журнал мережевих з'єднань;
- Dns.log - журнал DNS-запитів;

- http.log - журнал HTTP-звернень;
- ssl.log - журнал SSL/TLS-сесій;
- files.log - журнал переданих файлів;
- notice.log - журнал подій, які потребують уваги;
- weird.log - журнал нетипової або некоректної поведінки протоколів.

Conn.log журнал є одним із найважливіших джерел інформації для аналізу мережевої поведінки. Він містить відомості про IP-адресу джерела та призначення, порти, протоколи, стани з'єднань, кількість переданих байтів і тривалість сесії. З цими даними можна визначити нові з'єднання, аналізувати обсяги трафіку та знаходити потенційні ознаки сканування хостів або нетипової активності.

Журнал Dns.log містить інформацію про DNS-запити, доменні імена, тип запиту, айпі адресу клієнта, DNS-сервер, відповідь DNS та коди результатів. Цей журнал використовується для аналізу DNS-доменів, виявлення нових підозрілих DNS-запитів, флуд DNS запитів та NXDOMAIN spike(неіснуючі запити).

Журнал Http.log містить інформацію про HTTP-запити, зокрема метод запиту, URI, host, User-Agent, статус відповіді та інші параметри web-взаємодії.

Notice.log має особливе значення, оскільки саме до нього Zeek записує логи, які він вважає важливими з погляду безпеки. Події можуть створюватись як стандартними скриптами Zeek та і кастомними скриптами. У межах поведінкового аналізу цей журнал може використовуватись як основний канал передачі аномалій до SIEM системи для подальшої обробки [11].

Окрім журналів, які безпосередньо використовуються для аналізу мережевої поведінки, Zeek також формує службові журнали. До них належать telemetry.log, stats.log, capture_loss.log, packet_filter.log, loaded_scripts.log, stdout.log та stderr.log. Ці журнали не є основним джерелом для побудови baseline або виявлення поведінкових аномалій, однак вони мають значення для контролю коректності роботи Zeek-сенсора.

Важливою перевагою Zeek є можливість створення власних скриптів. За допомогою Zeek scripting language можна описувати правила аналізу мережевої поведінки, перевіряти події на відповідність baseline, формувати повідомлення про

аномалії та передавати їх у notice.log. Це робить Zeek зручним інструментом не лише для пасивного моніторингу, а й для побудови власних поведінкових детекторів.

Zeek може використовуватись у поведінковому аналізі для виявлення таких подій:

- поява нової комунікаційної пари;
- поява нового сервісу для відомої пари хостів;
- звернення до нетипового порту;
- DNS-запит до нового або підозрілого домену;
- велика кількість DNS-запитів;
- сплеск NXDOMAIN-відповідей;
- періодичні з’єднання з однаковим інтервалом;
- передача великого обсягу даних;
- активність поза нормальними часовими межами.

Крім того, на основі Zeek реалізуються кастомні детектори, які аналізують поточні події та формують повідомлення про аномалії.

Схематично роль Zeek у системі поведінкового аналізу наведено на рис. 1.6.

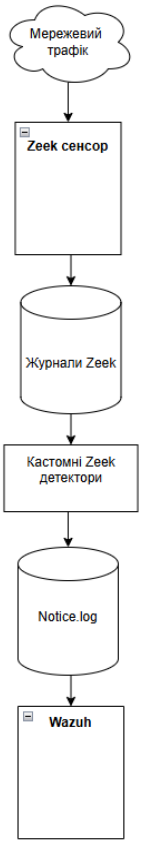


Рисунок 1.6 – Роль Zeek у системі поведінкового аналізу мережевого трафіку

Отже, Zeek є важливим компонентом системи виявлення аномалій мережевого трафіку. Він забезпечує глибокий пасивний аналіз мережевої активності, формує структуровані журнали, дозволяє створювати кастомні правила аналізу та генерує події які можуть бути передані до SIEM системи. У поєднанні з baseline підходом Zeek дозволяє виявляти не лише окремі відомі атаки, а й відхилення від нормальної поведінки корпоративної мережі.

1.6. Wazuh як платформа агрегації, кореляції та візуалізації подій безпеки

Для побудови системи виявлення аномалії було обрано SIEM систему Wazuh. Це платформа для моніторингу безпеки, яка забезпечує збір журналів подій, їх декодування, впровадження правил для виведення алертів, формування алертів, кореляцію подій і відображення результатів у дашборд.

Основні компоненти Wazuh. Wazuh Manager, Wazuh Indexer, Wazuh Dashboard, Wazuh Agent. Ці компоненти виконують свою роботу поступово. Wazuh Agent збирає журнали з вузлів та передає їх на Manager, він виконує обробку подій,

застосовує декодери та правила. Wazuh Indexer забезпечує зберігання та індексацію даних, а Wazuh Dashboard використовується для пошуку, аналізу та візуалізації алертів. В нашій роботі однією з найважливіших частин було написання правил та декодерів. Декодери виступають важливим елементом, так як саме вони визначають, які саме поля потрібно витягнути з вхідного журналу. У випадку з Zeek-подіями такими полями можуть бути тип подій, айпі адреса джерела, айпі адрес призначення, порт, DNS-запит, кількість подій або обсяг переданих даних. Надалі ці поля використовуються правилами Wazuh для визначення типу аномалії.

Правила Wazuh дозволяють класифікувати події та призначити їм відповідний рівень критичності, MITRE ATT&CK айді, поля які витягуються з Zeek логу, завдяки яким ми можемо кастомно налаштувати алерти, та його відображення, також можна групувати події за типами, що спрощує подальший аналіз [12].

Окремо треба вказати важливу функцію, як кореляція подій. Вона дозволяє аналізувати не лише одиничні алерти, а й повторюваність або взаємозв'язок кількох подій. Наприклад, декілька DNS-аномалій з одного джерела або поєднання з беконінгом та передачею великого обсягу даних можуть свідчити про більш серйозний інцидент.



Рисунок 1.7 – Приклад спрацювання кореляції подій Wazuh

Схематично процес обробки Zeek-події у Wazuh наведено на рис. 1.8

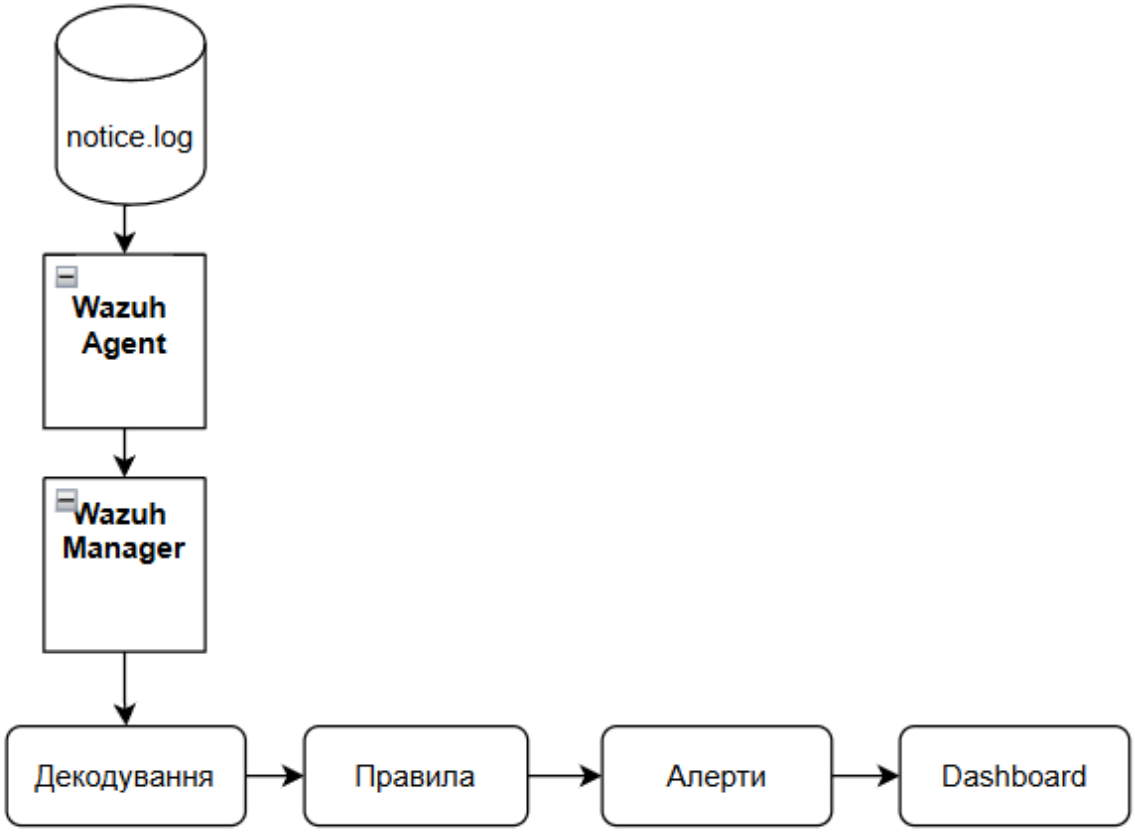


Рисунок 1.8 – Процес обробки Zeek подій у Wazuh SIEM

1.7. MITRE ATT&CK у контексті класифікації мережевих аномалій

MITRE ATT&CK – база знань про класифікацію подій безпеки для аналізування можливих дій зловмисника. Вона містить опис тактик, технік, які можуть застосовуватись під час кібератак, і дозволяє пов’язувати окремі події безпеки з типовими етапами атаки.

В контексті виявлення поведінкових аномалій, ця база знань використовується не як інструмент безпосередного виявлення, а як засіб класифікації та пояснення сформованих алертів. Застосування даної бази знань допомагає ознайомитись з конкретною подією з більш технічної та тактичної сторони. Ця класифікація є корисною для аналітика системи, яка дозволяє не розглядати кожен алерт ізольовано, а й оцінювати його як потенційну частину певного сценарію атак. Саме тому для систематизації результатів моніторингу доцільно використовувати загальноприйняті моделі опису тактик і технік атак.

Приклади відповідності між мережевими аномаліями та техніками MITRE ATT&CK наведено в таблиці 1.4

Таблиця 1.4

Таблиця між аномаліями та техніками MITRE ATT&CK

Мережева аномалія	Можлива тактика	Приклад техніки MITRE ATT&CK
Сканування порту	Discovery	Network Service Discovery
Мережевий beaconing	Command and Control	Application Layer Protocol
DNS-аномалії	Command and Control	DNS
Передача великого обсягу трафіку	Exfiltration	Exfiltration Over C2 Channel
SSH з недозволеного сегменту мережі	Lateral Movement	SSH
Комунікація нової пари пристроїв	Discovery / Lateral Movement	Network Service Discovery / Remote - Services

MITRE ATT&CK виконує роль допоміжного механізму класифікації можливої аномалії у поведінковому аналізі мережевого трафіку. Використання цієї бази даних дозволяє перетворити технічні алерти на більш зрозумілі для аналізу подій, пов’язані з типовими тактиками й техніками атак. Це підвищує практичну цінність системи моніторингу та полегшує подальше розслідування інцидентів безпеки.

Висновки до першого розділу

У першому розділі було розглянуто теоретичні основи виявлення аномалій в мережевому трафіку та визначено, що аномалія є відхиленням від типової та дозволеної поведінки трафіку в мережі, яка може проявлятись нетиповими сервісами, новими з’єднаннями, підозрілими DNS-запитами та іншими аномальними подіями всередині мережі.

Проаналізовано відмінности сигнатурного та поведінкового підходів до виявлення загроз в мережі. Було встановлено, що сигнатурний підхід є ефективнішим у для виявлення відомих атак, однак для виявлення нових, модифікованих або прихованих сценаріїв він є обмеженим та потребує більшого вдосконалення. На відміну від сигнатурного підходу, поведінковий дозволяє виявляти відхилення від нормальної активності мережі, він є більш гнучким та

автоматизованим, що робить його доцільним для побудови системи виявлення аномалій трафіку корпоративної мережі.

Розглянуто основні методи поведінкового аналізу, зокрема пороговий та статистичну агрегацію подій, аналіз новизни, DNS -аналіз, аналіз на базі ентропії, аналіз періодичності з'єдання та кореляцію подій безпеки. Було встановлено, що для реалізації в корпоративну мережу найбільш доцільним буде використовувати поєднання baseline поведінки(нормальної), статистичних правил, аналізу DNS - запитів, виявлення періодичної активності та SIEM кореляцію.

Було окремо розглянуто концепцію baseline підходу нормальної поведінки мережі. Встановлено, що baseline дозволяє формалізувати типову активність всередині корпоративної мережі, та використовувати її як еталон для виявлення відхилень.

Було визначено роль Zeek як засобу пасивного моніторингу мережі та Wazuh як платформи централізованої обробки, збору, кореляції та візуалізації подій безпеки. Zeek забезпечує аналіз мережі, формування структурованих журналів, створення notice подій, в той час як Wazuh приймає ці події, декодує їх, детектить співпадіння, застосовує правила, формує алерти, кореляції подій та відображає результати у зручному для нас виді в дашборді.

Також було окремо розглянуто поєднання класифікацій тактик та технік MITRE ATT&CK в правилах Wazuh, як додаткову інформацію, що підвищує інформативність результатів моніторингу та допомагає краще інтерпретувати можливі дії зловмисника.

За результатами першого розділу встановленно, що для розробки системи виявлення аномалій трафіку корпоративної мережі доцільно використовувати поєднання пасивного аналізу трафіку, збору нормальної поведінки мережі, створення кастомниих детекторів, статистичної агрегації, кореляції подій та централізованої візуалізації.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ЗАСАДИ ТА ПРОЄКТУВАННЯ СИСТЕМИ

В другому розділі буде розглянуто теоретичні та проєктні засади побудови системи виявлення аномалій трафіку корпоративної мережі на основі поведінкового аналізу. Основна увага буде приділена архітектурі Zeek як засобу пасивного аналізу мережевого трафіку, можливостям Wazuh системи для централізованої обробки подій безпеки, принципам інтеграції Zeek та Wazuh, побудові моделі нормальної поведінки мережі та визначенню критеріїв аномальності.

Система яка буде запропонована в цьому розділі базується на поєднанні двох основних компонентів. Першим компонентом виступає Zeek, який аналізує мережевий трафік, формує структуровані журнали та генерує події про потенційні аномалії. Другим компонентом виступає Wazuh, який приймає Zeek логи, виконує їх обробку, декодування, застосовує правила та формулює алерти або кореліцію алертів з відображенням у дашборді.

2.1. Архітектура мережевого аналізатора Zeek

У проєктованій системі Zeek виконує роль мережевого сенсора, який здійснює пасивний аналіз трафіку корпоративної мережі. Його основне завдання полягає у перетворенні мережевої активності на структуровані події, які можуть бути використанні для подальшої побудови baseline поведінки, виявлення відхилень та подальшої передачі логів подій до Wazuh.

На відміну від засобів активного захисту, Zeek не втручається в передавання трафіку, а працює лише в режимі спостереження. Це дозволяє йому аналізувати мережеву активність без зміни логіки роботи корпоративної мережі. Такий підхід є доцільним для побуди системи поведінкового аналізу, оскільки його основна увага приділяється не блокуванню окреми пакетів, а збору даних про поведінку хостів і виявленню відхилень від нормальної поведінки в мережі.

Для аналізу трафіку між основними сегментами корпоративної мережі, зібрані дані Zeek’ом застосовуються для визначення нормальної поведінки в

мережі для формування baseline. Для того, щоб в подальшому на основі цього збору будувати спрацювання детекторів на аномальну поведінку корпоративної мережі.

Загальна логіка використання Zeek у проєктованій системі полягає у виконанні таких етапів:

- пасивне отримання мережевого трафіку;
- формування структурованих журналів подій;
- використання логів для побудови baseline нормальної поведінки;
- аналіз поточних подій кастомними Zeek скриптами;
- формування notice подій при виявленні аномалій;
- передача подій до Wazuh'у для подальшої обробки та перевірки;

Ключовим елементом у цій логіці є notice log, оскільки саме через нього події, сформовані кастомними скриптами Zeek-детекторами, передаються до Wazuh. Такий підхід дозволяє не передавати у систему кореляції подій усі мережеві журнали без винятку, а зосередитись на подіях, які попередньо вже було опрацьовані, відібрані як потенційно важливі, що можуть ставити підозрілу активність з погляду безпеки мережі [13].



Рисунок 2.1 – Приклад виведення логів у файл notice.log Zeek аналізатора

Таким чином, у межах розроблюваної системи, Zeek виконує роль першого рівня аналізу. Він забезпечує збір і первинну обробку інформації мережевих подій, формує дані для нормальної поведінки мережі, генерує повідомлення аномалій, які надалі обробляються у Wazuh. Це дозволяє розділити функції системи. Zeek відповідає за глибокий аналіз трафіку, в той час як Wazuh відповідає за централізовану обробку, кореляцію та візуалізацію результатів.

2.2. Можливості Zeek для поведінкового аналізу

Платформа аналізатора мережевого трафіку Zeek є зручним інструментом не тільки із за можливості пасивного аналізу та інтеграції з SIEM системами, а також із за власної логіки обробки подій. У межах проєктованої системи він використовується для аналізу поточного трафіку, порівнянню подій із baseline нормальної поведінки та формування повідомлень про виявленні аномалії.

Під час аналізу трафіку Zeek формує події високого рівня, наприклад появи нової пари комунікації в мережі. Кастомні скрипти можуть реагувати на такі події, перевіряти їх параметри та приймати рішення про те, чи відповідає активність нормальній поведінці мережі.

У проєктованій системі Zeek-скрипти використовуються для перевірки декількох типів поведінкових відхилень. До них належать поява нової комунікаційної пари, поява нового сервісу для вже відомої пари хостів, SSH-підключення не з адміністративного сегмента, активність поза нормальними годинами, підозрілі DNS-запити, повторювані DNS-запити, сплеск NXDOMAIN-відповідей, port scan, network beaconing та передача великого обсягу даних.

Найважливішими механізмами у спроєктованій мережі на базі Zeek для виявлення аномалій в поведінковому аналізі виступає Notice та Sumstats Framework. Notice Framework використовується для формування подій, які потребують уваги з погляду боку безпеки. Якщо кастомний детектор виявляє відхилення від нормальної поведінки, він створює notice-подію, яка містить вже інформацію про сформовані логи на основі спрацювань детекторів або скриптів та записується у файл notice.log, і може бути передана у Wazuh для централізованої обробки [14].

```
NOTICE([$note=SSH_From_Non_Admin_VLAN,
        $msg=fmt("SSH attempt from non-admin VLAN: %s -> %s", src, dst),
        $identifier=fmt("%s-%s", src, dst)]);
```

Рисунок 2.2 – Приклад формувань notice подій у Zeek за допомогою Notice Framework

Ця подія містить поля note, msg, src, dst та інші параметри, які подалі декодуються у Wazuh.

Для виявлення аномалій, які проявляються через накопичення або повторюваність подій, доцільно використовувати статистичну агрегацію. У Zeek для цього може застосовуватися SumStats Framework. Він дозволяє підраховувати кількість подій за певний проміжок часу, групувати їх за IP-адресами, доменами або іншими параметрами та формувати notice-події при перевищенні визначених порогів. На відміну від простих перевірок, SumStats дає можливість аналізувати накопичення активності, кількість звернень, кількість унікальних значень, повторюваність запитів або максимальні значення певних параметрів. Такий підхід є важливим для пошуку аномалій, які стають підозрілими не через одну дію, а через їхню кількість або частоту [15].

```
SumStats::create([$name="sumstats.portscan.detector",
                  $epoch=60sec,
                  $reducers=set(portscan),
                  $threshold=PORT_SCAN_UNIQUE_PORTS,
                  $threshold_val(key: SumStats::Key, result: SumStats::Result) =
                    {
                    return result["sumstats.portscan"]$unique + 0.0;
                    },
                  $threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
                    {
                    local r = result["sumstats.portscan"];
```

Рисунок 2.3 – Приклад використання SumStats Framework для виявлення Port Scan

У межах цього проєкту корпоративної мережі, статистична агрегація використовується саме для виявлення таких аномалій, як сканування портів, DNS -флуд, пересилання або отримання великої кількості пакетів. Якщо від одного хоста за короткий час надходить велика кількість унікальних портів, то це може свідчити про сканування, або наприклад велика кількість надісланих DNS -запитів до одного домену може розглядатись як DNS -флуд. Якщо обсяг переданих даних перевищує нормальний рівень, подія може бути класифікована як передача великої кількості даних.

Окрему роль відіграє перевірка поточних подій на відповідність baseline. Zeek детектори порівнюють параметри мережевої активності з попередньо сформованими даними нормальної поведінки, якщо комунікаційної пари, сервісу, DNS домену, або часової активності немає в нормальній поведінці, така подія може свідчити про потенційне відхилення.

Також важливо, що Zeek працює по реальному TCP/IP стеку та на протокольних аналізаторах. Він аналізує фізичний, мережевий, транспортний та протокольний канали, а потім вже будує структуровані логи.

Схематичну послідовність поведінкового аналізу трафіку засобами Zeek наведено на рис. 2.4.

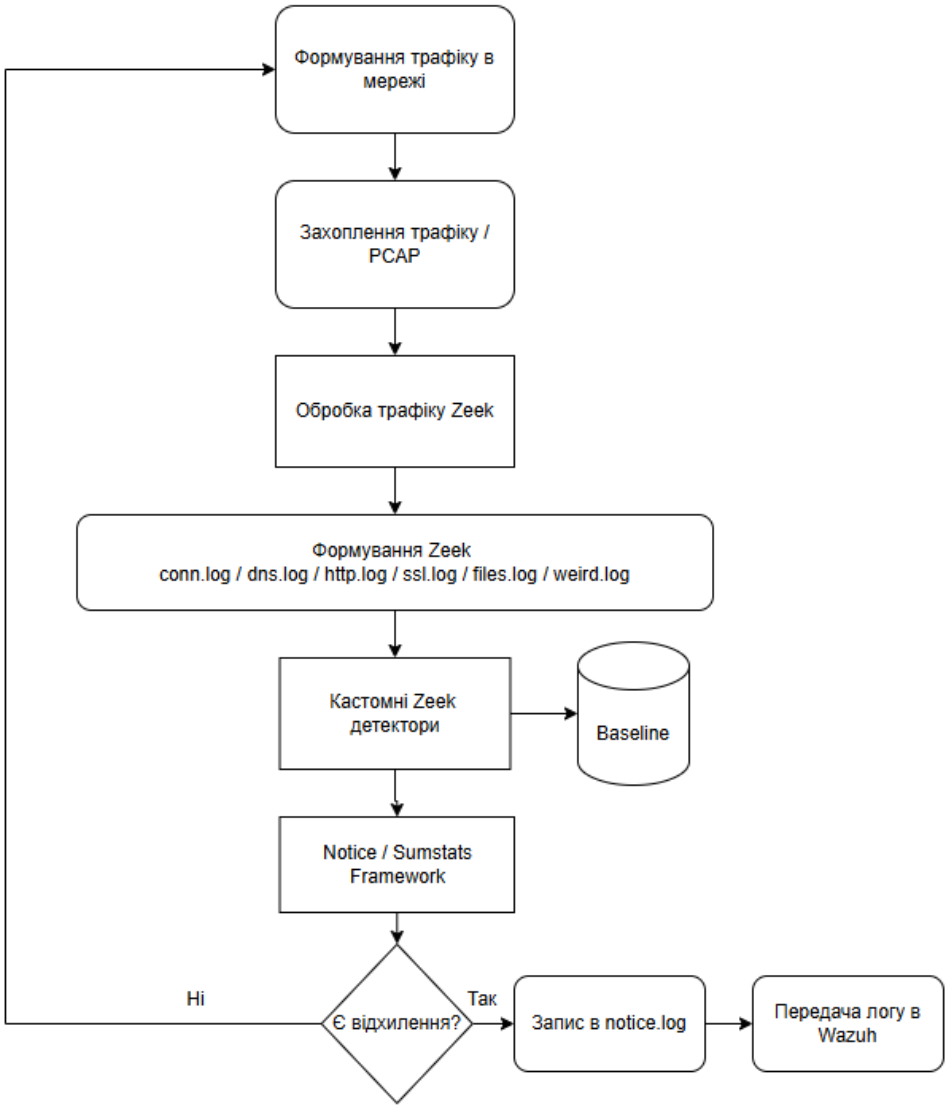


Рисунок 2.4 – Послідовність поведінкового аналізу трафіку засобами Zeek

На першому етапі в мережі формується легітимна або тестова мережева активність. Далі трафік захоплюється сенсором або аналізується у вигляді PCAP-файлу. Zeek обробляє цей трафік і формує структуровані журнали подій: Conn.log, Dns.log, http.log, ssl.log, files.log та weird.log. На основі цих журналів кастомні Zeek-детектори виконують перевірку подій із використанням baseline нормальної поведінки та статистичної агрегації SumStats. Якщо виявлено відхилення,

формується подія в notice.log, яка надалі передається у Wazuh для декодування, застосування правил, кореляції та візуалізації.

Таким чином, можливості Zeek дозволяють реалізувати гнучку систему поведінкового аналізу мережевого трафіку. Завдяки кастомним скриптам, Notice Framework, SumStats Framework і baseline-перевіркам Zeek може виявляти як одиничні відхилення, так і статистичні аномалії, що проявляються через повторюваність або накопичення подій.

2.3. Архітектура Wazuh та його роль у проєктованій системі

У проєктованій системі Wazuh виконує роль платформи централізованої обробки інформації, кореляції та візуалізації подій безпеки. Якщо Zeek відповідає за пасивний аналіз мережевого трафіку та формування подій про потенційні аномалії, то Wazuh виконує забезпечення та подальшу обробку цих подій, застосування правил, формування алертів, і представлення результатів у зручному для аналізу вигляді.

Сам по собі Wazuh як SIEM/HIDS платформа, може використовувати для збору журналів із подіями з різних джерел. У межах цієї корпоративної мережі Wazuh використовується разом з Zeek, що дозволяє поєднувати глибокий мережевий аналіз із централізованою обробкою та візуалізацією подій.

Основними компонентами Wazuh, є такі як: Wazuh Indexer, Wazuh Manager, Wazuh Dashboard та Wazuh Agent. Останній використовується як відповідний інструмент на хості, який формує мережевий аналіз та передає інформацію до головного Wazuh Manager'а. У межах цієї корпоративної мережі Zeek виступає Wazuh Agent'ом та передає Zeek-події, зокрема записів notice.log до Wazuh Manager'а.

Wazuh Manager є центральним компонентом системи. Він приймає події від агентів, виконує їх попередню обробку, застосовує декодери та правила. Саме на цьому етапі вхідна Zeek-подія перетворюється на структурований алерт. Для цього з журналу витягуються ключові поля, наприклад тип Zeek-події, IP-адреса джерела,

IP-адреса призначення, порт, DNS-запит, кількість подій або обсяг переданих даних.

Після формування алертів Wazuh Manager передає їх на подальше зберігання та індексацію. Для цього у складі Wazuh server використовується Filebeat. Filebeat виконує роль проміжного компонента між Wazuh Manager і Wazuh Indexer: він читає сформовані Wazuh-алерти та архівні події, після чого передає їх до Wazuh Indexer для індексації. Таким чином, Filebeat не виконує аналітичну обробку подій і не замінює Wazuh Agent, а забезпечує доставку вже оброблених подій від Wazuh Manager до Wazuh Indexer.

Wazuh Indexer використовується для зберігання та індексації подій. Завдяки цьому алерти можна шукати, фільтрувати, групувати та використовувати для побудови візуалізацій. Wazuh Dashboard забезпечує графічний інтерфейс для аналізу подій, перегляду алертів, побудови таблиць, графіків і панелей моніторингу [16].

У контексті цієї роботи особливе значення мають декодери Wazuh. Вони потрібні для того, щоб правильно обробляти Zeek-події у форматі JSON і витягувати з них необхідні поля. Наприклад, для подій із notice.log важливо отримати значення note, src, dst, p, sub та інших параметрів. Надалі ці поля використовуються правилами Wazuh для визначення типу аномалії.

Правила Wazuh визначають, як саме система має реагувати на певні події. Наприклад, якщо Zeek сформував notice-подію про port scan, DNS query flood, suspicious DNS query або SSH-підключення не з адміністративного сегмента, Wazuh-правило класифікує цю подію, задає рівень критичності та формує відповідний алерт. Також у правилах може задаватися відповідність технікам MITRE ATT&CK.

Окремою функцією Wazuh є кореляція подій. Вона дозволяє аналізувати не лише одиничні алерти, а й повторюваність або взаємозв’язок декількох подій. Наприклад, багаторазові спроби SSH-підключення з одного джерела, повторні port scan-події або поєднання network beaconing із передачею великого обсягу даних

можуть бути підставою для формування більш критичного кореляційного інциденту.

У проєктованій системі Wazuh виконує такі основі функції як:

- приймання Zeek-подій із notice.log;
- декодування JSON-журналів;
- витягування ключових полів подій;
- застосування кастомний правил виявлення;
- формування алертів;
- прив’язка подій до MITRE ATT&CK;
- кореляція декількох пов’язаних подій;
- відображення результатів у Wazuh Dashboard;

Загальну архітектуру Wazuh у проєктованій системі наведено на рис. 2.5

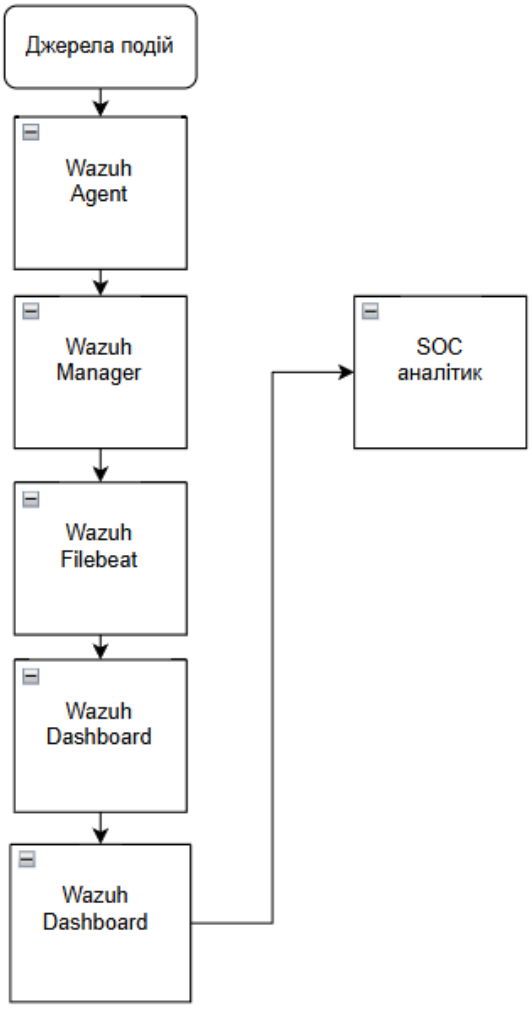


Рисунок 2.5 – Загальна архітектура Wazuh у проєктованій системі

Таким чином, Wazuh забезпечує другий рівень обробки подій у системі виявлення аномалій. Якщо Zeek відповідає за формування змістовних мережевих подій, то Wazuh забезпечує їх централізоване приймання, декодування, класифікацію, кореляцію, зберігання та візуалізацію. Така архітектура дозволяє побудувати повний цикл аналізу: від появи аномальної активності в мережі до формування алерту та його подальшого аналізу аналітиком

2.4. Інтеграція Zeek із Wazuh

Інтеграція Zeek та Wazuh у проєктуванні системи виявлення аномалій мережевого трафіку займає важливу роль, так як, саме цей етап забезпечує перехід від мережевого аналізу, який виконується засобами Zeek, до централізованої обробки подій у Wazuh.

Основною ідеєю інтеграції є те, що Zeek не передає у Wazuh весь масив мережевих журналів без попереднього відбору, замість цього, основним каналом передавання подій виступає журнал notice.log. До нього потрапляють лише ті події, які були сформовані на мережевому аналізаторі як такі, що потребують уваги з погляду безпеки. Такий підхід дозволяє зменшити кількість другорядних журналів у Wazuh та зосередитись на подальшу обробку саме потенційних аномалій.

У проєктованій системі подія потрапляє до notice.log після спрацювання кастомного Zeek-детектора. Наприклад, детектор може виявити нову комунікаційну пару, новий сервіс для відомої пари хостів, SSH-підключення не з адміністративного сегмента, підозрілий DNS-запит, DNS query flood, NXDOMAIN spike, port scan, network beaconing або передачу великого обсягу даних. У кожному з цих випадків Zeek формує notice-подію з відповідним типом, повідомленням та додатковими параметрами.

Ключовим полем для класифікації Zeek-подій у Wazuh є Zeek_note. Воно містить тип події, сформованої Zeek-детектором. Саме за цим значенням Wazuh може визначити, яке правило потрібно застосувати до конкретного запису. Наприклад, у проєктованій системи може набувати значення SumStatsDetectors::SumStats_Port_Scan, яке вказує на подію сканування портів,

DNSAnomalyDetector::Suspicious_DNS_Query - на підозрілий DNS-запит, а BeaconingDetector::Network_Beaconing - на періодичну мережеву активність, характерну для beaconing.

Для коректної інтеграції Zeek із Wazuh важливо, щоб журнали Zeek формувалися у структурованому форматі. У межах проєктованої системи для цього використовується JSON-формат логів. Це дозволяє Wazuh обробляти події не як звичайний текстовий рядок, а як набір окремих полів, які можуть бути використані декодерами та правилами [17].

Увімкнення JSON-формату журналів Zeek здійснюється за допомогою параметра:

redef LogAscii::use_json=T;

Завдяки такому налаштуванню записи notice.log містять структуровані поля, зокрема тип події, повідомлення, джерело, призначення та додаткові параметри. Це спрощує подальше декодування подій у Wazuh і дозволяє створювати правила, які працюють із конкретними значеннями, а не лише з текстовим описом події.

Для коректного відображення подій Zeek у Wazuh, потрібно виконати їх нормалізацію. Під нормалізацією мається на увазі перетворення технічного запису notice.log у набір структурованих полів, які можуть бути використані правилами Wazuh. Це дозволяє системі не просто зафіксувати факт появи повідомлення, а й визначити типи аномалії.

Особливе значення має поле note, яке після декодування використовується як Zeek_note. Воно є основним ідентифікатором типу Zeek-події та дозволяє Wazuh-правилам відрізняти різні класи аномалій. Наприклад, події port scan, DNS query flood, network beaconing або large transfer мають різні значення Zeek_note, тому можуть оброблятися окремими правилами з різним рівнем критичності та описом [16].

У таблиці 2.1 наведено відповідність полів Zeek подіям, які використовуються у Wazuh.

Таблиця 2.1

Відповідності полів Zeek подіям в інтеграції з Wazuh

Поле Zeek-події	Поле після декодування у Wazuh	Призначення
note	Zeek_note	Визначення типу події
msg	Zeek_msg	Текстовий опис події
sub	Zeek_sub	Додатковий контекст події, домен, порт, кількість повторень
src	srcip	IP-адреса джерела аномальної активності
dst	dstip	IP-адреса призначення
p параметр у sub	dstport	Порт призначення або сервіс
dns запит у sub	dnsquery	Доменне ім’я, яке використовується у DNS - аномаліях
Кількісний параметр у sub	dns_count, unique_ports, transfer_bytes	Значення, яке використовується для статистичних правил
Тип правила Wazuh	rule.id, rule.description	Ідентифікація сформованого Wazuh-алерту
MITRE Mapping	rule.mitre.*	Віднесення події до класифікації MITRE ATT&CK

Таким чином, у межах інтеграції Zeek із Wazuh відбувається поетапне перетворення події. Спочатку мережева активність аналізується Zeek і за потреби перетворюється на notice-подію. Далі ця подія передається у Wazuh у структурованому JSON-форматі. Після декодування Wazuh отримує окремі поля, які використовуються правилами для класифікації аномалії. У результаті технічне повідомлення Zeek перетворюється на повноцінний Wazuh-алерт із рівнем критичності, описом, групою правил і можливим зіставленням із MITRE ATT&CK.

2.5. Модель нормальної поведінки корпоративної мережі

Модель нормальної поведінки у корпоративній мережі для пошуку аномалій, займає важливу роль, так як, без наявності легітимного трафіку та отримання інформації про легітимні дії в мережі, ми не зможемо ефективно визначати аномалії, які можуть становити загрозу для нашої мережі.

У межах проєктованої системи baseline розглядається не як абстрактне статистичне поняття, а як набір попередньо визначених параметрів, які описують типову активність хостів у корпоративній мережі. Саме цією моделлю порівнюються поточні мережеві події під час роботи Zeek-детекторів.

На відміну від сигнатурного підходу, де виявлення ґрунтується на пошуку відомого шаблону атаки, підхід нормальної поведінки мережі орієнтований на визначення того, що є нормальним для конкретної мережі. Якщо поточна подія не відповідає цій моделі, вона може розглядатись як потенційне відхилення.

Саме такий підхід є особливо корисним для спроєктованої корпоративної мережі, де структура взаємодії між вузлами зазвичай має певну закономірність, користувацькі станції звертаються до серверів, адміністративні хости мають окремі права доступу, а моніторингові системи виконують збір і аналіз подій.

У проєктованій системі модель нормальної поведінки формується на основі аналізу мережевих журналів Zeek. На початковому етапі в мережі генерується легітимна активність, яка відповідає очікуваній роботі корпоративної інфраструктури. Zeek фіксує цю активність у вигляді структурованих логів, після чого дані використовуються для побудови baseline. Надалі цей baseline підключається до кастомних Zeek-детекторів і використовується для перевірки поточних подій.

До основних елементів baseline у межах цієї роботи належать:

- Дозволені комунікаційні пари між хостами;
- Типові сервіси та порти для відомих пар хостів;
- Дозволені dns-домени;
- Нормальні часові межі активності в системі;
- Допустимі джерела адміністративного доступу;
- Порогові значення для статистичних аномалій;
- Допустимі обсяги переданих даних;

Першим елементом baseline є модель комунікаційних пар. Вона визначає, які хости зазвичай взаємодіють між собою. Наприклад, користувацький хост може звертатися до серверного сегмента для доступу до web-сервісу або DNS-сервера.

Якщо у процесі роботи з’являється нова пара джерело-призначення, якої не було в baseline, така подія може бути визначена як відхилення. Це може виявляти появу нових мережевих взаємодій, які можуть бути пов’язані з розвідкою, lateral movement або несанкціонованою активністю.

Другим елементом є модель дозволених сервісів. Навіть якщо комунікаційна пара вже відома, це не означає, що будь-який порт або сервіс між цими хостами є нормальним. Наприклад, взаємодія користувацького хоста із сервером по DNS або HTTP може бути штатною, але поява нового порту для цієї ж пари може свідчити про зміну поведінки. Тому baseline має враховувати не лише IP-адреси, а й сервіси, які є типовими для конкретної пари хостів.

Окрему частину baseline становить DNS-поведінка. Для користувацьких і серверних вузлів визначаються домени, які є очікуваними в межах нормальної роботи. Якщо хост починає звертатися до нового доменного імені або домен має ознаки підозрілої структури, така подія може бути передана на подальший аналіз. DNS-базова модель є важливою, оскільки DNS-запити часто використовуються як індикатор командно-контрольної активності, beaconing або інших прихованих мережевих взаємодій.

Часова складова baseline використовується для виявлення активності поза нормальними межами. У корпоративному середовищі більшість користувацької активності відбувається в очікувані години роботи. Якщо хост проявляє активність у нетиповий час, така подія сама по собі не завжди є ознакою атаки, але може свідчити про відхилення від звичайної поведінки. У поєднанні з іншими подіями це може мати важливе значення для аналізу інциденту.

Для адміністративного доступу baseline визначає, з яких сегментів мережі дозволено виконувати підключення до критичних вузлів. У межах проєктованої системи особлива увага приділяється SSH-доступу. Якщо SSH-підключення відбувається не з адміністративного сегмента, така подія розглядається як порушення політики доступу. Це дозволяє відокремити штатні адміністративні дії від потенційно небажаних спроб підключення.

Крім якісних параметрів, baseline містить також кількісні характеристики. До них належать порогові значення для статистичних аномалій: кількість унікальних портів за певний час, кількість повторюваних DNS-запитів, кількість NXDOMAIN-відповідей або обсяг переданих даних. Такі параметри використовуються для виявлення подій, які стають підозрілими не через сам факт виникнення, а через частоту, повторюваність або накопичення [18].

Узагальнену структуру baseline у проєктованій системі наведено в табл. 2.2

Таблиця 2.2

Елементи baseline нормальної поведінки

Елементи baseline	Джерело даних	Що описує	Приклад використання
Комунікаційні пари	Conn.log	Звичні взаємодії між хостами	Нова пара джерело-призначення
Дозволені сервіси	Conn.log	Типові порти для відомих хост пар	Новий порт для вже відомої пари
DNS-домени	Dns.log	Доменні імена, характерні для нормальної роботи	Запит до нового або підозрілого домену
Часова активність	Conn.log	Нормальні години активності хостів	Активність поза робочим часом
Адміністративний доступ	Conn.log	Дозволені джерела ssh-підключень	SSH-доступ не з адміністративного сегменту
Статистичні пороги	Поточні події Zeek / Sumstats	Кількість повторень або унікальних значень	Порт сканування, DNS -флуд, відсутній DNS - домен
Обсяг трафіку	Conn.log оброблені Python-скриптом;	Допустимий обсяг переданого трафіку	Передача великої к-сті трафіку

Важливою особливістю запропонованої моделі є те, що baseline не є універсальним для будь-якої мережі. Він формується з урахуванням конкретної топології, адресації, ролей хостів і нормальних сценаріїв взаємодії. Тому одна й та сама дія в різних мережах може мати різну оцінку. Наприклад, SSH-підключення з

адміністративного сегмента може вважатися штатною подією, тоді як аналогічне підключення з користувацького сегмента може бути класифіковане як відхилення.

З погляду реалізації нормальну поведінку можна поділити на дві частини. Перша містить в собі статичні або напівстатичні значення. Друга частина містить в собі динамічні або порогові значення, які використовуються для статистичного аналізу.

Саме такий поділ нормальної поведінки мережі дозволяє виявляти новизни, тобто появи раніше не характерних комунікацій або сервісів та виявлення накопичувальних аномалій на основі динамічного методу, які проявляються перевищенням порогів або повторюваності подій.

Приклад логових файлів для збору нормальної поведінки в мережі у проєктованій системі наведено на рис. 2.6

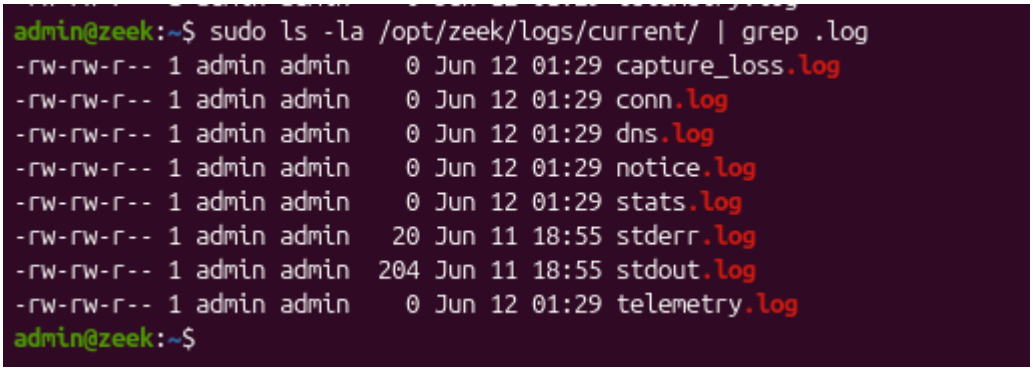


Рисунок 2.6 – Приклад логових файлів для збору нормальної поведінки в мережі у проєктованій системі

Таким чином, модель нормальної поведінки є центральним елементом у проєктованій системи та визначає які взаємодії, сервіси, запити, часові межі та кількісні показники вважаються допустимими. На основі цієї моделі детектори можуть виявляти як поодинокі відхилення, так і статистичні аномалії.

2.6. Проєктування загальної архітектури системи виявлення аномалій трафіку

Після визначення ролі Zeek, Wazuh, baseline та механізмів інтеграції між ними доцільно перейти до проєктування загальної архітектури мережі та системи виявлення аномалій трафіку.

Проєктована система будується на основі сегментованої мережевої архітектури з розподіленням політик, функціональними призначеннями і сценаріями взаємодії між вузлами мережі. Такий підхід дозволяє чітко визначити очікувані сценарії взаємодії для виявлення поведінкових аномалій. Сегментація має важливу роль у поведінковому аналізі, оскільки нормальна активність мережі залежить від того, до якого сегменту належать елементи мережі.

У межі проєктованої системи виділяються серверний, користувацький, адміністративний, моніторинговий і зовнішній сегменти. Серверний сегмент використовується для розміщення внутрішніх сервісів корпоративної мережі, користувацький для моделювання роботи звичайної робочої станції, яка буде виконувати роль “працівника” в середині мережі. Адмністративний сегмент для легітимного керування всередині інфраструктури, доступу до моніторингових даних та доступу до всіх робочих станцій для здійснення контролю та налаштування. Моніторинговий сегмент для розміщення засобів аналізу та обробки подій, зовнішній сегмент для імітації активності, яка надходить з-за меж корпоративної мережі в цілях отримання інформації в відкритому та обмеженому доступі.

Основною логічного розділення мережі на сегменти є VLAN. За допомогою VLAN окремі групи хостів відокремлюються одна від одної на мережевому рівні, це дозволяє задати чітку структуру взаємодії між сегментами та надалі використовувати її як частину baseline нормальної поведінки [19].

Серверний сегмент призначений для розміщення внутрішних сервісів, таких як:

- Тестовий веб додаток DVWA;
- Локальний DNS сервіс;
- Реверсний проксі на базі NGINX;

Тестовий веб-додаток DVWA, а також локальний DNS-сервіс використовується для формування типової HTTP-активності в мережі. NGINX виконує роль перенаправлення до потрібного веб-додатку.

Завдяки цим сервісам у мережі формуються HTTP-звернення, DNS-запити та передавання даних між сервером та хостами, які надалі аналізуються Zeek аналізатором.

Користувацький сегмент представляє робочу станцію звичайного користувача, для якого типовою є поведінка в робочі часи активності, з зверненнями до внутрішнього веб-сервера, виконання DNS-запитів та взаємодія з дозволеними сервісами. Саме з цього сегмента можуть моделюватись як нормальні сценарії роботи, так і поведінкові відхилення, такі як, передавання великого обсягу даних, повторювані DNS-запити, спроба доступу до заборонених сервісів, виконання внутрішньої розвідки мережі тощо.

Зовнішній сегмент використовується для моделювання активності з боку машини атаки та зовнішнього користувача веб сервісами. У корпоративній мережі він дозволяє відтворити ситуації, коли підозріла активність надходить не з внутрішніх корпоративних VLAN, а із зовнішнього напрямку. Для цього на маршрутизаторі використовується nat, який буде моделювати формування зовнішніх звернень, сканування, підключення до внутрішніх ресурсів та інші тестові сценарії. Для того, щоб надалі такий трафік аналізувався мережевим аналізатором.

Контроль доступу між сегментами реалізується за допомогою налаштуванням списків контролю доступу, які визначають дозволені або заборонені типи трафіку між окремими VLAN, а також використовуються для контрольних тестових сценаріїв, коли окремі дозволені правила можуть імітувати потенційно небезпечні напрями взаємодії, які надалі використовуються для моделювання розвідки, сканування або спроб доступу до внутрішніх ресурсів.

Для моніторингового сегменту окреме значення має механізм SPAN, який виконує роль передавання копій мережевиї пакетів до пасивного аналізатора трафіка на базі Zeek, який не має втручатись у передавання трафіку всередині корпоративної мережі, для цього трафік з потрібних сегментів дзеркалюється на інтерфейс Zeek сенсора. У проєктованій архітектурі дзеркалювання трафіку не передбачається для всіх сегментів без винятку, зокрема моніторинговий та

адміністративний сегмент не розглядаються як основні джерела трафіку для SPAN. Таке рішення зумовлене міркування безпеки та відповідає підходу мінімізації збору даних у реальних корпоративних мережах, тому що, адміністративний сегмент може містити службовий трафік керування, підключення адміністраторів, облікові дані та іншу чутливу інформацію. Тому в межах роботи SPAN використовується вибірково - для передавання до Zeek-сенсора копії трафіку з тих сегментів, які є найбільш важливими для поведінкового аналізу та тестування аномалій. Насамперед це взаємодія користувацького, серверного та зовнішнього сегментів. Такий підхід дозволяє аналізувати основні сценарії мережевої активності, не перевантажуючи систему службовим трафіком і не збільшуючи ризики, пов'язані з надмірним збором чутливих даних [20].

Узагальнену топологію лабораторної мережі, яка використовується для побудови системи виявлення аномалій трафіку, наведено на рис. 2.7. Схема відображає поділ мережі на функціональні зони, використання маршрутизатора з NAT та ACL, комутатора з VLAN-сегментацією, а також підключення моніторингового сегмента через механізм SPAN.

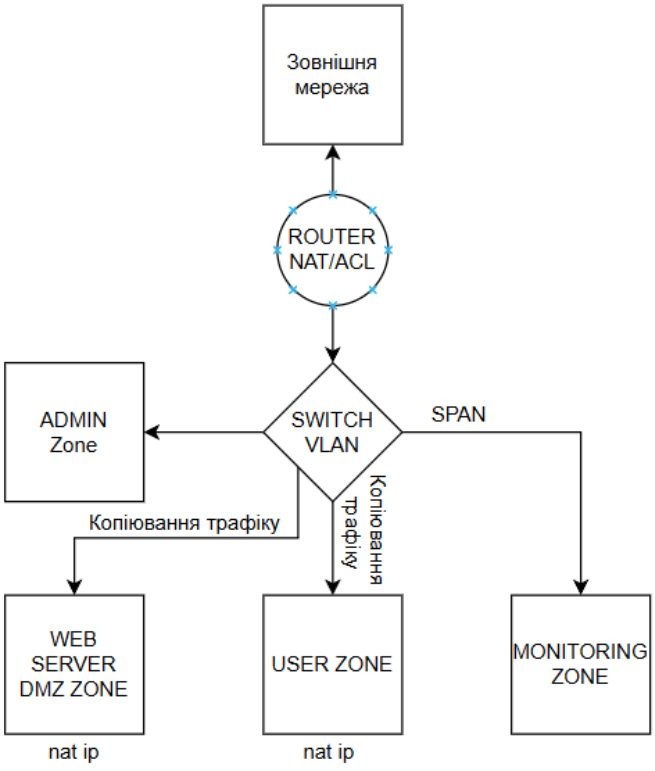


Рисунок 2.7 – Узагальнена топологія корпоративної мережі та підключення контуру моніторингу

2.7. Проєктування кастомних Zeek детекторів для виявлення аномалій

Кастомні Zeek детектори є основним механізмом первиного виявлення аномалій у проєктованій системі. Їх завдання полягає в аналізі поточної активності, порівнянні з baseline нормальною поведінкою в мережі або статистичними умовами та формуванні notice-подій у разі виявлення відхилень.

Усі детектори у проєктованій системи використовують єдиний підхід до передавання результатів. Виявлена аномалія записується у notice.log, через Notice Framework.

Кожен детектор відповідає за окремий тип поведінкового відхилення. Такий підхід дозволяє розділити логіку аналізу за напрямами: нові мережеві взаємодії, нові сервіси, порушення політики доступу, DNS-аномалії, активність поза нормальним часом, періодичні з'єднання, сканування портів і передавання великого обсягу даних.

У проєктованій системі передбачено такі кастомні Zeek-детектори.

Таблиця 2.3

Кастомні Zeek детектори проєктованої системи

Детектор	Джерело даних	Умова виявлення	Результат
New_Communication_Pair	Conn.log	Пара src – dst відсутня в baseline	Подія про нову комунікаційну пару
New_Service_For_Known_Pair	Conn.log	Для відомої пари хостів з'явився новий порт або сервіс	Подія про новий сервіс
SSH_Non_Admin_VLAN	Conn.log	SSH-підключення виконується не з адміністративного сегмента	Подія про порушення політики SSH-доступу
Activity_Outside_Normal_Hours	Conn.log	Активність хоста виникає поза нормальними часовими показниками	Подія про активність у нетиповий час
New_DNS_Domain	Dns.log	DNS-запит до домену, відсутнього у baseline	Подія про новий DNS-домен
Suspicious_DNS_Query	Dns.log	Домен має підозрілу структуру	Подія про підозрілий DNS-запит
DNS_NXDOMAIN_Spike	Dns.log	Велика кількість NXDOMAIN відповідей від одного джерела	Подія про сплеск невдалих DNS-запитів

Продовження таблиці 2.3

Детектор	Джерело даних	Умова виявлення	Результат
Network_Beaconing	Conn.log	Повторювані з'єднання зі стабільним часовим інтервалом	Подія про network beaconing
Sumstats_Port_Scan	Zeek Sumstats	Перевищення порогу унікальних портів за визначений час	Подія про сканування портів
Sumstats_DNS_Query_Flood	Zeek Sumstats	Велика кількість повторюваних DNS-запитів до одного домену	Подія про DNS Query flood
Sumstats_Large_Transfer	Zeek sumstats	Перевищення порогу обсягу переданих даних	Подія про велику кількість трафіку

Таким чином, кастомні Zeek-детектори забезпечують перший рівень виявлення аномалій у системі. Вони аналізують мережеві події, порівнюють їх із baseline або статистичними умовами та формують структуровані notice-події, які надалі обробляються у Wazuh.

2.8. Проєктування Wazuh-декодерів, правила та кореляцій подій

Після формування notice-подій на стороні Zeek необхідно забезпечити їх коректну обробку у Wazuh. У проєктованій системі Wazuh виконує другий рівень аналізу: приймає події з notice.log, декодує їх, витягує ключові поля, застосовує правила виявлення, формує алерти та виконує кореляцію пов'язаних подій.

Обробка подій з Zeek'а у Wazuh складається з трьох основних етапів. На першому розбирається подія як JSON-запис. На другому кастомні декодери витягують із неї поля, необхідні для правил. На третьому Wazuh правила визначають тип аномалії, критичності, опис алерту, описанням технік та тактик Mitre ATT&CK, а також виконується кореляція подій, якщо спрацьовує правило, яке поєднує в собі дві або більше аномалії.

Основним полем для класифікації Zeek-подій є Zeek_note. Воно містить тип notice-події, сформованої Zeek-детектором. Саме за цим полем Wazuh визначає, яке спеціалізоване правило має спрацювати. Наприклад, події SumStatsDetectors::SumStats_Port_Scan, DNSAnomalyDetector::Suspicious_DNS_Query і

BeaconingDetector::Network_Beaconing обробляються різними правилами, оскільки належать до різних типів аномальної поведінки.

Крім Zeek_note, для аналізу використовуються й інші поля: srcip, dstip, dstport, dnsquery, dns_score, dns_count, nxdomain_count, unique_ports, transfer_bytes та Zeek_sub. Завдяки цим полям Wazuh може формувати не загальні повідомлення, а інформативні алерти, у яких видно джерело активності, цільовий вузол, домен, порт, кількість повторень або обсяг переданих даних [21].

Спеціалізовані правила Wazuh проєктуються окремо для кожного типу аномалії. Вони перевіряють значення Zeek_note, використовують поля, отримані декодерами, і формують відповідний алерт. Узагальнений перелік правил наведено в таблиці 2.4.

Таблиця 2.4

Проєктовані Wazuh правила для обробки Zeek аномалій

ID правила	Zeek note	Тип аномалії	Основні поля	Призначення
110000	Network_Beaconing	Network beaconing	Srcip, dstip, dstport, Zeek_sub	Виявлення періодичної мережевої активності
110002	New_Communication_Pair	Нова комунікаційна пара	Scrip, dstip	Фіксація нового напряму взаємодії хостів
110003	SSH_Non_From_Admn_VLAN	SSH не з адмін зони	Scrip, dstip, dstport	Виявлення порушення політики адміністративного доступу
110004	Suspicious_Dns_Query	Підозрілий запит DNS	Dnsquery, dns_score,srcip, dstip	Виявлення нетипової або DGA подібної DNS поведінки
110005	New_DNS_Domain	Новий DNS-домен	Dnsquery, srcip, dstip	Виявлення домену, відсутнього в baseline
110006	Activity_Outside_Normal_Hours	Активність поза нормальним часом	Srcip, dstip, Zeek_sub	Фіксація часової поведінки аномалії

Продовження таблиці 2.4

ID правила	Zeek note	Тип аномалії	Основні поля	Призначення
110007	New_Service_For_Known_Pair	Новий сервіс для відомої пари	Srcip, dstip, dstport	Виявлення нового порту або сервісу
110100	Sumstats_Port_Scan	Сканування портів	Srcip, dstip, unique_ports	Виявлення сканування портів
110101	DNS_NXDOMAIN_Spike	NXDOMAIN Spike	Srcip, dstip, nxdomain_count	Виявлення великої кількості невдалих DNS-запитів
110102	Sumstats_DNS_Query_Flood	Флуд dns запитами	Srcip, dstip, dns_count	Виявлення повторних DNS-запитів
110103	Sumstats_Large_Transfer	Пересилання або отримання великої кількості трафіку	Scrip, transfer_bytes, Zeek_sub	Виявлення передавання великого обсягу даних

Окремим елементом проєктування є визначення рівнів критичності для правил. Події, які лише вказують на зміну baseline, можуть мати середній рівень важливості. Натомість події, пов’язані зі скануванням портів, порушенням SSH-політики, beaconing або передаванням великого обсягу даних, мають вищу аналітичну значущість, оскільки можуть свідчити про активну фазу розвідки, несанкціонований доступ або потенційну ексфільтрацію.

Кореляційні правила Wazuh використовуються для виявлення ситуацій, коли кілька окремих алертів разом утворюють більш серйозний сценарій. Одинична подія може мати середній рівень важливості, але її поєднання з іншими аномаліями може свідчити про складнішу підозрілу активність.

Наведені кореляційні правила продемонстровані у таблиці 2.5

Таблиця 2.5

Кореляційні правила Wazuh

ID кореляції	Базові події	Логіка кореляції	Ймовірна атака
110200	Beaconing + великий обсяг трафіку	Періодичний зв'язок із вузлом та подальша велика передача даних	C2 + ексфільтрація
110201	New Communication Pair + Port scan	Новий напрям комунікації і багато унікальних портів	Розвідка мережі
110202	Activity Outside Hours + SSH Non From Admin VLAN	ССШ доступ з неадміністративного сегменту та активність у нетиповий час (актуально для користувача в сегменті USERS)	Несанкціонований доступ + горизонтальне переміщення в мережі

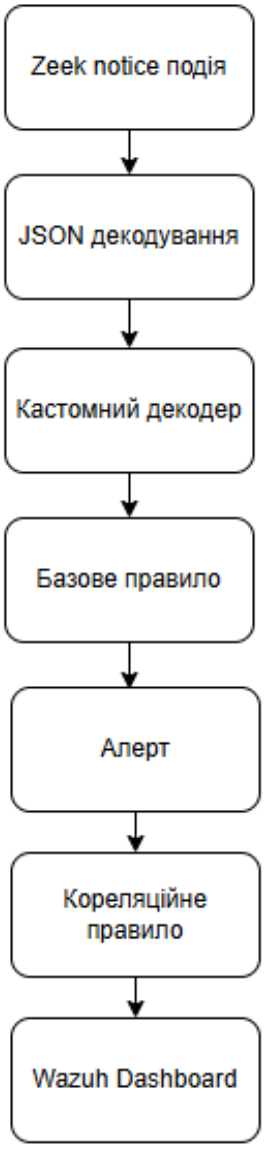


Рисунок 2.8 – Логіка формування Wazuh-алертів і кореляційних подій

Висновки до другого розділу

У другому розділі було спроєктовано систему виявлення аномалій трафіку корпоративної мережі на основі поведінкового аналізу. Основу увагу приділено не лише окремим засобам моніторингу, а їхні взаємодії в межах єдиної архітектури. Wazuh використовується як платформа для обробки, класифікації, кореляції та візуалізації, а Zeek як сенсор пасивного аналізу мережевого трафіку.

Було визначено роль Zeek у проєктовній системі, спроєктовано розташування та реалізовано ідею механізму SPAN для копіювання мережевого трафіку.

Також було спроєктовано архітектуру Wazuh у межах системи виявлення аномалій. Визначено, що Wazuh Agent зчитує Zeek notice.log, Wazuh Manager виконує декодування та застосування правил, після чого оброблені алерти передаються через Filebeat до Wazuh Indexer і відображаються у Wazuh Dashboard. Такий підхід дозволяє перетворити події Zeek на структуровані Wazuh-алерти, придатні для аналізу, пошуку, фільтрації, побудови візуалізацій та кореляції.

У розділі було описано логіку нормальної поведінки корпоративної мережі – baseline.

Окрему увагу було приділено загальній архітектурі мережі, передбачено сегментацію на VLAN, для контролю взаємодії між ними було спроєктовано використання ACL, які визначають дозволені та заборонені напрями трафіку. NAT було спроєктовано для імітації виходу в зовнішню мережу та моделювання взаємодії із зовнішньою машиною атаки.

Було спроєктовано набір кастомних Zeek-детекторів для виявлення різних типів поведінкових аномалій.

Крім того, було спроєктовано Wazuh-декодери, правила та кореляції подій. Для Zeek-подій передбачено базове правило, від якого наслідуються спеціалізовані правила для окремих типів аномалій.

Сформовано проєктну основу системи виявлення аномалій трафіку.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

У третьому розділі розгляно практичну реалізацію та налаштування проєктованої мережі з системою виявлення аномалій на основі поведінкового аналізу.

Практична частина роботи включає налаштування сегментованої мережевої інфраструктури, встановлення доцільних операційних систем для проєктованої мережі, розгортання Zeek сенсора та Wazuh SIEM платформи, їх налаштування та інтеграція, підготовку журналів у JSON форматі, формування baseline нормальної поведінки в корпоративній мережі, набір деректорія для виявлення поведінкових аномалій, правила обробки та декодування notice.log у Wazuh.

У межах тестування перевіряється виявлення таких типів аномальної активності, як нова комунікаційна пара, новий сервіс для відомої пари хостів, SSH-доступ не з адміністративного сегмента, активність поза нормальними годинами, DNS-аномалії, сканування портів, network beaconing, DNS query flood та передача/отримання великої кількості трафіку. Окремо перевіряється логіка кореляції подій, яка дозволяє пов’язувати окремі спрацювання правил у більш змістовні сценарії, наприклад C2-активність з подальшою передачею великого обсягу даних або розвідку мережі після появи нового напряму комунікації.

3.1. Вибір програмного середовища та операційних систем для розгортання проєкту

Для розгортання проєктованої корпоративної мережі було обрано середовище GNS3 та VMWare Workstation.

GNS3 використовується для моделювання мережевої інфраструктури проєктованої корпоративної мережі. У межах цієї роботи це середовище застосовано для розгортання віртуального маршрутизатора Cisco IOSv, комутатора Cisco IOU L2, створення VLAN-сегментів, налаштування маршрутизації між мережами, списків контролю доступу, NAT та SPAN. Завдяки цьому в GNS3 буде

змодельовано логіку передавання трафіку між користувацьким, серверним, адміністративним, моніторинговим і зовнішнім сегментами мережі.

VMware Workstation використовується для запуску віртуальних машин, які виконують ролі окремих вузлів мережі: web-сервера, користувацької станції, адміністративного хоста, Zeek-сенсора, Wazuh-сервера, GNS3VM для запуску всередині неї віртуальних мережевих пристроїв, зокрема Cisco IOSv та Cisco IOU L2, та зовнішнього атакувального вузла. Таким чином, GNS3 забезпечує моделювання мережевої частини, а VMware Workstation - роботу операційних систем і сервісів, необхідних для тестування системи виявлення аномалій.

Для практичної реалізації системи виявлення аномалій трафіку, потрібно підібрати операційні системи, які працюватимуть відповідно до ролей кожного вузла корпоративної мережі. Основними критеріями вибору були сумісність із використовуваними інструментами, стабільність роботи, можливість мережевого налаштування, підтримка необхідних служб, а також зручність інтеграції Zeek та Wazuh.

Оскільки система будується на основі пасивного аналізу мережевого трафіку та подальшої обробки подій безпеки, для ключових компонентів було обрано Linux-системи. Це пояснюється тим, що Zeek, Wazuh Agent, Wazuh Manager, Filebeat та більшість мережевих служб найкраще підтримуються саме в Linux-середовищі. Крім того, Linux забезпечує гнучке керування мережевими інтерфейсами, журналами, службами та правами доступу, що є важливим для побудови проєкту корпоративної мережі.

Для серверний та моніторингових платформ проєкту корпоративної мережі було обрано операційні системи на базі Ubuntu-Server. Вибір зумовлений стабільністю цієї платформи, широкою підтримкую серверного програмного забезпечення, зручністю адміністрування мережевих служб і сумісністю з основними компонентами проєкту, зокрема Zeek, Wazuh Agent, Wazuh Manager та Filebeat.

Для серверного вузла було також використано операційну систему на базі Ubuntu Server. На цьому вузлі подалі буде розгорнуто Web-сервер NGINX,

тестовий веб додаток DVWA, а також службу DNS -сервера dnsmasq. Зокрема NGINX та DVWA використовуються для створення HTTP звернень та взаємодії між користувацьким та серверним сегментом мережі. DNS -сервіс dnsmasq використовується для генерації DNS -запитів, що дозволяє сформувати нормальну поведінку мережеі baseline та надалі перевірити роботу Zeek-детекторів і правил Wazuh на відхилення від цієї поведінки.

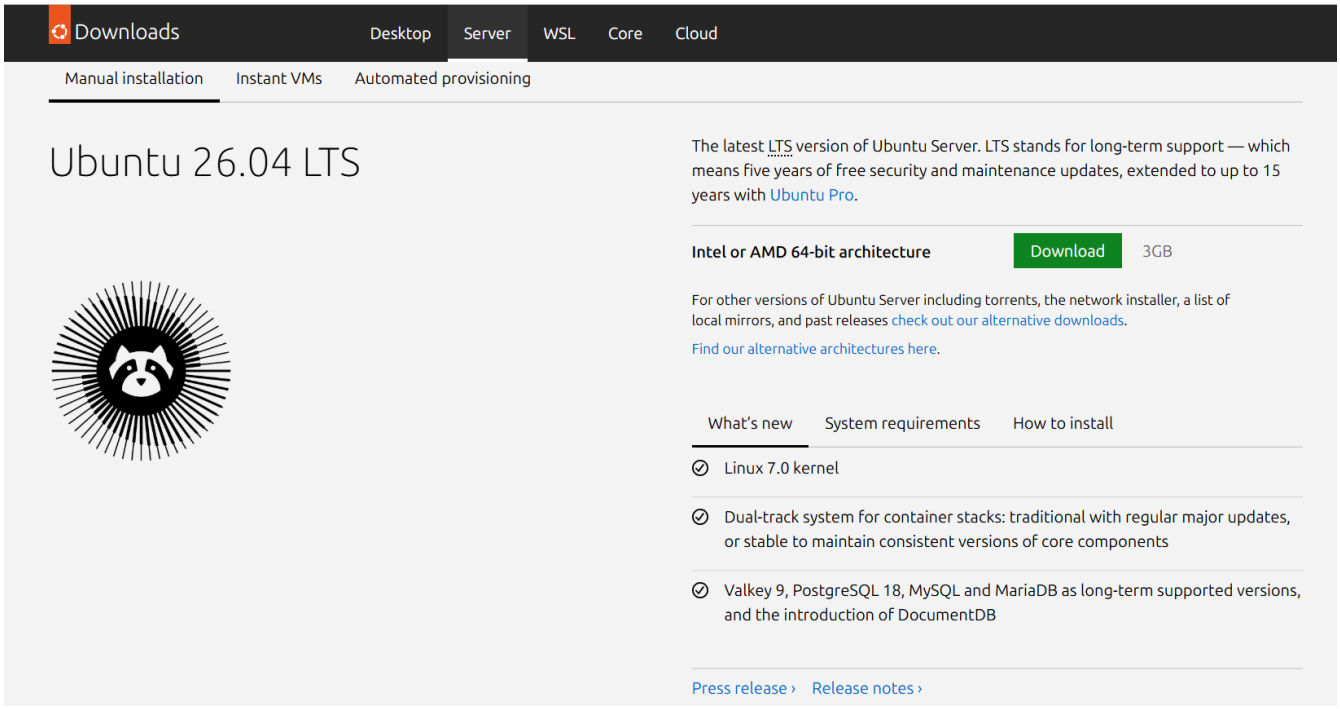


Рисунок 3.1 – Офіційне джерело завантаження образу ОС Ubuntu Server 26.04

Для зовнішньої машини атаки було використано Kali Linux, оскільки ця операційна система спеціалізується на тестуванні безпеки та містить набір інструментів для моделювання мережевої розвідки, сканування портів, перевірки web-сервісів і генерації аномального трафіку. У межах цієї роботи Kali Linux використовується не як частина системи захисту, а як джерело контрольованої атакувальної активності для перевірки спрацювання Zeek-детекторів і Wazuh-правил.



Рисунок 3.2 – Офіційне джерело завантаження образу ОС Kali Linux

Користувацький та адміністративний сегменти лабораторного стенду було розгорнуто на базі Ubuntu Desktop. Вибір цієї операційної системи зумовлений наявністю графічного інтерфейсу, зручністю виконання типових користувацьких дій і простотою адміністрування вузлів під час тестування. Користувацька робоча станція використовується для імітації звичайної активності в корпоративній мережі, зокрема звернень до web-сервера, DNS-запитів і мережевої взаємодії з дозволеними сервісами. Адміністративний вузол застосовується для керування корпоративною мережою та моделювання легітимного адміністративного доступу до всіх сегментів мережі.

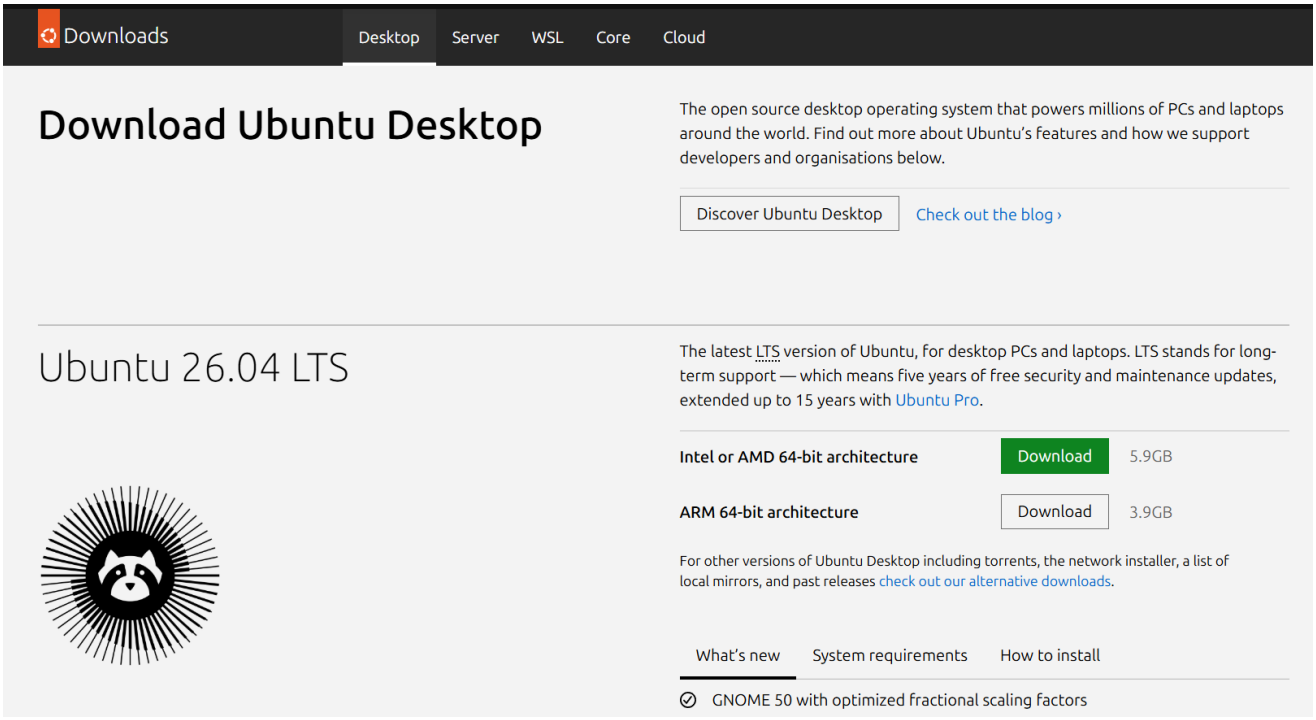


Рисунок 3.3 – Офіційне джерело завантаження образу ОС Ubuntu Desktop 26.04

Для керування мережевою інфраструктурою корпоративної мережі було використано віртуалізовані образи мережевого обладнання Cisco. Як маршрутизатор було обрано образ Cisco IOSv на базі Cisco IOS Software, IOSv Software (VIOS-ADVENTERPRISEK9-M), Version 15.9(3)M6. Для комутаційного рівня було використано образ Cisco IOU L2 з системним образом i86bi_linux_l2-advipservicesk9-ms.high_iron.

Вибір таких платформ зумовлений тим, що вони підтримують основні функції, необхідні для побудови проєктної корпоративної мережі. Маршрутизатор Cisco IOSv використовується для організації маршрутизації між мережевими сегментами, налаштування списків контролю доступу, реалізації NAT та керування взаємодією між внутрішніми VLAN і зовнішнім сегментом. Це дозволяє моделювати як звичайну міжмережеву взаємодію, так і контрольовані сценарії тестування, пов'язані зі скануванням, зверненням до внутрішніх сервісів або виходом у зовнішню мережу.

Таблиця 3.1

Огляд технологій на обраних апаратних платформах

Компонент	Образ	Основні функції у проєкті
Маршрутизатор	Cisco IOSv, VIOS-ADVENTERPRISEK9-M, 15.9(3)M6	Маршрутизація між VLAN, ACL, NAT та взаємодія із зовнішніми сегментами
Комутатор	Cisco IOU L2, i86bi_linux_l2-advipservicesk9-ms.high_iron	VLAN, access/trunk порти, SPAN, підключення вузлів мережі

3.2. Налаштування мережевої інфраструктури проєкту корпоративної мережі

Після вибору операційних систем і мережевих платформ було виконано налаштування мережевої інфраструктури проєктованої корпоративної мережі. На цьому етапі було реалізовано логічне розділення мережі на VLAN, налаштовано маршрутизацію між сегментами, списки контролю доступу, NAT для імітації взаємодії із зовнішнім сегментом, та SPAN для передавання копії трафіку на мережевий сенсор пасивного аналізу Zeek.

Узагальнену топологію корпоративної мережі, що використовується для реалізації системи виявлення аномалій трафіку, наведено на рисунку 3.4

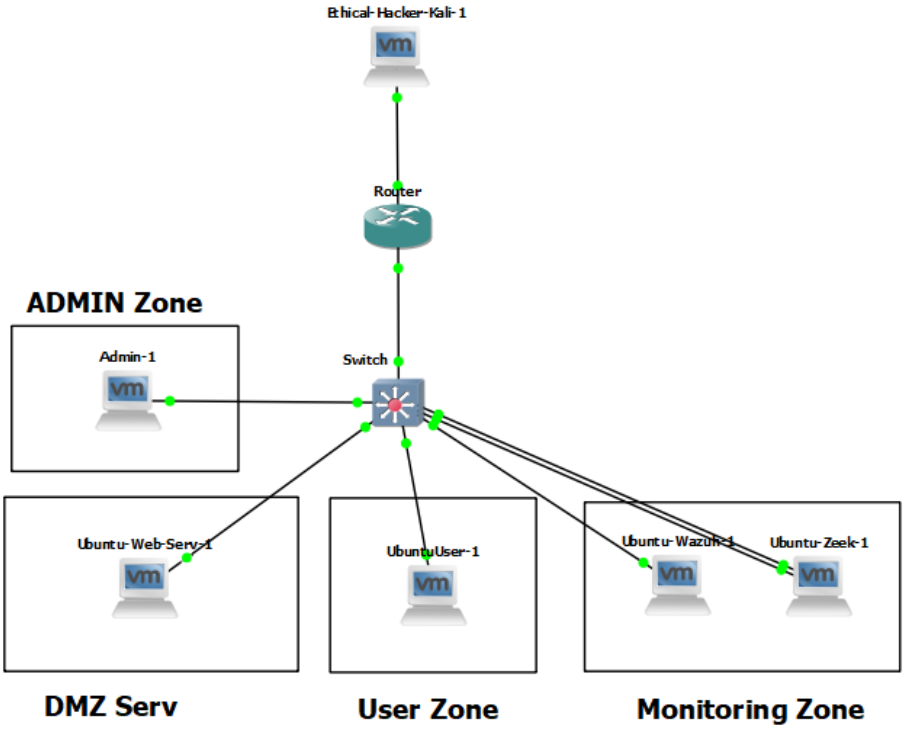


Рисунок 3.4 – Узагальнена топологія проєктованої корпоративної мережі

Топологія відображає поділ мережі на функціональні сегменти такі як: серверний, користувацький, моніторинговий, адміністративний та зовнішній. Центральними компонентами є маршрутизатор та комутатор.

Адресацію та призначення основних сегментів корпоративної мережі наведено в таблиці 3.2.

Таблиця 3.2

Адресація сегментів корпоративної мережі

Сегмент	VLAN/мережа	Основні вузли	Призначення
Server VLAN	VLAN 10 / 192.168.10.0 / 28	192.168.10.10	Серверний вузол: NGINX, DNSMASQ, DVWA
User VLAN	VLAN 20 / 192.168.20.0 / 24	192.168.20.2	Користувацька робоча станція
Monitoring VLAN	VLAN 30 / 192.168.30.0 / 29	192.168.30.2-3	Wazuh Manager, Zeek сенсор
Admin VLAN	VLAN 40 / 192.168.40.0 / 30	192.168.40.2	Адміністративний хост
Зовнішній сегмент	10.255.255.0 / 24	10.255.255.254	Зовнішній сегмент користування та атакування в корпоративній мережі

Для логічного розділення внутрішньої мережі на функціональні зони на комутаторі було створено VLAN-зони. Фрагмент конфігурації наведено на рисунку 3.5.

```
Switch#show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Et1/2, Et1/3, Et2/0, Et2/1
                                                Et2/2, Et2/3, Et3/0, Et3/1
                                                Et3/2
10   VLAN0010                         active    Et0/1
20   VLAN0020                         active    Et0/2
30   VLAN0030                         active    Et0/3, Et1/0
40   VLAN0040                         active    Et1/1
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
```

Рисунок 3.5 – Налаштовані VLAN сегменти на комутаторі

Після створення VLAN було виконано призначення портів комутатора відповідно до сегментів, порти до яких підключені концеві вузли налаштовано в режимі access, та відповідно між вузлами передачі трафіку в мережі налаштовано trunk режим.

```
interface Ethernet0/0
 switchport trunk allowed vlan 10,20,30,40
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
interface Ethernet0/1
 switchport access vlan 10
 switchport mode access
!
interface Ethernet0/2
 switchport access vlan 20
 switchport mode access
!
interface Ethernet0/3
 switchport access vlan 30
 switchport mode access
!
interface Ethernet1/0
 switchport access vlan 30
 switchport mode access
!
interface Ethernet1/1
 switchport access vlan 40
 switchport mode access
```

Рисунок 3.6 – Призначені порти комутатора до VLAN сегментації

На маршрутизаторі Cisco було налаштовано підінтерфейси для кожного VLAN. Кожен інтерфейс має власний тег 802.q та айпі адресу, яка використовується як шлюз за замовчуванням для відповідного сегмента. Ці налаштування наведені на рисунку 3.7.

```
Router#show run | section interface
interface GigabitEthernet0/0
 no ip address
 duplex auto
 speed auto
 media-type rj45
interface GigabitEthernet0/0.10
 encapsulation dot1Q 10
 ip address 192.168.10.1 255.255.255.240
 ip access-group WEB_SERV out
 ip nat inside
 ip virtual-reassembly in
interface GigabitEthernet0/0.20
 encapsulation dot1Q 20
 ip address 192.168.20.1 255.255.255.0
 ip access-group ANTI_SPOOFING_USER_VLAN in
 ip nat inside
 ip virtual-reassembly in
interface GigabitEthernet0/0.30
 encapsulation dot1Q 30
 ip address 192.168.30.1 255.255.255.248
 ip access-group Wazuh_Zeek out
interface GigabitEthernet0/0.40
 encapsulation dot1Q 40
 ip address 192.168.40.1 255.255.255.248
 ip access-group ADMIN_OUT in
 ip access-group ADMIN_IN out
interface GigabitEthernet0/1
 ip address 10.255.255.1 255.255.255.0
 ip nat outside
 ip virtual-reassembly in
 duplex auto
 speed auto
 media-type rj45
```

Рисунок 3.7 – Налаштування підінтерфейсів маршрутизатора

Для контролю взаємодії між мережевими сегментами було налаштовано списки контролю доступу ACL. Вони визначають дозволені та заборонені напрями трафіку між VLAN. У межах цієї роботи ACL також використовуються для створення умов тестування, за яких окремі напрями взаємодії залишаються доступними для моделювання мережевої розвідки, сканування портів та звернення до внутрішніх сервісів. Налаштування таких списків доступу зображені на рисунку 3.8.

```
Router#show access-lists
Standard IP access list 10
    10 permit 192.168.40.2
Standard IP access list 20
    10 permit 192.168.20.0, wildcard bits 0.0.0.255
Extended IP access list ADMIN_IN
    10 permit tcp 192.168.30.0 0.0.0.7 192.168.40.0 0.0.0.7 established
    20 permit tcp 192.168.10.0 0.0.0.15 192.168.40.0 0.0.0.7 established
Extended IP access list ADMIN_OUT
    10 permit udp 192.168.40.0 0.0.0.7 any eq domain (5 matches)
    20 permit tcp 192.168.40.0 0.0.0.7 any eq www
    30 permit tcp 192.168.40.0 0.0.0.7 any eq 443
    40 permit tcp 192.168.40.0 0.0.0.7 any eq 22
    50 permit icmp 192.168.40.0 0.0.0.7 any echo
    60 permit tcp 192.168.40.0 0.0.0.7 any eq 8080
    70 deny ip any any
Extended IP access list ANTI_SPOOFING_USER_VLAN
    10 deny ip 192.168.40.0 0.0.0.7 any
    20 deny ip 192.168.30.0 0.0.0.7 any
    30 permit ip any any (4 matches)
Extended IP access list WEB_SERV
    10 permit tcp any 192.168.10.0 0.0.0.15 eq 443
    20 permit tcp any 192.168.10.0 0.0.0.15 eq 8080
    30 permit tcp any 192.168.10.0 0.0.0.15 eq www
    40 permit tcp host 192.168.40.2 192.168.10.0 0.0.0.15 eq 22
    50 permit tcp any 192.168.10.0 0.0.0.15 established
    60 permit tcp any 192.168.10.0 0.0.0.15 eq domain
    70 permit udp any 192.168.10.0 0.0.0.15 eq domain
    80 permit udp 192.168.10.0 0.0.0.15 any
    90 permit tcp any 192.168.10.0 0.0.0.15 range 1 100
Extended IP access list Wazuh_Zeek
    10 permit tcp host 192.168.40.2 192.168.30.0 0.0.0.7 eq 22
    20 permit tcp host 192.168.40.2 192.168.30.0 0.0.0.7 eq 443
    30 permit tcp host 192.168.40.2 192.168.30.0 0.0.0.7 eq www
    40 permit tcp host 192.168.40.2 192.168.30.0 0.0.0.7 eq 5601
    50 permit tcp host 192.168.20.2 192.168.30.0 0.0.0.7 eq 22
Router#
```

Рисунок 3.8 – Налаштування списків доступу для контролю трафіку між сегментами

Для взаємодії зовнішнього сегменту з внутрішнім, було налаштовано NAT. У межах проєктованої корпоративної мережі він використовується для імітації виходу у зовнішню мережу та взаємодії з Kali-машиною, яка застосовується для генерації контрольованої атакувальної активності, а також для взаємодії з дозволеними сервісами сегменту Веб-сервера. Дані налаштування наведені на рисунку 3.9.

```
Router#show run int gig0/1
Building configuration...

Current configuration : 159 bytes
!
interface GigabitEthernet0/1
 ip address 10.255.255.1 255.255.255.0
 ip nat outside
 ip virtual-reassembly in
 duplex auto
 speed auto
 media-type rj45
end

Router#show run | section nat
 ip nat inside
 ip nat inside
 ip nat outside
ip nat inside source list 20 interface GigabitEthernet0/1 overload
ip nat inside source static 192.168.10.10 10.255.255.10
```

Рисунок 3.9 – Налаштована nat трансляції адреси для імітації виходу в зовнішню мережу

Одне з найголовніших налаштувань даної проєктованої корпоративної мережі – налаштування SPAN механізму для передавання копії мережевого трафіку на Zeek-сенсор. Він отримує копію трафіку з визначених VLAN сегментів і виконує пасивний аналіз без втручання в передавання пакетів. Дані налаштування наведені на рисунку 3.10.

```
Switch#show monitor session 1
Session 1
---------
Type                   : Local Session
Source VLANs           :
    Both               : 10,20
Destination Ports      : Et3/3
    Encapsulation      : Native
```

Рисунок 3.10 – Налаштований SPAN механізм для копіювання трафіку на мережевий аналізатор

Таким чином у межах налаштування мережевої інфраструктури було реалізовано VLAN сегментацію, trunk з’єднання між комутатором і маршрутизатором, маршрутизацію між VLAN, ACL, NAT та SPAN. Отримана мережева основа дозволяє надалі розгорнути Zeek-сенсор, організувати збір копії трафіку, сформувати baseline нормальної поведінки в корпоративній мережі та передавати виявленні notice події до Wazuh.

3.3. Розгортання та налаштування Zeek-сенсора

На етапі коли мережа є функціональною та виконане налаштування дозволяє ефективно використовувати корпоративну мережу, можна переходити до розгортання Zeek-сенсора. Як вже було описано в попередньому розділі, Zeek працює в пасивному режимі та отримує копію мережевого трафіку через механізм SPAN, налаштований на комутаторі.

Перед встановлення Zeek-сенсора було оновлено системні пакети та підготовлено середовище для встановлення необхідного програмного забезпечення.

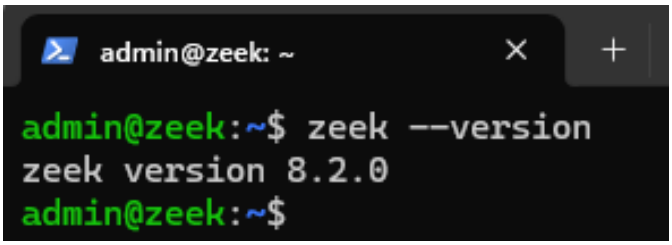


Рисунок 3.11 – Виконана інсталяції Zeek аналізатора на операційну систему

Результат виконання підтверджує, що Zeek успішно встановлений у системі та готовий до подальшого налаштування як сенсор пасивного аналізу мережевого трафіку.

Для роботи було визначено мережевий інтерфейс, на який надходить копія мережевого трафіку зі SPAN сесії. Перевірку доступних мережевих інтерфейсів виконано через команду: ip a, та наведено на рисунку 3.12.

```
admin@zeek:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:2f:c2:8c brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.30.3/29 brd 192.168.30.7 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe2f:c28c/64 scope link
       valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:2f:c2:96 brd ff:ff:ff:ff:ff:ff
    altname enp2s5
    inet6 fe80::20c:29ff:fe2f:c296/64 scope link
       valid_lft forever preferred_lft forever
```

Рисунок 3.12 – Виконання перевірки мережевих інтерфейсів доступних для пасивного аналізу трафіку

Таким чином, після визначення мережевого інтерфейса доступного для налаштування пасивного аналізу трафіку, переходимо до налаштування потрібного інтерфейсу для аналізування копійованого трафіку. Для даних налаштувань переходимо до конфігураційного файлу node.cfg. Налаштування мережевого інтерфейсу на Zeek аналізатора наведено на рисунку 3.13.

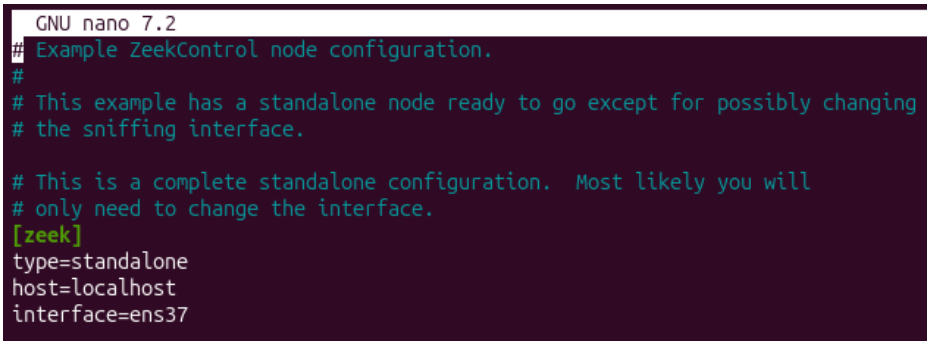


Рисунок 3.13 – Налаштування мережевого інтефейсу на моніторинг

У цьому фрагменті параметр type=standalone означає, що Zeek працює в автономному режимі на одному вузлі. Параметр host=localhost вказує, що процес Zeek запускається локально на моніторинговому вузлі. Параметр interface визначає мережевий інтерфейс, на який надходить копія трафіку зі SPAN-сесії.

Після виконання налаштування моніторингу на мережевому інтерфейсі перевіряємо коректність конфігурацію Zeek та запускаємо мережевий аналізатор на аналізування копій мережевого трафіку.

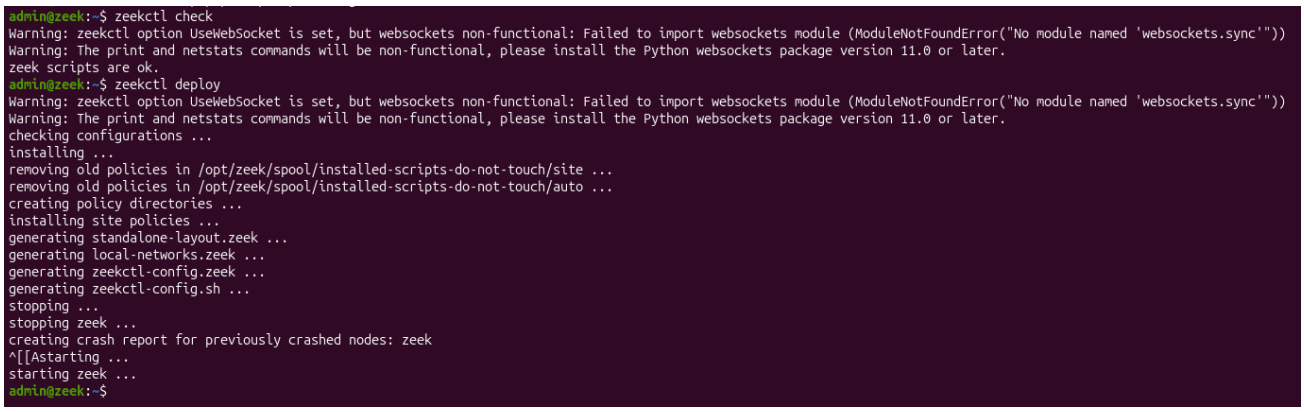


Рисунок 3.14 – Перевірка налаштованої конфігурації та запуск сенсора

Після запуску сенсора, перевіряємо чи він формує логи мережевого трафіку.



Рисунок 3.15 – Перевірка створених журналів на основі аналізу мережевого трафіку

У результаті формуються журнали логів, зокрема Conn.log, який містить в собі інформацію про мережеві з’єднання, Dns.log, який містить в собі dns запити, http.log, який містить в собі http-взаємодію.

Таким чином, у межах цього етапу було підтвердженно встановлення Zeek, визначено інтерфейси прослуховування для отримання копії трафіку через SPAN, виконане базове налаштування сенсора, перевірено його запуск та налаштування.

3.4. Розгортання Wazuh: manager, indexer, dashboard

Після розгортання Zeek-сенсора наступним етапом є налаштування Wazuh як центральної платформи обробки подій безпеки. У межах проєктованої системи Zeek відповідає за пасивний аналіз мережевого трафіку та формування журналів, а Wazuh використовується для приймання подій, їх декодування, застосування правил, формування алертів, зберігання результатів і подальшої візуалізації.

У межах проєктованої корпоративної мережі Wazuh разом Zeek розміщено в моніторинговому сегменті. Таке розміщення дозволяє відокремити компоненти моніторингу від користувацьких і серверних вузлів, а також централізовано приймати події від систем, що беруть участь у виявленні аномалій трафіку.

Основними компонентами Wazuh, які використовуються в цій роботі наведені в таблиці 3.3.

Таблиця 3.3

Основні компоненти Wazuh

Компонент	Призначення
Wazuh Manager	Приймання подій від агентів, декодування, застосування правил і формування алертів
Wazuh Indexer	Зберігання оброблених подій і забезпечення пошуку по них
Wazuh Dashboard	Веб-інтерфейс для перегляду, фільтрації та візуалізації подій
Filebeat	Передавання оброблених Wazuh-алертів від Manager до Indexer
Wazuh Agent	Передавання журналів із контрольованих вузлів до Wazuh Manager

Важливо зазначити, що Filebeat не замінює Wazuh Agent і не приймає журнали Zeek напряму, його роль передавати вже оброблені події і алерти від Wazuh manager до Wazuh indexer.

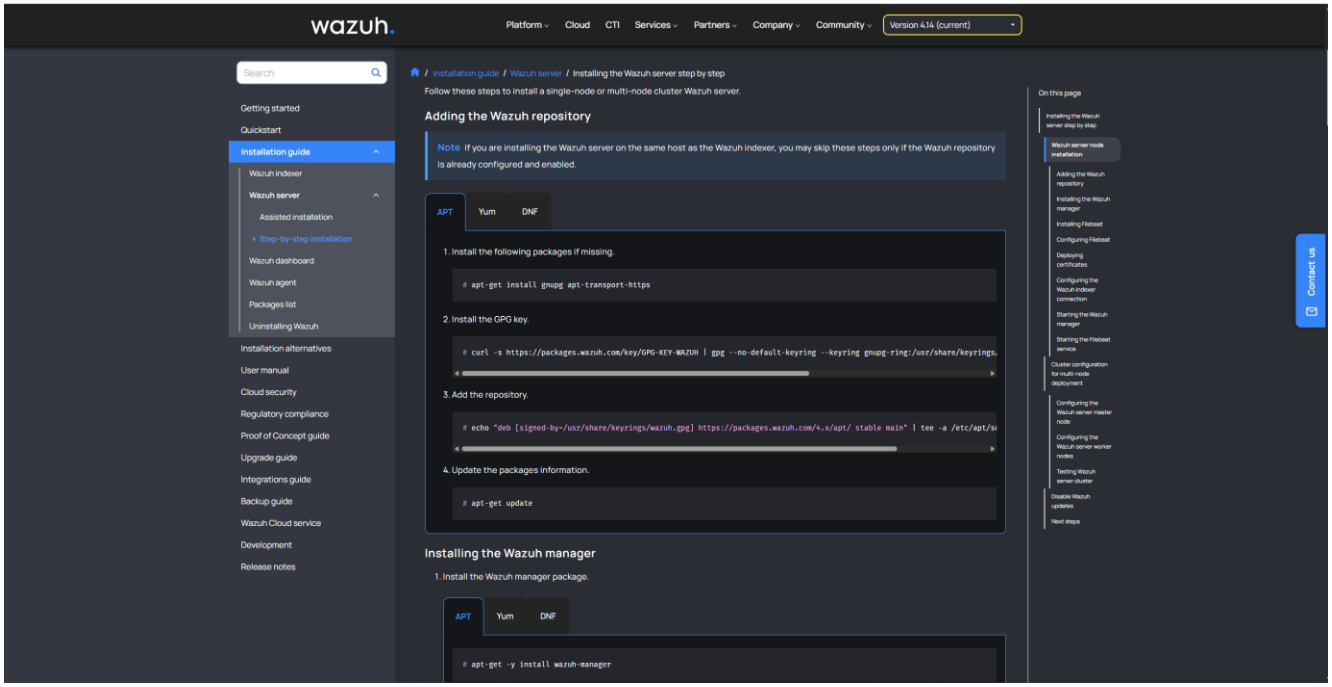


Рисунок 3.16 – Офіційне джерело Wazuh з інструкцією інсталяції системи

Після встановлення компонентів Wazuh було виконано перевірку стану основних служб, насамперед перевірено роботу Wazuh Manager, оскільки саме він є центральним компонентом обробки подій.

```
admin@wazuh:~$ sudo systemctl status wazuh-manager
● wazuh-manager.service - Wazuh manager
     Loaded: loaded (/usr/lib/systemd/system/wazuh-manager.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-06-14 14:46:43 CEST; 6s ago
    Process: 1585 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 223 (limit: 7107)
     Memory: 654.7M (peak: 660.4M)
        CPU: 15.525s
     CGroup: /system.slice/wazuh-manager.service
             ├─1475 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
             ├─1476 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
             ├─1477 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
             ├─1480 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
             ├─1483 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
             ├─1528 /var/ossec/bin/wazuh-authd
             ├─1543 /var/ossec/bin/wazuh-db
             ├─1579 /var/ossec/bin/wazuh-execd
             ├─1703 /var/ossec/bin/wazuh-analysisd
             ├─1718 /var/ossec/bin/wazuh-syscheckd
             ├─1765 /var/ossec/bin/wazuh-remoted
             ├─1803 /var/ossec/bin/wazuh-logcollector
             ├─1831 /var/ossec/bin/wazuh-monitord
             ├─1841 /var/ossec/bin/wazuh-modulesd
             └─2987 dpkg -s iptables

Jun 14 14:46:38 wazuh env[1585]: Started wazuh-syscheckd...
Jun 14 14:46:39 wazuh env[1585]: Started wazuh-remoted...
Jun 14 14:46:40 wazuh env[1585]: Started wazuh-logcollector...
Jun 14 14:46:40 wazuh env[1585]: Started wazuh-monitord...
Jun 14 14:46:40 wazuh env[1838]: 2026/06/14 14:46:40 wazuh-modulesd:router: INFO: Loaded router module.
Jun 14 14:46:40 wazuh env[1838]: 2026/06/14 14:46:40 wazuh-modulesd:content_manager: INFO: Loaded content_manager module.
Jun 14 14:46:40 wazuh env[1838]: 2026/06/14 14:46:40 wazuh-modulesd:inventory-harvester: INFO: Loaded Inventory harvester module.
Jun 14 14:46:41 wazuh env[1585]: Started wazuh-modulesd...
Jun 14 14:46:43 wazuh env[1585]: Completed.
Jun 14 14:46:43 wazuh systemd[1]: Started wazuh-manager.service - Wazuh manager.
admin@wazuh:~$
```

Рисунок 3.17 – Перевірка роботи Wazuh Manager

Результат виконання команди підтверджує, що служба Wazuh Manager запущена та працює коректно.

```
admin@wazuh:~$ sudo /var/ossec/bin/wazuh-control status
wazuh-clusterd not running...
wazuh-modulesd is running...
wazuh-monitord is running...
wazuh-logcollector is running...
wazuh-remoted is running...
wazuh-syscheckd is running...
wazuh-analysisd is running...
wazuh-maild not running...
wazuh-execd is running...
wazuh-db is running...
wazuh-authd is running...
wazuh-agentlessd not running...
wazuh-integratord not running...
wazuh-dbd not running...
wazuh-csyslogd not running...
wazuh-apid is running...
admin@wazuh:~$
```

Рисунок 3.18 – Перевірка основних процесів Wazuh

Ця команда дозволяє переглянути стан основних процесів Wazuh, зокрема процесів приймання подій, аналізу, застосування правил і взаємодії з агентами.

Наступним компонентом виступає Wazuh Indexer. Він використовується для зберігання оброблених подій безпеки та забезпечує можливість швидкого пошуку, фільтрації й подальшого відображення даних у Wazuh Dashboard.

```
admin@wazuh:~$ sudo systemctl status wazuh-indexer
● wazuh-indexer.service - wazuh-indexer
     Loaded: loaded (/usr/lib/systemd/system/wazuh-indexer.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-06-14 14:46:54 CEST; 7min ago
       Docs: https://documentation.wazuh.com
   Main PID: 969 (java)
      Tasks: 72 (limit: 7107)
     Memory: 1.5G (peak: 1.5G)
        CPU: 1min 13.075s
     CGroup: /system.slice/wazuh-indexer.service
             └─969 /usr/share/wazuh-indexer/jdk/bin/java -Xshare:auto -Dopensearch.networkaddress.cache
```

Рисунок 3.19 – Перевірка роботи Wazuh Indexer

Після цього було перевірено роботу Wazuh Dashboard. Цей компонент забезпечує графічний доступ до подій і алертів, дозволяє виконувати пошук за полями, аналізувати події за часовими інтервалами, переглядати деталі спрацювання правил і будувати візуалізації для подальшого аналізу результатів роботи системи.

```
admin@wazuh:~$ sudo systemctl status wazuh-dashboard
● wazuh-dashboard.service - wazuh-dashboard
     Loaded: loaded (/etc/systemd/system/wazuh-dashboard.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-06-14 14:46:12 CEST; 9min ago
   Main PID: 841 (node)
      Tasks: 11 (limit: 7107)
     Memory: 295.1M (peak: 398.5M)
        CPU: 14.774s
     CGroup: /system.slice/wazuh-dashboard.service
             └─841 /usr/share/wazuh-dashboard/node/bin/node --no-warnings --max-http-header-size=
```

Рисунок 3.20 – Перевірка роботи Wazuh Dashboard

Окремо було перевірено роботу Filebeat, який забезпечує передавання оброблених алертів до Wazuh Indexer.

```
admin@wazuh:~$ sudo systemctl status filebeat
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
     Loaded: loaded (/usr/lib/systemd/system/filebeat.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-06-14 14:46:13 CEST; 10min ago
       Docs: https://www.elastic.co/products/beats/filebeat
   Main PID: 961 (filebeat)
      Tasks: 8 (limit: 7107)
     Memory: 54.3M (peak: 54.8M)
        CPU: 437ms
     CGroup: /system.slice/filebeat.service
             └─961 /usr/share/filebeat/bin/filebeat --environment systemd -c /etc/filebeat/filebeat.yml --
```

Рисунок 3.21 – Перевірка стану служби Filebeat

Після перевірки всіх служб виконано перевірку веб-інтерфейсу Wazuh Dashboard.

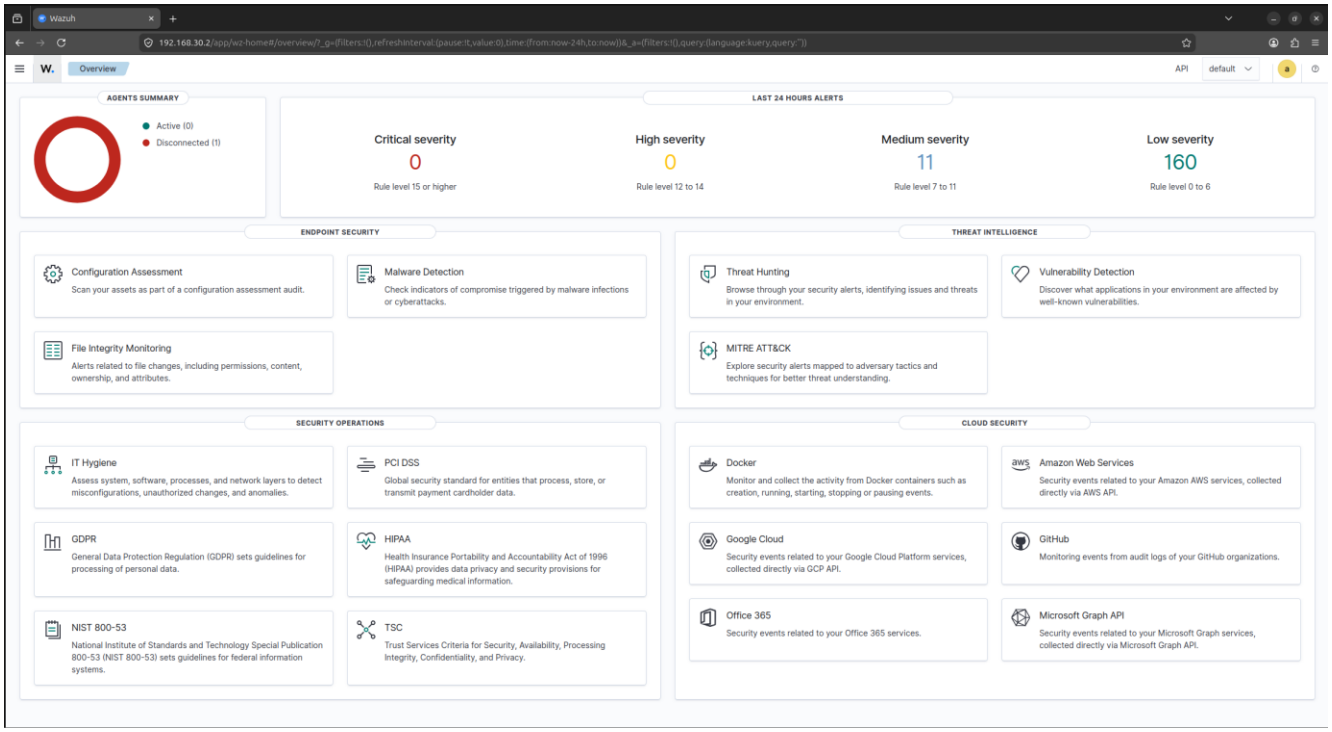


Рисунок 3.22 – Перевірка веб-інтерфейсу Wazuh Dashboard

Таким чином, у межах цього етапу було розгорнуто та перевірено роботу основних компонентів Wazuh: Wazuh Manager, Wazuh Indexer, Wazuh Dashboard і Filebeat. Також було підтверджено можливість роботи з агентами, які використовуються для передавання журналів із контрольованих вузлів до Wazuh Manager. Це створює основу для наступного етапу - інтеграції Zeek із Wazuh, пересилання журналу notice.log, створення кастомних декодерів і перевірки формування структурованих алертів.

3.5. Інтеграція Zeek із Wazuh, пересилання журналів, базове декодування та тестування

Основним джерелом Zeek подій для інтеграції з Wazuh є журнал notice.log. Саме в ньому фіксуються події, які створюються Zeek або кастомними детекторами для виявлення підозрілої або нетипової активності. Для передавання журналів Zeek до Wazuh використовується Wazuh Agent, встановлений на вузлі із Zeek-сенсором. Агент не виконує аналіз мережевого трафіку самостійно. Його завдання полягає у зчитуванні визначених журналів і передаванні подій до Wazuh Manager для подальшої обробки. Для передавання журналів Zeek до Wazuh було використано Wazuh Agent, встановлений на Zeek сенсорі.

```
admin@zeek:~$ sudo systemctl status wazuh-agent
[sudo] password for admin:
● wazuh-agent.service - Wazuh agent
     Loaded: loaded (/usr/lib/systemd/system/wazuh-agent.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-06-14 15:26:07 CEST; 1min 0s ago
    Process: 1047 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 32 (limit: 2205)
     Memory: 398.6M (peak: 400.8M)
        CPU: 22.108s
     CGroup: /system.slice/wazuh-agent.service
```

Рисунок 3.23 – Перевірка роботи встановленого на Zeek Wazuh Agent

Першим етапом інтеграції стало встановлення Wazuh Agent на Zeek-вузлі. Після встановлення агент було налаштовано на взаємодію з Wazuh Manager, який у межах проєктованої корпоративної мережі розміщено в моніторинговому сегменті за адресою 192.168.30.2.

```
 GNU nano 7.2
<!--
  Wazuh - Agent - Default configuration for ubuntu 24.04
  More info at: https://documentation.wazuh.com
  Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <client>
    <server>
      <address>192.168.30.2</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
    <config-profile>ubuntu, ubuntu24, ubuntu24.04</config-profile>
    <notify_time>20</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>
  </client>
```

Рисунок 3.24 – Налаштування підключення Zeek з Wazuh Agent до Wazuh Manager

У цьому фрагменті параметр address визначає IP-адресу Wazuh Manager, до якого агент передає події. Порт 1514 використовується для обміну подіями між агентом і менеджером, а параметр protocol визначає транспортний протокол передавання.

```
admin@wazuh:~$ sudo /var/ossec/bin/agent_control -l

Wazuh agent_control. List of available agents:
   ID: 000, Name: wazuh (server), IP: 127.0.0.1, Active/Local
   ID: 001, Name: zeek, IP: any, Active

List of agentless devices:
```

Рисунок 3.25 – Перевірка підключення Zeek сенсора до Wazuh Manager

Після того, як підключення виконано успішно, виконуємо налаштування агента, для зчитування notice.log та його подальше передавання до Wazuh Manager

```
<localfile>
 <log_format>json</log_format>
 <location>/opt/zeek/logs/current/notice.log</location>
</localfile>
```

Рисунок 3.26 – Налаштування Zeek для передавання notice подій до Wazuh Manager

У цьому фрагменті параметр log_format визначає формат журналу як JSON. Це відповідає попередньо налаштованому формату журналів Zeek і дозволяє Wazuh коректно обробляти події як структуровані записи. Параметр location вказує шлях до журналу Zeek notice.log, який має зчитуватися агентом. На етапі первинної перевірки важливо переконатися, що Zeek-події надходять до Wazuh у JSON-форматі та можуть бути оброблені системою.

Для цього налаштуємо базове правило Wazuh, яке буде спрацьовувати, для того, щоб надалі працювали кастомні правила.

```
<group name="zeek,">
  <rule id="100900" level="0">
    <decoded_as>json</decoded_as>
    <description>Zeek alerts</description>
  </rule>
```

Рисунок 3.27 – Базове правило Wazuh

Для перевірки використовується інструмент wazuh-logtest, який дозволяє протестувати окрему подію з журналу Zeek. Для цього використовуємо команду wazuh-logtest.

```
admin@wazuh:~$ sudo /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.14.5
Type one log per line

{ "ts": 1781169224.876952, "note": "SumStatsDetectors::SumStats_DNS_Query_Flood", "msg": "SumStats DNS repeated query/flood det

**Phase 1: Completed pre-decoding.
        full event: '{ "ts": 1781169224.876952, "note": "SumStatsDetectors::SumStats_DNS_Query_Flood", "msg": "SumStats DNS rep
for": 60.0 }'

**Phase 2: Completed decoding.
        name: 'json'
        parent: 'json'

**Phase 3: Completed filtering (rules).
        id: '100900'
        level: '0'
        description: 'Zeek alerts'
        groups: '['zeek']'
        firedtimes: '1'
        mail: 'False'
```

Рисунок 3.28 – Результат тестування

Результат тестування показав, що подія успішно проходить етап попереднього декодування. На другому етапі Wazuh визначає формат події як json, що підтверджує коректність налаштування журналів Zeek у JSON-форматі. На третьому етапі спрацьовує базове правило 100900 з описом Zeek alerts, яке використовується для приймання Zeek-подій у Wazuh.

Після налаштування зчитування журналу notice.log події Zeek передаються до Wazuh Manager за такою логікою: Wazuh Agent – Wazuh Manager – JSON – Правила Wazuh – Wazuh алерт

3.6. Побудова Baseline нормальної поведінки корпоративної мережі

Для формування baseline попередньо було згенеровано легітимний трафік у корпоративній мережі. Користувацький вузол звертався до внутрішнього web-сервера, виконував DNS-запити через сервер DNS/dnsmasq, взаємодіяв із дозволеними сервісами, а адміністративний доступ здійснювався з відповідного адміністративного сегмента. Така активність використовувалася як приклад нормальної поведінки мережі.

Журнал Conn.log використовується для визначення нормальних комунікаційних пар, типових портів, сервісів між вузлами. Журнал Dns.log використовується для формування списку доменів, які зустрічаються під час легітимної активності.

Обробкою активності та подальшим записом у файл нормальної поведінки виконується за допомогою Python скрипта, який збирає інформацію з двох логів та створює baseline з типом .Zeek

```
  GNU nano 7.2
#!/usr/bin/env python3
import json
import gzip
import ipaddress
from pathlib import Path
from collections import defaultdict

ZEEK_LOG_DIRS = [
    Path("/opt/zeek/logs/current"),
    Path("/opt/zeek/logs/2026-06-08"),
    Path("/opt/zeek/logs/2026-06-09"),
    Path("/opt/zeek/logs/2026-06-10"),
]
OUTPUT_FILE = "/home/admin/baselines_script/baseline_data.zeek"

INTERNAL_NETS = [
    ipaddress.ip_network("192.168.10.0/28"),
    ipaddress.ip_network("192.168.20.0/24"),
    ipaddress.ip_network("192.168.30.0/29"),
    ipaddress.ip_network("192.168.40.0/30"),
]

known_pairs = set()
known_services = set()
known_dns_domains = set()
active_hours = defaultdict(set)
transfer_sizes = defaultdict(list)


def is_internal(ip):
    try:
        ip_obj = ipaddress.ip_address(ip)
        return any(ip_obj in net for net in INTERNAL_NETS)
    except Exception:
        return False


def open_log(path):
    if str(path).endswith(".gz"):
        return gzip.open(path, "rt", encoding="utf-8", errors="ignore")
    return open(path, "r", encoding="utf-8", errors="ignore")


def read_json_logs(pattern):
    for log_dir in ZEEK_LOG_DIRS:
        if not log_dir.exists():
            continue

        for path in log_dir.glob(pattern):
            with open_log(path) as f:
                for line in f:
                    line = line.strip()
                    if not line or not line.startswith("{"):
                        continue
                    try:
                        yield json.loads(line)
                    except Exception:
                        continue

def process_conn():
    for rec in read_json_logs("conn*"):
        src = rec.get("id.orig_h")
```

Рисунок 3.29 – Python скрипт збору baseline нормальної поведінки

Наведений повний скрипт подана в додатку Б.

Логіка побудови baseline виглядає так: Zeek Conn.log – Dns.log – Python скрипт – baseline файл – Zeek детектори

```
  GNU nano 7.2
@load base/utils/site

module Baseline;

export {
    const known_pairs: set[string] = {
        "192.168.20.2 192.168.10.10",
        "192.168.40.2 192.168.10.10",
        "192.168.40.2 192.168.20.2",
        "192.168.40.2 192.168.30.3",
        "192.168.40.2 192.168.30.2",
        "192.168.30.3 192.168.30.1",
        "192.168.30.3 192.168.30.2",
        "192.168.30.3 192.168.40.2",
        "192.168.10.1 192.168.10.10",
        "192.168.20.1 192.168.20.2",
        "192.168.30.1 192.168.30.2",
    } &redef;

    const known_services: set[string] = {
        "192.168.20.2 192.168.10.10 53 udp",
        "192.168.20.2 192.168.10.10 53 tcp",
        "192.168.20.2 192.168.10.10 8080 tcp",
        "192.168.20.2 192.168.10.10 80 tcp",
        "192.168.40.2 192.168.10.10 22 tcp",
        "192.168.40.2 192.168.20.2 22 tcp",
        "192.168.40.2 192.168.30.2 22 tcp",
        "192
        "192.168.40.2 192.168.30.3 22 tcp",
    } &redef;

    const known_dns_domains: set[string] = {
        "dvwa.alexxx.net",
        "github.alexxx.net",
        "mail.alexxx.net",
        "10.10.168.192.in-addr.arpa",
        "_vlmcs._tcp.itdep.local",
        "_webdav._tcp.local",
        "au.download.windowsupdate.com",
        "crl.microsoft.com",
        "dvwa.alexxx.net",
        "fe2.update.microsoft.com",
        "isatap",
        "teredo.ipv6.microsoft.com",
        "time.windows.com",
        "update.test.local",
        "update.test.local.itdep.local",
        "watson.microsoft.com",
        "workgroup",
        "wpad",
        "wpad.itdep.local",
        "ws-a-1",
        "www.bing.com",
        "www.download.windowsupdate.com",
        "www.microsoft.com",
    } &redef;

    const normal_active_hours: table[string] of set[count] = {
        ["192.168.10.10"] = set(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23),

        ["192.168.20.2"] = set(8,9,10,11,12,13,14,15,16,17,18),
```

Рисунок 3.30 – Сформований результат нормальної поведінки в корпоративній мережі

Основні типи активності, які використовуються для формування baseline, наведено в таблиці 3.4.

Таблиця 3.4

Основні типи нормальної активності в мережі

Тип активності	Джерело	Призначення	Опис
HTTP-звернення	USER VLAN	SERVER VLAN	Звернення користувацького вузла до web-сервера NGINX/DVWA
DNS-запити	USER VLAN	SERVER VLAN	Запити до DNS/DNSmasq для імітації звичаної роботи користувача
Адміністративний доступ	ADMIN VLAN	Внутрішні вузли	Легітимне адміністрування систем із дозволеного сегмента
Службова взаємодія	Внутрушні сегменти	Дозволені вузли	Звичайна мережева взаємодія між визначеними хостами
Передавання даних у межах норми	USER VLAN / SERVER VLAN	Дозволені сервіси	Передавання трафіку в межах допустимих обсягів

Окремо в baseline враховуються нормальні години активності. Це дозволяє виявляти мережеві події, які відбуваються у нетиповий час. Наприклад, якщо користувацький вузол починає активно взаємодіяти з іншими системами вночі, така активність може бути класифікована як відхилення від нормальної поведінки.

Для статистичних детекторів у baseline або конфігурації детекторів задаються порогові значення. Вони використовуються для виявлення подій, які стають підозрілими не через сам факт появи, а через накопичення за певний проміжок часу. До таких випадків належать велика кількість унікальних портів під час сканування, повторювані DNS-запити до одного домену або передавання великого обсягу даних.

3.7. Реалізація кастомних Zeek-детекторів аномалій

Після побудови Baseline нормальної поведінки є можливість реалізації кастомних Zeek детекторів, які будуть порівнювати поточну мережеву активність з

виявленими подіями. У межах цієї роботи Zeek виконує роль не тільки аналізатору трафіку, а й як платформа реалізації поведінкової логіки виявлення аномалій.

У роботі реалізовано декілька груп детекторів, як статистичних так і динамічниих. Перша група базується на порівнянні активності з baseline, друга група базується на накопиченні подій за певний проміжок часу.

Основні кастомні детектори які були реалізовані в межах роботи було розміщено в site директорії Zeek, після чого вони були підключені через файл local .Zeek, це дозволяє Zeek автоматично застосовувати та завантажувати необхідні детектори під час запуску сенсора.

```
  GNU nano 7.2

@load policy/tuning/json-logs.zeek

@load ./baseline_data.zeek
@load ./common.zeek
@load ./new_pair_detector.zeek
@load ./ssh_policy_detector.zeek
#@load ./portscan_detector.zeek
#@load ./transfer_anomaly_detector.zeek
@load ./beaconing_detector.zeek
@load ./dns_anomaly_detector.zeek

@load ./sumstats_detectors.zeek
@load ./active_hours_detector.zeek

@load ./new_service_detector.zeek
@load ./suspicious_user_agent_detector.zeek

#filters
@load ./common_filters
```

Рисунок 3.31 – Підключеня реалізованих Zeek-детекторів до файлу local.Zeek

Оскільки повний код детекторів займає значний обсяг, у цьому розділі наведено лише фрагмент одного із скриптів, такий скрипт детектора виявляє динамічні відхилення, які використовуються для таких аномалій, як: сканування портів, dns-query-flood, beaconing, NXDOMAIN Spike та пересилання великого обсягу даних.

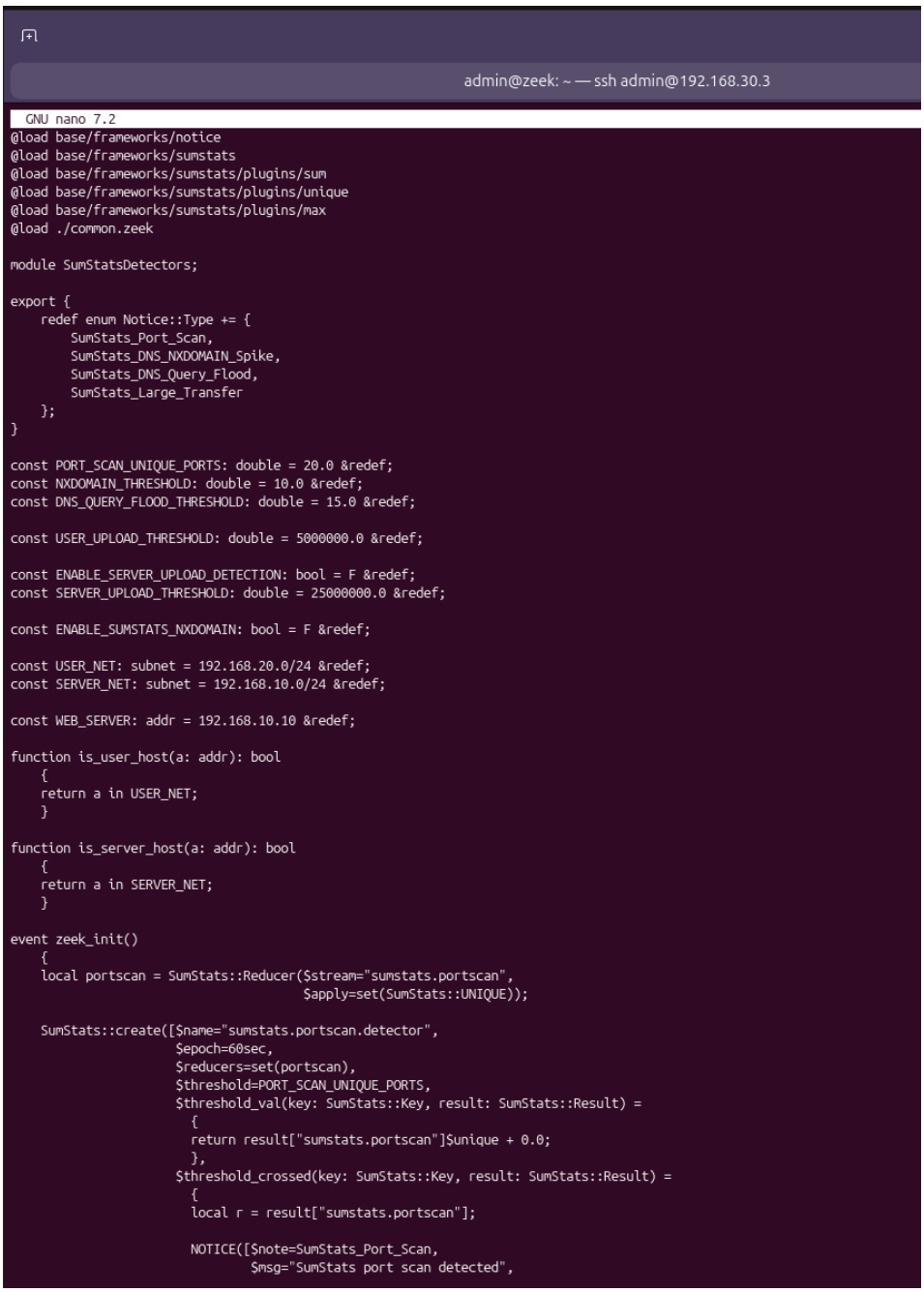


Рисунок 3.32 – Фрагмент реалізації Zeek-детектора на базі Sumstats Framework

У модулі SumStatsDetectors визначено власні типи notice-подій: SumStats_Port_Scan, SumStats_DNS_NXDOMAIN_Spike, SumStats_DNS_Query_Flood та SumStats_Large_Transfer. Саме ці типи подій надалі потрапляють у журнал notice.log і використовуються Wazuh для класифікації відповідних аномалій.

У скрипті також задано порогові значення для детекторів. Наприклад, port scan визначається при перевищенні 20 унікальних портів за часовий інтервал, DNS query flood - при 15 повторюваних DNS-запитах, а великий користувацький upload

- при перевищенні 5000000 байт. Окремо передбачено поріг для server upload - 25000000 байт.

У події Zeek_init() створюються основні SumStats-агрегатори. Для port scan використовується SumStats::UNIQUE, який рахує кількість унікальних портів для пари джерело-ціль. Якщо кількість унікальних портів перевищує заданий поріг, Zeek формує notice-подію SumStats_Port_Scan. У полі sub додатково записується цільовий вузол і кількість унікальних портів, що надалі дозволяє Wazuh-декодерам витягнути це значення в окреме поле.

Для DNS query flood використовується агрегатор SumStats::SUM, який підраховує кількість повторюваних DNS-запитів від одного вузла до одного домену. Якщо кількість таких запитів перевищує поріг, формується notice-подія SumStats_DNS_Query_Flood, у якій вказується домен і кількість повторень.

У детектора великого передавання даних використовується Sumstats::MAX, що дозволяє фіксувати максимальний обсяг даних, переданих у межах з'єднання.

Після написання скриптів виконуємо перевірку коректності їх налаштувань та написання.

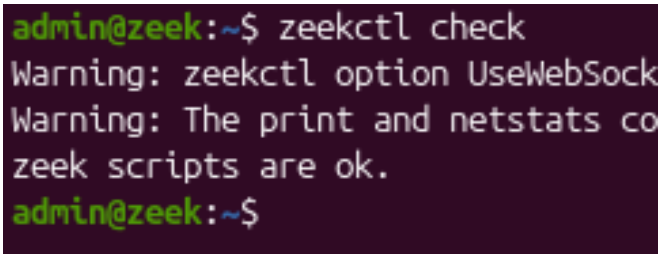


Рисунок 3.33 – Виконана перевірка конфігураційних файлів та скриптів/детекторів Zeek сенсора

3.8. Налаштування Wazuh декодерів, правила та кореляції подій

Основна задача декодерів полягає у перетворенні полів Zeek-події на поля, зручні для подальшої обробки правилами Wazuh. Події Zeek у журналі notice.log містять стандартні поля note, msg, sub, src, dst та інші. Для правил Wazuh ці значення було приведено до більш зручного вигляду: Zeek_note, Zeek_msg, Zeek_sub, srcip, dstip, dstport, dnsquery, dns_count, unique_ports, transfer_bytes.

Фрагмент логіки декодування Zeek подій наведено в рисунку 3.34.

```
GNU nano 7.2                                                    /var/ossec

<decoder name="zeek-json-sumstats-large-transfer">
  <parent>json</parent>
  <prematch>SumStatsDetectors::SumStats_Large_Transfer</prematch>
  <regex type="pcre2">"note":"([^"]+)".*"sub":"(flow=([^ ]+) max_bytes=([0-9]+))".*"src":"([^"]+)"</regex>
  <order>zeek_note,zeek_sub,transfer_flow,transfer_bytes,srcip</order>
</decoder>

<decoder name="zeek-json-dns-nxdomain-spike">
  <parent>json</parent>
  <prematch>DNS_NXDOMAIN_Spike</prematch>
  <regex type="pcre2">"note"\s*:\s*"([^"]+)".*"sub"\s*:\s*"(count=([0-9]+))".*"src"\s*:\s*"([^"]+)".*"dst"\s*:\s*"([^"]+)"</regex>
  <order>zeek_note,zeek_sub,nxdomain_count,srcip,dstip</order>
</decoder>

<decoder name="zeek-json-network-beaconing">
  <parent>json</parent>
  <prematch>BeaconingDetector::Network_Beaconing</prematch>
  <regex type="pcre2">"note":"([^"]+)".*"sub":"([^"]+)".*"src":"([^"]+)".*"dst":"([^"]+)".*"p":([0-9]+)</regex>
  <order>zeek_note,zeek_sub,srcip,dstip,dstport</order>
</decoder>

<decoder name="zeek-json-sumstats-dns-query-flood">
  <parent>json</parent>
  <prematch>SumStatsDetectors::SumStats_DNS_Query_Flood</prematch>
  <regex type="pcre2">"note":"([^"]+)".*"sub":"(query=([^ ]+) count=([0-9]+))".*"src":"([^"]+)"</regex>
  <order>zeek_note,zeek_sub,dnsquery,dns_count,srcip</order>
</decoder>

<decoder name="zeek-json-dns-suspicious-query">
  <parent>json</parent>
  <prematch>DNSAnomalyDetector::Suspicious_DNS_Query</prematch>
  <regex type="pcre2">"note":"([^"]+)".*"sub":"(query=([^ ]+) score=([0-9.]+))".*"src":"([^"]+)".*"dst":"([^"]+)"</regex>
  <order>zeek_note,zeek_sub,dnsquery,dns_score,srcip,dstip</order>
</decoder>

<decoder name="zeek-json-dns-new-domain">
  <parent>json</parent>
  <prematch>DNSAnomalyDetector::New_DNS_Domain</prematch>
  <regex type="pcre2">"note":"([^"]+)".*"sub":"(query=([^"]+))".*"src":"([^"]+)".*"dst":"([^"]+)"</regex>
  <order>zeek_note,zeek_sub,dnsquery,srcip,dstip</order>
</decoder>

<decoder name="zeek-json-new-pair">
  <parent>json</parent>
  <prematch>NewPairDetector::New_Communication_Pair</prematch>
  <regex type="pcre2">"note":"([^"]+)".*"src":"([^"]+)".*"dst":"([^"]+)"</regex>
  <order>zeek_note,srcip,dstip</order>
</decoder>
```

Рисунок 3.34 – Фрагмент файлу з декодуванням notice подій

У наведеному фрагменті можемо спостерігати розбір поля sub, який передає контекс подій з Zeek детекторів. Для DNS query флуду з поля sub витягується домене ім’я та кількість повторюваних запитів, а для порт скан – цільовий вузол і кількість унікальних портів. Це дозволяє надалі використовувати ці значення у Wazuh правилах, описах алертів і фільтрах дашборду.

Після створення декодерів було налаштовано базове правило для приймання Zeek подій, це правило є стандартним та надається офіційним джерелом Wazuh, при інтеграції з Zeek. Воно використовується кожен раз при отриманні нових логів з notice.log файлу, використовується як батьківське правило, для всіх інших правил.

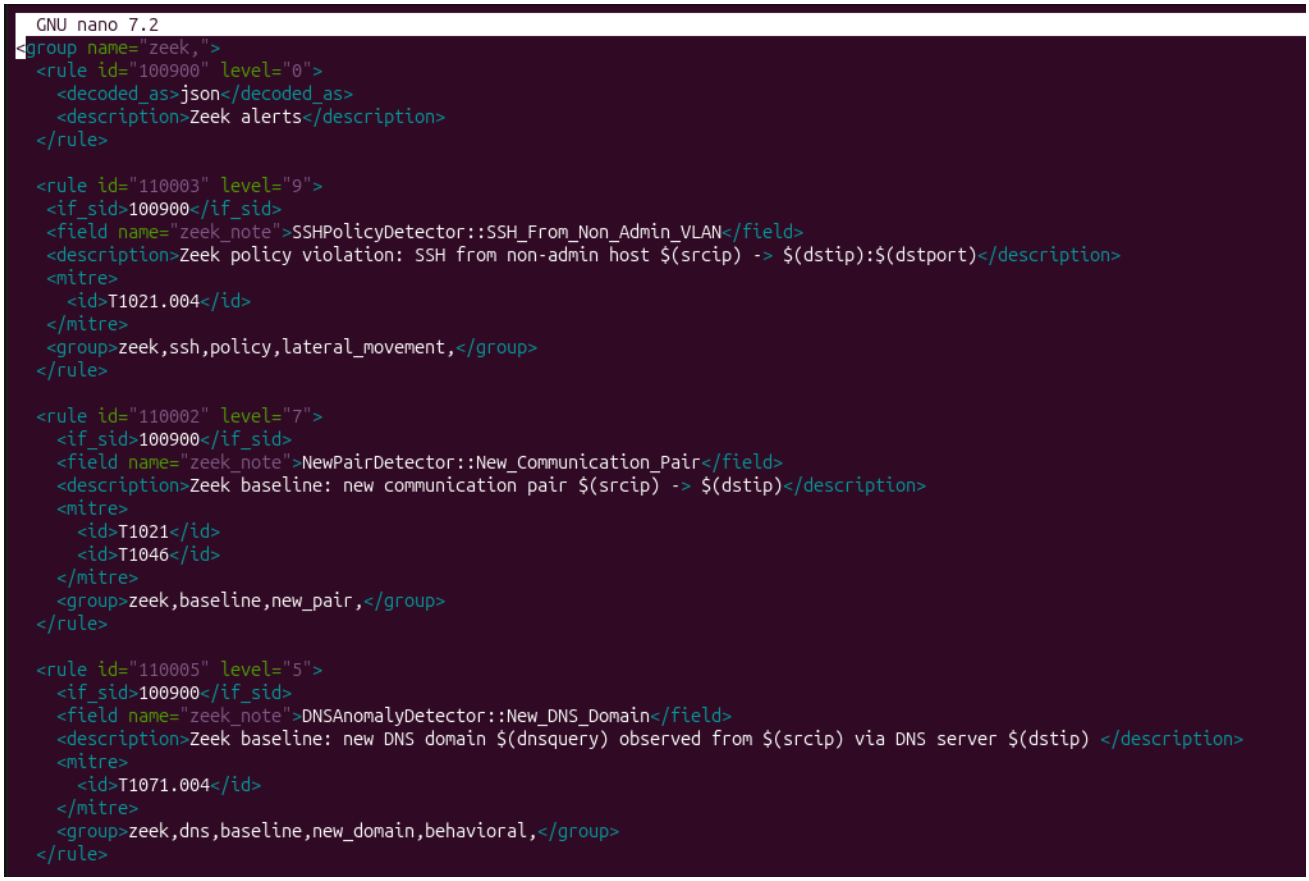


Рисунок 3.35 – Фрагмент файлу правил Wazuh

Першим у фрагменті визначено базове правило 100900 з рівнем 0. Воно спрацьовує на Zeek-події у JSON-форматі та має опис Zeek alerts. Це правило не формує окремий інцидент безпеки, а використовується як основа для подальших спеціалізованих правил. Саме тому інші правила містять конструкцію <if_sid>100900</if_sid>, яка означає, що вони застосовуються лише до подій, попередньо розпізнаних як Zeek-події.

Далі наведено правило 110003, яке відповідає за виявлення SSH-доступу не з адміністративного сегмента. Воно спрацьовує, якщо поле Zeek_note має значення SSHPolicyDetector::SSH_From_Non_Admin_VLAN. В описі алерту використовуються змінні $(srcip), $(dstip) та $(dstport), що дозволяє одразу бачити джерело підключення, цільовий вузол і порт. Для цього правила також додано прив’язку до MITRE ATT&CK T1021.004, що відповідає віддаленому доступу через SSH.

Правило 110002 використовується для класифікації події NewPairDetector::New_Communication_Pair. Воно спрацьовує у випадку появи

нової комунікаційної пари, якої немає в baseline нормальної поведінки. Такий алерт може свідчити про новий напрям взаємодії між вузлами, що є важливим для виявлення розвідки або lateral movement. У правилі зазначено техніки MITRE T1021 і T1046, які пов'язані з віддаленим доступом і мережевим скануванням.

Правило 110005 призначене для обробки події DNSAnomalyDetector::New_DNS_Domain. Воно спрацьовує тоді, коли Zeek фіксує DNS-запит до домену, відсутнього у baseline. В описі алерту додатково виводяться доменне ім'я $(dnsquery), IP-адреса джерела $(srcip) та DNS-сервер $(dstip). Для цього правила використано техніку MITRE T1071.004, яка пов'язана з використанням DNS як прикладного протоколу для комунікації.

Надалі на рисунку 3.36 наведено спрацювання спеціалізованого правила Wazuh про несанкціонований доступ до SSH через Logtest з витягом від переданого log файлу Zeek notice.log.

```
admin@wazuh:~$ sudo /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.14.5
Type one log per line

{"ts":1781126412.723435,"proto":"tcp","note":"SSHPolicyDetector::SSH_From_Non_Admin_VLAN","msg":"SSH attempt from non-admin
":300.0}

**Phase 1: Completed pre-decoding.
        full event: '{"ts":1781126412.723435,"proto":"tcp","note":"SSHPolicyDetector::SSH_From_Non_Admin_VLAN","msg":"SSH at
est":[],"suppress_for":300.0}'

**Phase 2: Completed decoding.
        name: 'json'
        parent: 'json'
        dstip: '192.168.10.10'
        dstport: '22'
        srcip: '10.255.255.254'
        zeek_note: 'SSHPolicyDetector::SSH_From_Non_Admin_VLAN'

**Phase 3: Completed filtering (rules).
        id: '110003'
        level: '9'
        description: 'Zeek policy violation: SSH from non-admin host 10.255.255.254 -> 192.168.10.10:22'
        groups: '['zeek', 'zeek', 'ssh', 'policy', 'lateral_movement']'
        firedtimes: '1'
        mail: 'False'
        mitre.id: '['T1021.004']'
        mitre.tactic: '['Lateral Movement']'
        mitre.technique: '['SSH']'
**Alert to be generated.
```

Рисунок 3.36 – Перевірка спрацювання правила 110003

Окрім окремих правил, у Wazuh було налаштовано кореляційні правила. Їх призначення полягає у тому, щоб пов'язувати між собою декілька різних подій, які в сукупності можуть свідчити про складніший сценарій атаки. В проєктованій мережі було розроблено правила трьох кореляцій. Перша кореляція про подію

мережевого beaconing + пересилання великої кількості трафіку. Друга кореляція про нову комунікацію в мережі та виявлення сканування портів. Третя кореляція про аномалію активності в нічний час та спробу несанкціонованого доступу до SSH не з адміністративного сегменту.

Одна з описаних кореляцій правил наведена на рисунку 3.37.

```
<rule id="110202" level="14" frequency="2" timeframe="600">
  <if_sid>110003</if_sid>
  <if_matched_sid>110006</if_matched_sid>
  <same_srcip />
  <same_dstip />
  <description>Zeek correlation: SSH access from non-admin VLAN during abnormal activity period $(srcip) to $(dstip)</description>
  <mitre>
    <id>T1021.004</id>
  </mitre>
  <group>zeek,zeek_correlation,ssh,remote_access,policy_violation,</group>
</rule>
```

Рисунок 3.37 – Наведений фрагмент кореляційного правила 110202

Кореляційне правило 110202 використовується для виявлення ситуації, коли SSH-підключення з неадміністративного сегмента відбувається поза нормальними годинами активності. Це правило поєднує два окремі спрацювання: ActiveHoursDetector::Activity_Outside_Normal_Hours та SSHPolicyDetector::SSH_From_Non_Admin_VLAN.

Окремо кожна з цих подій може мати середній рівень критичності. Наприклад, активність поза робочим часом не завжди є атакою, а SSH-з’єднання може бути результатом помилки конфігурації або тестування. Однак їх поєднання в межах одного джерела суттєво підвищує ризик, оскільки така активність може свідчити про несанкціонований доступ, спробу обходу адміністративної політики або початковий етап lateral movement.

Надалі проведемо перевірку спрацювання цього правила за допомогою Wazuh-logtest. Спрацювання наведено на рисунку 3.38.



Рисунок 3.38 – Тестове спрацювання кореляційного правила

Таким чином, у межах цього етапу було налаштовано Wazuh-декодери, базове правило для Zeek-подій, спеціалізовані правила для окремих типів поведінкових аномалій, а також кореляційні правила для виявлення складніших сценаріїв. Це дозволяє перетворити Zeek notice-події на структуровані Wazuh-алерти з описом, рівнем критичності, витягнутими полями та прив’язкою до MITRE ATT&CK.

3.9. Сценарії експериментів і верифікації детекції

Після всіх вдалих налаштувань проєкту корпоративної мережі, можно переходити до тестування та верифікації подій. Перевірка виконувалась за загальною схемою. Генерація тестової активності – формування Zeek-подій – формування за наявності аномалії, та спрацювання детектору – передавання інформації Wazuh Agent’у – спрацювання Wazuh правила – аналіз результату.

Для кожного спрацювання перевірялось три ключові моменти: поява відповідної події у notice.log, коректне декодування події у Wazuh, та спрацювання потрібного правила з відповідним ідентифікатором.

Першим сценарієм було перевірено виявлення нової комунікаційної пари, яка була відсутня у baseline нормальної поведінки мережі. Для цього до

користувацького сегменту було підключено станцію з IP-адресою 192.168.20.3, яка попередньо не була зафіксована у базовій моделі дозволених мережевих взаємодій.

У результаті звернення нового вузла до інших сегментів мережі у Wazuh було сформовано алерт за правилом 110002, що відповідає виявленню нової комунікаційної пари. Проведене тестування наведено на рисунку 3.39.

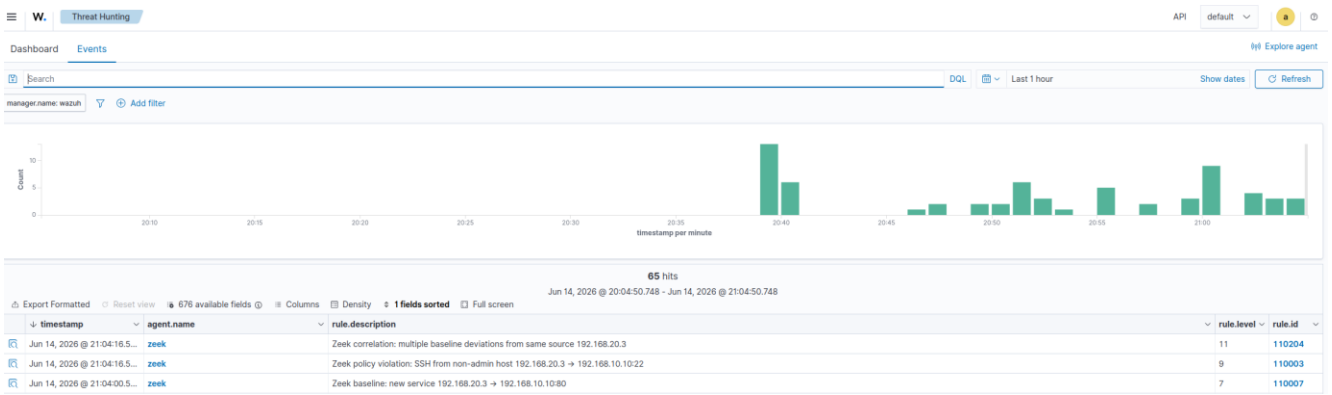


Рисунок 3.39 – Спрацювання алерту про аномалію з користувацьким вузлом

Отриманий алерт свідчить про те, що система виявила відхилення від попередньо сформованої моделі нормальної поведінки. Додатково під час тестування було зафіксовано спробу доступу з користувацького сегменту до адміністративного сегменту, що призвело до спрацювання правила 110003. Таким чином, у межах одного сценарію було підтверджено можливість виявлення як нового джерела мережевої активності, так і порушення політики доступу між сегментами мережі.

У підсумку одночасна поява нового вузла в користувацькому сегменті та спроба доступу до адміністративного сегменту призвели до формування кореляційного алерту. Це свідчить не лише про окреме спрацювання правил 110002 та 110003, а й про виявлення комплексної поведінкової девіації, коли декілька подій у межах одного сценарію вказують на потенційно підозрілу активність у мережі.

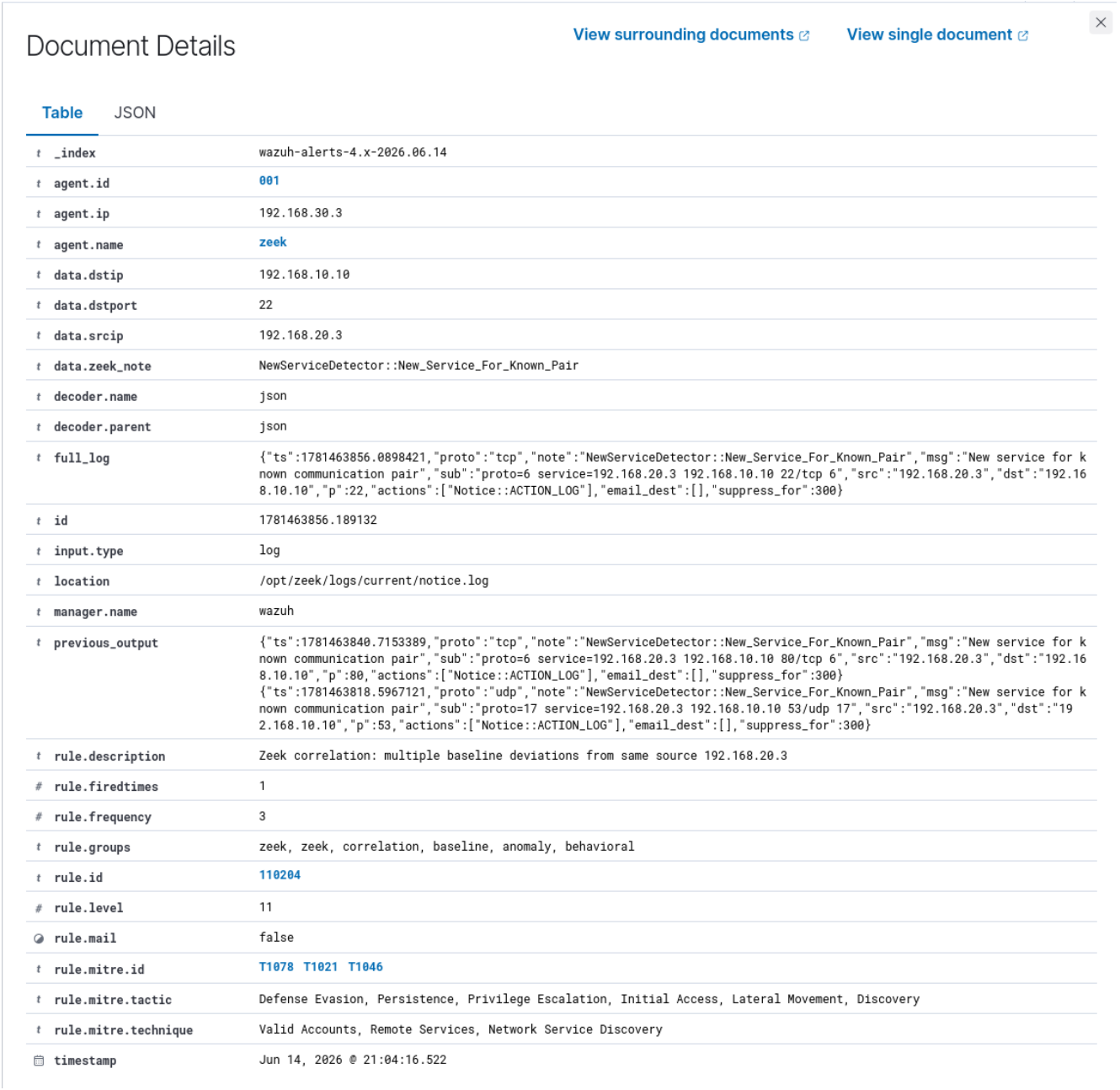


Рисунок 3.40 – Повне виведення спрацьованого кореляційного правила про девіацію подій у мережі

Сформований кореляційний алерт виводить нам всю докладну інформацію про подію, від айпі адрес джерела здійснення та вектору аномалії, тип зафіксованої аномалії, її опис, в якому міститься інформація з Zeek детектора до правил MITRE ATT&CK.

Наступний експеримент було проведено для перевірки кореляційного сценарію, пов’язаного з активністю користувацького вузла поза нормальними годинами та спробою SSH-доступу до серверного вузла з неадміністративного сегмента. У межах цього сценарію користувацький хост ініціював мережеву

активність у нетиповий час, а також виконав спробу підключення до web-сервера за протоколом SSH. Оскільки SSH-доступ у проєктованій корпоративній мережі має здійснюватися лише з адміністративного сегмента, така подія розглядається як порушення політики доступу. Даний експеримент наведений на рисунку 3.41

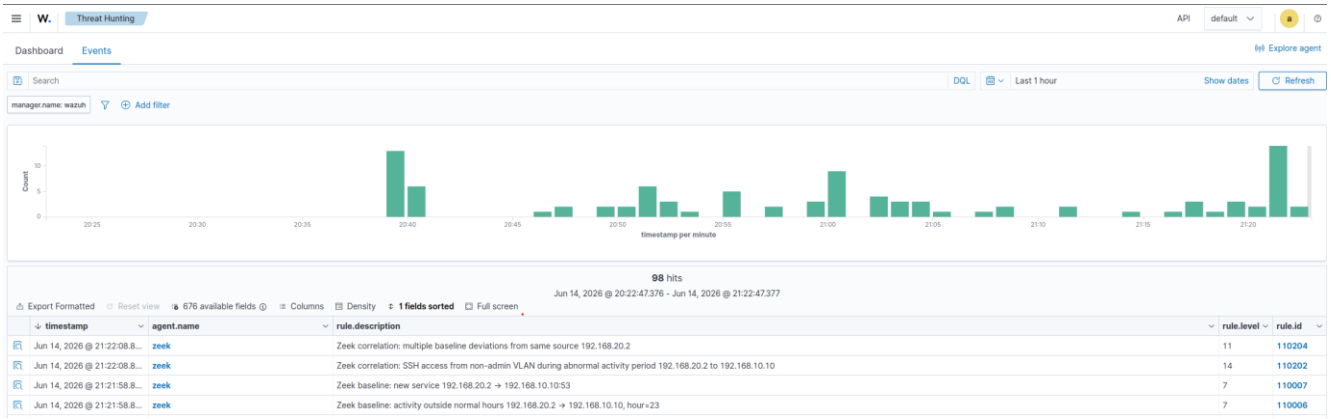


Рисунок 3.41 – Формування алертів про поведінкову аномалію всередині мережі

У результаті Zeek сформував дві пов’язані notice-події: ActiveHoursDetector::Activity_Outside_Normal_Hours та SSHPolicyDetector::SSH_From_Non_Admin_VLAN. Після передавання цих подій до Wazuh вони були класифіковані окремими правилами 110006 і 110003. Оскільки обидві події були пов’язані з одним джерелом і відбулися в межах заданого часового інтервалу, Wazuh додатково сформував кореляційний алерт 110202, який вказує на потенційно несанкціонований доступ у нетиповий час.

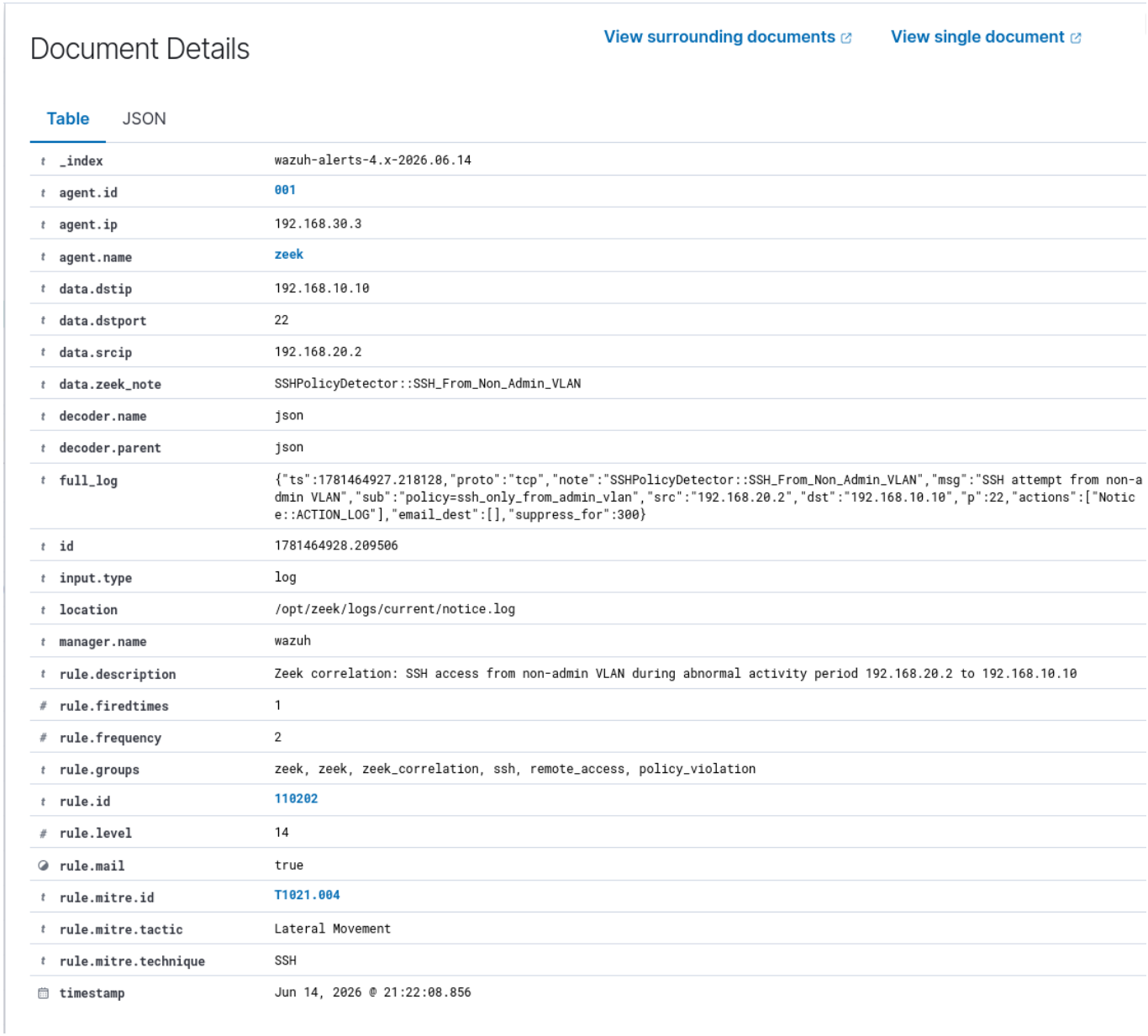


Рисунок 3.42 – Детальний вивід інформації про кореляцію події

Наступний експеримент був виконаний для перевірки DNS-аномалій. У межах цього сценарію з користувацького вузла було згенеровано DNS-запит до домену, який був відсутній у baseline нормальної поведінки мережі.

```
programist@andrew-kovalski-programist:~$ TS=$(date +%s)
dig @192.168.10.10 new-domain-$TS.nxlab.test A +tries=1 +time=1

; <<>> DiG 9.20.18-1ubuntu2-Ubuntu <<>> @192.168.10.10 new-domain-1781466284.nxlab.test A +tries=1 +time=1
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 30317
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;new-domain-1781466284.nxlab.test. IN   A

;; Query time: 5 msec
;; SERVER: 192.168.10.10#53(192.168.10.10) (UDP)
;; WHEN: Sun Jun 14 21:44:44 CEST 2026
;; MSG SIZE  rcvd: 61

programist@andrew-kovalski-programist:~$
```

Рисунок 3.43 – Виконання перевірки аномалії з новим DNS домейном

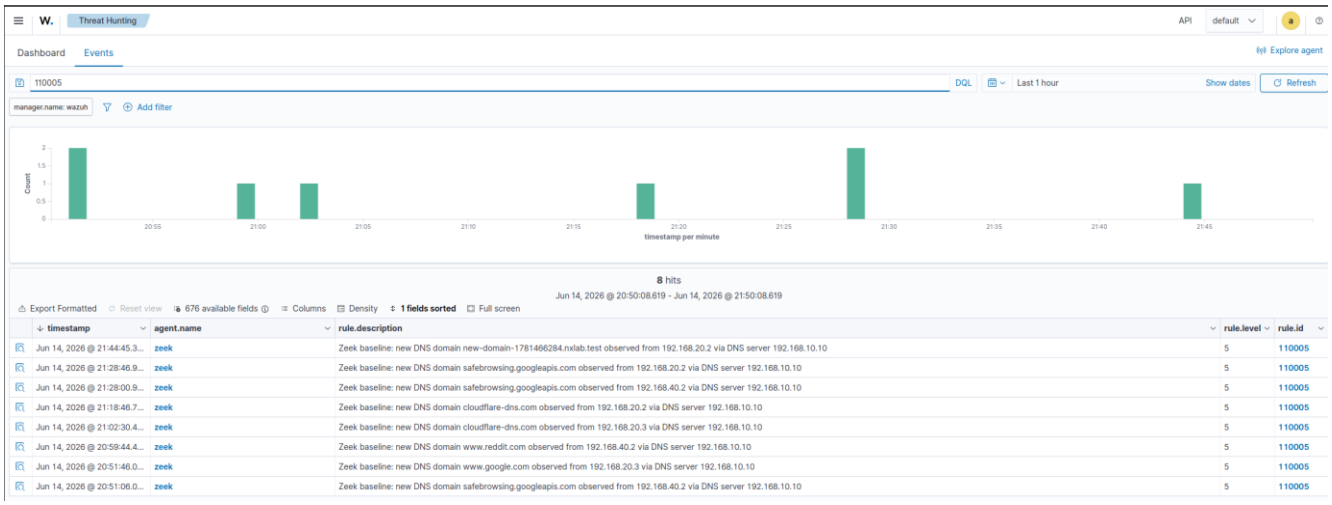


Рисунок 3.44 – Спрацювання алерту на новий DNS запит

Наступний експеримент було проведено для перевірки виявлення підозрілого DNS-запиту. У межах цього сценарію з користувацького вузла було згенеровано DNS-запит із нетиповою структурою доменного імені. Такий запит може містити велику кількість символів, незвичну послідовність літер і цифр або мати ознаки автоматично згенерованого домену. Подібна DNS-активність відрізняється від звичайних запитів користувача та може свідчити про спробу використання DNS для прихованої комунікації або звернення до підозрілого ресурсу.

```
programist@andrew-kovalski-programist:~$ TS=$(date +%s)
dig @192.168.10.10 a9x8q6w7s7dsa82i827dsa923-$TS.nxlab.test A +tries=1 +time=1

; <<>> DiG 9.20.18-1ubuntu2-Ubuntu <<>> @192.168.10.10 a9x8q6w7s7dsa82i827dsa923-1781466738.nxlab.test A +tries=1 +time=1
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 41428
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;a9x8q6w7s7dsa82i827dsa923-1781466738.nxlab.test. IN A

;; Query time: 6 msec
;; SERVER: 192.168.10.10#53(192.168.10.10) (UDP)
;; WHEN: Sun Jun 14 21:52:18 CEST 2026
;; MSG SIZE  rcvd: 76

programist@andrew-kovalski-programist:~$
```

Рисунок 3.45 – Проведення імітації атаки з підозрілим DNS -запитом

В результаті було отримано алерт, з правилом 110004. Детальний вивід наведений на рисунку 3.46.

Document Details

View surrounding documents | View single document

Table | JSON

t _index	wazuh-alerts-4.x-2026.06.14
t agent.id	001
t agent.ip	192.168.30.3
t agent.name	zeek
t data.dns_score	4
t data.dnsquery	a9x8q6w7s7dsa82i827dsa923-1781466738.nxlab.test
t data.dstip	192.168.10.10
t data.srcip	192.168.20.2
t data.zeek_note	DNSAnomalyDetector::Suspicious_DNS_Query
t data.zeek_sub	query=a9x8q6w7s7dsa82i827dsa923-1781466738.nxlab.test score=4
t decoder.name	json
t decoder.parent	json
t full_log	{"ts":1781468836.1909521,"note":"DNSAnomalyDetector::Suspicious_DNS_Query","msg":"Suspicious DNS query detected","sub":"query=a9x8q6w7s7dsa82i827dsa923-1781466738.nxlab.test score=4","src":"192.168.20.2","dst":"192.168.10.10","actions":["Notice::ACTION_LOG"],"email_dest":[],"suppress_for":600}
t id	1781468836.280805
t input.type	log
t location	/opt/zeek/logs/current/notice.log
t manager.name	wazuh
t rule.description	Zeek suspicious DNS query a9x8q6w7s7dsa821827dsa923-1781466738.nxlab.test from 192.168.20.2 via DNS server 192.168.10.10, score=4
# rule.firedtimes	1
t rule.groups	zeek, zeek, dns, dga, behavioral, suspicious_dns_query
t rule.id	110004
# rule.level	8
rule.mail	false

Рисунок 3.46 – Виведений алерт про підозрілий DNS запит

Наступний експеримент було проведено для перевірки виявлення DNS query flood. У межах цього сценарію з користувацького вузла було згенеровано багаторазові DNS-запити до одного доменного імені протягом короткого проміжку часу. Така активність відрізняється від звичайної DNS-поведінки користувача, оскільки характеризується повторюваністю та перевищенням заданого статистичного порогу.

```
programist@andrew-kovalski-programist:~$ for i in $(seq 1 20); do
  dig @192.168.10.10 flood-test.example.com A +tries=1 +time=1 > /dev/null
done
programist@andrew-kovalski-programist:~$
```

Рисунок 3.47 – Проведення імітації атаки з dns-flood

Для виявлення цього сценарію в Zeek використовується статистичний детектор на основі SumStats. Він підраховує кількість DNS-запитів до одного домену від одного джерела за визначене часове вікно. Після перевищення встановленого порогу Zeek формує notice-подію SumStatsDetectors::SumStats_DNS_Query_Flood.

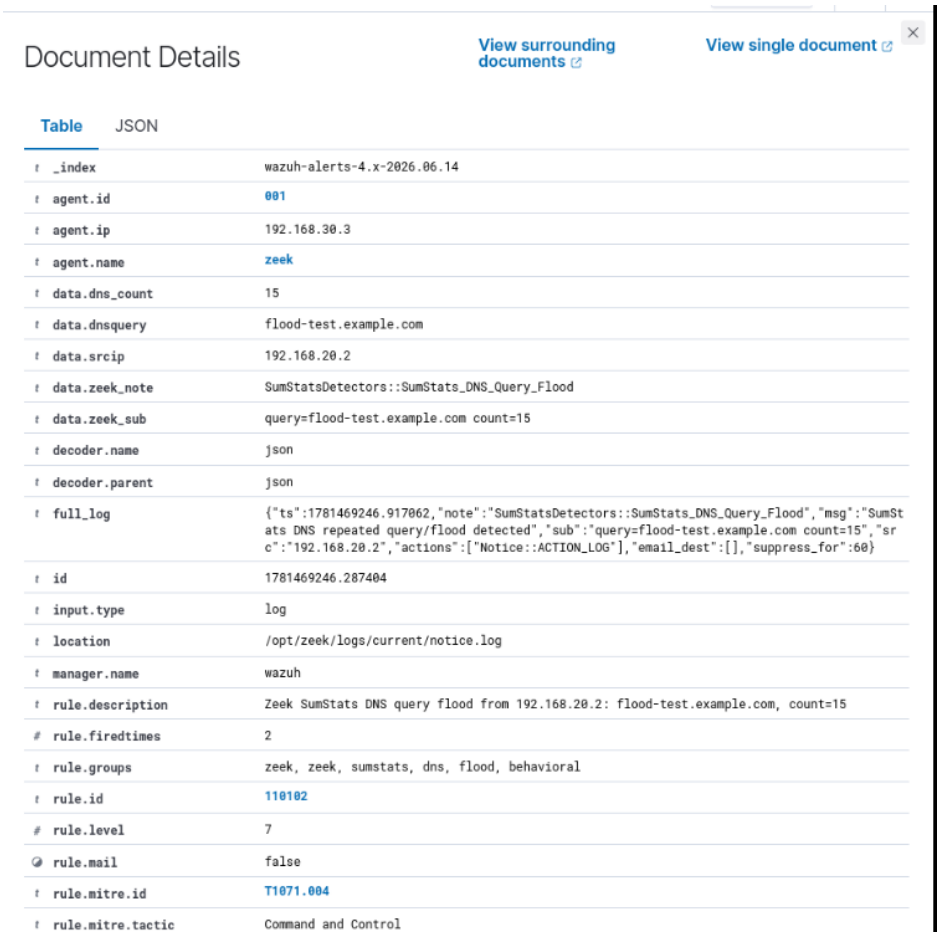


Рисунок 3.48 – Виведений алерт про DNS флуд аномалію

Наступний експеримент було проведено для перевірки виявлення DNS NXDOMAIN spike. У межах цього сценарію з користувацького вузла було згенеровано серію DNS-запитів до неіснуючих доменних імен. У результаті DNS-сервер повертав відповіді типу NXDOMAIN, які означають, що запитуваний домен не існує.

```
programist@andrew-kovalski-programist:~$ TS=$(date +%s)
for i in $(seq 1 12); do
  dig @192.168.10.10 no-such-$i-$TS.nxlab.test A +tries=1 +time=1 > /dev/null
  sleep 0.2
done
programist@andrew-kovalski-programist:~$
```

Рисунок 3.49 – Проведення імітації DNS-Spike

Document Details

View surrounding documents | View single document

agent.id	001
agent.ip	192.168.30.3
agent.name	zeek
data.dstip	192.168.10.10
data.nxdomain_count	10
data.srcip	192.168.20.2
data.zeek_note	DNSAnomalyDetector::DNS_NXDOMAIN_Spike
data.zeek_sub	count=10
decoder.name	json
decoder.parent	json
full_log	{"ts":1781470082.33876,"note":"DNSAnomalyDetector::DNS_NXDOMAIN_Spike","msg":"DNS NXDOMAIN spike detected","sub":"count=10","src":"192.168.20.2","dst":"192.168.10.10","actions":["Notice::ACTION_LOG"],"email_dest":[],"suppress_for":300}
id	1781470082.300970
input.type	log
location	/opt/zeek/logs/current/notice.log
manager.name	wazuh
rule.description	Zeek SumStats NXDOMAIN spike from 192.168.20.2 via DNS Server 192.168.10.10, count=10
rule.firedtimes	1
rule.groups	zeek, zeek, sumstats, dns, dga, nxdomain
rule.id	110101
rule.level	8
rule.mail	false
rule.mitre.id	T1568
rule.mitre.tactic	Command and Control
rule.mitre.technique	Dynamic Resolution
timestamp	Jun 14, 2026 @ 22:48:02.956

Рисунок 3.50 – Вивід алерту про DNS Spike

Велика кількість таких відповідей за короткий проміжок часу може свідчити про підозрілу DNS-активність, наприклад перебір доменних імен, помилки автоматизованого скрипта або спроби звернення до доменів, згенерованих шкідливим програмним забезпеченням.

Наступний експеримент був проведений для перевірки виявлення комбінованої аномальної активності, яка складається з періодичного мережевого з'єднання та подальшої передачі великого обсягу даних. Такий сценарій імітує поведінку C2-комунікації, коли вузол періодично звертається до зовнішнього або контрольного сервера, а після цього виконує передачу значного обсягу інформації.

Атакована машина Kali Linux виступала тестовим сервером, який приймав періодичні beacon запити та отримання файлу великого розміру, що наведено на рисунку 3.51.

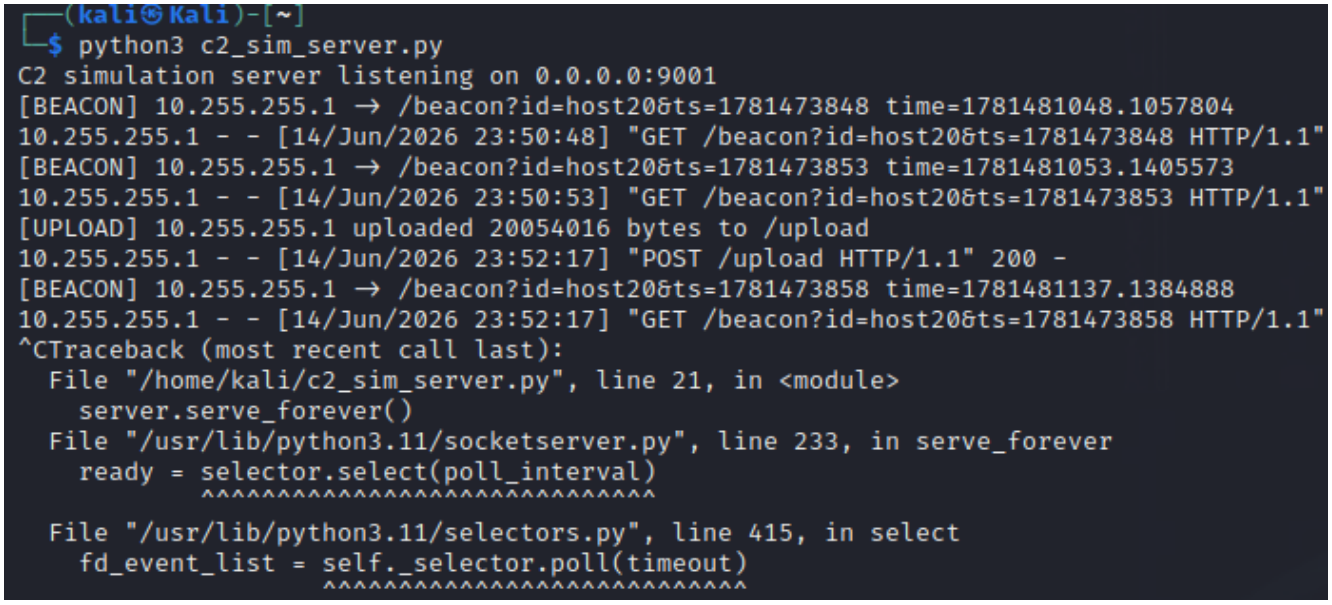


Рисунок 3.51 – Kali linux в ролі сервера на який було вивантажено дані користувацького вузла

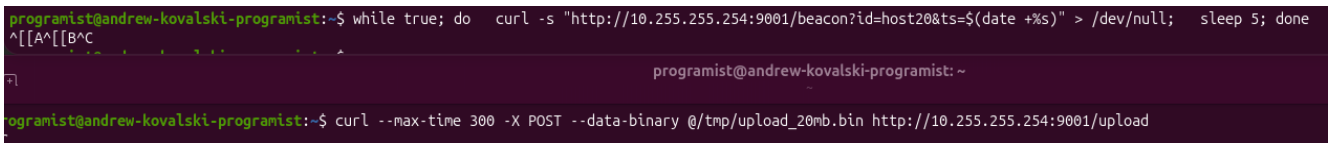


Рисунок 3.52 – Проведення імітації з'єднання з атакуючою машиною внаслідок зараження

На основі цих двох спрацювань, було визвано кореляційне правило. Кореляційне правило наведене на рисунку 3.53.

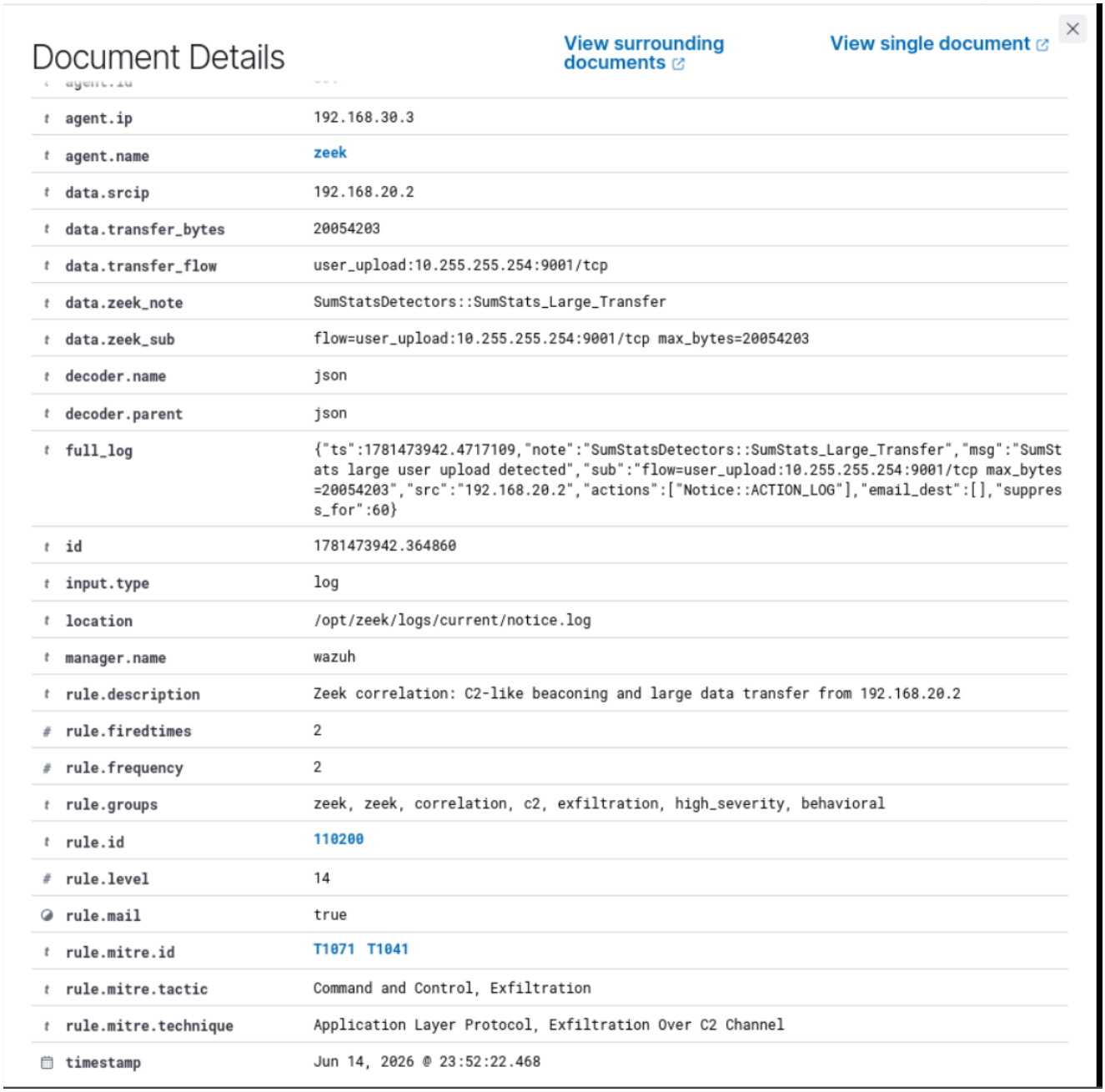


Рисунок 3.53 – Спрацювання кореляційного правила Beaconing + large transfer

Проведений експеримент підтвердив можливість виявлення не лише окремих аномалій, а й їх кореляцію, що підвищує рівень поведінкової аномалії в корпоративній мережі.

Наступний експеримент був проведений для перевірки виявлення сканування портів у корпоративній мережі. Такий тип активності є характерним для етапу розвідки, коли зловмисник намагається визначити доступні сервіси на цільовому вузлі перед подальшим виконанням атаки.

```
┌──(kali㉿Kali)-[~]
└─$ sudo nmap -sS -Pn -p 1-100 10.255.255.10
Starting Nmap 7.94 ( https://nmap.org ) at 2026-06-14 23:56 UTC
Nmap scan report for 10.255.255.10
Host is up (0.041s latency).
Not shown: 97 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
53/tcp open  domain
80/tcp open  http
MAC Address: 0C:7D:E4:5A:00:01 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 14.46 seconds
```

Рисунок 3.54 – Проведення сканування портів Веб-серверу з атакуючої машини

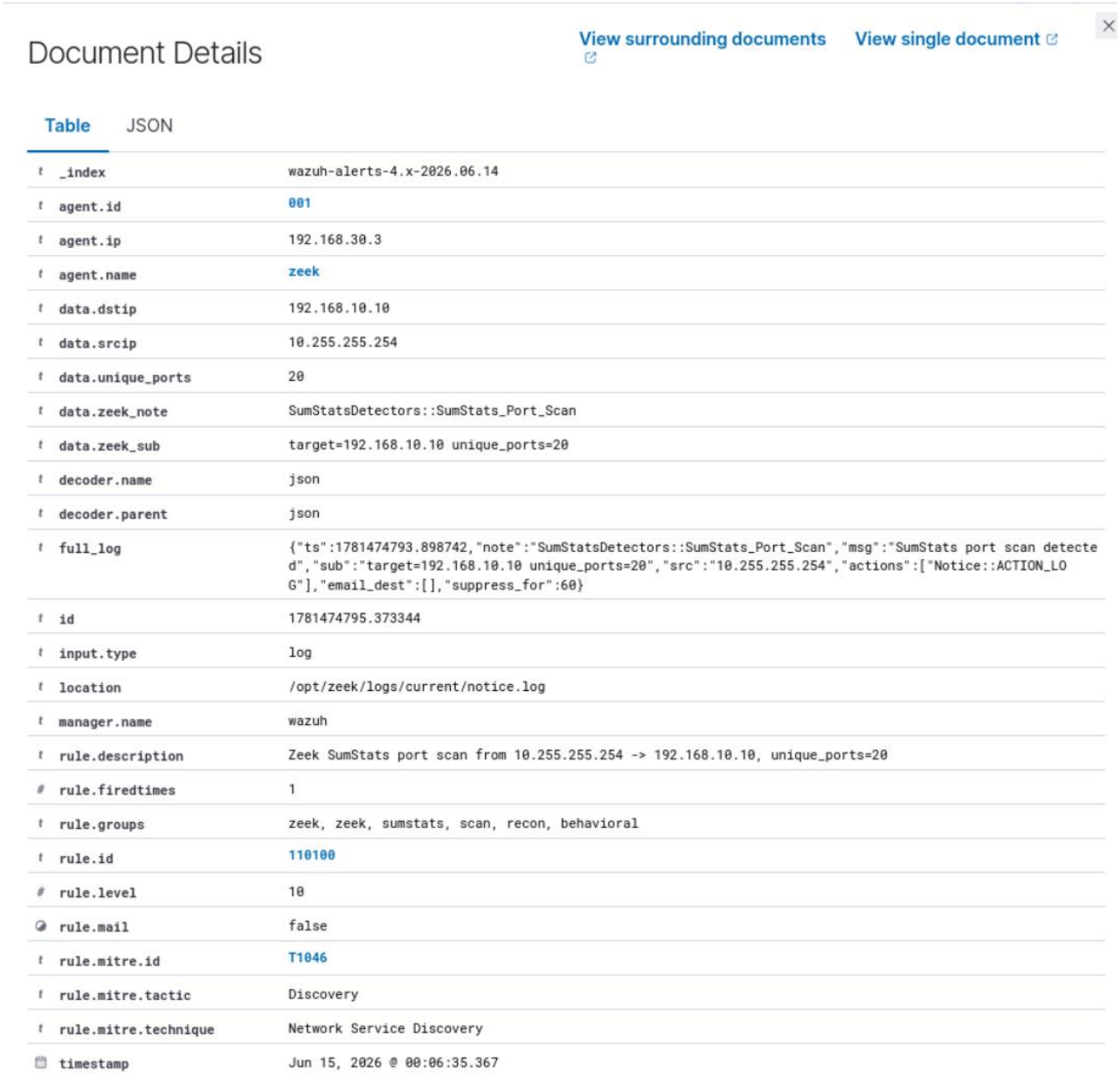


Рисунок 3.55 – Спрацювання правила про поведінкову аномалію в мережі

Наступний експеримент був проведений для перевірки можливості виявлення сканування портів внутрішнього хоста корпоративної мережі. У межах

тестового сценарію з вузла атакувальника було виконано звернення до великої кількості портів цільового сервера за короткий проміжок часу. Така активність імітує етап мережевої розвідки, під час якого зловмисник намагається визначити відкриті порти та доступні сервіси на внутрішньому вузлі.

```
programist@andrew-kovalski-programist:~$ sudo nmap -sS -Pn -p 1-100 100 192.168.10.10
Starting Nmap 7.98 ( https://nmap.org ) at 2026-06-15 00:17 +0200
Failed to resolve "100".
Nmap scan report for 192.168.10.10
Host is up (0.030s latency).
Not shown: 97 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
53/tcp open  domain
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 9.67 seconds
programist@andrew-kovalski-programist:~$
```

Рисунок 3.56 – Проведення розвідки мережі з внутрішнього вузла

Document Details

View surrounding documents | View single document

Table | JSON

_index	wazuh-alerts-4.x-2026.06.14
agent.id	001
agent.ip	192.168.30.3
agent.name	zeek
data.dstip	192.168.10.10
data.srcip	192.168.20.3
data.unique_ports	20
data.zeek_note	SumStatsDetectors::SumStats_Port_Scan
data.zeek_sub	target=192.168.10.10 unique_ports=20
decoder.name	json
decoder.parent	json
full_log	{"ts":1781475446.267482,"note":"SumStatsDetectors::SumStats_Port_Scan","msg":"SumStats port scan detected","sub":"target=192.168.10.10 unique_ports=20","src":"192.168.20.3","actions":["Notice::ACTION_LOG"],"email_dest":[],"suppress_for":60}
id	1781475447.387131
input.type	log
location	/opt/zeek/logs/current/notice.log
manager.name	wazuh
rule.description	Zeek correlation: new communication pair and port scan activity from 192.168.20.3
rule.firedtimes	1
rule.frequency	2
rule.groups	zeek, zeek, correlation, recon, new_pair, scan, behavioral
rule.id	110201
rule.level	13
rule.mail	true
rule.mitre.id	T1046 T1021
rule.mitre.tactic	Discovery, Lateral Movement
rule.mitre.technique	Network Service Discovery, Remote Services
timestamp	Jun 15, 2026 @ 00:17:27.937

Рисунок 3.57 – Спрацювання кореляційного правила нового комунікаційного хоста та сканування портів

У результаті виконання експерименту було сформовано кореляційний алерт, який об’єднав подію появи нової комунікаційної пари з подальшим скануванням портів внутрішнього хоста. Це дозволяє розглядати активність не як ізольоване мережеве з’єднання, а як послідовність дій, характерну для мережевої розвідки.

Окремо варто зазначити, що для частини Wazuh-правил було виконано прив’язку до технік MITRE ATT&CK. Це дозволяє розглядати сформовані алерти не лише як окремі технічні події, а як можливі етапи атаки. Наприклад, port scan і нова комунікаційна пара можуть бути пов’язані з тактикою Discovery, SSH-доступ з неадміністративного сегмента - з Lateral Movement, DNS-аномалії та beaconing - з Command and Control, a large transfer - з Exfiltration

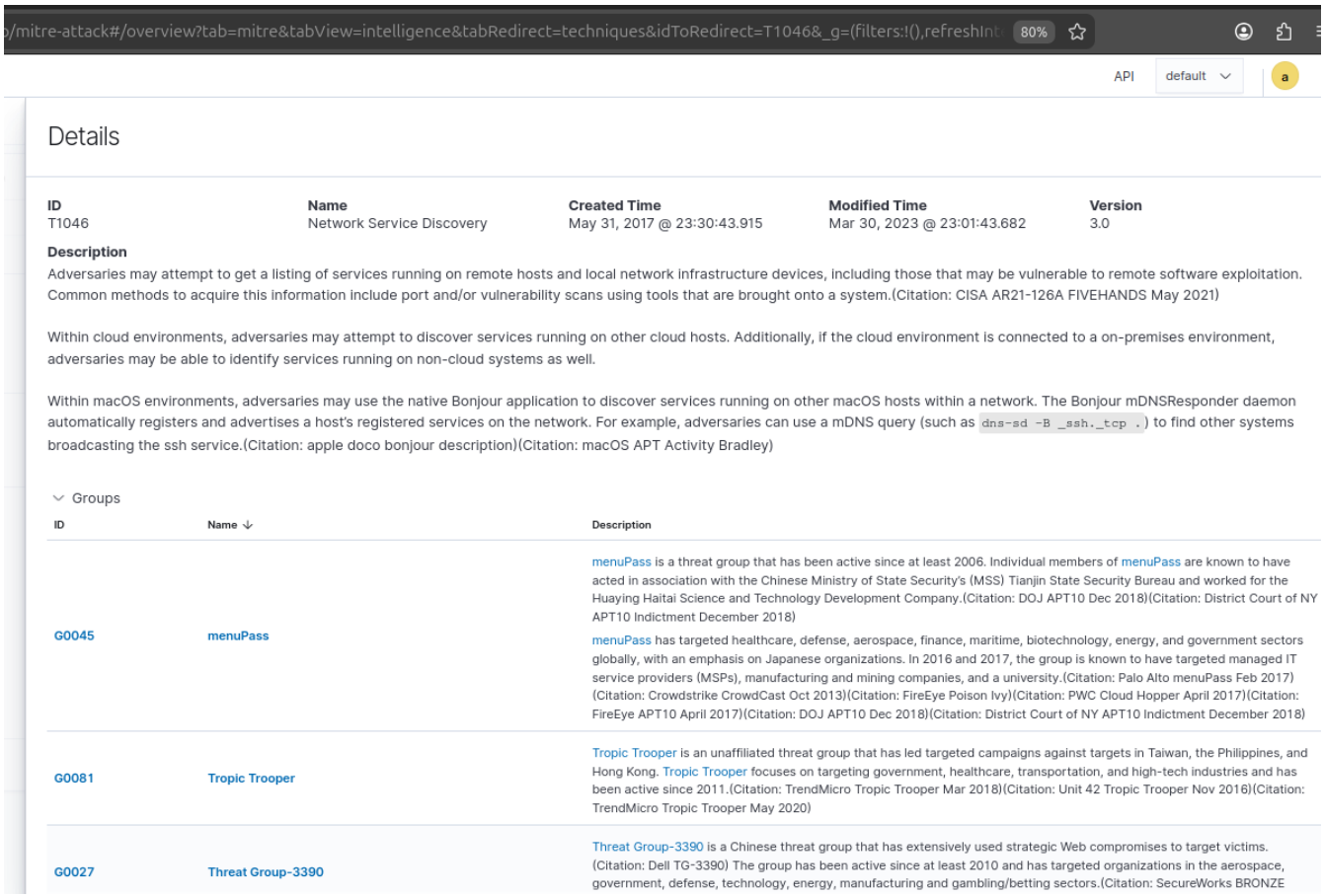


Рисунок 3.58 – Перегляд тактики MITRE ATT&CK до алерту сканування портів

Наступним етапом стала перевірка відображення подій, пов’язаних із поведінковими аномаліями, у Wazuh Dashboard. Для цього було створено окремі інформаційні панелі для аналізу звичайних алертів та кореляційних правил. Перша панель використовувалася для перегляду окремих спрацювань Zeek-детекторів,

зокрема DNS-аномалій, port scan, large transfer, network beaconing та інших подій.

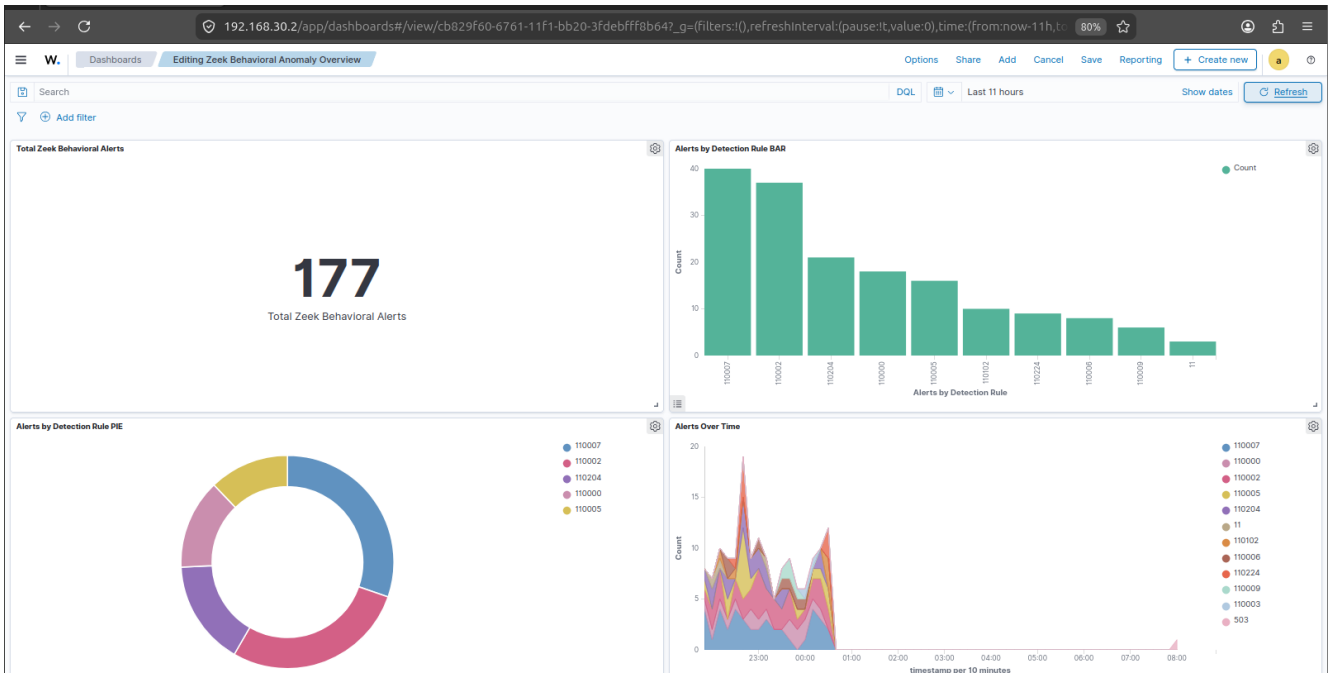


Рисунок 3.59 – Wazuh Dashboard з виявленням поодиноких поведінкових аномалій

У верхній частині панелі відображається загальна кількість сформованих алертів за вибраний часовий інтервал. У наведеному прикладі було зафіксовано 354 алерти, що відповідає результатам проведених експериментів.

Діаграми розподілу за правилами детекції дозволяють визначити, які типи аномалій спрацьовували найчастіше. Це є корисним для аналізу активних сценаріїв тестування та виявлення правил, які можуть потребувати додаткового налаштування.

Часова шкала показує, що алерти виникали переважно у вигляді окремих піків активності. Це відповідає послідовному виконанню експериментальних сценаріїв, таких як DNS-аномалії, port scan, large transfer і network beaconing.

Нижня частина Dashboard відображає основні IP-адреси джерел і призначення. Це дозволяє швидко визначити, які вузли найчастіше генерували аномальну активність і на які вузли вона була спрямована. Зокрема, серверний вузол 192.168.10.10 виступає однією з основних цілей, оскільки більшість тестових сценаріїв була пов’язана з доступом до web-сервера, скануванням портів і передаванням даних.

Друга панель була призначена для аналізу кореляційних алертів, які поєднують декілька пов’язаних подій в один більш критичний сценарій атаки.

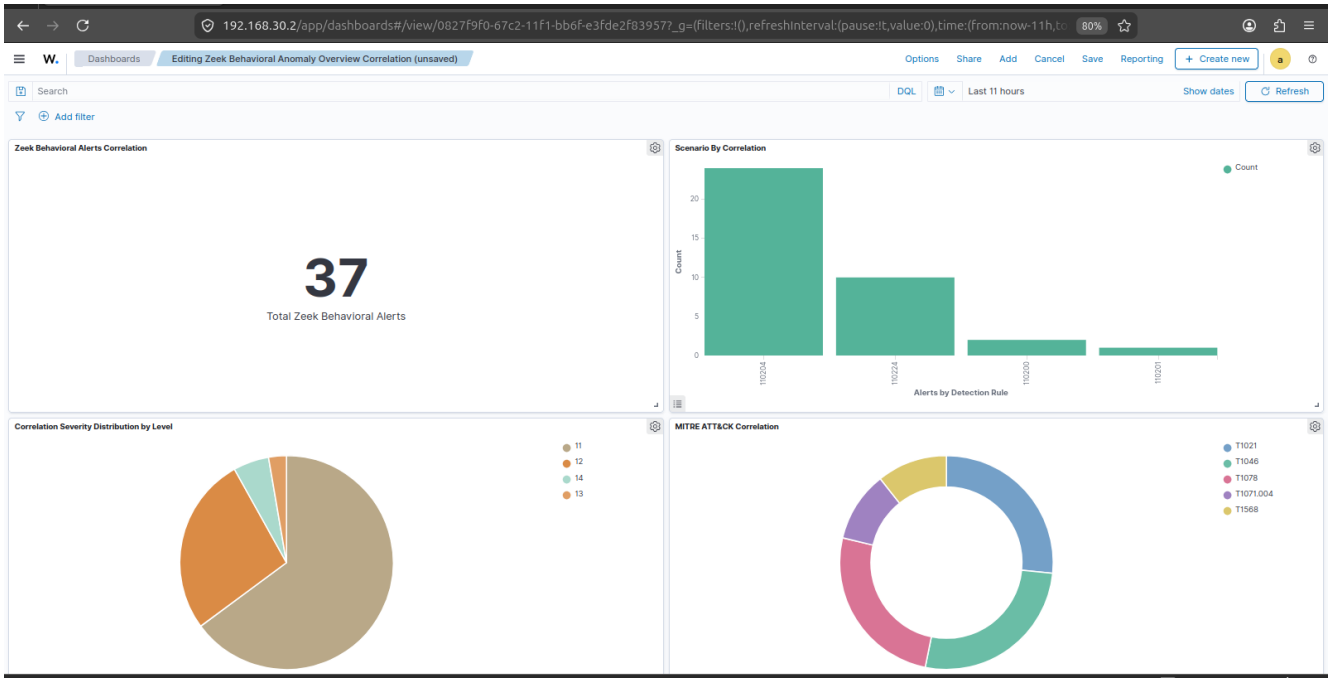


Рисунок 3.60 – Wazuh Dashboard з виведенням кореляційних аномалій

На відміну від загального Dashboard, ця панель орієнтована саме на події, які були сформовані в результаті поєднання декількох пов’язаних аномалій.

У верхній частині панелі відображається загальна кількість кореляційних алертів за вибраний часовий інтервал. У наведеному прикладі було сформовано 37 таких подій. Діаграма “Scenario by Correlation” показує розподіл алертів за ідентифікаторами кореляційних правил, що дозволяє визначити, які сценарії спрацьовували найчастіше.

Окремо наведено розподіл кореляційних алертів за рівнем критичності. Більшість подій має підвищений рівень небезпеки, що є логічним, оскільки кореляційні правила поєднують декілька окремих поведінкових відхилень в один більш значущий сценарій.

Також на панелі відображено розподіл алертів за техніками MITRE ATT&CK. Це дозволяє оцінити, з якими етапами атаки можуть бути пов’язані сформовані події, зокрема з lateral movement, discovery, command and control або іншими техніками. Такий підхід підвищує аналітичну цінність Dashboard, оскільки

дозволяє розглядати алерти не лише як окремі технічні події, а як частину можливого сценарію атаки.

Таким чином, створений Dashboard для кореляційних правил дозволяє швидко оцінити кількість складних інцидентів, їх розподіл за сценаріями, рівень критичності та відповідність технікам MITRE ATT&CK.

3.10. Оцінка результатів проведених експериментів

Після проведення експериментів було виконано оцінку результатів роботи системи виявлення поведінкових аномалій трафіку. Оцінювання здійснювалося на основі перевірки спрацювання Zeek-детекторів, Wazuh-правил та кореляційних сценаріїв для заздалегідь змодельованої аномальної активності.

Для кожного сценарію перевірялося, чи була сформована відповідна Zeek notice-подія, чи була вона передана до Wazuh, чи спрацювало потрібне правило та чи містив сформований алерт необхідні поля для подальшого аналізу. Таким чином було перевірено повний ланцюг виявлення: від мережевої активності до відображення структурованого алерту у Wazuh Dashboard.

У межах роботи для оцінювання результатів використовувалися такі поняття:

- True Positive (TP) – змодельована аномалія була коректно виявлена системою;
- False Positive (FP) – система сформувала алерт для легітимної активності;
- False Negative (FN) – змодельована аномалія не була виявлена системою.

Таблиця 3.5

Оцінка спрацювання алертів

Сценарій	Правило Wazuh	Фактичний результат	Оцінка
Нова комунікаційна пара	110002	Алерт сформовано	TP
SSH не з адміністративного сегмента	110003	Алерт сформовано	TP
Активність поза нормальними годинами	110006	Алерт сформовано	TP
Новий DNS -домен	110005	Алерт сформовано, можливі FP для легітимних нових доменів	TP/FP
Підозрілий DNS query	110004	Алерт сформовано	TP
DNS query flood	110102	Алерт сформовано після перевищення порогу	TP
DNS NXDOMAIN Spike	110101	Алерт сформовано	TP
Новий сервіс для відомої при	110007	Алерт сформовано, можливі FP при появі легітимного сервісу	TP/FP
Сканування портів	110100	Алерт сформовано після перевищення порогу унікальних портів	TP
Трафік великого обсягу	110103	Алерт сформовано при великому upload	TP
Network Beaconing	110000	Алерт сформовано, можливі FP для періодичних внутрішніх сервісів	TP/FP
SSH + активність поза нормальними годинами	110202	Кореляційний алерт сформовано	TP
Нова комунікаційна пара + порт скан	110201	Кореляційний алерт сформовано	TP
Beaconing + великий обсяг трафіку	110200	Кореляційний алерт сформовано	TP

Отримані результати показали, що реалізована система коректно виявляє основні змодельовані сценарії аномальної активності. Окремі правила дозволяють фіксувати конкретні поведінкові відхилення, а кореляційні правила підвищують інформативність алертів, оскільки поєднують декілька пов’язаних подій в один більш критичний сценарій

Для New_DNS_Domain false positive можуть виникати під час звернення до легітимного домену, який не був зафіксований під час формування baseline. Для

Network_Beaconing надлишкові спрацювання можуть бути пов’язані з регулярною активністю внутрішніх хостів, наприклад DNS-запитами, службовими перевірками або оновленнями. Для New_Service_For_Known_Pair false positive можливі у випадку появи нового легітимного сервісу або зміни нормальної поведінки вузлів.

Таким чином, результати експериментального дослідження підтверджують, що система Zeek + Wazuh здатна виявляти поведінкові відхилення в корпоративній мережі, формувати структуровані алерти та надавати достатній контекст для подальшого аналізу подій безпеки.

Додатково було проаналізовано відповідність сформованих алертів технікам MITRE ATT&CK. Зокрема, port scan і нові комунікаційні напрями пов’язані з тактикою Discovery, SSH-доступ - з Lateral Movement, DNS-аномалії та beaconing - з Command and Control, a large transfer - з Exfiltration.

3.11. Аналіз False Positives і налаштування порогів

Після оцінки результатів експериментального дослідження було виконано аналіз можливих хибнопозитивних спрацювань та визначено напрями налаштування системи під конкретну корпоративну мережу. Для поведінкового аналізу цей етап є важливим, оскільки система працює не лише з фіксованими сигнатурами, а й з відхиленнями від baseline нормальної поведінки.

У межах проведених експериментів більшість змодельованих аномалій були коректно класифіковані як true positive. Водночас окремі правила показали вищу чутливість до змін нормальної активності. Найбільше потенційних false positives було пов’язано з такими детекторами:

- DNSAnomalyDetector::New_DNS_Domain;
- BeaconingDetector::Network_Beaconing;
- NewServiceDetector::New_Service_For_Known_Pair.

Для детектора нового DNS-домену хибнопозитивні спрацювання можуть виникати тоді, коли користувач або система звертається до легітимного домену, який не був зафіксований під час формування baseline. Це можуть бути домени

оновлень, хмарних сервісів, CDN, зовнішніх web-ресурсів або служб синхронізації. Формально такий домен є новим для baseline, однак не обов'язково є шкідливим.

Для зменшення кількості таких спрацювань доцільно періодично оновлювати baseline DNS-активності та додавати підтверджені легітимні домени до дозволеного списку. При цьому важливо не додавати всі нові домени автоматично, оскільки це може знизити ефективність виявлення реальних DNS-аномалій.

Детектор network beaconing також може створювати надлишкові спрацювання, оскільки деякі внутрішні хости або служби мають регулярну періодичну активність. Наприклад, це можуть бути DNS-запити, перевірки доступності сервісів, оновлення або фонові з'єднання прикладного програмного забезпечення. Така активність може бути схожою на beaconing, хоча фактично є легітимною.

Для тюнінгу цього детектора необхідно враховувати не лише стабільність інтервалу між з'єднаннями, а й напрям взаємодії, IP-адресу призначення, порт, тип сервісу та роль вузла в мережі. Регулярні службові з'єднання внутрішніх хостів до довірених ресурсів можуть бути додані до винятків або враховані під час оновлення baseline.

Детектор нового сервісу для відомої пари хостів може спрацьовувати у випадку появи нового легітимного сервісу або зміни конфігурації мережі. Наприклад, якщо між двома вузлами раніше використовувався лише один порт, а після оновлення або зміни сервісу з'являється новий порт, система сформує алерт. Такий підхід є корисним для виявлення малих або повільних спроб розвідки, однак потребує періодичного уточнення baseline.

Окремого налаштування потребують статистичні пороги. Для port scan важливим є поріг кількості унікальних портів за визначений часовий інтервал. Якщо поріг занадто низький, система може помилково класифікувати звичайну активність як сканування. Якщо поріг занадто високий, частина повільних або обмежених сканувань може залишитися непоміченою.

Для DNS query flood потрібно враховувати нормальну поведінку клієнтів і служб, які можуть багаторазово звертатися до одного домену. У таких випадках

доцільно аналізувати, чи є домен легітимним, чи повторюваність запитів є типовою для конкретного вузла, і за потреби коригувати поріг або список винятків.

Для large transfer важливо враховувати напрям передавання даних. У реалізованій системі великий download із web-сервера не розглядається як аномалія, оскільки він може відповідати нормальній роботі сервісу. Натомість великий upload від користувацького вузла має більшу аналітичну цінність, оскільки може свідчити про потенційне передавання даних у нетиповому напрямі.

Для адаптації системи до конкретної корпоративної мережі доцільно виконувати такі дії:

- періодично оновлювати baseline після змін у мережі;
- додавати підтверджені легітимні DNS-домени до дозволеного списку;
- уточнювати перелік дозволених сервісів між відомими парами хостів;
- враховувати службову періодичну активність внутрішніх вузлів;
- коригувати пороги для port scan, DNS query flood, NXDOMAIN spike та large transfer;
- аналізувати повторювані false positives перед зміною правил;
- використовувати кореляційні правила для підвищення цінності алертів і зменшення шуму.

Окремим напрямом тюнінгу та подальшого розвитку системи є її масштабування для більшої корпоративної мережі. У такому випадку доцільно доповнити Zeek і Wazuh іншими засобами мережевого захисту, зокрема IDS/IPS та міжмережевим екраном. Якщо Zeek забезпечує глибокий пасивний аналіз трафіку та виявлення поведінкових відхилень, то IDS/IPS може додатково виявляти сигнатурні атаки, експлойти, шкідливі payload-и та відомі шаблони атак. Це дозволяє поєднати поведінковий і сигнатурний підходи до виявлення загроз.

Міжмережевий екран у такій архітектурі може використовуватися не лише для фільтрації трафіку між сегментами, а й для зменшення кількості потенційно небезпечних напрямів взаємодії. Наприклад, правила firewall можуть обмежувати доступ до адміністративних сервісів лише з адміністративної VLAN, блокувати

нетипові вихідні з’єднання, контролювати доступ до серверного сегмента та зменшувати площину атаки.

Для великих мереж також доцільно використовувати кілька сенсорів моніторингу в різних сегментах або на ключових мережевих вузлах. Це дозволить не перевантажувати один сенсор і забезпечить кращу видимість трафіку між VLAN, серверними зонами, користувацькими сегментами та зовнішніми напрямами. У такому випадку Wazuh може виконувати роль централізованої платформи для збору, кореляції та візуалізації подій із різних джерел.

Таким чином, для масштабування запропонованої системи доцільно поєднувати Zeek як інструмент поведінкового аналізу, IDS/IPS як засіб сигнатурного виявлення, firewall як механізм контролю доступу та Wazuh як центральну платформу для обробки й кореляції подій. Такий підхід дозволяє підвищити повноту виявлення загроз, зменшити кількість хибнопозитивних спрацювань і краще адаптувати систему до реальної корпоративної інфраструктури.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено проєкт системи виявлення аномалій трафіку корпоративної мережі на основі поведінкового аналізу. Основна ідея полягала в тому, щоб поєднати Zeek як сенсор пасивного аналізу мережевого трафіку та Wazuh як платформу для централізованої обробки подій, формування алертів, кореляції та візуалізації результатів.

У першому розділі було розглянуто теоретичні основи виявлення аномалій мережевого трафіку. Було визначено, що поведінковий аналіз є доцільним підходом для виявлення таких подій, які не завжди можна зафіксувати звичайними сигнатурними засобами. Окрему увагу було приділено поняттю baseline нормальної поведінки, оскільки саме порівняння поточної активності з попередньо сформованою моделлю дозволяє знаходити відхилення в роботі мережі.

У другому розділі було обґрунтовано архітектуру системи та спроєктовано мережеву інфраструктуру з поділом на користувацький, серверний, адміністративний, моніторинговий і зовнішній сегменти. Для цього використовувалися VLAN, ACL, NAT і SPAN. VLAN забезпечували логічне розділення мережі, ACL застосовувалися для контролю доступу між сегментами, NAT – для імітації зовнішньої взаємодії, а SPAN – для передавання копії трафіку на Zeek-сенсор.

У практичній частині було розгорнуто Zeek у пасивному режимі та налаштовано формування журналів у JSON-форматі. Також було налаштовано основні компоненти Wazuh: Manager, Indexer, Dashboard, Filebeat і Wazuh Agent. Агент на Zeek-вузлі зчитував notice.log, у якому фіксувалися події, створені кастомними Zeek-детекторами.

Важливим етапом стала побудова baseline нормальної поведінки мережі. Для цього використовувалися журнали Zeek, зокрема Conn.log і Dns.log, які оброблялися Python-скриптом. На основі цих даних було сформовано перелік відомих комунікаційних пар, сервісів, DNS-доменів та інших параметрів нормальної активності. Надалі ці дані використовувалися для виявлення відхилень.

У межах роботи було створено набір кастомних Zeek-детекторів. Вони дозволяли виявляти нову комунікаційну пару, новий сервіс для відомої пари хостів, SSH-доступ з неадміністративного сегмента, активність поза робочими годинами, новий DNS-домен, підозрілі DNS-запити, DNS query flood, NXDOMAIN spike, port scan, network beaconing та large transfer. Частина детекторів працювала на основі baseline, а частина – на основі статистичної агрегації подій за допомогою Zeek SumStats.

Для інтеграції Zeek із Wazuh було створено кастомні декодери та правила. Декодери витягували з Zeek notice-подій потрібні поля: IP-адреси джерела й призначення, порти, DNS-запити, кількісні показники та додатковий контекст. Завдяки цьому у Wazuh формувалися структуровані алерти, які було зручно аналізувати в Dashboard.

Окремо було налаштовано кореляційні правила Wazuh. Вони дозволили об’єднувати кілька окремих відхилень в один більш критичний сценарій. Наприклад, було реалізовано кореляцію network beaconing разом із large transfer, нової комунікаційної пари разом із port scan, а також активності поза робочими годинами разом із SSH-доступом з неадміністративного сегмента. Це зробило систему більш корисною з практичного погляду, оскільки вона почала показувати не тільки окремі події, а й можливу послідовність дій зловмисника.

Для частини правил також було виконано прив’язку до MITRE ATT&CK. Це дозволило пов’язати алерти з такими тактиками та техніками, як Discovery, Command and Control, Exfiltration та Lateral Movement. Завдяки цьому результати детекції стали зрозумілішими з точки зору аналізу інцидентів.

Експериментальна перевірка показала, що запропонований підхід працює коректно. Більшість змодельованих аномалій були виявлені системою та класифіковані як true positive. Стабільно спрацьовували правила для SSH-доступу з неадміністративного сегмента, активності поза робочими годинами, port scan, DNS query flood, NXDOMAIN spike, large transfer, а також кореляційні правила. Це підтвердило працездатність усього ланцюга: мережевий трафік - Zeek notice-подія - передавання до Wazuh - декодування - правило - алерт у Dashboard.

Під час тестування також було визначено можливі джерела false positive. Найбільш чутливими виявилися детектори нового DNS-домену, network beaconing та нового сервісу для відомої пари хостів. Це логічно, оскільки такі правила реагують на відхилення від baseline, а в реальній мережі нормальна активність може змінюватися. Тому для практичного використання системи потрібно періодично оновлювати baseline, уточнювати дозволені домени й сервіси та коригувати пороги під конкретну мережу.

Також у роботі було створено інформаційні панелі у Wazuh Dashboard. Вони дозволяють переглядати кількість алертів, розподіл подій за правилами, часову динаміку, основні IP-адреси, рівні критичності та відповідність MITRE ATT&CK. Це показує, що система може використовуватися не лише для фіксації окремих спрацювань, а й для подальшого аналізу інцидентів.

Отримані результати підтверджують, що поєднання Zeek і Wazuh може бути використане як основа для виявлення поведінкових аномалій у корпоративній мережі. Перевагою такого підходу є те, що в одному середовищі поєднуються baseline-аналіз, статистичні методи, правила Wazuh, кореляція подій і візуалізація результатів.

Подальший розвиток системи можна пов’язати з використанням ML-моделей поверх Zeek-логів, наприклад Isolation Forest або autoencoders. Також доцільною є інтеграція з MISP або іншими threat intelligence-фідами для збагачення IP-адрес, доменів та інших індикаторів компрометації. Ще одним напрямом розвитку може бути SOAR-автоматизація реагування на критичні кореляційні алерти та додавання EDR-рівня для поєднання мережевої і хостової телеметрії.

Отже, у кваліфікаційній роботі було досягнуто поставленої мети, а саме розроблено проєкту корпоративної мережі з системою виявлення аномалій трафіку на основі поведінкового аналізу із застосуванням SIEM платформи Wazuh та мережевого аналізатору трафіку Zeek.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Maseer Z. K., Kadhim Q. K., Al-Bander B., Yusof R., Saif A. Meta-analysis and systematic review for anomaly network intrusion detection systems: detection methods, dataset, validation methodology, and challenges. IET Networks. 2024. Vol. 13, № 5–6. P. 339–376. DOI: 10.1049/ntw2.12128. URL: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/ntw2.12128 (дата звернення: 15.06.2026).
2. Lyu M., Gharakheili H. H., Sivaraman V. A Survey on Enterprise Network Security: Asset Behavioral Monitoring and Distributed Attack Detection. IEEE Access. 2024. Vol. 12. P. 89363–89383. DOI: 10.1109/ACCESS.2024.3419068. URL: https://doi.org/10.1109/ACCESS.2024.3419068 (дата звернення: 3.06.2026).
3. Chandola V., Banerjee A., Kumar V. Anomaly Detection: A Survey. ACM Computing Surveys. 2009. Vol. 41, № 3. Article 15. P. 1–58. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882 (дата звернення: 5.06.2026).
4. Hsiao H.-W., Lee Y.-Z. Entropy-Based Anomaly Detection for Cybersecurity Threats in Network Traffic // Multidisciplinary Social Networks Research. MISNC 2025 / ed. by V. Garcia Diaz, I.-H. Ting, K. Wang. Cham : Springer, 2026. Communications in Computer and Information Science. Vol. 2729. P. 106–117. DOI: 10.1007/978-3-032-09945-7_10. URL: https://doi.org/10.1007/978-3-032-09945-7_10 (дата звернення: 07.06.2026)
5. Корченко А. О. Методи ідентифікації аномальних станів для систем виявлення вторгнень : монографія / А. О. Корченко. Київ : ЦП «Компринт», 2019. 361 с.
6. Threat Hunting: DNS Queries Use Case. *ReliaQuest*. 16.08.2021. URL: https://reliaquest.com/blog/threat-hunting-use-case-dns-queries (дата звернення: 09.06.2026).
7. Sharma S. Detecting and Controlling Hidden DNS Tunnel Attacks. *Fidelis Security*. 05.03.2025. URL: https://fidelissecurity.com/threatgeek/learn/dns-tunneling-detection (дата звернення: 09.06.2026).

8. Maosa H., Ouazzane K., Ghanem M. C. A Hierarchical Security Event Correlation Model for Real-Time Threat Detection and Response. Network. 2024. Vol. 4, № 1. P. 68–90. DOI: 10.3390/network4010004. URL: https://doi.org/10.3390/network4010004 (дата звернення: 10.06.2026).

9. Baselining. *Black Fog*. URL: https://www.blackfog.com/cybersecurity-101/baselining/ (дата звернення: 06.06.2026).

10. What is Zeek?. *Zeek*. URL: https://docs.zeek.org/en/master/about/what.html (дата звернення: 09.06.2026).

11. weird.log and notice.log. *Zeek*. URL: https://docs.zeek.org/en/master/reference/logs/weird-and-notice.html#notice-log (дата звернення: 10.06.2026).

12. MITRE ATT&CK framework. *Wazuh*. URL: https://documentation.wazuh.com/current/user-manual/ruleset/mitre.html (дата звернення: 10.06.2026).

13. Monitoring With Zeek. *Zeek*. URL: https://docs.zeek.org/en/master/about/monitoring.html#storage-and-review (дата звернення: 10.06.2026).

14. Notice Framework. Zeek. URL: https://docs.zeek.org/en/master/frameworks/notice.html (дата звернення: 10.06.2026).

15. Summary Statistics. *Zeek*. URL: https://docs.zeek.org/en/master/frameworks/sumstats.html#overview (дата звернення: 10.06.2026).

16. 11. Wazuh Architecture. Wazuh. URL: https://documentation.wazuh.com/current/getting-started/architecture.html (дата звернення: 10.06.2026).

17. Wazuh. Network security monitoring with Wazuh and Zeek. Wazuh Blog. URL: https://wazuh.com/blog/network-security-monitoring-with-wazuh-and-Zeek/

18. Creating a Network Traffic Baseline and Detecting Anomalies with Zeek: An Advanced SOC Analyst Guide. Ramblings of a CyberSecurity Nerd. 2025. URL:

https://socdfir.com/2025/07/08/creating-a-network-traffic-baseline-and-detecting-anomalies-with-zeek-an-advanced-soc-analyst-guide/ (дата звернення: 12.06.2026).

19. Cisco. Configure Inter VLAN Routing with the Use of an External Router. Cisco Support Documentation. URL: https://www.cisco.com/c/en/us/support/docs/lan-switching/inter-vlan-routing/14976-50.html

20. SPAN, RSPAN, ERSPAN. Cisco Learning Network. URL: https://learningnetwork.cisco.com/s/article/span-rspan-erspan (дата звернення: 10.06.2026).

21. Wazuh. Rules Syntax. Wazuh Documentation. URL: https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/rules.html

22. Про захист інформації в інформаційно-комунікаційних системах : Закон України від 05.07.1994 № 80/94-ВР // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/80/94-%D0%B2%D1%80

23. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection — Information security management systems — Requirements. International Organization for Standardization, 2022. URL: https://www.iso.org/standard/27001 (дата звернення: 12.06.2026).

24. Scarfone K., Mell P. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94. Gaithersburg : National Institute of Standards and Technology, 2007. DOI: 10.6028/NIST.SP.800-94. URL: https://doi.org/10.6028/NIST.SP.800-94 (дата звернення: 12.06.2026).

25. Dempsey K., Chawla N. S., Johnson A., Johnston R., Jones A. C., Orebaugh A., Scholl M., Stine K. Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations. NIST Special Publication 800-137. Gaithersburg : National Institute of Standards and Technology, 2011. DOI: 10.6028/NIST.SP.800-137. URL: https://doi.org/10.6028/NIST.SP.800-137 (дата звернення: 12.06.2026)

ДОДАТКИ

Local.Zeek

```
admin@Zeek:~$ sudo cat /opt/Zeek/share/Zeek/site/local.Zeek
##! Local site policy. Customize as appropriate.
##!
##! This file will not be overwritten when upgrading or reinstalling!

# Installation-wide salt value that is used in some digest hashes, e.g., for
# the creation of file IDs. Please change this to a hard to guess value.
redef digest_salt = "Please change this value.";

# This script logs which scripts were loaded during each run.
@load misc/loaded-scripts

# Estimate and log capture loss.
@load misc/capture-loss

# Enable logging of memory, packet and lag statistics.
@load misc/stats

# For TCP scan detection, we recommend installing the package from
# 'https://github.com/ncsa/bro-simple-scan'. E.g., by installing it via
#
#     zkg install ncsa/bro-simple-scan

# Detect traceroute being run on the network. This could possibly cause
# performance trouble when there are a lot of traceroutes on your network.
# Enable cautiously.
#@load misc/detect-traceroute

# Generate notices when vulnerable versions of software are discovered.
# The default is to only monitor software found in the address space defined
# as "local".  Refer to the software framework's documentation for more
# information.
@load frameworks/software/vulnerable

# Detect software changing (e.g. attacker installing hacked SSHD).
@load frameworks/software/version-changes

# This adds signatures to detect cleartext forward and reverse windows shells.
@load-sigs frameworks/signatures/detect-windows-shells

# Load all of the scripts that detect software in various protocols.
@load protocols/ftp/software
@load protocols/smtp/software
@load protocols/ssh/software
@load protocols/http/software
# The detect-webapps script could possibly cause performance trouble when
# running on live traffic.  Enable it cautiously.
#@load protocols/http/detect-webapps

# This script detects DNS results pointing toward your Site::local_nets
# where the name is not part of your local DNS zone and is being hosted
# externally.  Requires that the Site::local_zones variable is defined.
@load protocols/dns/detect-external-names

# Script to detect various activity in FTP sessions.
@load protocols/ftp/detect

# Scripts that do asset tracking.
@load protocols/conn/known-hosts
@load protocols/conn/known-services
@load protocols/ssl/known-certs

# This script enables SSL/TLS certificate validation.
@load protocols/ssl/validate-certs

# This script prevents the logging of SSL CA certificates in x509.log
@load protocols/ssl/log-hostcerts-only

# If you have GeoIP support built in, do some geographic detections and
# logging for SSH traffic.
@load protocols/ssh/geo-data
# Detect hosts doing SSH bruteforce attacks.
@load protocols/ssh/detect-bruteforcing
# Detect logins using "interesting" hostnames.
@load protocols/ssh/interesting-hostnames

# Detect SQL injection attacks.
@load protocols/http/detect-sql-injection

#### Network File Handling ####

# Enable MD5 and SHA1 hashing for all files.
@load frameworks/files/hash-all-files

# Detect SHA1 sums in Team Cymru's Malware Hash Registry.
@load frameworks/files/detect-MHR

# Extend email alerting to include hostnames
@load policy/frameworks/notice/extend-email/hostnames

# Extend the notice.log with Community ID hashes
# @load policy/frameworks/notice/community-id

# Enable logging of telemetry data into telemetry.log and
# telemetry_histogram.log.
@load frameworks/telemetry/log

# Uncomment the following line to enable detection of the heartbleed attack. Enabling
# this might impact performance a bit.
# @load policy/protocols/ssl/heartbleed

# Uncomment the following line to enable logging of Community ID hashes in
# the Conn.log file.
# @load policy/protocols/conn/community-id-logging
```

```
# Uncomment the following line to enable logging of connection VLANs. Enabling
# this adds two VLAN fields to the Conn.log file.
# @load policy/protocols/conn/vlan-logging

# Uncomment the following line to enable logging of link-layer addresses. Enabling
# this adds the link-layer address for each connection endpoint to the Conn.log file.
# @load policy/protocols/conn/mac-logging

# Uncomment this to source zkg's package state
# @load packages

@load policy/tuning/json-logs.Zeek

@load ./baseline_data.Zeek
@load ./common.Zeek
@load ./new_pair_detector.Zeek
@load ./ssh_policy_detector.Zeek
#@load ./portscan_detector.Zeek
#@load ./transfer_anomaly_detector.Zeek
@load ./beaconing_detector.Zeek
@load ./dns_anomaly_detector.Zeek

@load ./sumstats_detectors.Zeek
@load ./active_hours_detector.Zeek

@load ./new_service_detector.Zeek
@load ./suspicious_user_agent_detector.Zeek

#filters
@load ./common_filters
```

Python baseline-script

```
#!/usr/bin/env python3
import json
import gzip
import ipaddress
from pathlib import Path
from collections import defaultdict

ZEEK_LOG_DIRS = [
Path("/opt/Zeek/logs/current"),
Path("/opt/Zeek/logs/2026-06-08"),
Path("/opt/Zeek/logs/2026-06-09"),
Path("/opt/Zeek/logs/2026-06-10"),
]
OUTPUT_FILE = "/home/admin/baselines_script/baseline_data.Zeek"

INTERNAL_NETS = [
ipaddress.ip_network("192.168.10.0/28"),
ipaddress.ip_network("192.168.20.0/24"),
ipaddress.ip_network("192.168.30.0/29"),
ipaddress.ip_network("192.168.40.0/30"),
]

known_pairs = set()
known_services = set()
known_dns_domains = set()
active_hours = defaultdict(set)
transfer_sizes = defaultdict(list)


def is_internal(ip):
try:
ip_obj = ipaddress.ip_address(ip)
return any(ip_obj in net for net in INTERNAL_NETS)
except Exception:
return False


def open_log(path):
if str(path).endswith(".gz"):
return gzip.open(path, "rt", encoding="utf-8", errors="ignore")
return open(path, "r", encoding="utf-8", errors="ignore")


def read_json_logs(pattern):
for log_dir in ZEEK_LOG_DIRS:
if not log_dir.exists():
continue

for path in log_dir.glob(pattern):
with open_log(path) as f:
for line in f:
line = line.strip()
if not line or not line.startswith("{"):
continue
try:
yield json.loads(line)
except Exception:
continue

def process_conn():
for rec in read_json_logs("conn*"):
src = rec.get("id.orig_h")
dst = rec.get("id.resp_h")
port = rec.get("id.resp_p")
proto = rec.get("proto")
ts = rec.get("ts")

if not src or not dst:
continue

if not is_internal(src):
continue

if is_internal(dst):
known_pairs.add((src, dst))
if port and proto:
known_services.add((src, dst, str(port), proto))

if ts:
hour = int(float(ts) % 86400 // 3600)
active_hours[src].add(hour)

orig = int(rec.get("orig_bytes") or 0)
resp = int(rec.get("resp_bytes") or 0)
total = orig + resp

if total > 0:
transfer_sizes[src].append(total)


def process_dns():
for rec in read_json_logs("dns*"):
src = rec.get("id.orig_h")
query = rec.get("query")

if not src or not query:
continue

if not is_internal(src):
continue

q = query.lower().rstrip(".")
known_dns_domains.add(q)


def avg(values):
if not values:
return 0
return int(sum(values) / len(values))


def write_baseline():
with open(OUTPUT_FILE, "w", encoding="utf-8") as f:
f.write("@load base/utils/site\n\n")
f.write("module Baseline;\n\n")
f.write("export {\n")

f.write("    const known_pairs: set[string] = {\n")
for src, dst in sorted(known_pairs):
```

```
f.write(f'        "{src} {dst}",\n')
f.write("    } &redef;\n\n")

f.write("    const known_services:
set[string] = {\n")
for src, dst, port, proto in
sorted(known_services):
f.write(f'        "{src} {dst} {port}
{proto}",\n')
f.write("    } &redef;\n\n")

f.write("    const
known_dns_domains: set[string] =
{\n")
for domain in
sorted(known_dns_domains):
f.write(f'        "{domain}",\n')
f.write("    } &redef;\n\n")

f.write("    const
normal_active_hours: table[string]
of set[count] = {\n")
for host, hours in
sorted(active_hours.items()):
h = ", ".join(str(x) for x in
sorted(hours))
f.write(f'        ["{host}"] =
set({h}),\n')
f.write("    } &redef;\n\n")

f.write("    const avg_transfer_size:
table[string] of count = {\n")
for host, values in
sorted(transfer_sizes.items()):
f.write(f'        ["{host}"] =
{avg(values)},\n')
f.write("    } &redef;\n")

f.write("}\n")


if __name__ == "__main__":
process_conn()
process_dns()
write_baseline()
print(f"[+] Baseline generated:
{OUTPUT_FILE}")
print(f"[+] known_pairs:
{len(known_pairs)}")
print(f"[+] known_services:
{len(known_services)}")
print(f"[+] known_dns_domains:
{len(known_dns_domains)}")

        baseline_data.Zeek
@load base/utils/site

module Baseline;

export {
    const known_pairs: set[string] =
{
        "192.168.20.2
192.168.10.10",
        "192.168.40.2
192.168.10.10",
        "192.168.40.2 192.168.20.2",
        "192.168.40.2 192.168.30.3",
        "192.168.30.3 192.168.30.1",
        "192.168.30.3 192.168.40.2",
            "192.168.10.1
192.168.10.10",
            "192.168.20.1
192.168.20.2",
            "192.168.30.1
192.168.30.2",
    } &redef;

    const known_services:
set[string] = {
        "192.168.20.2 192.168.10.10
53 udp",
        "192.168.20.2 192.168.10.10
53 tcp",
        "192.168.20.2 192.168.10.10
8080 tcp",
        "192.168.40.2 192.168.10.10
22 tcp",
        "192.168.40.2 192.168.20.2
22 tcp",
        "192.168.40.2 192.168.30.3
22 tcp",
    } &redef;

    const known_dns_domains:
set[string] = {
        "dvwa.alexxx.net",
            "github.alexxx.net",
            "mail.alexxx.net",
        "10.10.168.192.in-addr.arpa",
        "_vlmcs._tcp.itdep.local",
        "_webdav._tcp.local",

"au.download.windowsupdate.com
",
        "crl.microsoft.com",
        "dvwa.alexxx.net",
        "fe2.update.microsoft.com",
        "isatap",
        "teredo.ipv6.microsoft.com",
        "time.windows.com",
        "update.test.local",
        "update.test.local.itdep.local",
        "watson.microsoft.com",
        "workgroup",
        "wpad",
        "wpad.itdep.local",
        "ws-a-1",
        "www.bing.com",

"www.download.windowsupdate.c
om",
        "www.microsoft.com",
    } &redef;

    const normal_active_hours:
table[string] of set[count] = {
        ["192.168.10.10"] =
set(0,1,2,3,4,5,6,7,8,9,10,11,12,13,
14,15,16,17,18,19,20,21,22,23),

        ["192.168.20.2"] =
set(8,9,10,11,12,13,14,15,16,17,18
),

        ["192.168.30.3"] =
set(0,1,2,3,4,5,6,7,8,9,10,11,12,13,
14,15,16,17,18,19,20,21,22,23),
    } &redef;

    const avg_transfer_size:
table[string] of count = {
        ["192.168.20.2"] = 504120,
        ["192.168.40.2"] = 301032,
        ["192.168.10.10"] =
12530021,
        ["192.168.30.3"] = 2432000,
    } &redef;
}

        Sumstats_detectors.Zeek
@load base/frameworks/notice
@load base/frameworks/sumstats
@load
base/frameworks/sumstats/plugins/
sum
@load
base/frameworks/sumstats/plugins/
unique
@load
base/frameworks/sumstats/plugins/
max
@load ./common.Zeek

module SumStatsDetectors;

export {
redef enum Notice::Type += {
SumStats_Port_Scan,
SumStats_DNS_NXDOMAIN_Spi
ke,
SumStats_DNS_Query_Flood,
SumStats_Large_Transfer
};
}

const
PORT_SCAN_UNIQUE_PORTS:
double = 20.0 &redef;
const
NXDOMAIN_THRESHOLD:
double = 10.0 &redef;
const
DNS_QUERY_FLOOD_THRESH
OLD: double = 15.0 &redef;
const
TRANSFER_MAX_THRESHOL
D: double = 5000000.0 &redef;
```

```
event Zeek_init()
{
local portscan = SumStats::Reducer($stream="sumstats.portscan", $apply=set(SumStats::UNIQUE));

SumStats::create([$name="sumstats.portscan.detector",
$epoch=60sec,
$reducers=set(portscan),
$threshold=PORT_SCAN_UNIQUE_PORTS,
$threshold_val(key: SumStats::Key, result: SumStats::Result) =
{
return result["sumstats.portscan"]$unique + 0.0;
},
$threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
{
local r = result["sumstats.portscan"];

NOTICE([$note=SumStats_Port_Scan,
$msg="SumStats port scan detected",
$src=key$host,
$sub=fmt("target=%s unique_ports=%d", key$str, r$unique),
$identifier=fmt("%s-%s", key$host, key$str),
$suppress_for=60sec]);
}]);

local nxd = SumStats::Reducer($stream="sumstats.dns.nxdomain", $apply=set(SumStats::SUM));

SumStats::create([$name="sumstats.dns.nxdomain.detector",
$epoch=60sec,
$reducers=set(nxd),
$threshold=NXDOMAIN_THRESHOLD,
$threshold_val(key: SumStats::Key, result: SumStats::Result) =
{
return result["sumstats.dns.nxdomain"]$sum;
},
$threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
{
local r = result["sumstats.dns.nxdomain"];

NOTICE([$note=SumStats_DNS_NXDOMAIN_Spike,
$msg="SumStats DNS NXDOMAIN spike detected",
$src=key$host,
$sub=fmt("count=%.0f", r$sum),
$identifier=fmt("%s", key$host),
$suppress_for=60sec]);
}]);

local dnsf = SumStats::Reducer($stream="sumstats.dns.query_count", $apply=set(SumStats::SUM));

SumStats::create([$name="sumstats.dns.query_flood.detector",
$epoch=60sec,
$reducers=set(dnsf),
$threshold=DNS_QUERY_FLOOD_THRESHOLD,
$threshold_val(key: SumStats::Key, result: SumStats::Result) =
{
return result["sumstats.dns.query_count"]$sum;
},
$threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
{
local r = result["sumstats.dns.query_count"];

NOTICE([$note=SumStats_DNS_Query_Flood,
$msg="SumStats DNS repeated query/flood detected",
$src=key$host,
$sub=fmt("query=%s count=%.0f", key$str, r$sum),
$identifier=fmt("%s-%s", key$host, key$str),
$suppress_for=60sec]);
}]);

local transfer = SumStats::Reducer($stream="sumstats.transfer.max", $apply=set(SumStats::MAX));

SumStats::create([$name="sumstats.transfer.detector",
$epoch=5min,
$reducers=set(transfer),
$threshold=TRANSFER_MAX_THRESHOLD,
$threshold_val(key: SumStats::Key, result: SumStats::Result) =
{
return result["sumstats.transfer.max"]$max;
},
$threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
{
local r = result["sumstats.transfer.max"];

NOTICE([$note=SumStats_Large_Transfer,
$msg="SumStats large transfer detected",
$src=key$host,
$sub=fmt("flow=%s max_bytes=%.0f", key$str, r$max),
$identifier=fmt("%s-%s", key$host, key$str),
$suppress_for=60sec]);
}]);
}

event connection_state_remove(c: connection)
{
local src = c$id$orig_h;
local dst = c$id$resp_h;
local p = c$id$resp_p;

if ( ! Common::is_internal(dst) )
return;

SumStats::observe("sumstats.portscan",
[$host=src, $str=fmt("%s", dst)],
[$str=fmt("%s", p)]);

local total_bytes: double = (c$orig$size + c$resp$size) + 0.0;

SumStats::observe("sumstats.transfer.max",
[$host=src, $str=fmt("%s:%s", dst, p)],
[$dbl=total_bytes]);
}

event DNS::log_dns(rec: DNS::Info)
{
if ( ! rec?$query )
return;
```

```
local src = rec$id$orig_h;
local dst = rec$id$resp_h;
local q = rec$query;

if ( Common::is_noise_addr(src) ||
Common::is_noise_addr(dst) )
return;

if                                    (
Common::is_noise_dns_query(q) )
return;


if ( ! Common::is_internal(src) )
return;

if     (     rec?$rcode_name     &&
rec$rcode_name                    ==
"NXDOMAIN" )
SumStats::observe("sumstats.dns.n
xdomain",
[$host=src],
[$num=1]);

SumStats::observe("sumstats.dns.q
uery_count",
[$host=src, $str=q],
[$num=1]);
}
```

Ssh_policy_detector.Zeek

```
@load base/frameworks/notice
@load ./common.Zeek

module SSHPolicyDetector;

export {
   redef enum Notice::Type += {

SSH_From_Non_Admin_VLAN
   };
}

event connection_state_remove(c:
connection)
   {
   local src = c$id$orig_h;
   local dst = c$id$resp_h;
   local p = c$id$resp_p;

   if ( ! Common::is_internal(dst) )
      return;

   if ( p == 22/tcp && !
Common::is_admin(src) )
      {

NOTICE([$note=SSH_From_Non
_Admin_VLAN,
            $msg="SSH attempt
from non-admin VLAN",
            $src=src,
            $dst=dst,
            $p=p,

$sub="policy=ssh_only_from_adm
in_vlan",
            $identifier=fmt("%s-
%s", src, dst),
            $suppress_for=5min]);
      }
   }
```

Active_hours_detector.Zeek

```
@load base/frameworks/notice
@load ./baseline_data.Zeek
@load ./common.Zeek

module ActiveHoursDetector;

export {
   redef enum Notice::Type += {

Activity_Outside_Normal_Hours
   };
}

event connection_state_remove(c:
connection)
   {
   local src = c$id$orig_h;
   local dst = c$id$resp_h;

   if ( ! Common::is_internal(src) )
      return;

   local src_s = fmt("%s", src);

   if ( src_s !in
Baseline::normal_active_hours )
      return;

  local current_hour =
(to_count(strftime("%H",
network_time())) + 2) % 24;

   if ( current_hour !in
Baseline::normal_active_hours[src
_s] )
      {

NOTICE([$note=Activity_Outside
_Normal_Hours,
            $msg="Host activity
outside normal hours",
            $src=src,
            $dst=dst,
            $sub=fmt("hour=%d",
current_hour),
            $identifier=fmt("%s-
hour-%d", src, current_hour),
            $suppress_for=30min]);
      }
   }
```

New_pair_detector.Zeek

```
@load base/frameworks/notice
@load ./baseline_data.Zeek
@load ./common.Zeek

module NewPairDetector;

export {
   redef enum Notice::Type += {
      New_Communication_Pair
   };
}

event connection_state_remove(c:
connection)
   {
   local src = c$id$orig_h;
   local dst = c$id$resp_h;

   if ( ! Common::is_internal(src) ||
! Common::is_internal(dst) )
      return;

   local pair = fmt("%s %s", src,
dst);

   if ( pair !in
Baseline::known_pairs )

NOTICE([$note=New_Communic
ation_Pair,
            $msg="New
communication pair detected",
            $src=src,
            $dst=dst,
            $sub=fmt("pair=%s",
pair),
            $identifier=pair,
            $suppress_for=5min]);
   }
```

New_service_detector.Zeek

```
@load base/frameworks/notice
@load ./baseline_data.Zeek
@load ./common.Zeek

module NewServiceDetector;

export {
   redef enum Notice::Type += {

New_Service_For_Known_Pair
   };
}

event connection_state_remove(c:
connection)
   {
   local src = c$id$orig_h;
   local dst = c$id$resp_h;
   local p = c$id$resp_p;
   local proto = c$id$proto;
```

```
   if ( c?$conn &&
c$conn?$conn_state )
   {
   if ( c$conn$conn_state != "SF"
&&
        c$conn$conn_state != "S1"
&&
        c$conn$conn_state != "S2"
&&
        c$conn$conn_state != "S3" )
       return;
   }

   if ( ! Common::is_internal(src) ||
! Common::is_internal(dst) )
       return;

   local service = fmt("%s %s %s
%s", src, dst, p, proto);

   if ( service !in
Baseline::known_services )
       {

NOTICE([$note=New_Service_Fo
r_Known_Pair,
            $msg="New service for
known communication pair",
            $src=src,
            $dst=dst,
            $p=p,
            $sub=fmt("proto=%s
service=%s", proto, service),
            $identifier=service,
            $suppress_for=5min]);
       }
   }
      Beaconing_detector.Zeek
@load base/frameworks/notice
@load ./common.Zeek

module BeaconingDetector;

export {
redef enum Notice::Type += {
Network_Beaconing
};
}

const     BEACON_TOLERANCE:
interval = 3sec &redef;
const     BEACON_MIN_STABLE:
count = 4 &redef;

global last_conn_time: table[string]
of time &write_expire=10min;
global              last_conn_interval:
table[string]          of          interval
&write_expire=10min;
global              conn_stable_count:
table[string]          of          count
&write_expire=10min;

event   connection_state_remove(c:
connection)
{
local src = c$id$orig_h;
local dst = c$id$resp_h;
local p = c$id$resp_p;

if ( ! Common::is_internal(src) )
return;

local key = fmt("%s %s %s", src,
dst, p);
local now_t = network_time();

if ( key in last_conn_time )
{
local     delta     =     now_t     -
last_conn_time[key];

if ( key in last_conn_interval )
{
local                diff                =
Common::interval_diff(delta,
last_conn_interval[key]);

if              (              diff              <=
BEACON_TOLERANCE              &&
delta > 5sec )
{
if ( key !in conn_stable_count )
conn_stable_count[key] = 0;

++conn_stable_count[key];

if   (   conn_stable_count[key]   >=
BEACON_MIN_STABLE )
NOTICE([$note=Network_Beacon
ing,
$msg="Network          beaconing
detected",
$src=src,
$dst=dst,
$p=p,
$sub=fmt("interval=%s
stable_count=%d",              delta,
conn_stable_count[key]),
$identifier=key,
$suppress_for=5min]);
}
else
conn_stable_count[key] = 0;
}

last_conn_interval[key] = delta;
}

last_conn_time[key] = now_t;
}

    Dns_anomaly_detector.Zeek
load base/frameworks/notice
@load ./baseline_data.Zeek
@load ./common.Zeek

module DNSAnomalyDetector;

export {
   redef enum Notice::Type += {
      New_DNS_Domain,
      Suspicious_DNS_Query,
      DNS_NXDOMAIN_Spike,
      DNS_Beaconing
   };
}

const
NXDOMAIN_THRESHOLD:
count = 10 &redef;
const
DNS_BEACON_TOLERANCE:
interval = 3sec &redef;
const
DNS_BEACON_MIN_STABLE:
count = 4 &redef;

global              nxdomain_count:
table[addr]          of          count
&write_expire=60sec;

global last_dns_time: table[string]
of time &write_expire=10min;
global              last_dns_interval:
table[string]          of          interval
&write_expire=10min;
global              dns_stable_count:
table[string]          of          count
&write_expire=10min;

function suspicious_dns_score(q:
string): count
   {
   local score: count = 0;

   if ( |q| > 25 )
      ++score;

   if ( /[0-9]/ in q )
      ++score;

   if ( /[a-z0-9]{12,}/ in q )
      ++score;

   if ( /[a-z][0-9][a-z]/ in q )
      ++score;

   return score;
   }

event              DNS::log_dns(rec:
DNS::Info)
   {
   if ( ! rec?$query )
      return;

   local q = rec$query;
   local src = rec$id$orig_h;
```

```
    local dst = rec$id$resp_h;


   if ( Common::is_noise_addr(src)
|| Common::is_noise_addr(dst) )
    return;

   if                                  (
Common::is_noise_dns_query(q) )
    return;


   if          (          q          !in
Baseline::known_dns_domains )

NOTICE([$note=New_DNS_Dom
ain,
            $msg="New          DNS
domain observed",
            $src=src,
            $dst=dst,
            $sub=fmt("query=%s",
q),
            $identifier=fmt("%s-
%s", src, q),
            $suppress_for=1hr]);

   if          (          q          !in
Baseline::known_dns_domains
&& suspicious_dns_score(q) >= 3 )

NOTICE([$note=Suspicious_DNS
_Query,
            $msg="Suspicious DNS
query detected",
            $src=src,
            $dst=dst,
            $sub=fmt("query=%s
score=%d",                    q,
suspicious_dns_score(q)),
            $identifier=fmt("%s-
%s", src, q),
            $suppress_for=10min]);

   if    (    rec?$rcode_name    &&
rec$rcode_name                 ==
"NXDOMAIN" )
      {
      if ( src !in nxdomain_count )
         nxdomain_count[src] = 0;

      ++nxdomain_count[src];

      if ( nxdomain_count[src] >=
NXDOMAIN_THRESHOLD )
         {

NOTICE([$note=DNS_NXDOMA
IN_Spike,
            $msg="DNS
NXDOMAIN spike detected",
            $src=src,
            $dst=dst,
$sub=fmt("count=%d",
nxdomain_count[src]),
            $identifier=fmt("%s",
src),
            $suppress_for=5min]);

         nxdomain_count[src] = 0;
         }
      }

   local key = fmt("%s %s", src, q);
   local now_t = network_time();

   if ( key in last_dns_time )
      {
      local delta = now_t -
last_dns_time[key];

      if ( key in last_dns_interval )
         {
         local diff =
Common::interval_diff(delta,
last_dns_interval[key]);

         if ( diff <=
DNS_BEACON_TOLERANCE
&& delta > 5sec )
            {
            if ( key !in
dns_stable_count )
               dns_stable_count[key]
= 0;


++dns_stable_count[key];

            if                    (
dns_stable_count[key]          >=
DNS_BEACON_MIN_STABLE )

NOTICE([$note=DNS_Beaconing,
               $msg="DNS
beaconing detected",
               $src=src,
               $dst=dst,

$sub=fmt("query=%s  interval=%s
stable_count=%d",    q,    delta,
dns_stable_count[key]),
               $identifier=key,

$suppress_for=10min]);
            }
         else
            dns_stable_count[key] =
0;
         }

      last_dns_interval[key] = delta;
      }

   last_dns_time[key] = now_t;
   }


      Local_decoder.xml

   !-- Local Decoders -->

<!-- Modify it at your will. -->
<!-- Copyright (C) 2015, Wazuh
Inc. -->

   <!--
   - Allowed static fields:
- location   - where the log came
from (only on FTS)
- srcuser    - extracts the source
username
   - dstuser    - extracts the
destination (target) username
- user       - an alias to dstuser
(only one of the two can be used)
   - srcip      - source ip
    - dstip      - dst ip
   - srcport    - source port
  - dstport    - destination port
    - protocol   - protocol
     - id        - event id
  - url        - url of the event
- action     - event action (deny,
drop, accept, etc)
- status     - event status (success,
failure, etc)
- extra_data - Any extra data
   -->

   <decoder name="Zeek-json-
sumstats-large-transfer">
     <parent>json</parent>

<prematch>SumStatsDetectors::Su
mStats_Large_Transfer</prematch
>
          <regex
type="pcre2">"note":"([^"]+)".*"s
ub":"(flow=([^ ]+) max_bytes=([0-
9]+))".*"src":"([^"]+)"</regex>

<order>Zeek_note,Zeek_sub,transf
er_flow,transfer_bytes,srcip</order
>
          </decoder>

   <decoder name="Zeek-json-dns-
nxdomain-spike">
     <parent>json</parent>

<prematch>DNS_NXDOMAIN_S
pike</prematch>
          <regex
type="pcre2">"note"\s*:\s*"([^"]+
)".*"sub"\s*:\s*"(count=([0-
9]+))".*"src"\s*:\s*"([^"]+)".*"dst"
\s*:\s*"([^"]+)"</regex>
```

```
<order>Zeek_note,Zeek_sub,nxdomain_count,srcip,dstip</order>
</decoder>

<decoder name="Zeek-json-network-beaconing">
<parent>json</parent>

<prematch>BeaconingDetector::Network_Beaconing</prematch>
<regex type="pcre2">"note":"([^"]+)".*"sub":"([^"]+)".*"src":"([^"]+)".*"dst":"([^"]+)".*"p":([0-9]+)</regex>

<order>Zeek_note,Zeek_sub,srcip,dstip,dstport</order>
</decoder>

<decoder name="Zeek-json-sumstats-dns-query-flood">
<parent>json</parent>

<prematch>SumStatsDetectors::SumStats_DNS_Query_Flood</prematch>
<regex type="pcre2">"note":"([^"]+)".*"sub":"(query=([^ ]+) count=([0-9]+))".*"src":"([^"]+)"</regex>

<order>Zeek_note,Zeek_sub,dnsquery,dns_count,srcip</order>
</decoder>

<decoder name="Zeek-json-dns-suspicious-query">
<parent>json</parent>

<prematch>DNSAnomalyDetector::Suspicious_DNS_Query</prematch>
<regex type="pcre2">"note":"([^"]+)".*"sub":"(query=([^ ]+) score=([0-9.]+))".*"src":"([^"]+)".*"dst":"([^"]+)"</regex>

<order>Zeek_note,Zeek_sub,dnsquery,dns_score,srcip,dstip</order>
</decoder>

<decoder name="Zeek-json-dns-new-domain">
<parent>json</parent>

<prematch>DNSAnomalyDetector::New_DNS_Domain</prematch>
<regex type="pcre2">"note":"([^"]+)".*"sub":"(query=([^"]+))".*"src":"([^"]+)".*"dst":"([^"]+)"</regex>

<order>Zeek_note,Zeek_sub,dnsquery,srcip,dstip</order>
</decoder>

<decoder name="Zeek-json-new-pair">
<parent>json</parent>

<prematch>NewPairDetector::New_Communication_Pair</prematch>
<regex type="pcre2">"note":"([^"]+)".*"src":"([^"]+)".*"dst":"([^"]+)"</regex>

<order>Zeek_note,srcip,dstip</order>
</decoder>

<decoder name="Zeek-json-sumstats-portscan">
<parent>json</parent>

<prematch>SumStatsDetectors::SumStats_Port_Scan</prematch>
<regex type="pcre2">"note":"([^"]+)".*"sub":"(target=([^ ]+) unique_ports=([0-9]+))".*"src":"([^"]+)"</regex>

<order>Zeek_note,Zeek_sub,dstip,unique_ports,srcip</order>
</decoder>

<decoder name="Zeek-json-active-hours">
<parent>json</parent>

<prematch>ActiveHoursDetector::Activity_Outside_Normal_Hours</prematch>
<regex type="pcre2">"note":"([^"]+)".*"sub":"([^"]+)".*"src":"([^"]+)".*"dst":"([^"]+)"</regex>

<order>Zeek_note,Zeek_sub,srcip,dstip</order>
</decoder>

<decoder name="Zeek-json-fields-port">
<parent>json</parent>
<prematch>"note":</prematch>
<regex type="pcre2">"note":"([^"]+)".*"src":"([^"]+)".*"dst":"([^"]+)".*"p":([0-9]+)</regex>

<order>Zeek_note,srcip,dstip,dstport</order>
</decoder>

<decoder name="Zeek-json-fields-noport">
<parent>json</parent>
<prematch>"note":</prematch>
<regex type="pcre2">"note":"([^"]+)".*"src":"([^"]+)".*"dst":"([^"]+)"</regex>

<order>Zeek_note,srcip,dstip</order>
</decoder>

<decoder name="Zeek-json-src-only">
<parent>json</parent>
<prematch>"note":</prematch>
<regex type="pcre2">"note":"([^"]+)".*"src":"([^"]+)"</regex>
<order>Zeek_note,srcip</order>
</decoder>

<decoder name="Zeek-json-sub">
<parent>json</parent>
<prematch>"sub":</prematch>
<regex type="pcre2">"sub":"([^"]+)"</regex>
<order>Zeek_sub</order>
</decoder>

Zeek_rules.xml
<group name="Zeek,">
<rule id="100900" level="0">

<decoded_as>json</decoded_as>
<description>Zeek alerts</description>
</rule>

<rule id="110003" level="9">
<if_sid>100900</if_sid>
<field name="Zeek_note">SSHPolicyDetector::SSH_From_Non_Admin_VLAN</field>
<description>Zeek policy violation: SSH from non-admin host $(srcip) -> $(dstip):$(dstport)</description>
<mitre>
<id>T1021.004</id>
</mitre>

<group>Zeek,ssh,policy,lateral_movement,</group>
</rule>
```

```
<rule id="110002" level="7">
<if_sid>100900</if_sid>
<field name="Zeek_note">NewPairDetector::New_Communication_Pair</field>
<description>Zeek baseline: new communication pair $(srcip) -> $(dstip)</description>
<mitre>
<id>T1021</id>
<id>T1046</id>
</mitre>

<group>Zeek,baseline,new_pair,</group>
</rule>

<rule id="110005" level="5">
<if_sid>100900</if_sid>
<field name="Zeek_note">DNSAnomalyDetector::New_DNS_Domain</field>
<description>Zeek baseline: new DNS domain $(dnsquery) observed from $(srcip) via DNS server $(dstip) </description>
<mitre>
<id>T1071.004</id>
</mitre>

<group>Zeek,dns,baseline,new_domain,behavioral,</group>
</rule>

<rule id="110006" level="7">
<if_sid>100900</if_sid>
<field name="Zeek_note">ActiveHoursDetector::Activity_Outside_Normal_Hours</field>
<description>Zeek baseline: activity outside normal hours $(srcip) -> $(dstip), $(Zeek_sub)</description>
<mitre>
<id>T1078</id>
</mitre>

<group>Zeek,baseline,active_hours,behavioral,</group>
</rule>

<rule id="110007" level="7">
<if_sid>100900</if_sid>
<field name="Zeek_note">NewServiceDetector::New_Service_For_Known_Pair</field>
<description>Zeek baseline: new service $(srcip) -> $(dstip):$(dstport)</description>
<mitre>
<id>T1046</id>
<id>T1021</id>
</mitre>

<group>Zeek,baseline,new_service,</group>
</rule>

<!-- DNS -->

<rule id="110001" level="8">
<if_sid>100900</if_sid>
<field name="Zeek_note">DNSAnomalyDetector::DNS_Beaconing</field>
<description>Zeek suspicious DNS query $(dnsquery) from $(srcip) via DNS server $(dstip), score=$(dns_score)</description>
<mitre>
<id>T1071.004</id>
</mitre>

<group>Zeek,dns,beaconing,c2,dga,behavioral,suspiciouts_query</group>
</rule>

<rule id="110004" level="8">
<if_sid>100900</if_sid>
<field name="Zeek_note">DNSAnomalyDetector::Suspicious_DNS_Query</field>
<description>Zeek suspicious DNS query $(dnsquery) from $(srcip) via DNS server $(dstip), score=$(dns_score)</description>
<mitre>
<id>T1568</id>
<id>T1071.004</id>
</mitre>

<group>Zeek,dns,dga,behavioral,suspicious_dns_query</group>
</rule>

<!-- BEACONING -->

<rule id="110000" level="9">
<if_sid>100900</if_sid>
<field name="Zeek_note">BeaconingDetector::Network_Beaconing</field>
<description>Zeek network beaconing detected $(srcip) -> $(dstip):$(dstport), $(Zeek_sub)</description>
<mitre>
<id>T1071</id>
</mitre>

<group>Zeek,beaconing,c2,</group>
</rule>

<!-- USER AGENT -->

<rule id="110009" level="10">
<if_sid>100900</if_sid>
<field name="Zeek_note">SuspiciousUserAgentDetector::Suspicious_HTTP_User_Agent</field>
<description>Zeek suspicious HTTP User-Agent from $(srcip)</description>
<mitre>
<id>T1595</id>
<id>T1190</id>
</mitre>

<group>Zeek,http,user_agent,recon,web_attack,</group>
</rule>

<rule id="110101" level="8">
<if_sid>100900</if_sid>
<field name="Zeek_note">DNSAnomalyDetector::DNS_NXDOMAIN_Spike</field>
<description>Zeek SumStats NXDOMAIN spike from $(srcip) via DNS Server $(dstip), count=$(nxdomain_count)</description>
<mitre>
<id>T1568</id>
</mitre>

<group>Zeek,sumstats,dns,dga,nxdomain</group>
</rule>

<!-- SUMSTATS -->

<rule id="110100" level="10">
<if_sid>100900</if_sid>
<field name="Zeek_note">SumStatsDetectors::SumStats_Port_Scan</field>
<description>Zeek SumStats port scan from $(srcip) -> $(dstip), unique_ports=$(unique_ports)
</description>
<mitre>
<id>T1046</id>
</mitre>
```

```
<group>Zeek,sumstats,scan,recon,
behavioral</group>
</rule>

<rule id="110102" level="7">
<if_sid>100900</if_sid>
<field
name="Zeek_note">SumStatsDete
ctors::SumStats_DNS_Query_Floo
d</field>
<description>Zeek SumStats
DNS query flood from $(srcip):
$(dnsquery),
count=$(dns_count)</description>
<mitre>
<id>T1071.004</id>
</mitre>

<group>Zeek,sumstats,dns,flood,b
ehavioral</group>
</rule>

<rule id="110103" level="9">
<if_sid>100900</if_sid>
<field
name="Zeek_note">SumStatsDete
ctors::SumStats_Large_Transfer</f
ield>
<description>Zeek SumStats
large transfer from $(srcip),
$(transfer_flow),
max_bytes=$(transfer_bytes)</des
cription>
<mitre>
<id>T1041</id>
<id>T1020</id>
</mitre>

<group>Zeek,sumstats,transfer,exf
iltration,behavioral</group>
</rule>

<!-- CORRELATION -->

<rule id="110203" level="12"
frequency="2" timeframe="900">
<if_sid>110003</if_sid>

<if_matched_sid>110100</if_mat
ched_sid>
<same_srcip />
<description>Zeek correlation:
port scan followed by SSH attempt
from same source
$(srcip)</description>
<mitre>
<id>T1046</id>
<id>T1021.004</id>
</mitre>

<group>Zeek,correlation,recon,ssh
,lateral_movement,behavioral,</gr
oup>
</rule>

<rule id="110202" level="14"
frequency="2" timeframe="600">
<if_sid>110003</if_sid>

<if_matched_sid>110006</if_mat
ched_sid>
<same_srcip />
<same_dstip />
<description>Zeek correlation:
SSH access from non-admin
VLAN during abnormal activity
period $(srcip) to
$(dstip)</description>
<mitre>
<id>T1021.004</id>
</mitre>

<group>Zeek,Zeek_correlation,ssh
,remote_access,policy_violation,</
group>
</rule>


<rule id="110201" level="13"
frequency="2" timeframe="900">
<if_sid>110100</if_sid>

<if_matched_sid>110002</if_mat
ched_sid>
<same_srcip />
<description>Zeek correlation:
new communication pair and port
scan activity from
$(srcip)</description>
<mitre>
<id>T1046</id>
<id>T1021</id>
</mitre>

<group>Zeek,correlation,recon,ne
w_pair,scan,behavioral,</group>
</rule>

<rule id="110224" level="12"
frequency="3" timeframe="900">

<if_matched_group>dns</if_matc
hed_group>
<same_srcip />
<description>Zeek correlation:
multiple DNS anomalies from
same source $(srcip)</description>
<mitre>
<id>T1071.004</id>
<id>T1568</id>
</mitre>

<group>Zeek,correlation,dns,dga,c
2,behavioral,</group>
</rule>

<rule id="110204" level="11"
frequency="3"
timeframe="1800">

<if_matched_group>baseline</if_
matched_group>
<same_srcip />
<description>Zeek correlation:
multiple baseline deviations from
same source $(srcip)</description>
<mitre>
<id>T1078</id>
<id>T1021</id>
<id>T1046</id>
</mitre>

<group>Zeek,correlation,baseline,
anomaly,behavioral,</group>
</rule>

<rule id="110200" level="14"
frequency="2"
timeframe="1800">
<if_sid>110103</if_sid>

<if_matched_sid>110000</if_mat
ched_sid>
<same_srcip />
<description>Zeek correlation:
C2-like beaconing and large data
transfer from
$(srcip)</description>
<mitre>
<id>T1071</id>
<id>T1041</id>
</mitre>

<group>Zeek,correlation,c2,exfiltr
ation,high_severity,behavioral,</gr
oup>
</rule>

</group>
```

Додаток Б

Router_config

```
Router#show run
Building configuration...

Current configuration : 5574 bytes
!
version 15.9
service timestamps debug datetime
msec
service timestamps log datetime
msec
no service password-encryption
!
hostname Router
!
boot-start-marker
boot-end-marker
!
!
!
no aaa new-model
!
!
!
mmi polling-interval 60
no mmi auto-configure
no mmi pvc
mmi snmp-timeout 180
!
!
!
!
!
no ip icmp rate-limit unreachable
!
!
!
ip dhcp excluded-address
192.168.20.1
!
ip dhcp pool IT_Departament
 network 192.168.20.0
255.255.255.0
 default-router 192.168.20.1
 domain-name itdep.local
!
!
!
no ip domain lookup
ip cef
no ipv6 cef
!
multilink bundle-name
authenticated
!
!
!
!
!
redundancy
!
no cdp log mismatch duplex
!
ip tcp synwait-time 5
!
!
!
!
!
!
!
!
!
!
!
!
!
interface GigabitEthernet0/0
 no ip address
 duplex auto
 speed auto
 media-type rj45
!
interface GigabitEthernet0/0.10
 encapsulation dot1Q 10
 ip address 192.168.10.1
255.255.255.240
 ip access-group WEB_SERV out
 ip nat inside
 ip virtual-reassembly in
!
interface GigabitEthernet0/0.20
 encapsulation dot1Q 20
 ip address 192.168.20.1
255.255.255.0
 ip access-group
ANTI_SPOOFING_USER_VLAN
in
 ip nat inside
 ip virtual-reassembly in
!
interface GigabitEthernet0/0.30
 encapsulation dot1Q 30
 ip address 192.168.30.1
255.255.255.248
 ip access-group Wazuh_Zeek out
!
interface GigabitEthernet0/0.40
 encapsulation dot1Q 40
 ip address 192.168.40.1
255.255.255.248
 ip access-group ADMIN_OUT in
 ip access-group ADMIN_IN out
!
interface GigabitEthernet0/1
 ip address 10.255.255.1
255.255.255.0
 ip nat outside
 ip virtual-reassembly in
 duplex auto
 speed auto
 media-type rj45
!
interface GigabitEthernet0/2
 no ip address
 shutdown
 duplex auto
 speed auto
 media-type rj45
!
interface GigabitEthernet0/3
 no ip address
 shutdown
 duplex auto
 speed auto
 media-type rj45
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
ip nat inside source list 20
interface GigabitEthernet0/1
overload
ip nat inside source static
192.168.10.10 10.255.255.10
!
ip access-list extended
ADMIN_IN
 permit tcp 192.168.30.0 0.0.0.7
192.168.40.0 0.0.0.7 established
 permit tcp 192.168.10.0 0.0.0.15
192.168.40.0 0.0.0.7 established
ip access-list extended
ADMIN_OUT
 permit udp 192.168.40.0 0.0.0.7
any eq domain
 permit tcp 192.168.40.0 0.0.0.7
any eq www
 permit tcp 192.168.40.0 0.0.0.7
any eq 443
 permit tcp 192.168.40.0 0.0.0.7
any eq 22
 permit icmp 192.168.40.0 0.0.0.7
any echo
 permit tcp 192.168.40.0 0.0.0.7
any eq 8080
 deny   ip any any
ip access-list extended
ANTI_SPOOFING_USER_VLAN
 deny   ip 192.168.40.0 0.0.0.7 any
 deny   ip 192.168.30.0 0.0.0.7 any
 permit ip any any
ip access-list extended
WEB_SERV
 permit tcp any 192.168.10.0
0.0.0.15 eq 443
```

```
 permit tcp any 192.168.10.0
0.0.0.15 eq 8080
 permit tcp any 192.168.10.0
0.0.0.15 eq www
 permit tcp host 192.168.40.2
192.168.10.0 0.0.0.15 eq 22
 permit tcp any 192.168.10.0
0.0.0.15 established
 permit tcp any 192.168.10.0
0.0.0.15 eq domain
 permit udp any 192.168.10.0
0.0.0.15 eq domain
 permit udp 192.168.10.0 0.0.0.15
any
 permit tcp any 192.168.10.0
0.0.0.15 range 1 100
ip access-list extended
Wazuh_Zeek
 permit tcp host 192.168.40.2
192.168.30.0 0.0.0.7 eq 22
 permit tcp host 192.168.40.2
192.168.30.0 0.0.0.7 eq 443
 permit tcp host 192.168.40.2
192.168.30.0 0.0.0.7 eq www
 permit tcp host 192.168.40.2
192.168.30.0 0.0.0.7 eq 5601
 permit tcp host 192.168.20.2
192.168.30.0 0.0.0.7 eq 22
!
ipv6 ioam timestamp
!
!
access-list 10 permit 192.168.40.2
access-list 20 permit 192.168.20.0
0.0.0.255
!
control-plane
!
banner exec ^C
**************************
**************************
*****************
* IOSv is strictly limited to use for
evaluation, demonstration and IOS
*
* education. IOSv is provided as-is
and is not supported by Cisco's
*
* Technical Advisory Center. Any
use or disclosure, in whole or in
part, *
* of the IOSv Software or
Documentation to any third party
for any       *
* purposes is expressly prohibited
except as otherwise authorized by
*
* Cisco in writing.
*
**************************
**************************
*****************
banner incoming
**************************
**************************
*****************
* IOSv is strictly limited to use for
evaluation, demonstration and IOS
*
* education. IOSv is provided as-is
and is not supported by Cisco's
*
* Technical Advisory Center. Any
use or disclosure, in whole or in
part, *
* of the IOSv Software or
Documentation to any third party
for any       *
* purposes is expressly prohibited
except as otherwise authorized by
*
* Cisco in writing.
*
**************************
**************************
*****************
banner login
**************************
**************************
*****************
* IOSv is strictly limited to use for
evaluation, demonstration and IOS
*
* education. IOSv is provided as-is
and is not supported by Cisco's
*
* Technical Advisory Center. Any
use or disclosure, in whole or in
part, *
* of the IOSv Software or
Documentation to any third party
for any       *
* purposes is expressly prohibited
except as otherwise authorized by
*
* Cisco in writing.
*
**************************
**************************
*****************^C
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 access-class 10 in
 login local
 transport input ssh
!
no scheduler allocate
!
end

Router#
```

Switch config

```
Switch#show run
Building configuration...

Current configuration : 2096 bytes
!
! Last configuration change at
09:15:25 UTC Mon Jun 15 2026
!
version 15.2
service timestamps debug datetime
msec
service timestamps log datetime
msec
no service password-encryption
service compress-config
!
hostname Switch
!
boot-start-marker
boot-end-marker
!
!
logging discriminator EXCESS
severity drops 6 msg-body drops
EXCESSCOLL
logging buffered 50000
logging console discriminator
EXCESS
!
no aaa new-model
!
no ip icmp rate-limit unreachable
!
!
!
no ip domain-lookup
no ip cef
no ipv6 cef
!
!
!
spanning-tree mode pvst
spanning-tree extend system-id
!
vlan internal allocation policy
ascending
!
ip tcp synwait-time 5
!
interface Ethernet0/0
switchport trunk allowed vlan
10,20,30,40
switchport trunk encapsulation
dot1q
switchport mode trunk
!
```

```
interface Ethernet0/1
 switchport access vlan 10
 switchport mode access
!
interface Ethernet0/2
 switchport access vlan 20
 switchport mode access
!
interface Ethernet0/3
 switchport access vlan 30
 switchport mode access
!
interface Ethernet1/0
 switchport access vlan 30
 switchport mode access
!
interface Ethernet1/1
 switchport access vlan 40
 switchport mode access
!
interface Ethernet1/2
!
interface Ethernet1/3
!
interface Ethernet2/0
!
interface Ethernet2/1
!
interface Ethernet2/2
!
interface Ethernet2/3
!
interface Ethernet3/0
!
interface Ethernet3/1
!
interface Ethernet3/2
!
interface Ethernet3/3
 switchport access vlan 30
 switchport mode access
!
interface Vlan1
 no ip address
 shutdown
!
interface Vlan10
 description VLAN-TO-WEB
 no ip address
!
interface Vlan20
 description IT Departament
 no ip address
!
interface Vlan30
 description SOC-VLAN
 no ip address
 shutdown
!
interface Vlan40
 description Admin
 no ip address
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
!
!
!
!
control-plane
!
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
!
monitor session 1 source vlan 10 ,
 20
monitor session 1 destination
 interface Et3/3
!
end

Switch#
```