

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

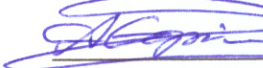
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи магістра
на тему: «Методи і засоби мережної цифрової
криміналістики для систем на базі Linux»

Виконала: студентка 2-го курсу групи КБм-24-1
спеціальності 125 «Кібербезпека та захист інформації»

 Олександра СІНІЦИНА

Керівник: завідувач кафедри КІ та КБ, к.т.н., доцент,

 Андрій ЄФІМЕНКО

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

 Володимир ВОРОТНІКОВ

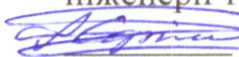
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека та захист інформації»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної

інженерії та кібербезпеки

 Андрій ЄФІМЕНКО

6 жовтня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу магістра

Студент: Сініцина Олександра Владиславівна

Тема роботи: **«Методи і засоби мережної цифрової криміналістики для систем на базі Linux»**

затверджена Наказом університету від 6 жовтня 2025 р. №566/с

Термін здачі студентом закінченої роботи 16 грудня 2025 р.

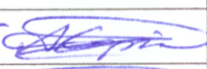
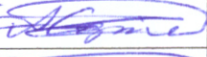
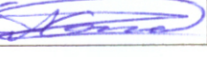
Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

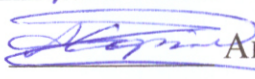
1. Архітектура та експериментальний стенд мережної цифрової криміналістики для систем на базі Linux
2. PCAP-файли мережного трафіку, журнали подій та лог-файли систем моніторингу й безпеки на базі Suricata

Обладнання та програмне забезпечення:

1. Гіпервізор Oracle VirtualBox 7.0 v.7.0.8 r156879 (Qt5.15.2)
2. Віртуальні машини: Wazuh v4.14.1 OVA, Ubuntu 24.04 2023, Kali Linux 2024.2, Kali Linux 2025.4, CSI Linux 24.10 (на базі Ubuntu 22.04.5 LTS).
3. Suricata v6.0.4
4. Wireshark v3.6.2
5. Network Miner v2.7.3

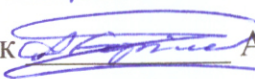
Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Андрій ЄФІМЕНКО	10.06.25 	10.06.2025 
2	Андрій ЄФІМЕНКО	10.06.25 	10.06.2025 
3	Андрій ЄФІМЕНКО	10.06.25 	10.06.2025 

Студент  Олександра СІНІЦИНА
Керівник  Андрій ЄФІМЕНКО

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	6.10.2025 – 7.10.2025	Виконано
2	Розробка плану роботи	8.10.2025 – 10.10.2025	Виконано
3	Збір та аналіз літератури	13.10.2025 – 17.10.2025	Виконано
4	Аналіз принципів та методів мережної криміналістики	20.10.2025 – 24.10.2025	Виконано
5	Аналіз інструментів та методів збору мережевих артефактів	27.10.2025 – 31.10.2025	Виконано
6	Розробка методики та архітектури експериментального стенду	03.11.2025 – 07.11.2025	Виконано
7	Реалізація та тестування системи моніторингу	10.11.2025 – 21.11.2025	Виконано
8	Аналіз отриманих мережних артефактів після проведення атак	24.11.2025 – 05.12.2025	Виконано
9	Оформлення пояснювальної записки	15.12.2025 – 19.12.2025	Виконано
10	Отримання відгуку та рецензії	22.12.2025	Виконано
11	Захист кваліфікаційної роботи	23.12.2025	Виконано

Студент  Олександра СІНІЦИНА
Керівник  Андрій ЄФІМЕНКО

АНОТАЦІЯ

Випускна кваліфікаційна робота спрямована на дослідження методів і засобів мережної цифрової криміналістики для систем на базі Linux, а також аналізу мережного трафіку та цифрових доказів з метою виявлення і дослідження мережних інцидентів.

Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаної літератури та додатків. Текстова частина викладена на 78 сторінках друкованого тексту. Пояснювальна записка має 14 сторінок додатків. Список використаних джерел містить 25 найменувань і займає 3 сторінки. В роботі наведено 104 рисунки. Загальний обсяг роботи – 104 сторінки.

Метою роботи є дослідження, аналіз та практичне впровадження методів і засобів мережної цифрової криміналістики для систем на базі Linux з метою підвищення ефективності виявлення мережних атак і розслідування інцидентів безпеки.

У роботі проаналізовано теоретичні основи мережної цифрової криміналістики та методи збору і аналізу мережних логів у Linux-середовищах. Розроблено експериментальний стенд локальної мережі, наведено сценарії налаштування пристроїв та інструментів для фіксації мережних подій. Проведено тестування функціонування стенду та оцінку ефективності методів збору і аналізу мережних артефактів.

Ключові слова: МЕРЕЖНА КРИМІНАЛІСТИКА, LINUX, ЦИФРОВІ ДОКАЗИ, АНАЛІЗ ТРАФІКУ, WIRESHARK, SURICATA, PCAP, ІНЦИДЕНТ БЕЗПЕКИ, ЛОГИ, ПОДІЇ.

ANNOTATION

The final qualification work is aimed at the study of methods and tools of network digital forensics for Linux-based systems, as well as the analysis of network traffic and digital evidence in order to detect and investigate network incidents.

The explanatory note consists of an introduction, three chapters, conclusions, a list of references, and appendices. The main text is presented on 78 pages of printed text. The explanatory note includes 14 pages of appendices. The list of references contains 25 items and occupies 3 pages. The work includes 104 pictures. The total volume of the work is 104 pages.

The purpose of the work is the study, analysis, and practical implementation of methods and tools of network digital forensics for Linux-based systems in order to improve the efficiency of detecting network attacks and investigating security incidents.

The work analyzes the theoretical foundations of network digital forensics and the methods of collecting and analyzing network logs in Linux environments. An experimental local network testbed has been developed, scenarios for configuring devices and tools for recording network events are provided. The functionality of the testbed and the effectiveness of methods for collecting and analyzing network artifacts have been tested.

Keywords: NETWORK FORENSICS, LINUX, DIGITAL EVIDENCE, TRAFFIC ANALYSIS, WIRESHARK, SURICATA, PCAP, SECURITY INCIDENT, LOGS, EVENTS.

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ МЕРЕЖНОЇ ЦИФРОВОЇ КРИМІНАЛІСТИКИ ДЛЯ СИСТЕМ НА БАЗІ LINUX	10
1.1. Аналіз принципів та методів цифрової криміналістики.....	10
1.2. Аналіз загроз, векторів та ознак атак на Linux-системи.....	15
1.3. Аналіз методів та інструментів збору мережних артефактів.....	18
1.4. Аналіз міжнародних стандартів цифрової криміналістики	21
РОЗДІЛ 2. РОЗРОБКА МЕТОДИКИ ТА ВИБІР ЗАСОБІВ МЕРЕЖНОЇ ЦИФРОВОЇ КРИМІНАЛІСТИКИ ДЛЯ LINUX-СИСТЕМ	24
2.1 Розробка методики мережної цифрової криміналістики для Linux- систем.....	24
2.2 Аналіз та вибір інструментальних засобів для розслідування мережних інцидентів у Linux	27
2.3 Проектування стенду для проведення досліджень та розслідування мережних інцидентів.....	34
РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ТА РОЗСЛІДУВАННЯ МЕРЕЖНИХ ІНЦИДЕНТІВ	38
3.1. Базове налагодження стенду для проведення розслідування мережних інцидентів.....	38
3.2. Налаштування підсистеми моніторингу мережного трафіку	44
3.3. Створення та налаштування спеціалізованих дашбордів для підсистеми моніторингу мережних подій.....	53
3.4. Проведення тестових атак та аналіз фіксації подій	61
3.5. Рекомендації щодо вдосконалення процесів проведення мережної криміналістики та експериментального стенду.	82
ВИСНОВКИ.....	85
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	86
ДОДАТКИ.....	89

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІБ – Інформаційна безпека

ОС – Операційна система

PCAP – Packet Capture

NIDS – Network-Based Intrusion Detection System

HIDS – Host-Based Intrusion Detection System

IDS – Intrusion Detection System

IPS – Intrusion Prevention System

DoS – Denial of Service

TCP – Transmission Control Protocol

HTTP – HyperText Transfer Protocol

ВСТУП

Актуальність теми. Мережна цифрова криміналістика стрімко перетворилася з вузькопрофільного методу розслідування на критично важливу складову кібербезпеки. Зі зростанням складності та масштабності кіберзагроз організації потребують повної видимості мережних комунікацій для виявлення атак, оперативного реагування на інциденти та формування доказової бази. Традиційні засоби безпеки часто не здатні виявляти приховані або нетипові загрози, тоді як мережна криміналістика забезпечує глибоке розуміння механізмів атак та їхніх артефактів. Її застосування дозволяє закривати критичні прогалини в обороні, забезпечувати надійнішу діяльність та підвищувати рівень захисту систем на базі Linux, які широко використовуються у корпоративному середовищі.

Метою кваліфікаційної роботи є дослідження, аналіз та практичне впровадження методів і засобів мережної цифрової криміналістики для систем на базі Linux з метою підвищення ефективності виявлення мережних атак і розслідування інцидентів безпеки.

Визначена мета обумовлює наступні завдання:

- проаналізувати існуючі методи та підходи мережної цифрової криміналістики;
- дослідити можливості Linux-засобів для виявлення та аналізу мережних інцидентів;
- вивчити роботу інструментів для збору й обробки мережних артефактів (PCAP-файлів, логів, телеметрії);
- розробити та налаштувати експериментальний стенд для фіксації, обробки та аналізу мережних подій;
- провести тестування методів в умовах моделювання мережних атак;
- оцінити ефективність використаних засобів та сформулювати рекомендації щодо їх практичного застосування.

Об’єктом дослідження є процеси збору, фіксації та аналізу мережного трафіку і мережних подій у Linux-середовищах, що виникають у результаті роботи систем або внаслідок мережних атак та інцидентів інформаційної безпеки.

Предметом дослідження є методи, засоби та технології мережної цифрової криміналістики для систем на базі Linux, зокрема методи аналізу PCAP-файлів, кореляції мережних подій та фіксації мережних атак із використанням інструментів моніторингу, IDS/IPS і систем управління подіями безпеки.

Результатом роботи є стенд мережної цифрової криміналістики для систем на базі Linux, що включає набір скриптів та інструментів для виявлення, фіксації та аналізу мережних артефактів. Проект реалізує можливість моделювати атаки, збирати PCAP-дані, здійснювати їх аналіз і формувати криміналістичні висновки щодо мережних інцидентів.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ МЕРЕЖНОЇ ЦИФРОВОЇ КРИМІНАЛІСТИКИ ДЛЯ СИСТЕМ НА БАЗІ LINUX

1.1. Аналіз принципів та методів цифрової криміналістики

1.1.1 Сутність, принципи та методи мережної цифрової криміналістики

Цифрова криміналістика – це галузь криміналістики, яка зосереджена на процесах збереження, ідентифікації, вилучення, відновленні, документування та аналізі комп’ютерних доказів, знайдених у цифрових носіях, з метою розслідування кіберзлочинів, аналізу атак та відновлення подій у цифровому просторі.^{[1] [23]}

Проте зі зростанням складності мережних інфраструктур та збільшенням кількості кібератак, що активно використовують мережні канали комунікації, виникла потреба у виокремленні окремої, більш спеціалізованої підгалузі.

Мережна криміналістика – це галузь, яка досліджує мережний трафік та дані, отримані під час передачі в середовищі. Вона передбачає моніторинг і перевірку мережного трафіку, його даних та журналів, які можуть бути використані для розслідування кіберзлочинів і порушень безпеки даних, а також охоплює реагування на інциденти, пошук загроз, перевірку відповідності, аналіз шкідливого програмного забезпечення та виявлення внутрішніх загроз.^{[1] [4]}

Мережна криміналістика має на меті виявлення, збереження та аналіз цифрових доказів, які можуть бути використані в судових розглядах. Вона дає змогу фахівцям із кібербезпеки виявляти активні атаки, досліджувати їхні методи та збирати необхідні докази. Аналізуючи мережні події, правоохоронні органи можуть відтворити хронологію інциденту й картину комунікацій між учасниками, а також виявити ознаки людської взаємодії, підроблення файлів чи використання ключових слів, що свідчать про злочинну діяльність.

Усі процеси збору та аналізу мережних доказів регулюються низкою фундаментальних принципів, які забезпечують юридичне значення та наукову достовірність результатів:

1) Цілісність доказів – усі мережні артефакти мають бути зібрані без змін, із фіксацією метаданих.

2) Відтворюваність – результати аналізу мають бути повторюваними незалежно від дослідника.

3) Безперервність ланцюга зберігання доказів (Chain of Custody) – кожен етап збору та передачі даних повинен бути задокументований.^[5]

4) Достовірність джерел – використовуються тільки перевірені джерела трафіку: мережні сенсори, IDS/IPS, firewall-логи, PCAP-записи.

5) Мінімальне втручання – дослідження не повинно впливати на роботу мережі або змінювати первинні дані.

Для аналізу подій у мережній криміналістиці застосовуються різні методи, які визначають, коли і як відбувається збір та дослідження мережного трафіку. Основними з них є два: криміналістика у реальному часі (live network forensics) та криміналістика після інциденту (post-incident forensics).

Криміналістика у реальному часі полягає у моніторингу та аналізі мережного трафіку в реальному часі, що дозволяє фахівцям спостерігати події під час їх виникнення і миттєво реагувати на підозрілу активність. Перевага цього методу є можливість запобігати збиткам і мінімізувати наслідки кіберінцидентів.

Криміналістика після інциденту використовується після того, як стався інцидент безпеки чи підозріла активність. Експерти збирають і аналізують журнали, дампи пакетів, записи з мережних пристроїв для відтворення подій. Це допомагає визначити причину інциденту, знайти вразливості та зібрати докази для судових справ чи підвищення рівня безпеки.

1.1.2 Основні етапи розслідування в мережній криміналістиці

Мережна криміналістика дотримується організованих методів, які гарантують повні розслідування та юридично прийнятні результати. Процес

зазвичай включає сім основних етапів, які проводять дослідники від першого виявлення інцидентів до фінальних звітів (рис. 1.1): ^{[1] [4] [23]}

Етап 1. Виявлення та ідентифікація.

Етап 2. Збір.

Етап 3. Збереження.

Етап 4. Дослідження.

Етап 5. Аналіз та кореляція.

Етап 6. Реагування.

Етап 7. Документування та звітність.

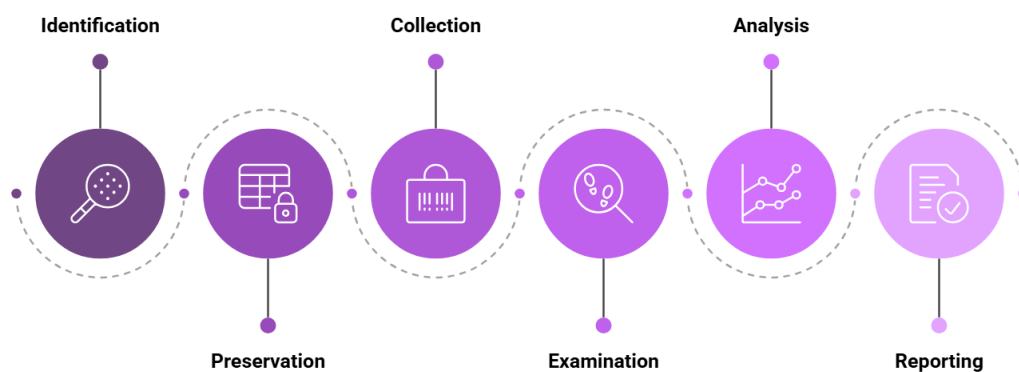


Рисунок 1.1 – Процес мережного криміналістичного розслідування

Етап «Виявлення та ідентифікація» полягає у визначенні обсягу розслідування та необхідних джерел даних: встановлюються межі розслідування і вирішується, які області мережі потребують дослідження. Для виявлення аномалій трафіку або ознак компрометації використовують системи моніторингу (IDS/IPS, SIEM, NetFlow-аналіз). Команди безпеки визначають, які докази потрібно зібрати, і обирають відповідні інструменти мережної криміналістики з урахуванням типів інцидентів та доступних ресурсів. ^[23]

Етап «Збір» передбачає, що відбувається безпосередній збір інформації з елементів мережної інфраструктури – маршрутизаторів, комутаторів, брандмауерів, систем моніторингу та інших джерел телеметрії. Отримання даних здійснюється з маршрутизаторів, файрволів і серверів: здійснюється фіксація

трафіку у форматах PCAP, збираються логи мережних пристроїв, DNS-записи та файли конфігурації firewall.^[23]

Етап «Збереження» передбачає забезпечення цілісності зібраних даних (наприклад, хешуванням), створення копій даних та збереження ланцюга доказів – це захищає всі важливі мережні дані від підробки або втрати. Дослідники створюють достовірні копії захоплених пакетів, записів потоків і журналів пристроїв, дотримуючись суворих ланцюгів зберігання, які гарантують, що докази можуть бути використані.^[23]

Етап «Дослідження» полягає в детальному вивченні зібраної інформації з використанням спеціалізованих інструментів аналізу PCAP та програм моніторингу мережі: дослідники проводять кореляцію мережних логів і пакетних дамів, відновлюють сесії й хронологію подій, шукають ознаки компрометації, незвичні шаблони трафіку та підозрілі комунікації, які можуть свідчити про шкідливу активність.

Етап «Аналіз та кореляція» полягає у пошуку ознак злову, передачі файлів або компрометації та відновленні ходу подій і методів атаки: застосовуються спеціалізовані інструменти для ідентифікації протоколів, відновлення сесій і виявлення підозрілих з'єднань. Проводиться аналіз метаданих, пошук шкідливих пакетів, сигнатур та інших індикаторів компрометації, а також перевірка наявності передачі даних на зовнішні ресурси. Результати дослідження інтерпретуються з метою встановлення методів атаки, часових рамок та оцінки впливу на бізнес; мережна криміналістика поєднує окремі фрагменти доказів у цілісну хронологію, що пояснює, як виник і розвивався інцидент. Результати аналізу поєднуються з даними з інших джерел (журнали ОС, IDS-логи, поштові записи, DNS-запити) для відновлення повної картини інциденту.^[23]

Етап «Реагування» передбачає використання результатів проведеного розслідування для нейтралізації виявлених загроз, усунення вразливостей і впровадження заходів, що запобігають повторенню подібних інцидентів у майбутньому. На цьому етапі результати аналізу мережної криміналістики

трансформуються у практичні дії, спрямовані на підвищення рівня кіберзахисту організації.

Етап «Документування та звітність» полягає в оформленні результатів розслідування у вигляді детального звіту: формується хронологія подій, збираються та додаються відповідні докази, фіксуються хеші файлів і наводяться висновки щодо походження та наслідків атаки. У разі судового розслідування такий звіт може слугувати цифровим доказом за умови дотримання процедур збереження цілісності даних і ланцюга зберігання доказів. [23]

1.1.3 Методи аналізу у мережній криміналістиці

З огляду на різноманіття форматів мережних даних, протоколів і сценаріїв атак, мережна криміналістика використовує сукупність взаємодоповнюючих методів аналізу. Кожен із них орієнтований на дослідження окремих аспектів мережної активності – статистичних характеристик трафіку, структур протоколів, поведінкових моделей або часових залежностей подій.

Основні методи мережної криміналістики, що застосовуються для аналізу мережного трафіку, включають:

- аналіз потоків трафіку (Traffic Flow Analysis);
- аналіз протоколів (Protocol Analysis);
- поведінковий аналіз (Behavioral Analysis);
- реконструкція часової шкали (Timeline Reconstruction);
- кореляційний аналіз (Correlation Analysis);
- розпізнавання шаблонів (Pattern Recognition). [23]

Розглянемо кожен із наведених методів детальніше.

Аналіз потоків трафіку (Traffic Flow Analysis) полягає у дослідженні шаблонів комунікацій між мережними вузлами з метою виявлення нетипової поведінки, що може свідчити про компрометацію систем. У межах цього підходу аналізуються частота з'єднань, обсяги переданих даних, часові характеристики

трафіку та параметри пунктів призначення з метою виявлення відхилень від нормальної активності.

Аналіз протоколів (Protocol Analysis) передбачає глибоку інспекцію мережних протоколів для розуміння механізмів взаємодії застосунків і виявлення зловживань або маніпуляцій протоколами. Під час аналізу досліджуються заголовки протоколів, вміст корисного навантаження та послідовності обміну повідомленнями з метою ідентифікації атак.

Поведінковий аналіз (Behavioral Analysis) забезпечує впорядкування мережних подій у хронологічній послідовності з метою відтворення розвитку атаки та визначення використаних зловмисником методів. Це дозволяє встановити послідовність дій атакувальника та шляхи його переміщення мережею.

Реконструкція часової шкали (Timeline Reconstruction) розміщує мережні події в хронологічному порядку, щоб зрозуміти розвиток атаки і використані методи. Цей метод допомагає дослідникам зрозуміти, як зловмисники переміщувалися мережею і які дії вони виконували.

Кореляційний аналіз (Correlation Analysis) спрямований на поєднання даних мережної криміналістики з інформацією, отриманою з інших засобів захисту, зокрема SIEM-систем, інструментів моніторингу кінцевих точок і джерел розвідувальних даних про загрози. Такий підхід забезпечує комплексне бачення інциденту безпеки.

Розпізнавання шаблонів (Pattern Recognition) використовує статистичний аналіз і методи машинного навчання, щоб знайти повторювану поведінку, яка свідчить про певні типи атак або методи зловмисників.

1.2. Аналіз загроз, векторів та ознак атак на Linux-системи

Linux-середовище є основою більшості сучасних серверних інфраструктур, у зв'язку з чим атаки на такі системи охоплюють як компрометацію хостових компонентів операційної системи, так і маніпулювання

мережними протоколами та сервісами. Виявлення та аналіз таких загроз здійснюється на основі дослідження системних і мережних журналів, аналізу поведінкових аномалій, а також детального вивчення мережного трафіку у форматі PCAP, що дозволяє реконструювати послідовність дій злоумисника.^[15]

Серед загроз, що спрямовані безпосередньо на ядро операційної системи, системні компоненти та серверні служби, можна виділити наступні:

- експлуатація вразливостей ядра та системних компонентів;
- компрометація серверних служб та демонів;
- privilege Escalation та приховані інструменти;^[2]
- помилки конфігурації (Misconfiguration).

Експлуатація вразливостей ядра та системних компонентів заключається у тому, що Linux-ядро та системні утиліти містять велику кількість потенційних точок входу. Критичні вразливості (Buffer Overflow, Privilege Escalation, Use-After-Free) дозволяють отримати виконання коду з правами root або здійснити втечу з контейнера. У мережній криміналістиці такі атаки проявляються як аномальні звернення до служб, зміни у структурі трафіку або нетипова активність у PCAP-файлах. Типовими прикладами є Dirty COW (CVE-2016-5195), віддалене виконання коду в Exim, а також вразливості утиліти sudo.^[25]

Компрометація серверних служб та демонів, таких як SSH, Apache/Nginx, FTP/SFTP, MySQL/PostgreSQL та Docker daemon, можуть бути скомпрометовані як через наявні вразливості, так і внаслідок некоректних конфігурацій.

Типові сценарії атак включають:

- brute-force на SSH;^[16]
- експлуатацію веб-вразливостей (RCE, SQL Injection, Command Injection);
- отримання доступу через ненадійні паролі або публічні SSH-ключі.^[2]

У мережних артефактах компрометація фіксується у вигляді повторюваних спроб автентифікації, шкідливих HTTP-payload та запитів з однаковими параметрами з різних IP-адрес.

Після отримання початкового доступу зломисники прагнуть підвищити привілеї через помилки в налаштуваннях `sudo`, вразливі `setuid`-файли або модулі ядра. На подальших етапах атаки можуть встановлюватися `rootkits` і `backdoors`, що приховують процеси, файли та мережні з'єднання.

Криміналістичними ознаками таких дій є:

- відкриті нестандартні порти;
- `outbound`-з'єднання до незвичних хостів;
- метадані процесів без відповідних виконуваних файлів.

Значна частина компрометацій Linux-систем зумовлена помилками конфігурації. Під час цифрової криміналістики такі проблеми виявляються шляхом порівняння фактичних конфігурацій із базовим профілем безпеки системи. Найпоширенішими помилками є:

- слабкі або типові паролі;
- зайві відкриті порти;
- надмірні дозволи на файлову систему;
- зберігання службових SSH-ключів у відкритих репозиторіях.

Також розглянемо атаки, орієнтовані на мережеву інфраструктуру та протоколи взаємодії в Linux-середовищах, а саме:

- сканування та збір інформації (`Reconnaissance`);
- атаки на мережні протоколи;
- атаки на веб-протоколи та `HTTP/HTTPS`;
- атаки на SSH-інфраструктуру.

Початковий етап більшості мережних атак пов'язаний із розвідкою цілі, зокрема скануванням портів, визначенням доступних сервісів та аналізом банерної інформації. Застосування таких інструментів, як `Nmap` або `Masscan` формує в мережному трафіку специфічний слід, а саме : спостерігається велика кількість `SYN`-пакетів, послідовні запити до широкого діапазону портів, а також численні короткотривалі з'єднання, які не переходять у повноцінні сесії.

Також Linux-мережі часто стають ціллю атак, спрямованих на експлуатацію базових мережних протоколів. До них належать:

- ARP spoofing/poisoning: перехоплення та перенаправлення трафіку;
- DNS spoofing: зміна відповіді резолвера;
- DHCP-атаки (starvation, Rogue DHCP): створення перешкод у мережі або перенаправлення трафіку через атакувальника;
- MITM-атаки з модифікацією переданих даних.

Одним з поширених векторів атак для Linux-систем компрометація веб-служб. Популярні сценарії включають brute-force веб-форм, експлуатацію RCE-вразливостей, запуск web-shell та Command Injection. У мережному трафіку це проявляється подовженими HTTP-запитами, нетиповими User-Agent, шкідливими параметрами у URL й підозрілим вмістом POST-запитів.

SSH є основним каналом адміністрування Linux-систем, що робить його поширеною ціллю атак. Найпоширенішими сценаріями є підбір облікових даних, атаки за словником, компрометація SSH-ключів із недостатнім рівнем захисту, а також використання SSH-з'єднань для прихованого тунелювання шкідливого трафіку. Журнали *auth.log* і мережні лог-файли відображають збільшену кількість невдалих спроб входу, спроби входу з підозрілих географічних зон, а також створення незвичних SSH-тунелів.

1.3. Аналіз методів та інструментів збору мережних артефактів

Системи виявлення вторгнення (IDS) призначені для контролю та аналізу подій у комп'ютерних системах і мережах, щоб своєчасно виявляти спроби несанкціонованого доступу, шкідливу активність або порушення безпеки. Вони виконують функції моніторингу, оцінки загроз і підтримки реагування на інциденти, допомагаючи захищати інформаційні ресурси організацій. [3]

Системи IDS можна класифікувати за різними ознаками, зокрема за місцем розташування (областю моніторингу) та за методом виявлення загроз. Основними типами за місцем розташування (областю моніторингу):

- HIDS (Host-based Intrusion Detection System)
- NIDS (Network-based Intrusion Detection System)
- Hybrid IDS

HIDS – це хостово-орієнтована система виявлення атак, яка спостерігає та аналізує події, що відбуваються тільки всередині системи. Вони встановлюються безпосередньо на кінцевих пристроях (хостах), таких як сервери, робочі станції та моніторять внутрішню активність системи: системні журнали, зміни файлів та конфігурацій, активність користувачів.^[3]

Приклади інструментів: OSSEC, Wazuh, AIDE, Tripwire, Falco.

NIDS – це мережево-орієнтована система виявлення атак, яка спостерігає та аналізує події, що відбуваються поза системою, у мережному трафіку. Вони встановлюються на стратегічних точках мережі (наприклад, після фаєрвола) та моніторять зовнішню активність мережі.^[3]

Приклади інструментів: Snort, Suricata, Zeek, Ntop.

Hybrid IDS – це комплексна система виявлення атак, яка спостерігає та аналізує події, що відбуваються як всередині окремих систем, так і поза ними, у мережному трафіку. Вона поєднує підходи HIDS та NIDS, використовуючи інформацію як з хостів, так і з мережі для більш повного та ефективного виявлення загроз.^[3]

Приклади інструментів: Системи, що інтегрують функціонал NIDS і HIDS (наприклад, Wazuh, інтегрований із мережевим моніторингом, або комерційні XDR-рішення, які використовують обидва підходи).

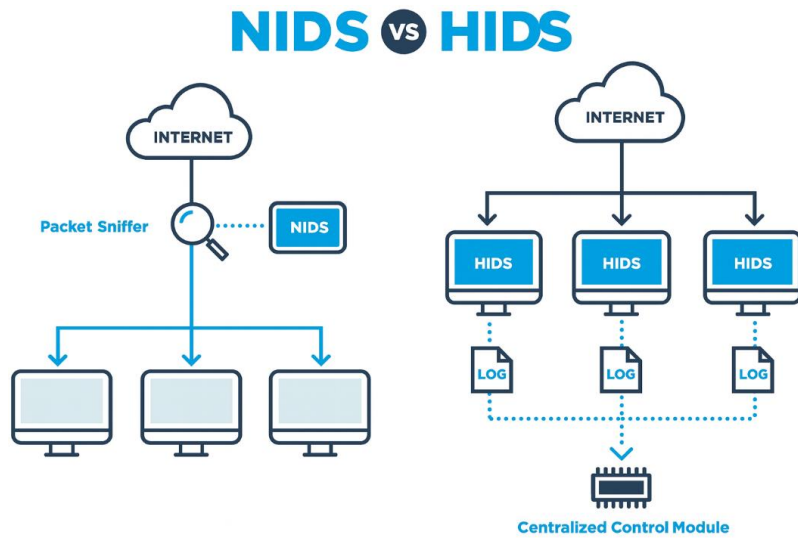


Рисунок 1.2 – Порівняльна схема роботи систем NIDS та HIDS

Розглянемо основні типи IDS, які класифікуються за методом виявлення загроз, що визначає спосіб аналізу подій і формування рішення про можливе вторгнення. [3] [6]

- На основі сигнатур (Signature-based IDS)
- На основі аномалій (Anomaly-based IDS)
- На основі правил (Rule-based IDS)

IDS на основі сигнатур – це система виявлення атак, яка спостерігає та аналізує події, порівнюючи їх із відомими шаблонами атак. Вона порівнює мережний трафік або системну активність із базою даних відомих "сигнатур" (шаблонів) атак. Ця система ефективна для виявлення відомих загроз, але не може виявити нові або модифіковані атаки.

IDS на основі аномалій – це система виявлення атак, яка спостерігає та аналізує події, шукаючи відхилення від норми. Вона створює "базовий" профіль нормальної поведінки системи чи мережі (наприклад, за допомогою машинного навчання або статистичного аналізу). Будь-яке значне відхилення від цього нормального профілю розцінюється як потенційне вторгнення (аномалія). Ця система може виявляти нові (zero-day) загрози, але може генерувати хибні спрацювання.

IDS на основі правил (Rule-based IDS) – це система виявлення вторгнень, які аналізує мережеву або системну активність відповідно до наперед заданих логічних правил типу «якщо умова – тоді дія». Такі системи використовують базу знань, що складається з набору фактів і правил логічного висновку, на основі яких приймається рішення про наявність підозрілої або шкідливої активності. Ефективність системи залежить від якості та повноти сформованих правил, а також від глибини розуміння адміністратором особливостей мережної інфраструктури.

1.4. Аналіз міжнародних стандартів цифрової криміналістики

Цифрова криміналістика як складова інформаційної безпеки ґрунтується на суворо регламентованих та стандартизованих підходах до ідентифікації, збору, збереження й аналізу цифрових доказів. Використання міжнародних стандартів забезпечує коректність, повторюваність і цілісність криміналістичних процедур, а також достовірність і юридичну допустимість отриманих цифрових доказів у судових процесах. Такі стандарти уніфікують процеси розслідування кіберінцидентів та визначають чіткі правила для всіх етапів криміналістичного дослідження в межах різних організацій і країн.

На сьогодні у сфері цифрової криміналістики застосовується низка міжнародних стандартів і рекомендаційних документів, зокрема:

- ISO/IEC 27037 – ідентифікація, збір, збереження та первинний аналіз цифрових доказів;
- ISO/IEC 27041 – забезпечення якості методів аналізу цифрових доказів;
- ISO/IEC 27042 – настанови щодо аналізу цифрових доказів;^[21]
- ISO/IEC 27043 – розслідування інцидентів інформаційної безпеки;
- ISO/IEC 17025 – вимоги до компетентності криміналістичних лабораторій;
- NIST SP 800-86 – інтеграція цифрової криміналістики у процес реагування на інциденти;^[4]

- RFC 3227 – керівництво щодо збору цифрових доказів;
- рекомендації міжнародних організацій SWGDE, INTERPOL та ENFSI.

Розглянемо детальніше стандарти, що безпосередньо застосовуються у сфері мережної цифрової криміналістики.

ISO/IEC 27037 є міжнародним стандартом, що містить рекомендації щодо методів ідентифікації, збору, збереження та первинного аналізу цифрових доказ. У контексті мережної криміналістики стандарт регламентує процедури, як слід правильно працювати з цифровими носіями, PCAP-файлами, журналами мережних подій і дампами трафіку та гарантувати, що зібрані докази залишаються автентичними та допустимими в судових розглядах.

NIST Special Publication 800-86 є спеціальним виданням Національного інституту стандартів і технологій США, яке надає детальні рекомендації щодо інтеграції цифрової криміналістики у процес реагування на інциденти. Основні положення цього стандарту охоплюють методологію збору й аналізу цифрових доказів, впровадження криміналістичних процесів та використання спеціалізованих інструментів цифрової криміналістики. Документ визначає послідовність дій із виявлення інцидентів, збору, збереження та аналізу цифрових доказів, а також регламентує застосування криміналістичних інструментів під час дослідження мережних атак.^[4]

RFC 3227 (Request for Comments 3227) є документом, розробленим Інженерною радою Інтернету (IETF), що містить практичні рекомендації щодо збору цифрових доказів у випадку кіберінцидентів. Він визначає перелік першочергових дій, необхідних для мінімізації втрати доказів, пріоритизації мережних даних та збереження їхньої цілісності й автентичності під час реагування на комп'ютерні інциденти.

ISO/IEC 27042 визначає процедури та критерії, що застосовуються безпосередньо до аналізу зібраних цифрових даних. Стандарт встановлює технічні та документальні вимоги до процесу аналізу, які забезпечують

відтворюваність результатів і коректність інтерпретації мережних артефактів, отриманих у ході розслідування кіберінцидентів.

Висновки до розділу 1:

У розділі було розглянуто теоретичні засади мережної цифрової криміналістики, її сутність, принципи та методи, що забезпечують достовірність і юридичну значущість цифрових доказів. Визначено основні етапи мережного криміналістичного розслідування, які дозволяють системно виявляти, зберігати та аналізувати мережні артефакти й відтворювати хронологію кіберінцидентів.

Проаналізовано ключові методи аналізу мережного трафіку, зокрема аналіз потоків і протоколів, поведінковий та кореляційний підходи, а також реконструкцію часових шкал подій. Окрему увагу приділено загрозам і векторам атак на Linux-системи, включно з атаками на мережні протоколи та серверні служби. Розглянуто засоби збору мережних даних і роль IDS-систем, а також міжнародні стандарти, що регламентують процеси цифрової криміналістики.

РОЗДІЛ 2. РОЗРОБКА МЕТОДИКИ ТА ВИБІР ЗАСОБІВ МЕРЕЖНОЇ ЦИФРОВОЇ КРИМІНАЛІСТИКИ ДЛЯ LINUX-СИСТЕМ

2.1 Розробка методики мережної цифрової криміналістики для Linux-систем

Методика мережної цифрової криміналістики для Linux-середовищ орієнтована на дослідження мережних взаємодій, трафіку, конфігурацій сервісів та мережних артефактів, що дозволяє встановити факт та механізм компрометації, визначити вектори атаки й виявити канали витоку даних. Вона передбачає стандартизовані процедури, інструменти і формат документування, які забезпечують юридичну коректність, збереження контексту й можливість кореляції артефактів для точного відновлення хронології подій та ідентифікації векторів компрометації.

Вона охоплює дві взаємопов'язані фази – збір мережних доказів та аналіз зібраних даних, а також стандартизовані процедури, інструменти та формат документування результатів. Розглянемо послідовно ключові етапи окреслених фаз.

Фаза 1. Збір мережних доказів забезпечує збереження контексту мережних подій та відтворюваність аналізу. Основна мета – отримати повний набір артефактів, пов'язаних із мережними взаємодіями, із фіксацією метаданих, хешів і часу зняття зразків. Фазу можна розділити на наступні етапи: [4] [5] [21]

1. Інвентаризація мережної конфігурації системи

- Збір відомостей про мережні інтерфейси.
- Перелік активних мережних сервісів та демони, що прослуховують порти.
- Експорт журналів мережних подій.
- Фіксація IP-адрес, портів, маршрутів, DNS-конфігурацій.
- Збір конфігурацій мережних сервісів.

2. Сканування вразливостей мережних сервісів

- Ідентифікація CVE, що стосуються мережних демони (OpenSSH, Apache, Samba, Kernel networking).

- Проведення локальних і зовнішніх сканів (Nmap, OpenVAS, Nikto).
- Виявлення неправильно налаштованих портів, служб і протоколів.

3. Мережний збір трафіку та метаданих

- Захоплення трафіку у форматі PCAP.
- Збір потоків у разі наявності відповідних агентів.
- Фіксація поточних з'єднань, відкритих портів, активних сокетів.
- Створення "знімка" мережної активності під час інциденту.

4. Перевірка цілісності і документування мережних артефактів

- Хешування PCAP-зразків, логів та конфігурацій.
- Фіксація інструментів, команд і параметрів.

Фаза 2. Аналіз мережних доказів полягає у кореляції артефактів з поведінковими аномаліями. Досліджуються зміни в правах доступу та конфігураціях, а також виявляються підозрілі файли й модулі. Мережний трафік детально аналізується з метою встановлення C2-зв'язків або витоків даних. Мета цієї фази – відновити хронологію подій, визначити механізми компрометації, шляхи поширення в мережі, ідентифікувати шкідливі артефакти та сформулювати практичні рекомендації щодо реагування й відновлення. Фазу доцільно поділити на такі етапи:

1. Кореляція логів мережних сервісів і побудова таймлайну

- Агрегація подій у хронологічний таймлайн
- Виявлення аномалій поведінки: незвичні входи, помилки, зміни конфігурацій.

- Виявлення змін у мережних конфігураціях, запуску тунелів чи проксі

2. Аналіз вразливостей мережної інфраструктури

- Перевірка встановлених версій мережних сервісів на наявність CVE.
- Оцінка можливості експлуатації віддалено чи локально.

- Аналіз мережного стеку Linux на предмет відомих вразливостей ядра.

3. Аналіз мережних прав і політик доступу

- Аудит iptables/nftables, SELinux/AppArmor профілів.
- Виявлення неправомірно відкритих портів або дозволених маршрутів.
- Виявлення ознак тунелювання трафіку або створення бекдор-портів.

4. Виявлення шкідливих мережних артефактів

- Пошук нетипових бінарних файлів, що створюють прослуховувані порти.

- Аналіз запусків reverse shell, тунелів.
- Перевірка наявності індикаторів мережних rootkitів.

5. Аналіз PCAP та поточкових даних ^[24]

- Виявлення C2-зв'язків, beaconing-трафіку, підозрілих DNS-запитів.
- Аналіз протоколів HTTP/HTTPS, SSH, SMB, DNS, TLS.
- Виявлення спроб ексфільтрації даних.
- Побудова карти взаємодій між хостами.

6. Перевірка цілісності мережного середовища й оцінка впливу

- Порівняння конфігурацій мережних сервісів із контрольними зразками.
- Ідентифікація змінених параметрів маршрутизації, firewall-правил.
- Визначення масштабу компрометації та каналів поширення.

Результати мережного розслідування оформлюються у структурованому звіті, який містить опис інциденту та його мету, перелік зібраних мережних артефактів, використані інструменти та методики аналізу, кореляційні таблиці й таймлайн мережних подій, а також висновки щодо виявлених аномалій, встановлених векторів атаки та оцінку потенційної шкоди з рекомендаціями щодо відновлення й підвищення мережної безпеки. До звіту додаються допоміжні матеріали: PCAP-файли, контрольні суми, конфігураційні дані та скрипти збору, що забезпечує повну відтворюваність аналізу та можливість незалежної перевірки отриманих результатів.

2.2 Аналіз та вибір інструментальних засобів для розслідування мережних інцидентів у Linux

Linux-системи мають широкий вибір утиліт і фреймворків, призначених для збору, збереження та аналізу цифрових артефактів, пов'язаних з мережними інцидентами, без порушення їхньої цілісності. Вони охоплюють інструменти для перехоплення та аналізу мережного трафіку, моніторингу мережних подій, збору логів з мережних пристроїв, а також засоби аудиту і відстеження змін у мережних конфігураціях. Правильне поєднання цих інструментів дозволяє відтворити хронологію мережних інцидентів, виявити підозрілу активність у мережі та зібрати допустимі доказові матеріали для подальшого розслідування.

З метою побудови ефективної підсистеми розслідування мережних інцидентів у Linux-середовищі було визначено основні типи інструментальних засобів:

- система виявлення вторгнень та кореляції подій (HIDS / SIEM);
- спеціалізовані дистрибутиви для цифрової криміналістики;
- засоби аналізу мережного трафіку та виявлення атак;
- інструменти детального аналізу мережного трафіку;
- програми для проведення атак.^{[6][7]}

Для реалізації підсистеми виявлення вторгнень та кореляції подій у Linux-середовищі обрано систему Wazuh – безкоштовну платформу безпеки з відкритим кодом. Вона поєднує можливості XDR (розширене виявлення та реагування) і SIEM (управління інформацією та подіями безпеки). Вона забезпечує централізований збір логів, моніторинг, аналіз і реагування на кіберзагрози у фізичних, віртуалізованих, контейнеризованих і хмарних середовищах.

Платформа дозволяє виявляти підозрілі дії, виконує глибокий аналіз даних, збираючи, агрегуючи та індексує інформацію з різних джерел безпеки для виявлення інцидентів, загроз та аномальної поведінки. Вона також забезпечує моніторинг цілісності файлів, відстежуючи зміни у файловій системі, зокрема

зміни вмісту, дозволів та атрибутів файлів, що дозволяє вчасно виявляти потенційні загрози. Wazuh підтримує виявлення вразливостей, аналізуючи операційні системи та встановлені програми на наявність відомих слабких місць, і допомагає користувачам усувати їх до моменту експлуатації. Окрім цього, Wazuh спрощує дотримання вимог стандартів безпеки (таких як PCI DSS, NIST тощо), автоматизуючи перевірки, формуючи детальні звіти й забезпечуючи прозорість процесів аудиту.

Архітектурними компонентами є Wazuh Server та Wazuh Agent.

- Wazuh Server: центральний сервер, який аналізує дані, отримані від агентів, виявляє аномалії та загрози.
- Wazuh Agent: легкий агент, який встановлюється на кінцеві точки (сервери, ноутбуки, віртуальні машини, контейнери). Він збирає дані, виявляє загрози та реагує на інциденти.

Wazuh інтегрується з іншими технологіями, що дозволяє візуалізувати дані та проводити їх аналіз. Завдяки відкритому коду, він є гнучким рішенням, яке можуть використовувати як малі підприємства, так і великі корпорації.

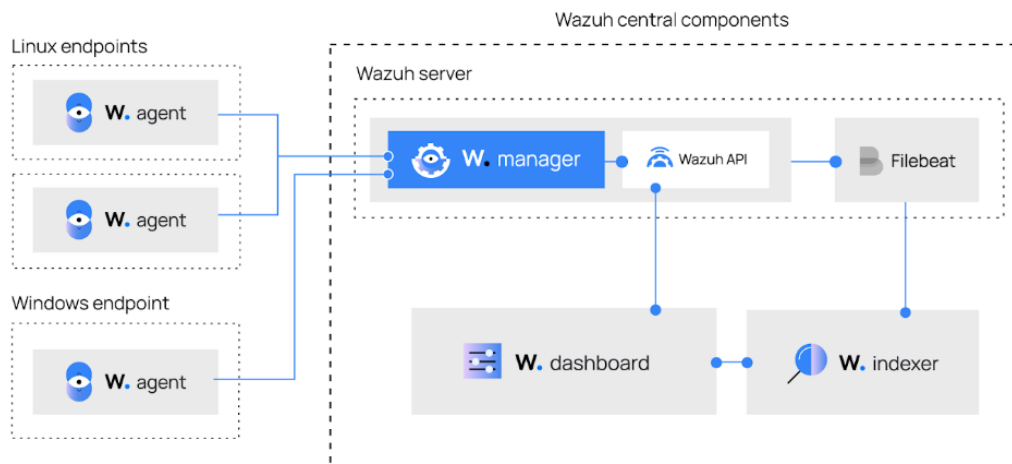


Рисунок 2.1 – Архітектура Wazuh-системи

Порівняльна таблиця переваг Wazuh з іншими системами наведена у додатку А.

Дистрибутиви DFIR – це спеціалізовані операційні системи, попередньо налаштовані для проведення цифрової криміналістики та аналізу інцидентів. Вони містять у своєму складі які містять велику колекцію DFIR-інструментів, що значно спрощує роботу експертів із цифрової криміналістики.

Одним із найпотужніших дистрибутивів є CSI Linux – операційна система, що використовує базову платформу Ubuntu 22.04 LTS, спеціально розроблена для завдань цифрової криміналістики, OSINT (розвідки з відкритих джерел) та реагування на інциденти. Вона поєднує в собі інструменти для аналізу даних, збирання доказів, моніторингу мережного трафіку, дослідження зловмисної активності, а також проведення OSINT-розслідувань. CSI Linux має зручне середовище для аналітиків, дозволяючи ефективно виконувати повний цикл розслідування – від збору інформації до формування звітів. [10] [11] [15]

Повний список інструментів, які встановлені на CSI Linux, знаходиться у додатку Б.

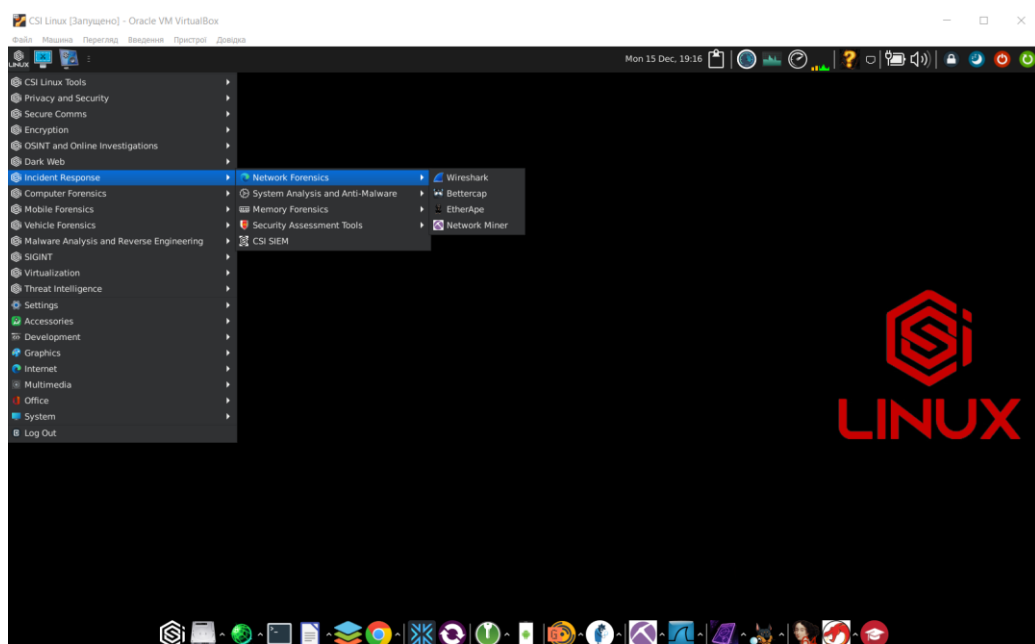


Рисунок 2.2 – Інтерфейс CSI Linux

Серед засобів аналізу мережного трафіку та виявлення атак можна виділити Suricata. Suricata – це високопродуктивна кросплатформна система мережної безпеки з відкритим вихідним кодом, призначена для виявлення атак,

моніторингу трафіку та збору мережних артефактів. Платформа забезпечує такі основні функціональні можливості:

- система виявлення вторгнень (IDS): аналізує мережний трафік у режимі реального часу та виявляє підозрілу або шкідливу активність на основі сигнатур і поведінкових правил;
- система запобігання вторгненням (IPS): крім виявлення, може активно блокувати шкідливі запити та захищати мережу від атак;
- мережна безпека та моніторинг (NSM): збирає деталізовані дані про мережні події, потоки, протоколи та файли для подальшого аналізу в рамках мережної цифрової криміналістики. [13] [14]

Архітектура функціонування Suricata наведена на рисунку 2.3.

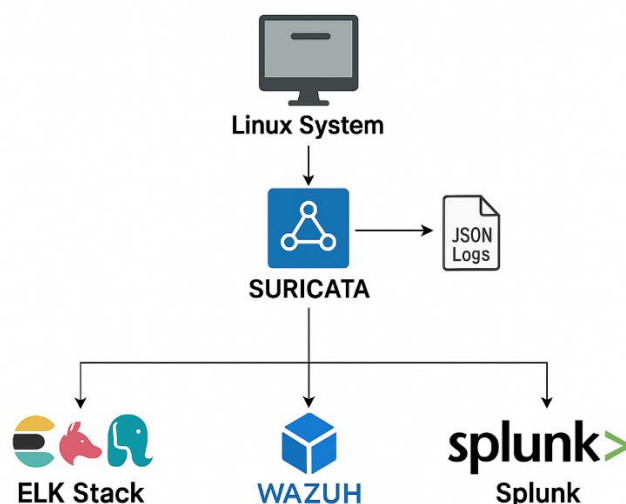


Рисунок 2.3 – Архітектура роботи Suricata

Порівняльна таблиця переваг Suricata наведена у додатку В.

Для детального аналізу захоплених pcap-файлів у роботі буде використано інструменти Wireshark та NetworkMiner.

Wireshark – це найпоширеніший інструмент для аналізу мережного трафіку, який дозволяє детально досліджувати пакети, протоколи та сесії. Він використовується у мережній криміналістиці для аналізу PCAP-файлів, реконструкції подій і виявлення аномалій. [15]

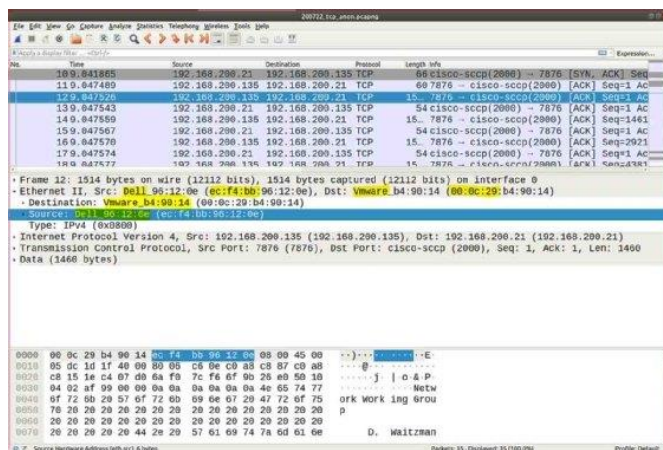
NetworkMiner – пасивний інструмент мережної криміналістики, орієнтований на витяг артефактів із захопленого трафіку. Він автоматично виділяє файли, зображення, DNS-запити, сертифікати та інші артефакти для подальшого аналізу.^[15]

Функціонал програм наведено у таблиці 2.1.

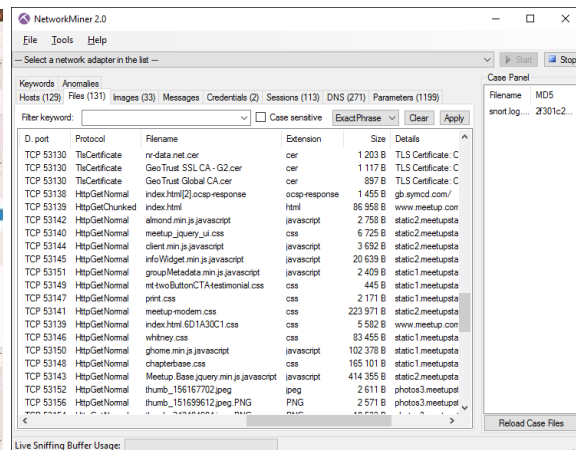
Таблиця 2.1

Порівняння основних функцій програм

Можливості	Wireshark	NetworkMiner
Тип аналізу	Інтерактивний, детальний аналіз пакетів	Пасивний аналіз, витяг артефактів
Основне призначення	Перегляд, декодування та аналіз мережних пакетів і протоколів	Витяг файлів, метаданих та інформації про хости з PCAP
Декодування протоколів	Так, широкий спектр (TCP/IP, HTTP, DNS, TLS тощо)	Обмежено, орієнтований на витяг артефактів
Аналіз трафіку в реальному часі	Так	Ні, лише з PCAP-файлів
Фільтрація і реконструкція трафіку	Так, сесії, потоки, пакети	Відновлення TCP-сесій і поточкових структур
Виявлення аномалій	Так	Частково (шкідливі об'єкти в трафіку)
Витяг файлів / метаданих	Потрібні додаткові плагіни або скрипти)	Так, автоматично
Ідентифікація хостів / MAC / ОС / портів	Обмежено	Так, автоматично
Використання в цифровій криміналістиці	Аналіз PCAP, реконструкція подій, деталізація пакетів	Витяг артефактів, підготовка доказів, аналіз файлів та сесій



A



Б

Рисунок 2.4 – Інтерфейс програм Wireshark (А) та NetworkMiner (Б)

Для повноцінного дослідження процесів виявлення та аналізу мережних інцидентів до складу експериментального стенду було включено низку інструментів, що дозволяють генерувати різні типи мережної активності та перевіряти реакцію систем моніторингу, а саме Hydra, Nmap, hping3, curl і ApacheBench. Розглянемо детальніше призначення та функції обраного програмного забезпечення.

1) *Hydra* – це інструмент для проведення атак типу brute-force, призначений для тестування стійкості сервісів автентифікації до підбору облікових даних. У роботі використовується для моделювання атак несанкціонованого доступу з метою перевірки ефективності засобів виявлення та фіксації мережних інцидентів. [16]

До основних функціональних можливостей інструмента належать:

- виконання brute-force та dictionary-атак на мережні сервіси;
- підтримка великої кількості протоколів (SSH, FTP, HTTP(S), Telnet, SMTP, POP3 та інші);
- можливість паралельного виконання атак для підвищення швидкості підбору;
- використання списків логінів і паролів для моделювання реалістичних сценаріїв атак;

- детальне логування результатів атак для подальшого аналізу.

2) *Nmap* – це інструмент для активного сканування мережі, який використовується для виявлення хостів, відкритих портів, запущених сервісів та їхніх версій. У роботі застосовується для моделювання розвідувальної фази атаки з метою перевірки можливостей систем моніторингу та фіксації підозрілої мережної активності.^[16]

Інструмент надає можливості для:

- визначення активних хостів у мережі, версій сервісів і ОС;
- сканування TCP та UDP портів;
- підтримка різних типів сканування (SYN scan, FIN scan, XMAS scan, UDP scan);
- використання скриптів Nmap Scripting Engine (NSE) для поглибленого аналізу безпеки.

3) *hping3* – це інструмент для генерації та аналізу мережних пакетів, який використовується для тестування мережної безпеки та моделювання атак відмови в обслуговуванні. У роботі застосовується для створення контрольованих мережних аномалій та аналізу реакції систем виявлення вторгнень.^[19]

Функціональними можливостями інструмента є:

- генерація TCP, UDP та ICMP пакетів з довільними параметрами;
- моделювання атак типу DoS та Flood;
- перевірка фільтрації пакетів і правил міжмережних екранів;
- аналіз відповідей мережних вузлів на нестандартні запити;
- підтримка високої швидкості відправлення пакетів для навантажувального тестування.

4) *curl* – це універсальний інструмент командного рядка для передачі даних з використанням мережних протоколів. У роботі використовується для моделювання автоматизованих HTTP-запитів до веб-сервера з метою аналізу реакції системи моніторингу та виявлення підозрілої мережної активності.^[20]

До основних функціональних можливостей інструмента належать:

- виконання HTTP/HTTPS-запитів з командного рядка;
- формування користувацьких HTTP-заголовків;
- автоматизація мережних звернень у скриптах і тестових сценаріях;
- можливість ідентифікації інструмента в трафіку за характерним User-Agent.

5) *ApacheBench* (ab) – це спеціалізований інструмент для навантажувального тестування веб-серверів, призначений для оцінки їх продуктивності та стійкості до великої кількості однотипних HTTP-запитів. У роботі використовується для моделювання інтенсивної автоматизованої активності, що має ознаки DoS-атаки або веб-сканування, з метою перевірки ефективності засобів виявлення інцидентів.^[17]

До основних функціональних можливостей інструмента належать:

- генерація великої кількості HTTP-запитів за короткий проміжок часу;
- підтримка паралельних з'єднань для створення навантаження на сервер;
- імітація масових звернень до одного або кількох ресурсів;
- отримання статистики щодо обробки запитів сервером;
- чітка ідентифікація в мережному трафіку за User-Agent ApacheBench.

2.3 Проектування стенду для проведення досліджень та розслідування мережних інцидентів

Для проведення дослідження та практичної перевірки методів мережної цифрової криміналістики було змодельовано та розгорнуто віртуальний стенд.

Проект стенду представлений на рисунку 2.4.

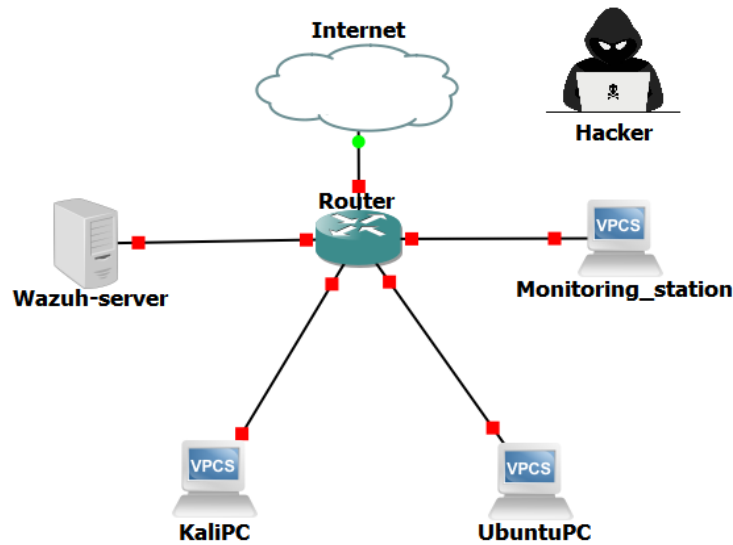


Рисунок 2.5 – Проект віртуального стенду

Структура середовища складається з таких основних компонентів:

- *Маршрутизатор (Router)* – забезпечує маршрутизацію трафіку між усіма вузлами локальної мережі та з'єднання із зовнішньою мережею Інтернет.

- *Wazuh-server* – основний сервер системи моніторингу та реагування на інциденти (SIEM). Він здійснює збір журналів безпеки з агентів, кореляцію подій, виявлення підозрілої активності та формування інцидентів безпеки. Результати аналізу передаються до Wazuh Dashboard для подальшої візуалізації та аналізу.

- *Monitoring station* – це аналітична станція, призначена для моніторингу, збору та аналізу мережного трафіку в реальному часі та загальної інформації про агенти мережі. На цій станції розгорнуто середовище CSI Linux, яке використовується для аналізу даних, отриманих від Wazuh-сервера та мережної системи виявлення вторгнень Suricata, а також для проведення детального криміналістичного аналізу мережних інцидентів.

У складі monitoring station функціонує мережна система виявлення вторгнень, яка аналізує відзеркалений мережний трафік цільової мережі. У разі виявлення підозрілої активності формуються алерти безпеки, які передаються до Wazuh-сервера. Одночасно автоматизаційний скрипт ініціює запис мережного

трафіку на основі активованого попередження у форматі PCAP для подальшого аналізу.

– *Hacker* – зовнішній зловмисник бази дистрибутива Kali Linux, який здійснює атаки на вузли мережі з метою перевірки працездатності системи моніторингу та коректності фіксації інцидентів.

– *KaliPC та UbuntuPC* є робочими станціями користувачів-жертв, на яких встановлено Wazuh Agent. Агенти здійснюють збір системних та безпекових подій і передають їх на Wazuh-server для подальшого аналізу.

Загальний принцип роботи віртуального стенду зображений на блок-схемі на рисунку 2.5.

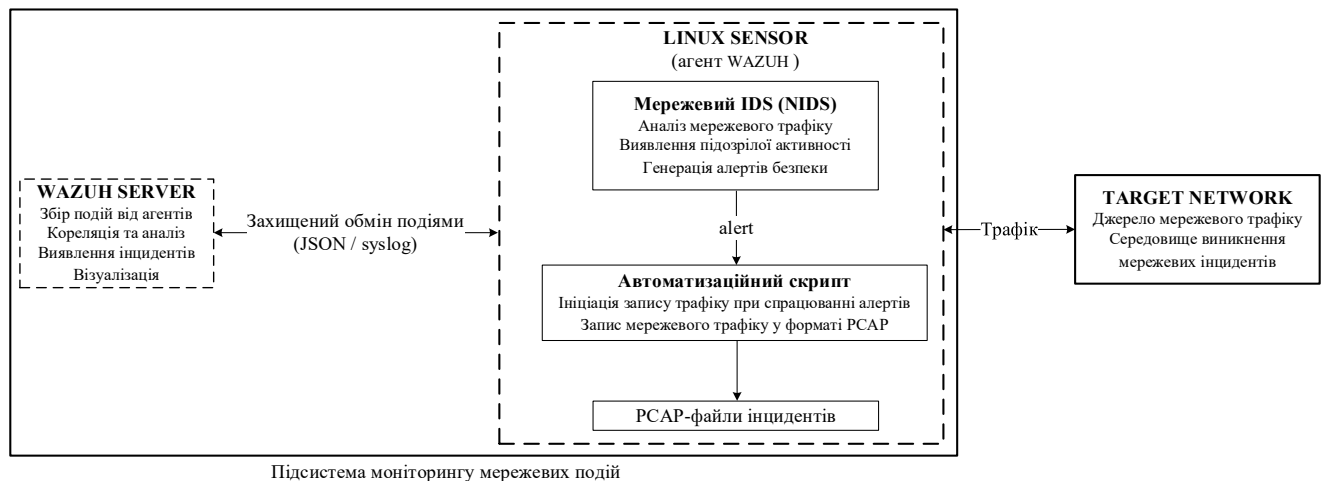


Рисунок 2.5 – Принцип роботи стенду

Для забезпечення коректної взаємодії компонентів експериментального стенду було виконано налаштування мережної адресації в межах однієї підмережі. Частині вузлів призначено статичні IP-адреси для стабільної роботи систем моніторингу та аналізу, іншим вузлам — динамічну адресацію. Таблиця адресації наведена у таблиці 2.2.

Таблиця адресації підмережі

№	Пристрій	Тип адреси	IP-адреса	Маска / Префікс
1	Router	статична	192.168.0.1	255.255.255.0 /24
2	Wazuh-server	статична	192.168.0.105	255.255.255.0 /24
3	Monitoring Station	статична	192.168.0.100	255.255.255.0 /24
4	Hacker	статична	192.168.0.108	255.255.255.0 /24
5	UbuntuPC	динамічна	192.168.0.106	255.255.255.0 /24
6	KaliPC	динамічна	192.168.0.103	255.255.255.0 /24

Висновки з розділу 2:

Було розроблено методику мережної цифрової криміналістики для Linux-середовищ, орієнтовану на виявлення, фіксацію та аналіз мережних інцидентів. Методика ґрунтується на поетапному підході, який включає збір мережного трафіку, журналів безпеки та їх подальшу кореляцію і аналіз. Обґрунтовано вибір інструментальних засобів, зокрема платформи Wazuh як центрального елемента моніторингу та реагування, системи Suricata для виявлення мережних атак, а також середовища CSI Linux для проведення криміналістичного аналізу. Запропоновано архітектуру стенду, що забезпечує автоматизований запис мережного трафіку у форматі PCAP у разі спрацювання алертів безпеки.

Використання зазначених інструментів у сукупності дозволяє сформувати різнотипні сценарії мережної взаємодії та забезпечує комплексний підхід до аналізу ефективності засобів мережної цифрової криміналістики.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ТА РОЗСЛІДУВАННЯ МЕРЕЖНИХ ІНЦИДЕНТІВ

3.1. Базове налагодження стенду для проведення розслідування мережних інцидентів

3.1.1 Налаштування Wazuh-серверу

З офіційного сайту було завантажено образ віртуальної машини Wazuh-серверу та імпортовано у VirtualBox. [6]

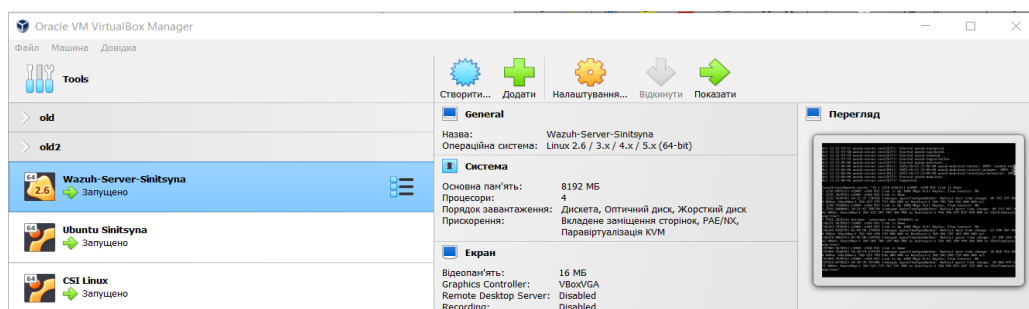


Рисунок 3.1 – Встановлення серверу

Після успішного імпорту машину було запущено, та ми увійшли під базовим користувачем:

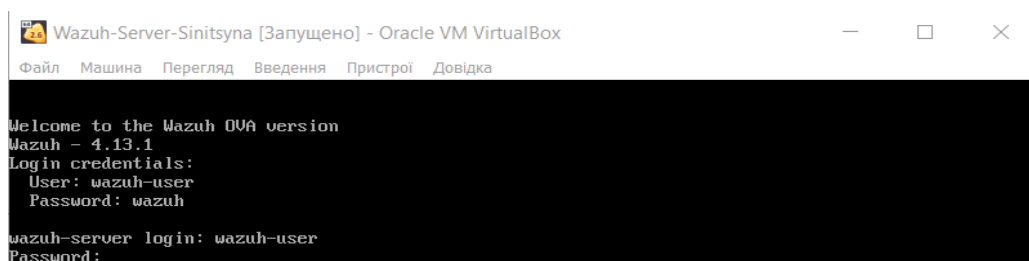


Рисунок 3.2 – Вхід під користувачем

Змінимо IP-адресу на статичну:

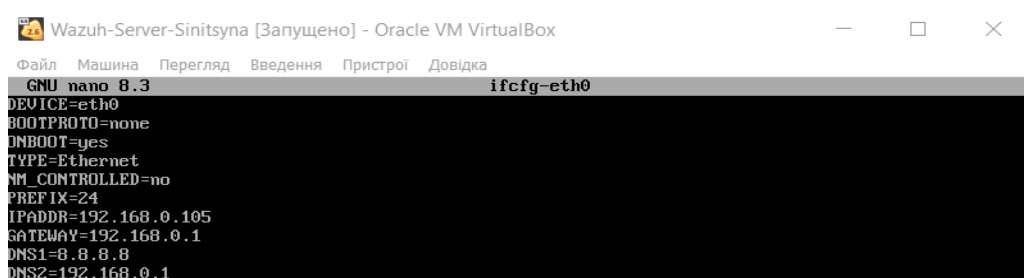


Рисунок 3.3 – Налаштування статичної адреси

```
[wazuh-user@wazuh-server network-scripts]$ sudo systemctl restart network
```

Рисунок 3.4 – Перезапуск мережі

```
[wazuh-user@wazuh-server network-scripts]$ 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
   inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:90:e1:91 brd ff:ff:ff:ff:ff:ff
   altname enp0s17
   inet 192.168.0.101/24 metric 1024 brd 192.168.0.255 scope global dynamic eth0
       valid_lft 5152sec preferred_lft 5152sec
   inet 192.168.0.105/24 brd 192.168.0.255 scope global secondary eth0
       valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe90:e191/64 scope link proto kernel_ll
       valid_lft forever preferred_lft forever
```

Рисунок 3.5 – Перевірка IP-адреси

Закріпивши веб-сайт за певною адресою, ми можемо отримати доступ до Wazuh dashboard. Для того, щоб переглянути сайт, введемо статичну IP-адресу сервера у рядок браузера:

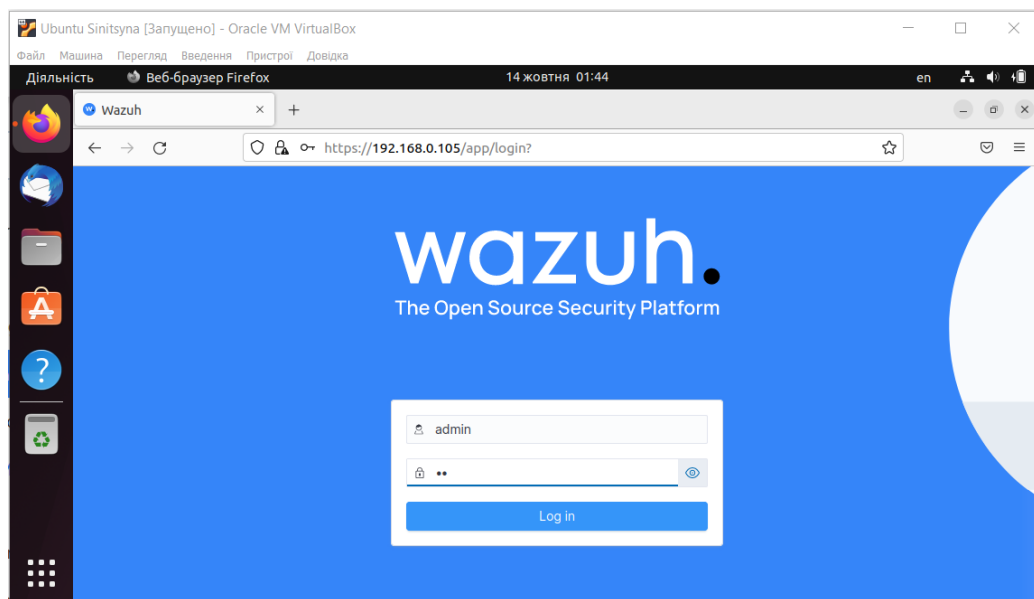


Рисунок 3.6 – Вхід на Wazuh dashboard

Після входу у Wazuh dashboard можемо побачити, що наш сервер онлайн і містить усі необхідні оновлення:

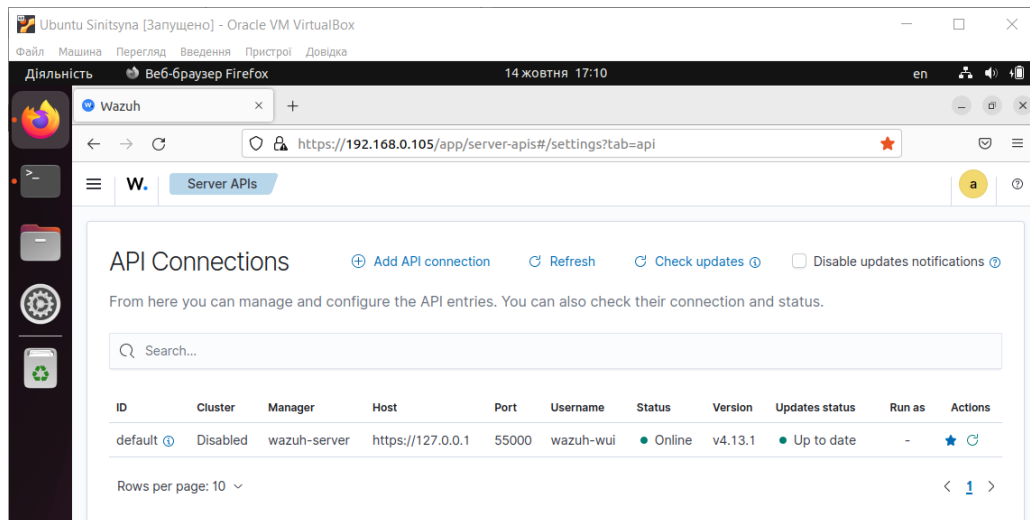


Рисунок 3.7 – Результат роботи серверу

3.1.2 Налаштування Wazuh-agent ^[6] ^[8]

Важливо, щоб агент і сервер Wazuh мали однакову версію, що забезпечує сумісність та коректну взаємодію. У нашому випадку це версія 4.13.1.

Для налаштування агента спочатку імпортуємо GPG-ключ Wazuh для перевірки підписів пакетів і надамо необхідні дозволи на файл ключа.

Команда: `sudo curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | sudo gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && sudo chmod 644 /usr/share/keyrings/wazuh.gpg`

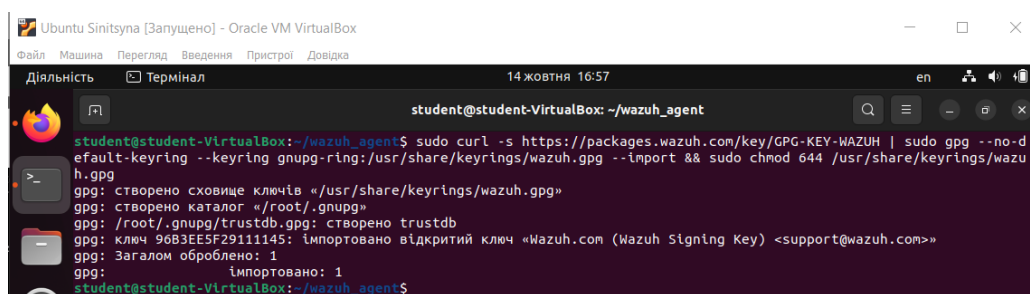


Рисунок 3.8 – Встановлення додаткових пакунків

Наступним кроком додаємо репозиторій Wazuh у список джерел пакетів системи та встановлюємо Wazuh-агент:

Команда: `sudo echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | sudo tee -a /etc/apt/sources.list.d/wazuh.list`

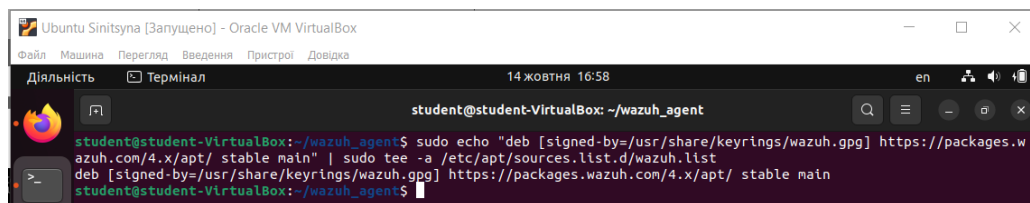


Рисунок 3.9 – Додавання репозиторію

Команда: `sudo apt install wazuh-agent`

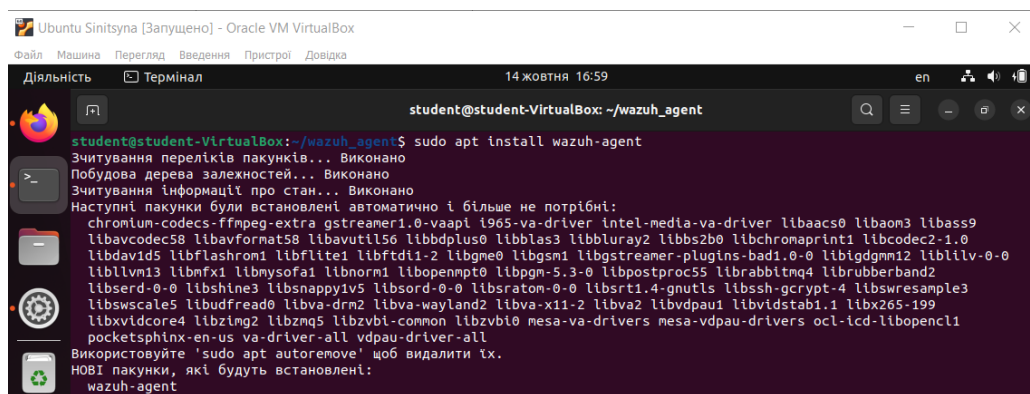


Рисунок 3.10 – Встановлення агента

Перевіримо успішність агента переглядом статусу служби:

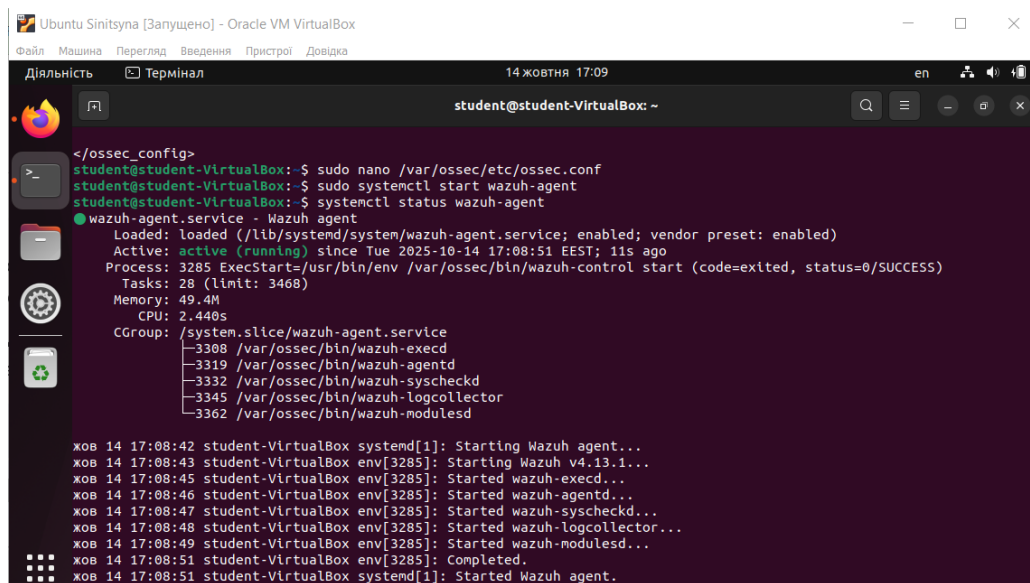


Рисунок 3.11 – Результат встановлення агента

Наступним кроком є налаштування адреси Wazuh-сервера у конфігураційному файлі агента:

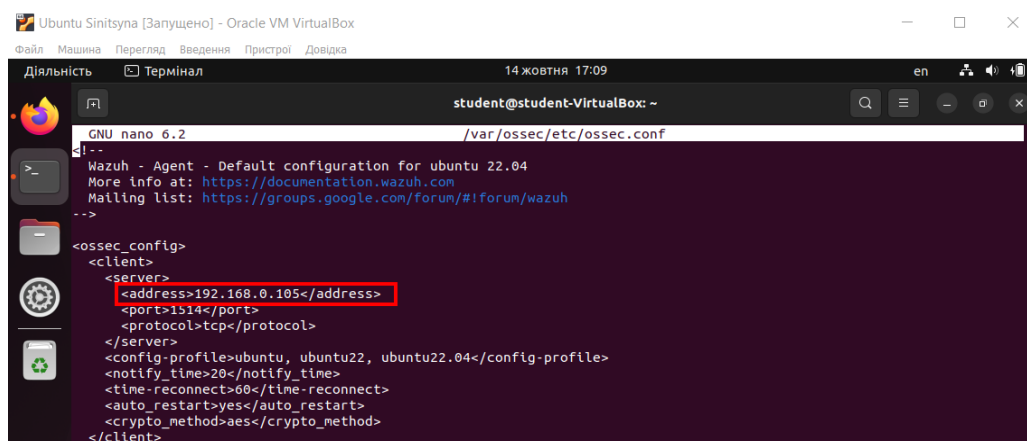


Рисунок 3.12 – Налаштування адреси серверу у конфігураційному файлі

Після успішної активації агента додаємо його на сервер для централізованого моніторингу.

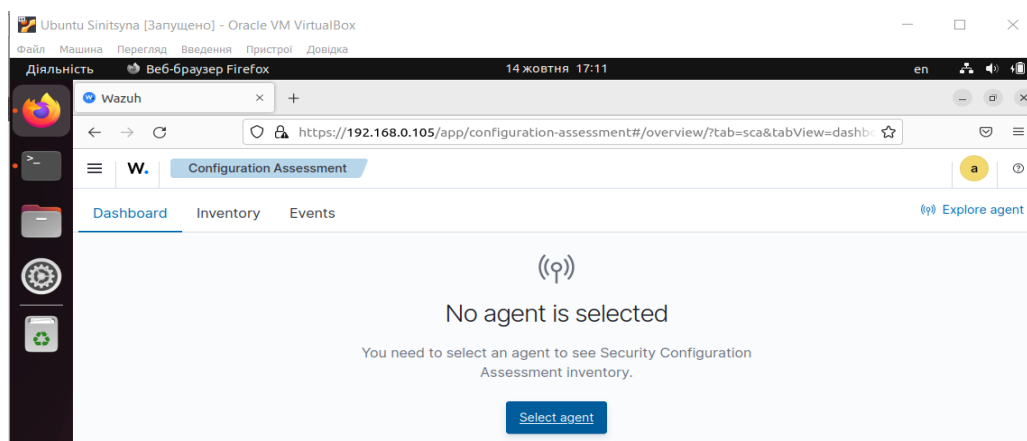


Рисунок 3.13 – Налаштування агента UbuntuPC

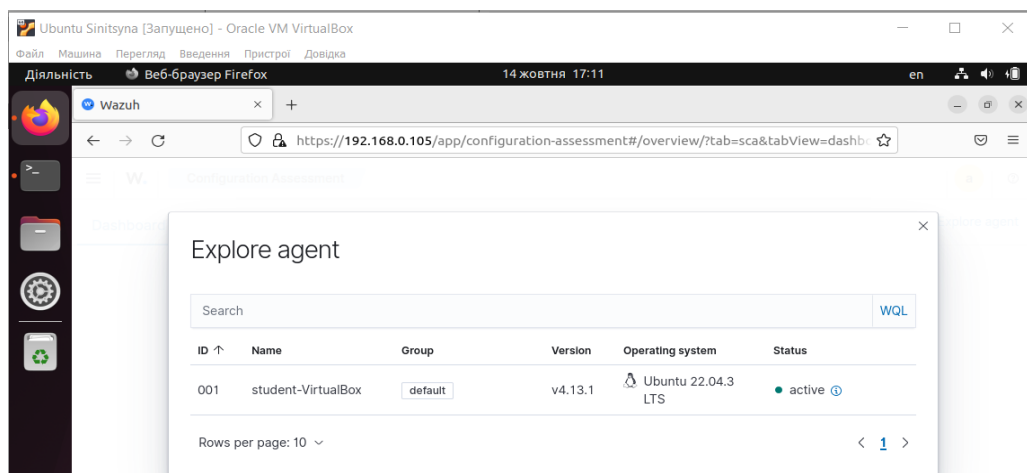


Рисунок 3.14 – Додавання агента UbuntuPC

Як результат, моніторинг відобразив нещодавні дії по встановленню пакунків на віртуальну машину.

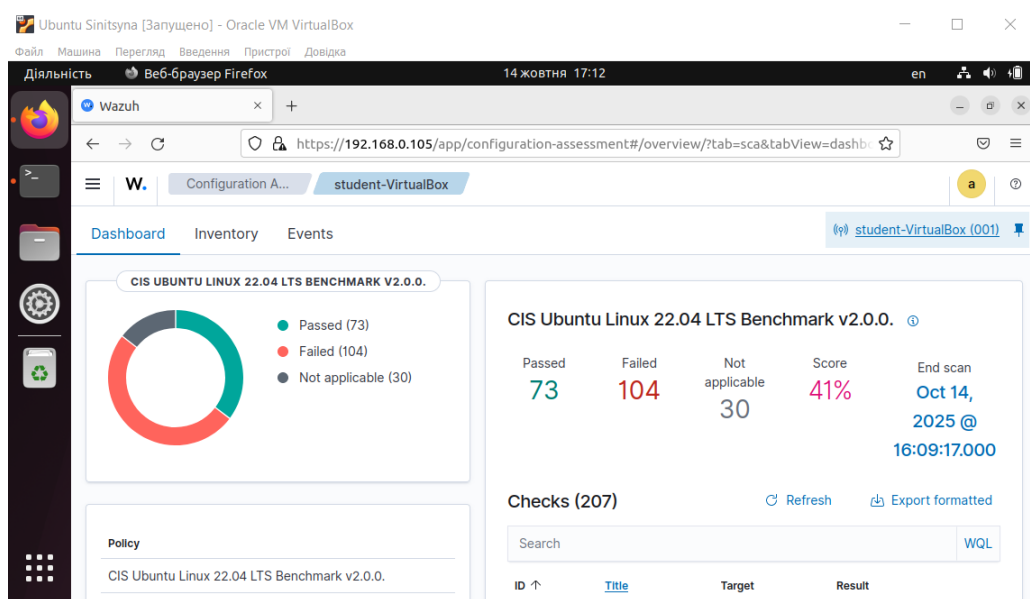


Рисунок 3.15 – Результат додавання агента UbuntuPC

Агенти для віртуальної машини KaliPC та Monitoring Station були налаштовані аналогічно. Для зручності моніторингу агентів також було створено 2 групи *servers* та *user_linux-agents*.

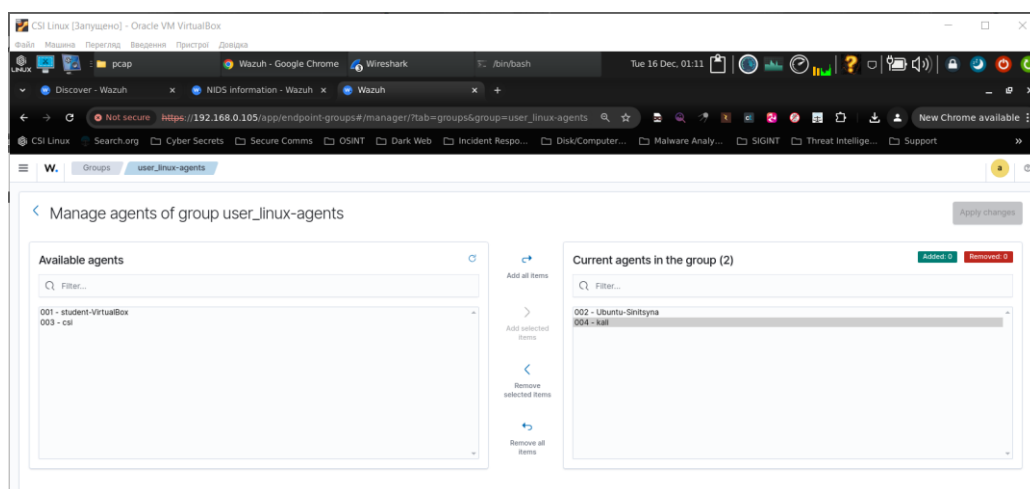


Рисунок 3.16 – Внесення агентів до групи *user_linux-agents*

Після конфігурації агенти автоматично додаються до моніторингу сервером.

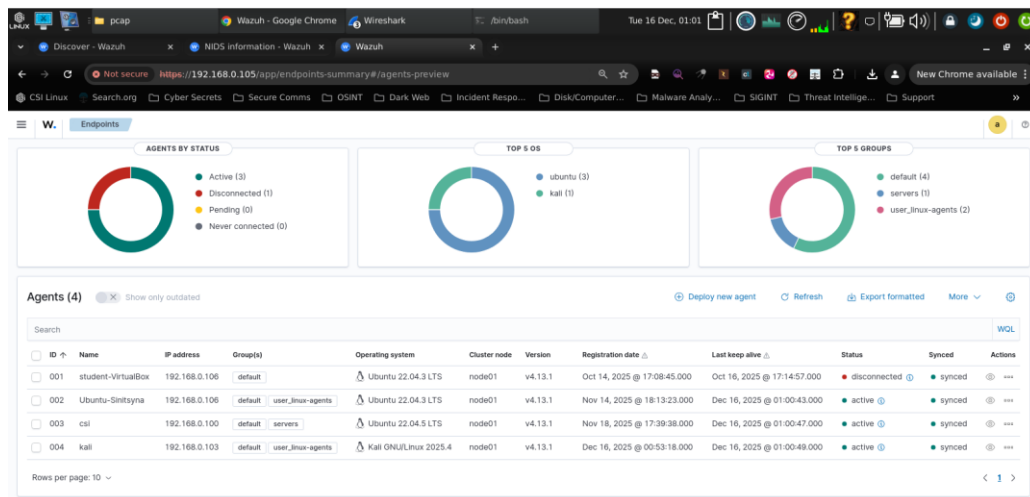


Рисунок 3.17 – Усі додані агенти моніторингу

3.2. Налаштування підсистеми моніторингу мережного трафіку

3.1.3 Інтеграція з Network IDS Suricata

Wazuh не є системою виявлення вторгнень, яка безпосередньо аналізує мережний трафік, так як основною функцією є моніторинг безпеки кінцевих точок, збір системних журналів, моніторинг цілісності файлів та оцінка вразливостей. Але Wazuh добре інтегрується з різними мережними системами виявлення вторгнень, тому для глибокого аналізу мережі ми підключимо Suricata до одного з Wazuh-агентів. [7]

1. Налаштування Suricata [12] [13] [14]

Команда: `sudo apt install suricata`

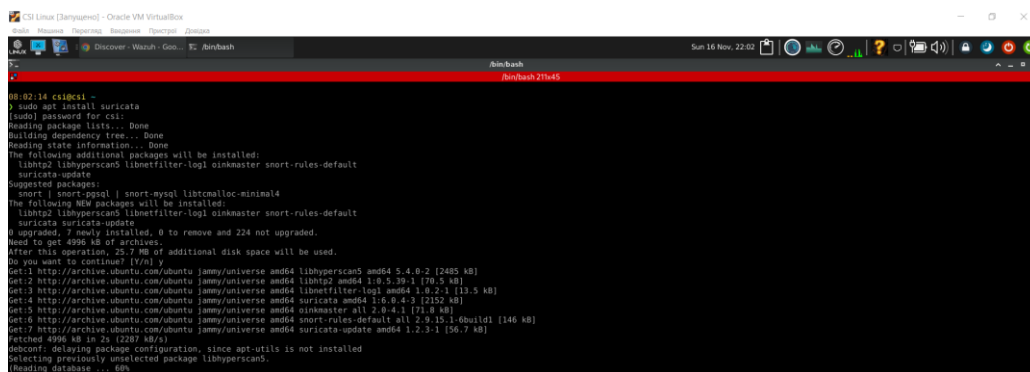


Рисунок 3.18 – Встановлення Suricata

У файлі `/etc/suricata/suricata.yaml` необхідно відредагувати параметри конфігурації. Зокрема, налаштовуються:

- мережні інтерфейси для моніторингу,
- типи та формат логів, які генеруватиме Suricata.

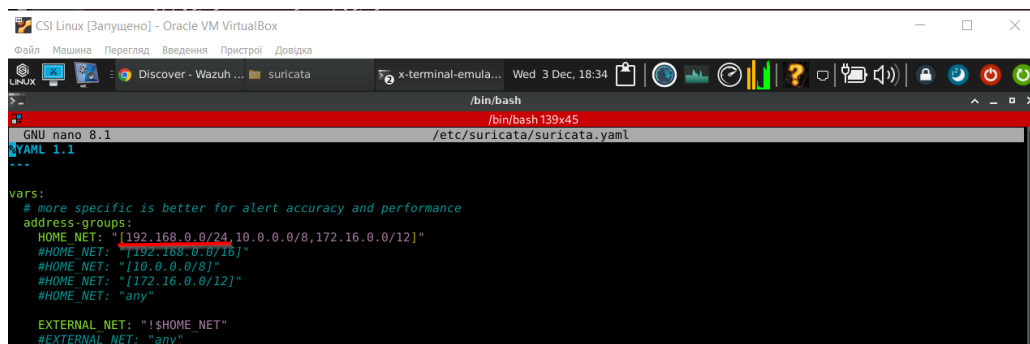


Рисунок 3.19 – Налаштування підмереж для моніторингу

Визначаємо основні типи лог-файлів, які будуть використовуватись для фіксації мережного трафіку, типи даних та інтерфейси для моніторингу. Основні журнали, які нас цікавлять:

- `fast.log` – швидкий і стислий перелік усіх спрацьовувань,
- `eve.json` – найбільш детальний журнал усіх подій.

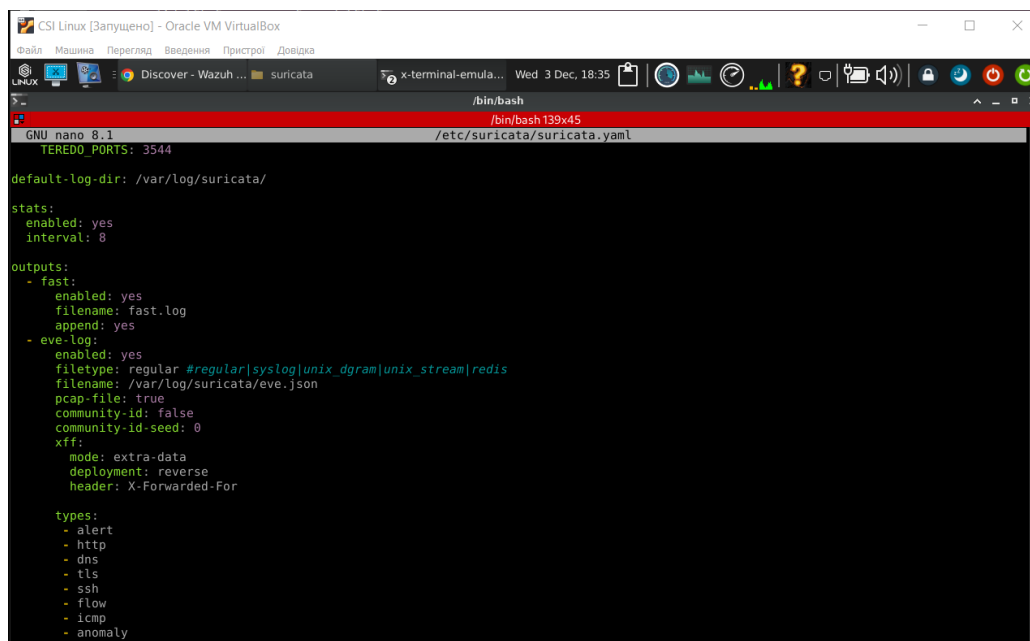


Рисунок 3.20 – Налаштування файлів у конфігураційному файлі

```
af-packet:
- interface: enp0s3
  threads: 1
  cluster-id: 99
  cluster-type: cluster_flow
  defrag: yes
- interface: enp0s8
  threads: 1
  cluster-id: 98
  cluster-type: cluster_flow
  defrag: yes
  use-mmap: yes
  mmap-lock: yes
```

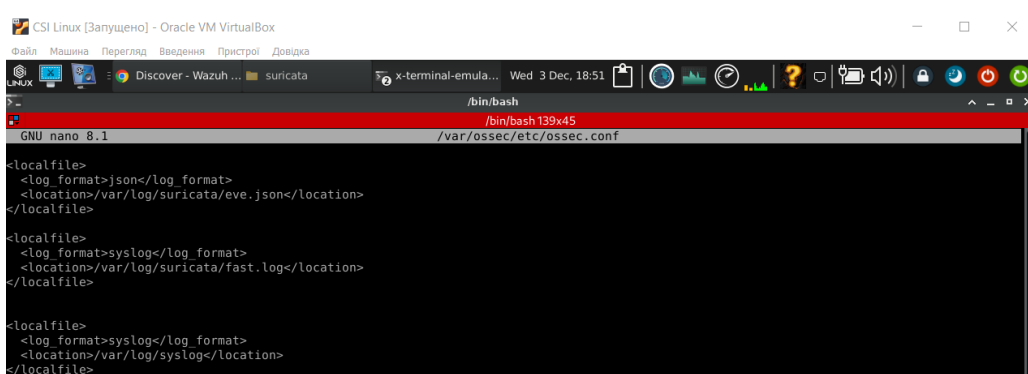
Рисунок 3.21 – Налаштування інтерфейсів у конфігураційному файлі

2. Інтеграція логів Suricata з Wazuh

Щоб журнали Suricata надходили до Wazuh Agent, у файл `/var/ossec/etc/ossec.conf` потрібно додати відповідний блок:

```
<localfile>
  <log_format>потрібний формат файлу</log_format>
  <location>шлях до файлу і назва</location>
</localfile>
```

Після цього агент зможе обробляти записи Suricata та передавати їх менеджеру Wazuh для подальшого аналізу.



```
GNU nano 8.1 /var/ossec/etc/ossec.conf
<localfile>
  <log format>json</log format>
  <location>/var/log/suricata/eve.json</location>
</localfile>

<localfile>
  <log format>syslog</log format>
  <location>/var/log/suricata/fast.log</location>
</localfile>

<localfile>
  <log format>syslog</log format>
  <location>/var/log/syslog</location>
</localfile>
```

Рисунок 3.22 – Налаштування журналів у конфігураційному файлі

Повні конфігураційні файли знаходяться у додатку Е.

3. Налаштування та створення правил Suricata на Wazuh Server:

Для забезпечення повноцінного мережного моніторингу на сервері зазвичай використовується набір стандартних правил Suricata. Вони дозволяють

фіксувати події різних типів трафіку й формувати артефакти для подальшого аналізу інцидентів. [9]

```
06:53:39 csi@csi ~
> sudo ls /etc/suricata/rules
app-layer-events.rules  dns-events.rules      ipsec-events.rules     mqtt-events.rules      smtp-events.rules
decoder-events.rules    files.rules            kerberos-events.rules  nfs-events.rules        stream-events.rules
dhcp-events.rules       http-events.rules      local.rules             ntp-events.rules        tls-events.rules
dnp3-events.rules       http2-events.rules     modbus-events.rules     smb-events.rules
```

Рисунок 3.23 – Перегляд доступних rule-files у Suricata

Для моніторингу було обрано наступні rule-files:

- dhcp-events.rules

Використовується для виявлення інцидентів, пов'язаних із Rogue DHCP-серверами, підміною адрес, а також флудингом DHCP-пакетами, що може спричинити відмову в обслуговуванні.

- decoder-events.rules

Виявляє низькорівневі аномалії у мережних заголовках (пошкоджені пакети, неправильні параметри), що дозволяє знаходити ознаки технічних експлойтів, нестандартних або модифікованих протоколів.

- stream-events.rules

Фіксує аномалії у встановленні та завершенні TCP/UDP/ICMP-сесій (наприклад, перевантаження, неочікувані FIN/RST), що є доказами мережних маніпуляцій та прихованого обміну.

- http-events.rules

Фіксує підозрілі події рівня HTTP, такі як аномальні User-Agent або неправильні методи, необхідні для аналізу Web-атак, фішингу та Command & Control зв'язку.

- tls-events.rules

Відстежує події SSL/TLS, що є ключовим для розслідування зашифрованого трафіку та виявлення MITM-атак.

- files.rules

Виявляє передачу файлів різних типів у трафіку, що критично важливо для ідентифікації передачі шкідливого програмного забезпечення та витоку даних.

– dns-events.rules

Корисний для виявлення DNS Tunneling, DGA-запитів та інших аномалій, пов'язаних із використанням DNS для обміну даними злоумисників.

Повний конфігураційний файл знаходиться у додатку Е.

```
default-rule-path: /etc/suricata/rules

rule-files:
- local.rules
- dhcp-events.rules
- decoder-events.rules
- stream-events.rules
- http-events.rules
- tls-events.rules
- files.rules
- dns-events.rules
```

Рисунок 3.24 – Налаштування rule-files у конфігураційному файлі *sudo nano /etc/suricata/suricata.yaml*

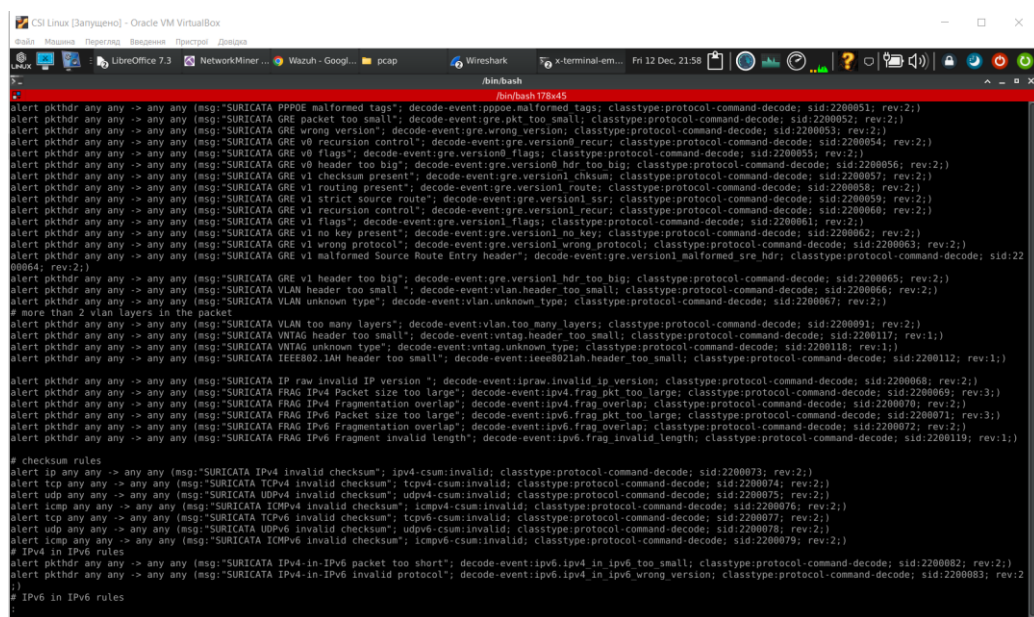


Рисунок 3.25 – Перегляд увімкнутих правил *cat /etc/suricata/rules/*.rules | less*

Також було створено файл local.rules для кастомних правил Suricata, які визначають специфічні типи атак у мережі.^[9]

Команда: sudo nano /etc/suricata/rules/local.rules

До файлу було додано три власних правила, яких немає у стандартних шаблонах, для відстеження специфічних подій:

- ICMP Ping: повідомлення про пінг-запити,
- SYN Flood: можлива DoS-атака на TCP-порт 80 з пороговою кількістю спрацьовувань (threshold) для відстеження підозрілої активності.

– Suspicious HTTP GET request (RAW TCP): виявлення HTTP GET-запитів у TCP-трафіку, що дозволяє фіксувати розвідувальну активність щодо веб-ресурсів сервера.

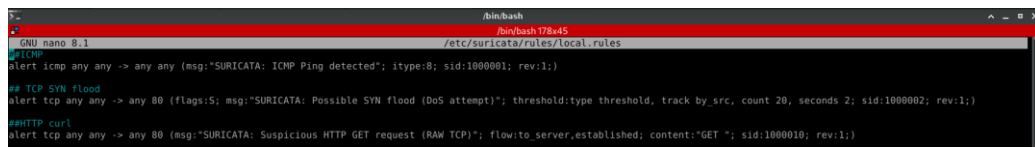


Рисунок 3.26 – Налаштування власних правил у файлі *local.rules*

4. Конфігурація логування повного мережного трафіку в PCAP-файл

Для додаткового моніторингу також можна налаштувати запис повного мережного трафіку у форматі PCAP, що дозволяє зберігати всі пакети для подальшого аналізу та розслідування інцидентів.

Для уникнення перевантаження стенду налаштування було вимкнено.

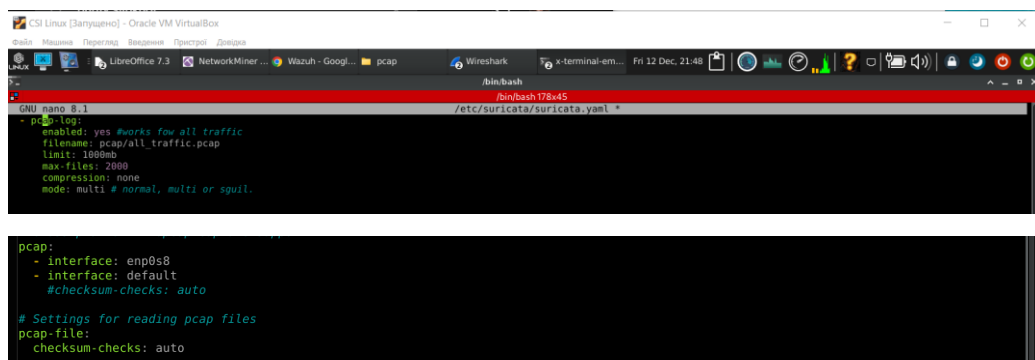


Рисунок 3.27 – Налаштування повного запису трафіку

Після усіх конфігураційних змін обов'язково виконуємо перезапуск suricata та wazuh-агента. Команди для перезапуску знаходяться у додатку Г.

3.1.4 Автоматизований запис PCAP під час мережної атаки

Для автоматичного збору мережного трафіку під час подій Suricata було реалізовано скриптову логіку, яка відстежує появу alert-подій та запускає запис PCAP-файлів тільки тоді, коли фіксується атака. Це дозволяє не заповнювати систему зайвими записами та отримувати лише релевантні фрагменти трафіку.

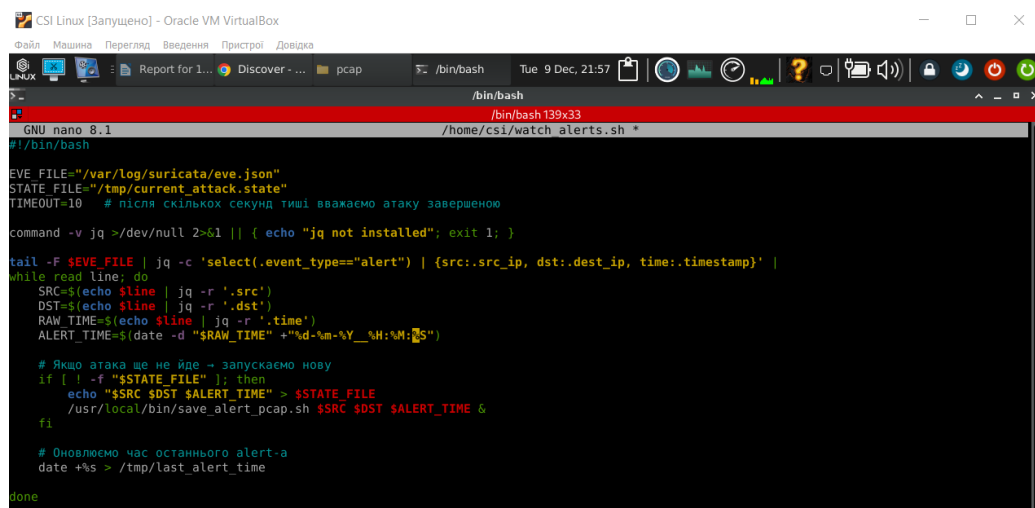
Рішення складається з двох скриптів:

1. `watch_alerts.sh` – моніторинг alert-подій, що надходять від Suricata.

Скрипт у режимі реального часу відстежує файл `eve.json`, фільтрує події типу "alert" і визначає момент початку та завершення атаки. [18] [23]

Основні функції:

- безперервно читає `eve.json` через `tail -F`;
- виділяє IP-адреси джерела та призначення, а також час події;
- запускає захоплення трафіку (`save_alert_pcap.sh`), якщо атака розпочалася;
- відстежує час останнього alert-а в системному тимчасовому файлі;
- завершує поточну сесію захоплення, якщо протягом заданого часу немає нових alert-ів.



```
#!/bin/bash

EVE_FILE="/var/log/suricata/eve.json"
STATE_FILE="/tmp/current_attack.state"
TIMEOUT=10 # після кількох секунд тиші вважаємо атаку завершеною

command -v jq >/dev/null 2>&1 || { echo "jq not installed"; exit 1; }

tail -F $EVE_FILE | jq -c 'select(.event_type=="alert") | {src:.src_ip, dst:.dest_ip, time:.timestamp}' |
while read line; do
    SRC=$(echo $line | jq -r '.src')
    DST=$(echo $line | jq -r '.dst')
    RAW_TIME=$(echo $line | jq -r '.time')
    ALERT_TIME=$(date -d "$RAW_TIME" +"%d-%m-%Y_%H:%M:%S")

    # Якщо атака ще не йде → запускаємо нову
    if [ ! -f "$STATE_FILE" ]; then
        echo "$SRC $DST $ALERT_TIME" > $STATE_FILE
        /usr/local/bin/save_alert_pcap.sh $SRC $DST $ALERT_TIME &
    fi

    # Оновлюємо час останнього alert-а
    date +%s > /tmp/last_alert_time
done
```

Рисунок 3.28 – Bash-скрипт `watch_alerts.sh`

2. `save_alert_pcap.sh` – збереження PCAP-файлу атаки.

Цей скрипт запускається з першого сценарію під час фіксації атаки. Він виконує цілеспрямоване захоплення трафіку за участю джерела й отримувача атаки. [18] [23]

Основні функції:

- створює каталог для збереження PCAP-файлів;
- формує файл виду `attack-DD-MM-YYYY__HH:MM:SS.pcap`;

- запускає tcpdump у фоновому режимі для запису тільки трафіку між IP;
- контролює процес захоплення;
- якщо протягом заданого часу (IDLE_TIMEOUT) не надходять нові alert-и – захоплення завершується;
- коректно зупиняє tcpdump та очищує стан системи.

```

GNU nano 8.1 /usr/local/bin/save_alert_pcap.sh
/bin/bash

SRC=$1
DST=$2
ATTACK_START=$3

INTERFACE="enp0s8"
OUTPUT_DIR="/var/log/suricata/pcap"
STATE_FILE="/tmp/current_attack.state"
LAST_ALERT_FILE="/tmp/last_alert_time"
IDLE_TIMEOUT=10 # скільки секунд без alertів = атака закінчилась

mkdir -p $OUTPUT_DIR

FILENAME="${OUTPUT_DIR}/attack-${ATTACK_START}.pcap"

echo "Starting attack capture: $FILENAME"

# Запускаємо tcpdump в бекграунді
tcpdump -i $INTERFACE host $SRC or host $DST -w $FILENAME &
TCPDUMP_PID=$!

# Основний цикл – чекати поки атака триває
while true; do
    sleep 1

    LAST=$(cat $LAST_ALERT_FILE 2>/dev/null)
    NOW=$(date +%s)

    # Якщо не було alert-ів певний час – атака завершена
    if (( NOW - LAST > IDLE_TIMEOUT )); then
        echo "Attack ended. Stopping tcpdump."
        kill $TCPDUMP_PID 2>/dev/null
        rm -f $STATE_FILE
        break
    fi
done

```

Рисунок 3.29 – Bash-скрипт *save_alert_pcap.sh*

Загальний принцип роботи системи дозволяє автоматично отримувати PCAP-докази інцидентів, уникаючи непотрібного запису повного трафіку в режимі 24/7:

1. Suricata реєструє подію типу alert у eve.json.
2. watch_alerts.sh фіксує перший alert та запускає save_alert_pcap.sh.
3. save_alert_pcap.sh починає запис PCAP-трафіку, пов'язаного з IP-адресами атаки.
4. Поки приходять нові події – захоплення триває.
5. Якщо події зникають на заданий інтервал – запис завершується і файл зберігається.

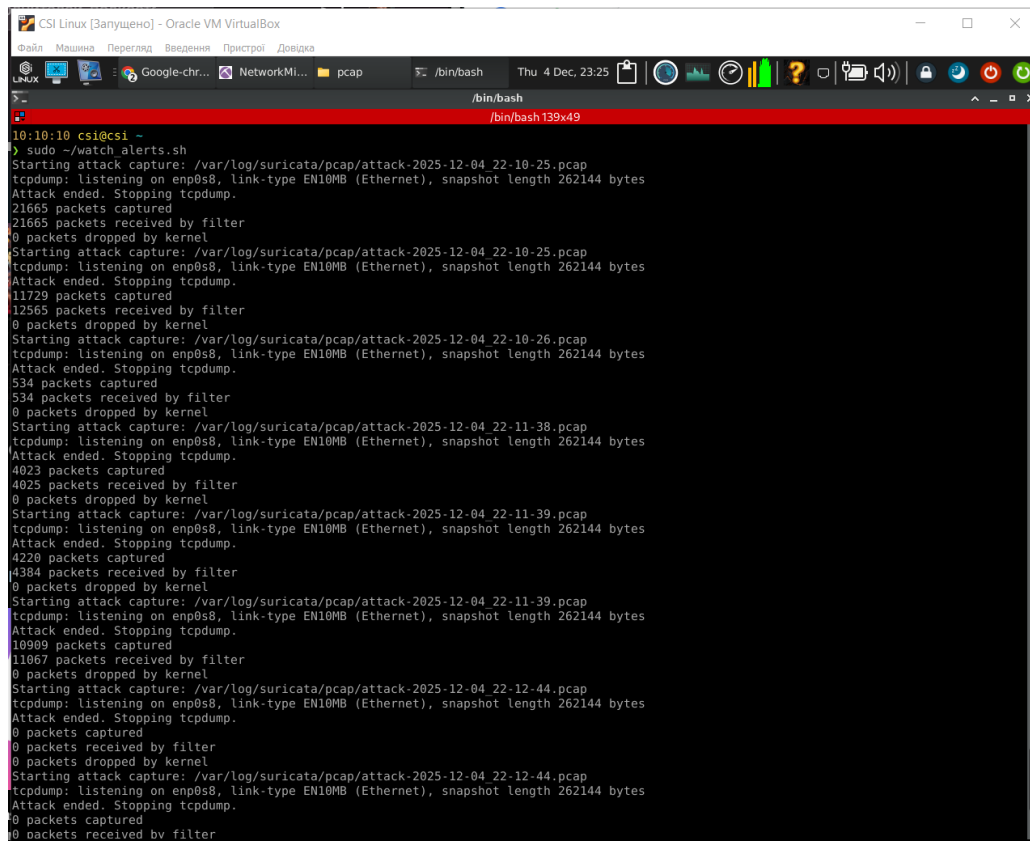


Рисунок 3.30 – Запуск і робота скрипта *watch_alerts.sh*

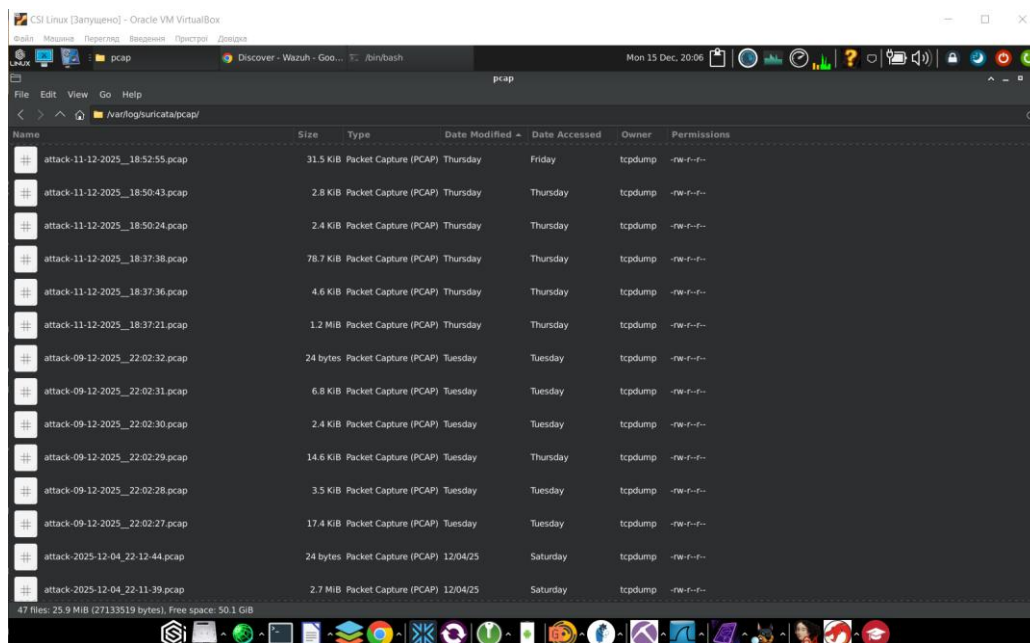


Рисунок 3.31 – Результат прослуховування та запису атак

Скрипти та значення змінних наведені у додатку Г.

3.3. Створення та налаштування спеціалізованих дашбордів для підсистеми моніторингу мережних подій

Для забезпечення оперативного контролю над мережними подіями в середовищі Wazuh було створено набір спеціалізованих дашбордів, орієнтованих на аналіз попереджень, які генерує інтегрована система IDS Suricata. Оскільки стандартні панелі Wazuh не забезпечують повного охоплення мережних інцидентів, було розроблено комплексну панель NIDS Information, що об'єднує загальні та функціональні огляди стану мережної безпеки.

Перед початком створення дашбордів було проаналізовано структуру сповіщень, які надходять до Wazuh від Suricata. Для цього на етапі попереднього аналізу використовувався інструмент Discover у Wazuh, що дає змогу переглядати сирі події, застосовувати фільтри та визначати ключові атрибути, які доцільно виносити на візуалізацію.

Кожне IDS-повідомлення містить низку інформативних полів, серед яких: час події (`predecoder.timestamp`), інформація про агент (`agent.ip`, `agent.name`, `agent.id`), мережні атрибути (`data.srcip`, `data.dstip`, `data.dest_port`), ідентифікатор та характеристики правила (`rule.id`, `rule.level`, `rule.description`, `rule.groups`), а також повний текст спрацювання (`full_log`). [6] [7] [8] [9]



Рисунок 3.32 – Оповіщення типу IDS, що надійшло від Suricata до Wazuh

Після аналізу структури сповіщень було розпочато формування дашбордів. Створення панелей здійснюється у кілька послідовних кроків:

1. У Wazuh відкриваємо розділ Visualize для створення нового дашборду та обираємо тип майбутньої візуалізації.

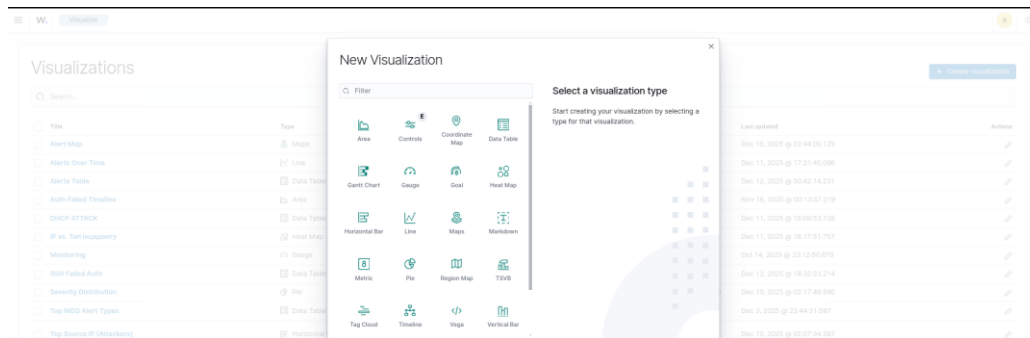


Рисунок 3.33 – Вибір типу графіку для дашборда

2. Оберемо ресурс, з якого будуть братися сповіщення. У нашому випадку це індекс *wazuh-alerts*, який містить усі події та алерти, згенеровані Wazuh та інтегрованими системами.

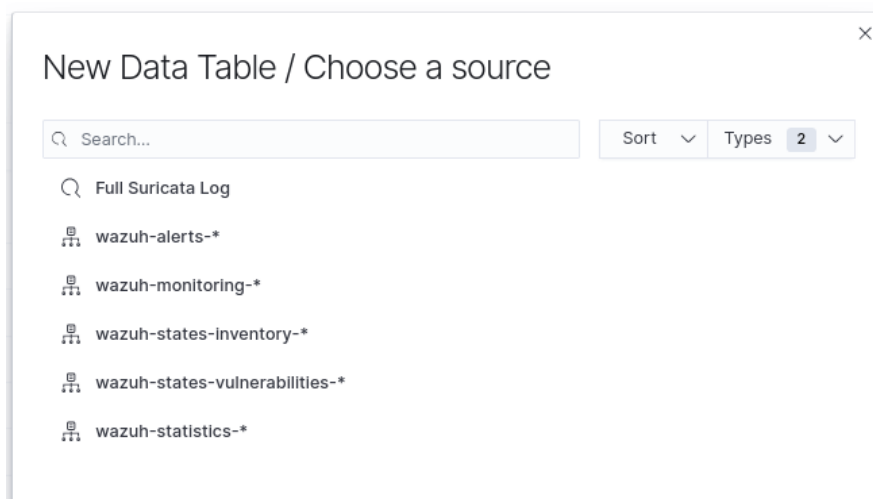


Рисунок 3.34 – Вибір ресурсу

3. Після вибору джерела даних налаштовуються параметри візуалізації залежно від типу графіка.

Для гістограми налаштування будуть виглядати наступним чином:

1. Data → Metrics & Axes → Y-axis

- Aggregation: Count
- Custom label: Count

2. Buckets → X-axis

- Aggregation: Terms

- Field: поле, за яким ми узагальнюємо
- Order by: Metric Count, Order: Descending
- Size: 20
- Group other values in separate bucket: включено
- Show missing values: за потреби
- Custom label: вказується назва осі

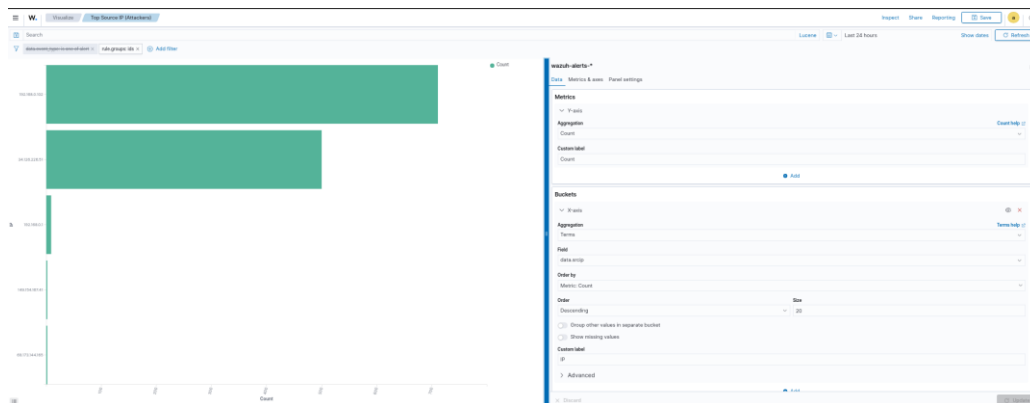


Рисунок 3.35 – Налаштування параметрів гістограми

Для створення таблиці налаштовуються наступні параметри:

1. Data → Metrics

- Metric: Count

2. Buckets → Split Rows

– задаємо розбивку рядків за обраними полями або діапазонами, наприклад, за датою, агентом чи іншими характеристиками події

Timestamp	Host	Agent IP	Source IP	Decoder	Rule ID	Rule Level	Rule Groups	Rule File	Journal Location	Count
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	34.138.226.51	snort	20101	6	ids	16	/var/log/wurkdata/first.log	United States 3
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	34.138.226.51	snort	20101	6	ids	17	/var/log/wurkdata/first.log	United States 3
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	34.138.226.51	snort	20101	6	ids	23	/var/log/wurkdata/first.log	United States 3
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	34.138.226.51	snort	20101	6	ids	25	/var/log/wurkdata/first.log	United States 3
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	34.138.226.51	snort	20101	6	ids	29	/var/log/wurkdata/first.log	United States 3
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	34.138.226.51	snort	20100	8	fts	2	/var/log/wurkdata/first.log	United States 1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	34.138.226.51	snort	20100	8	ids	2	/var/log/wurkdata/first.log	United States 1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	34.138.226.51	snort	20102	10	ids	1	/var/log/wurkdata/first.log	United States 1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	88.173.144.165	snort	20100	8	fts	1	/var/log/wurkdata/first.log	Germany 1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	us	192.168.0.10	88.173.144.165	snort	20100	8	ids	1	/var/log/wurkdata/first.log	Germany 1

Рисунок 3.36 – Налаштування параметрів таблиці

Для створення географічної карти використовується тип візуалізації Documents. У налаштуваннях карти задаються:

- Geospatial field — поле з координатами (наприклад, GeoLocation.location),
- Number of documents — кількість подій для відображення,
- за потреби додаються Filters та Tooltips для зручного перегляду детальної інформації про кожну подію, а також можна змінити стиль відображення.

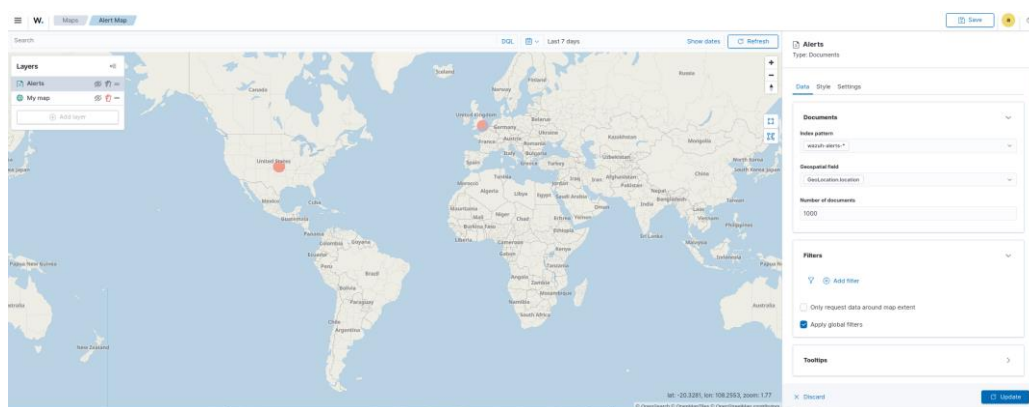


Рисунок 3.37 – Налаштування параметрів для карти

Оскільки поле *full_log* містить великий обсяг тексту, його неможливо відобразити безпосередньо у стандартних візуалізаціях дашборду. Тому використаємо підхід Saved Search + Dashboard:

- Створюємо Saved Search у Discover з полями *full_log* та *timestamp*.

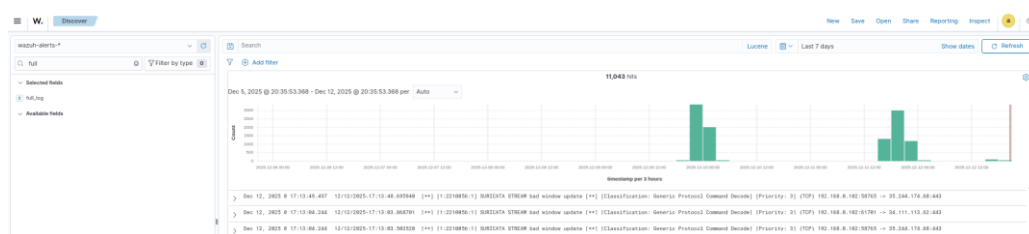


Рисунок 3.38 – Фільтрація за полем *full_log*

- Saved Search додаємо на потрібний дашборд у вигляді таблиці, що дозволяє прокручувати, сортувати та шукати записи.

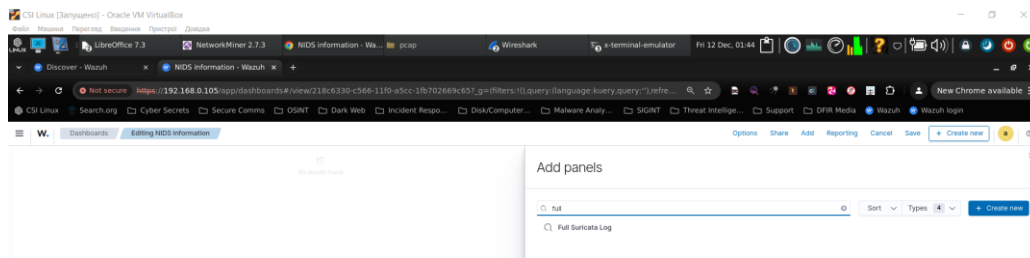


Рисунок 3.39 – Пошук потрібного Saved Search

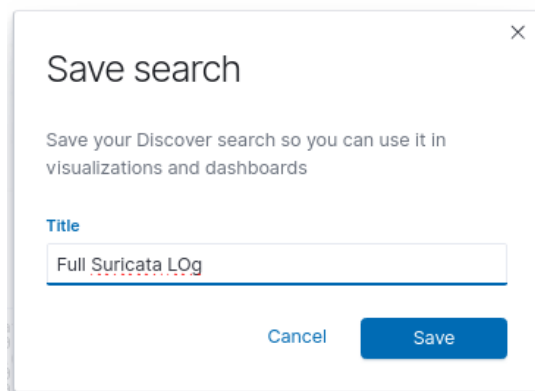


Рисунок 3.40 – Збереження пошуку по обраному фільтру

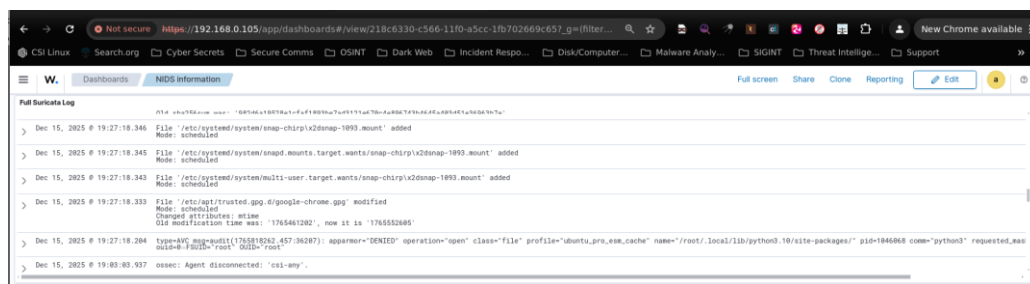


Рисунок 3.41 – Результат додавання дашборду на панель

Розроблена інформаційна панель NIDS Information складається з набору дашбордів, які охоплюють як загальний моніторинг стану системи, так і спеціалізовані функціональні огляди. До дашбордів загального огляду можна віднести:

- Total IDS Alerts: показує загальну кількість попереджень за обраний період.

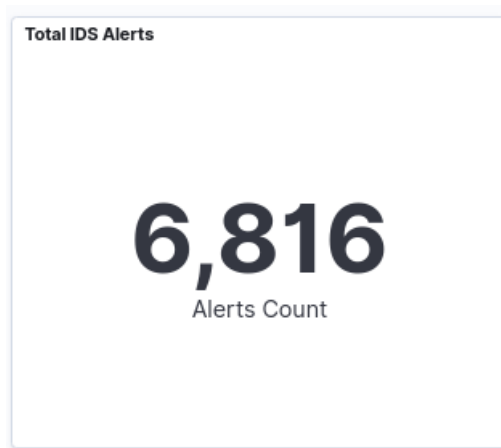


Рисунок 3.42 – Дашборд Total IDS Alerts

- Alerts Over Time: графік, що відображає активність попереджень у часі.

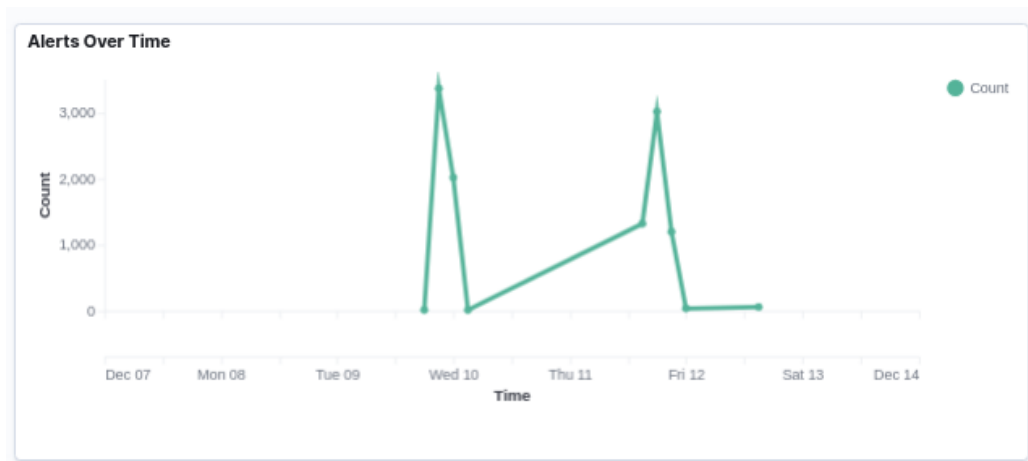


Рисунок 3.43 – Дашборд Alerts Over Time

- Full Suricata Log: деталізовані лог-записи про мережеву активність та попередження.



Рисунок 3.44 – Дашборд Full Suricata Log

- Top Source IP (Attackers): гістограма, що виділяє IP-адреси, які згенерували найбільшу кількість попереджень.

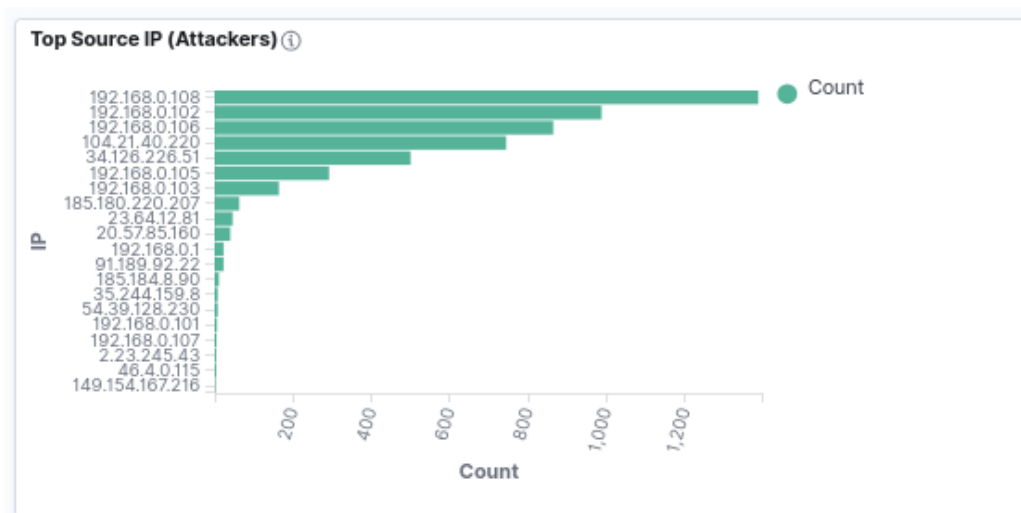


Рисунок 3.45 – Дашборд Top Source IP (Attackers)

– Severity Distribution: кругова діаграма, що показує, як попередження розподіляються за рівнями правил, де рівні 0-16 показують важливість сповіщень. Інтерпретація рівнів rule.level знаходиться у додатку Д.

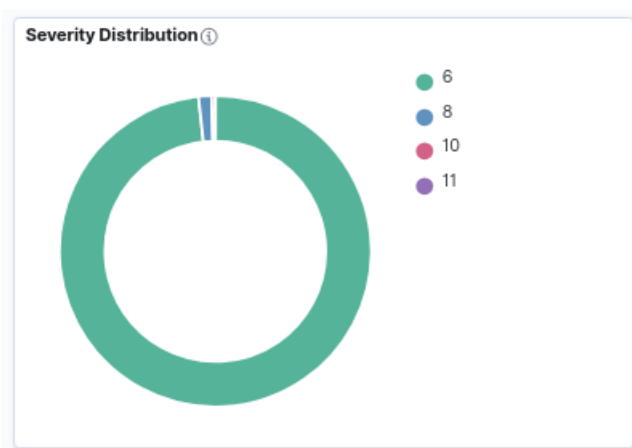


Рисунок 3.46 – Дашборд Severity Distribution

– Alerts Table: деталізований журнал, що надає контекст щодо виявлених подій

Timestamp	Host	Agent IP	Source IP	Decoder Name	Rule ID	Rule Level	Rule Groups	Rule First Times	Journal Location	GeoIP Country	Count
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	104.21.40.220	snort	20101	6	ids	8	/var/log/suricata/fast.log	United States	2
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	104.21.40.220	snort	20101	6	ids	9	/var/log/suricata/fast.log	United States	2
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	104.21.40.220	snort	20101	6	ids	10	/var/log/suricata/fast.log	United States	2
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	104.21.40.220	snort	20101	6	ids	11	/var/log/suricata/fast.log	United States	2
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	104.21.40.220	snort	20101	6	ids	12	/var/log/suricata/fast.log	United States	2
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	104.21.40.220	snort	20100	8	fts	11	/var/log/suricata/fast.log	United States	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	104.21.40.220	snort	20100	8	ids	11	/var/log/suricata/fast.log	United States	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	104.21.40.220	snort	20152	10	ids	1	/var/log/suricata/fast.log	United States	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2	cal	192.168.0.101	54.39.128.230	snort	20101	6	ids	1	/var/log/suricata/fast.log	Canada	1

Рисунок 3.47 – Дашборд Alerts Table

– Alert Map: географічна мапа, що відображає місцезнаходження джерел атак/попереджень. Важливо зазначити, що такі геолокації визначається на основі IP-адреси джерела з'єднання шляхом зіставлення з GeoIP-базами (наприклад, GeoLite2) та має орієнтовний характер, оскільки реальне місцезнаходження атакувальника може бути приховане за допомогою VPN, проксі або інших засобів анонімізації.



Рисунок 3.48 – Дашборд Alert Map

До функціональних дашбордів можна віднести:

– SSH Failed Auth: Таблиця, спеціалізована для попереджень, пов'язаних з протоколом SSH.

MITRE technique	Alert Name	Source IP	Source P...	Agent Name	Fired Times	Category	Count
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33222	Ubuntu-Sinitsyna	1,232	authentication_failed	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33222	Ubuntu-Sinitsyna	1,232	invalid_login	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33222	Ubuntu-Sinitsyna	1,232	sshd	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33222	Ubuntu-Sinitsyna	1,232	syslog	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33222	Ubuntu-Sinitsyna	1,257	authentication_failed	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33222	Ubuntu-Sinitsyna	1,257	invalid_login	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33222	Ubuntu-Sinitsyna	1,257	sshd	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33222	Ubuntu-Sinitsyna	1,257	syslog	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33236	Ubuntu-Sinitsyna	1,233	authentication_failed	1
Password Guessing	sshd: Attempt to login usi	192.168.0.108	33236	Ubuntu-Sinitsyna	1,233	invalid_login	1

Рисунок 3.49 – Дашборд SSH Failed Auth

– DHCP attacks table: Таблиця, що містить деталізовані записи про потенційні атаки DHCP.

Time	Source IP	Destination IP	Source Country	Rule Description	Rule Level: Descen...	Rule Fired Times	Agent Name	Count
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.103.62.83	54.7.215.40	South Korea	IDS event.	6	6,529	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.131.52.90	36.121.45.360	Australia	IDS event.	6	728	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.133.140.120	218.8.229.430	Australia	IDS event.	6	6,824	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.163.16.48	187.192.191.780	Taiwan	IDS event.	6	6,173	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.2.56.48	219.98.26.870	China	IDS event.	6	809	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.37.48.26	73.39.50.730	Philippines	IDS event.	6	6,237	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.51.189.68	18.99.82.240	China	IDS event.	6	6,136	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.58.123.124	202.249.104.1240	China	IDS event.	6	7,157	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.6.158.1	253.109.39.420	India	IDS event.	6	6,463	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.81.76.18	8.124.191.390	China	IDS event.	6	594	csi	1

Рисунок 3.50 – Дашборд DHCP attacks table

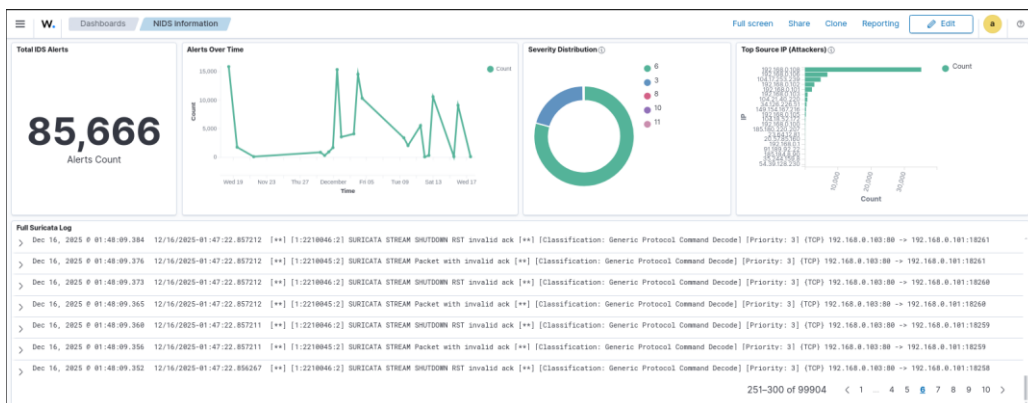


Рисунок 3.51 – Фінальний вигляд панелі NIDS Information

Timestamp	Host	Agent IP	Source IP	Decoder Name	Rule ID	Rule Level	Rule Groups	Rule Fired Times	Journal Location	GeoIP Country	Count
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.100	69.173.144.138	snort	20101	6	ids	165	/var/log/suricata/ftlog	Germany	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.100	185.125.190.96	snort	20100	8	fts	1	/var/log/suricata/ftlog	United Kingdom	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.100	185.125.190.96	snort	20100	8	ids	1	/var/log/suricata/ftlog	United Kingdom	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.103	69.173.144.165	snort	20100	8	fts	1	/var/log/suricata/ftlog	Germany	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.103	69.173.144.165	snort	20100	8	ids	1	/var/log/suricata/ftlog	Germany	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.103	69.173.144.165	snort	20101	6	ids	12	/var/log/suricata/ftlog	Germany	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.104	149.154.167.41	snort	20100	8	fts	5	/var/log/suricata/ftlog	United Kingdom	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.104	149.154.167.41	snort	20100	8	fts	6	/var/log/suricata/ftlog	United Kingdom	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.104	149.154.167.41	snort	20100	8	ids	5	/var/log/suricata/ftlog	United Kingdom	1
Dec 1, 2025 @ 00:00:00.000 to Dec 12, 2025	csi	192.168.0.104	149.154.167.41	snort	20100	8	ids	6	/var/log/suricata/ftlog	United Kingdom	1

MITRE technique	Alert Name	Source IP	Source P...	Agent Name	Fired Times	Category	Count
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33222	Ubuntu-Sintyna	1,257	authentication_failed	1
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33222	Ubuntu-Sintyna	1,232	invalid_login	1
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33222	Ubuntu-Sintyna	1,232	snort	1
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33222	Ubuntu-Sintyna	1,232	snort	1
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33222	Ubuntu-Sintyna	1,257	authentication_failed	1
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33222	Ubuntu-Sintyna	1,257	invalid_login	1
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33222	Ubuntu-Sintyna	1,257	snort	1
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33222	Ubuntu-Sintyna	1,257	snort	1
Password Guessing	snort: Attempt to login using a non	192.168.0.108	33226	Ubuntu-Sintyna	1,233	authentication_failed	1

Рисунок 3.52 – Фінальний вигляд панелі NIDS Information (продовження)

3.4. Проведення тестових атак та аналіз фіксації подій

Тестування розробленого лабораторного стенду здійснювалося відповідно до запропонованої методології мережної цифрової криміналістики для Linux-середовищ. Метою тестування було перевірити коректність фіксації мережних подій, повноту збору артефактів, узгодженість логів між різними джерелами та здатність інструментів відтворити хронологію інциденту.

Процедура тестування складалася з двох фаз, що відображають загальну структуру розробленої методики:

1. Фаза збору мережних доказів,
2. Фаза аналізу й кореляції зібраних артефактів.

Виконаємо декілька контрольованих атак, що охоплюють основні типи мережних загроз для Linux-середовищ.

1. Port and service scanning

Для моделювання інциденту використано активне сканування мережі з використанням Nmap. Застосовано SYN-сканування (-sS), що є одним із найпоширеніших методів прихованого виявлення відкритих TCP-портів, а також опцію -A, яка вмикає поглиблену ідентифікацію сервісів, ОС та трасування маршруту. Такий тип активності характерний для етапу розвідки хакера і часто передує більш цільовим атакам на конкретні мережні сервіси. [15]

Команда: *sudo nmap -sS -A 192.168.0.106/24*



```
(kali@kali)~$ sudo nmap -sS -A 192.168.0.106/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-11 18:37 EET
Nmap scan report for 192.168.0.1
Host is up (0.0027s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.0 (protocol 2.0)
|_ ssh-hostkeys
|_ 1024 bd:c8:c2:61:51:63:4c:77:bd:3f:6c:de:27:2d:ce:27 (DSA)
53/tcp    open  domain   PowerDNS Recursor 5.2.4
|_ dns-nsid:
|_ NSID: fttbdsnslv3 (66747462646e7331c767633)
|_ id.server: fttbdsnslv3
|_ bind.version: PowerDNS Recursor 5.2.4 (built Jul 17 2025 12:22:40 by root@localhost)
80/tcp    open  http     TP-LINK router http config
|_ http-title: Opening...
1900/tcp  open  upnp     TP-LINK router upnpd
MAC Address: 40:E0:00:00:57:54 (TP-Link Limited)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: printer|WAP|storage-misc|general purpose|switch|PBX
Running (JUST GUESSING): Canon embedded (97%), Mercusys embedded (97%), IBM embedded (92%), Fujitsu Siemens embedded (92%), Wind River VxWorks (92%), Avaya embedded (92%), Nortel embedded (88%), Xerox embedded (88%)
OS CPE: cpe:/h:canon:imageRUNNER_C5185 cpe:/h:mercusys:ac12g cpe:/h:ibm:dc59900 cpe:/h:fujitsu:externus_dx80 cpe:/o:windriver:vxworks cpe:/h:avaya:4526gtx cpe:/h:nortel:cs1000m cpe:/h:xerox:phaser_8560dt
Aggressive OS guesses: Canon imageRUNNER C5185 printer or Mercusys AC12G WAP (97%), Canon imageRUNNER C2380 or C2880i or Xerox Phaser 8860MFP printer (92%), Fujitsu Externus DX80 or IBM DCS9900 NAS device (92%), VxWorks (92%), Avaya 4526GTX switch (92%), Nortel CS1000M VoIP PBX or Xerox Phaser 8560DT printer (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE
HOP RTT ADDRESS
1 2.70 ms 192.168.0.1
```

Рисунок 3.53 – Робота команди сканування мережі

Система Suricata NIDS зафіксувала характерні ознаки активного сканування: велика кількість SYN-пакетів до різних TCP-портів, спроби

визначення версій служб, а також пакети, що відповідають поведінці OS Detection. У журналі Suricata можуть фіксуватись алерти з категорій "Nmap Scan", "Attempted Information Leak", "OS fingerprinting".

Це підтверджує, що система коректно розпізнала нетипову для звичайного трафіку активність, яка потенційно свідчить про початковий етап атаки.

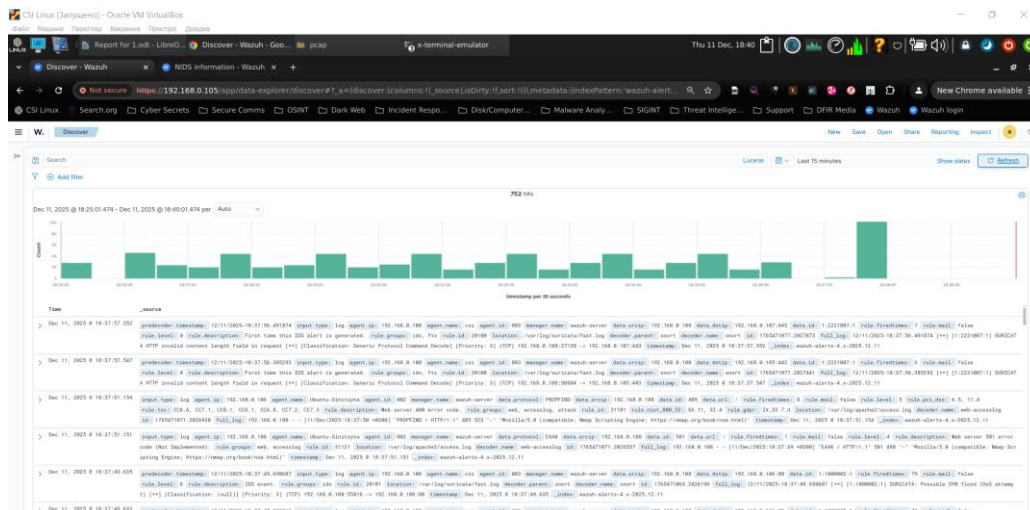


Рисунок 3.54 – Відображення зафіксованих попереджень



Рисунок 3.55 – Відображення зафіксованих попереджень

На захопленому фрагменті трафіку спостерігається домінування TCP-пакетів (понад 96 % від загальної кількості), що є типовим для процедур сканування портів і сервісів. Наявність значної кількості короткоживучих TCP-з'єднань, а також різноманіття протоколів прикладного рівня (зокрема SSH, HTTP, SMB) свідчить про етап мережної розвідки, під час якого здійснюється виявлення активних хостів, відкритих портів і доступних сервісів. [24]

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	3900	100.0	1199041	572 k	0	0	0
Ethernet II	100.0	3900	100.0	1199041	572 k	0	0	0
Internet Protocol Version 4	99.7	3888	6.5	77760	37 k	0	0	0
User Datagram Protocol	1.4	56	0.0	448	213	0	0	0
NetBIOS Name Service	0.5	20	0.0	514	245	0	0	0
Domain Name System	0.5	20	0.2	2104	1004	20	2104	1004
Data	0.8	30	0.6	6919	3302	30	6919	3302
Transmission Control Protocol	96.4	3761	87.0	1042999	497 k	2767	671718	320 k
Transport Layer Security	18.3	714	87.0	802782	383 k	711	755592	360 k
SSH Protocol	3.1	119	2.5	29444	14 k	119	29444	14 k
Remote Procedure Call	0.1	2	0.1	732	349	0	0	0
Portmap	0.1	2	0.1	660	315	2	660	315
NetBIOS Session Service	0.3	11	0.2	1855	885	2	198	94
SMB2 (Server Message Block Protocol version 2)	0.1	4	0.1	1376	656	4	1376	656
SMB (Server Message Block Protocol)	0.1	5	0.0	245	116	5	245	116
Hypertext Transfer Protocol	2.8	110	6.3	75621	36 k	87	15121	7218
Line-based text data	0.5	21	4.7	55917	26 k	21	77751	27 k
HTML Form URL Encoded	0.1	2	0.0	176	84	2	176	84
Domain Name System	0.1	4	0.0	272	129	4	272	129
Data	0.9	37	0.0	11945	5702	37	11945	5702
Internet Control Message Protocol	1.8	71	1.0	11412	5447	56	8068	3851
NetBIOS Name Service	0.1	2	0.0	100	47	2	100	47
Data	0.3	13	2.704	2704	1290	13	2704	1290
Address Resolution Protocol	0.3	12	0.0	498	237	12	498	237

Рисунок 3.56 – Ієрархія протоколів захопленого трафіку

У подальшому аналізі захопленого PCAP-трафіку простежується характерний для агресивного сканування Nmap комплексний шаблон активності. Хост 192.168.0.108 надсилає масові SYN-запити до різних адрес і портів без завершення TCP-рукоштовування, одночасно фіксуються TCP Retransmission, повторне використання портів, нестандартні TCP-флаги, зміни у вікні, MSS та TCP-опціях, що відповідає stealth-скануванню.

No.	Time	Source	Destination	Protocol	Length	Info
35	2025-12-11 18:37:38.187000	192.168.0.108	192.168.0.1	TCP	60	43677 → 22 [SYN] Seq=43677 Win=0 Len=0 MSS=1460 WS=1 TSV=1204967295 TSecr=0
36	2025-12-11 18:37:38.187000	192.168.0.108	192.168.0.1	TCP	60	43677 → 22 [SYN] Seq=43677 Win=0 Len=0 MSS=1460 WS=1 TSV=1204967295 TSecr=0
37	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
38	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
39	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	60	43678 → 22 [SYN] Seq=43678 Win=0 Len=0 MSS=1460 WS=1 TSV=1204967295 TSecr=0
40	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
41	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
42	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
43	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
44	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
45	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
46	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
47	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
48	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
49	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
50	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
51	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
52	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
53	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
54	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
55	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
56	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
57	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
58	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
59	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
60	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
61	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0
62	2025-12-11 18:37:38.286800	192.168.0.108	192.168.0.1	TCP	74	43678 → 22 [SYN] Seq=0 Win=63 Len=0 MSS=1460 WS=1 SACK_PERM=1 TSV=1204967295 TSecr=0

Рисунок 3.57 – Захоплений трафік за допомогою скрипта (уривок)

Для підтвердження ідентифікації початкової фази TCP-сеансів застосуємо фільтр `tcp.flags.syn == 1 and tcp.flags.ack == 0`. [24]

No.	Time	Source	Destination	Protocol	Length	Info
942	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
943	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
944	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
945	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
946	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
947	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
948	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
949	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
950	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
951	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
952	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
953	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
954	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
955	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
956	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
957	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
958	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
959	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
960	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
961	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
962	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
963	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
964	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
965	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
966	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
967	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
968	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
969	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128
970	2025-12-11 18:37:48.423800	192.168.0.108	192.168.0.1	TCP	74	57784 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM=1 TSV=1208811874 TSecr=0 WS=128

Паралельно спостерігається мультипротокольна активність, притаманна режиму -А: численні ICMP Echo-request/Echo-reply та ICMP “Destination unreachable”, що свідчать про виявлення активних/неактивних хостів; UDP-пакети на службові порти, які відповідають UDP-скануванню. Така сумарна мережна поведінка підтверджує, що Nmap здійснює повний аудит мережі з одночасною перевіркою доступності хостів, відкритості портів та збиранням системно-ідентифікаційних артефактів.

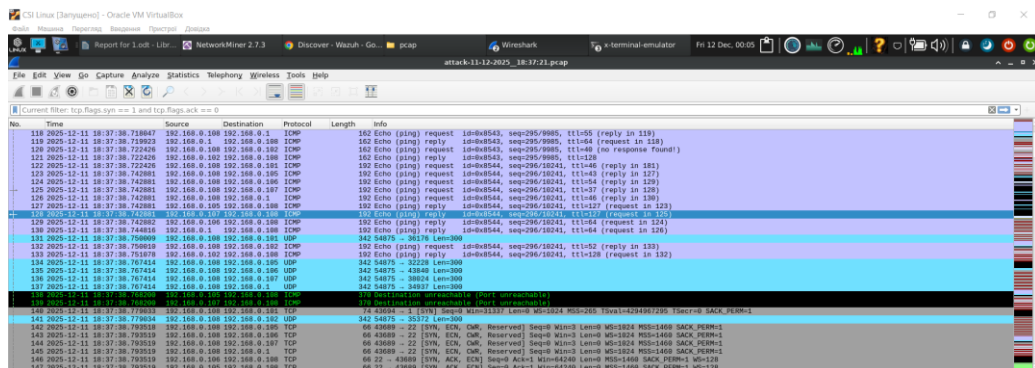


Рисунок 3.59 – Захоплений трафік за допомогою скрипта (уривок)

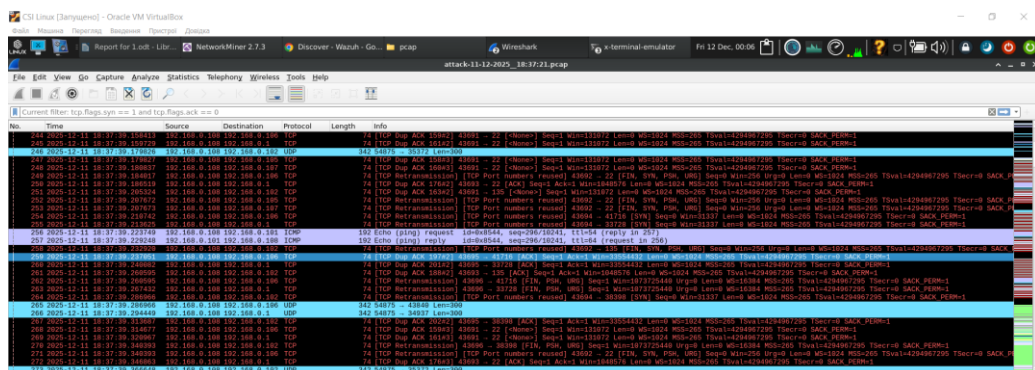


Рисунок 3.60 – Захоплений трафік за допомогою скрипта (уривок)

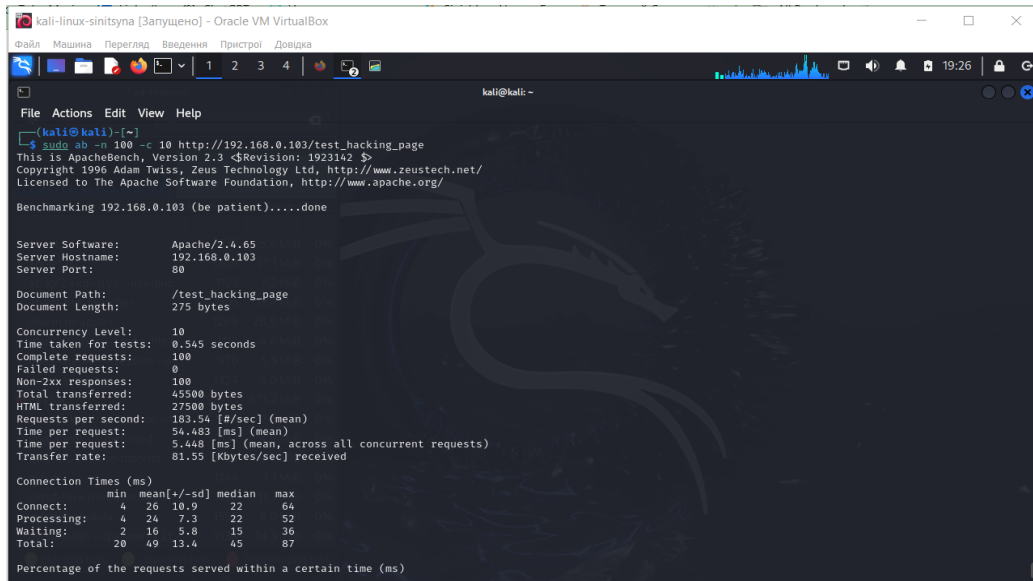
2. Web Reconnaissance

Для моделювання інциденту використано атаку типу HTTP-розвідки веб-ресурсів (Web Reconnaissance), яка полягає у надсиланні серії HTTP GET-запитів до веб-сервера Apache з метою виявлення доступних або неіснуючих веб-ресурсів, аналізу структури URI та оцінювання конфігурації веб-сервісу. Така активність може передувати подальшим цілеспрямованим атакам на веб-додатки та використовується зловмисниками на етапі розвідки.

Команди:

sudo ab -n 100 -c 10 http://192.168.0.103/test_hacking_page, ^[17]

sudo curl http://192.168.0.103/test_hacking_page ^[20]



```
kali@kali:~$ sudo ab -n 100 -c 10 http://192.168.0.103/test_hacking_page
This is ApacheBench, Version 2.3 <$Revision: 1923142 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.0.103 (be patient).....done

Server Software:      Apache/2.4.65
Server Hostname:      192.168.0.103
Server Port:          80

Document Path:        /test_hacking_page
Document Length:       275 bytes

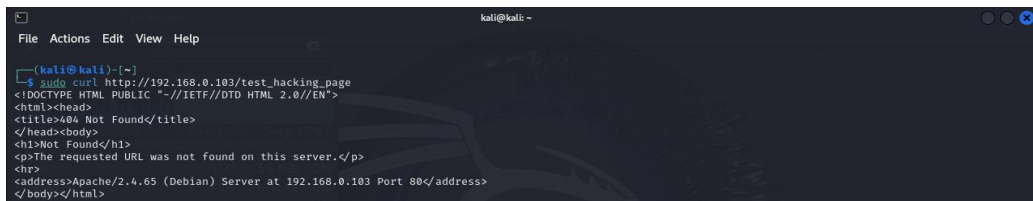
Concurrency Level:     10
Time taken for tests:   0.545 seconds
Complete requests:     100
Failed requests:        0
Non-2xx responses:     100
Total transferred:     45500 bytes
HTML transferred:      27500 bytes
Requests per second:   183.54 [#/sec] (mean)
Time per request:      54.483 [ms] (mean)
Time per request:      5.448 [ms] (mean, across all concurrent requests)
Transfer rate:          81.55 [Kbytes/sec] received

Connection Times (ms)
  min  mean[+/-sd] median   max
Connect:  4    26  10.9   22    64
Processing:  4    24   7.3   22    52
Waiting:  2    16   5.8   15    36
Total:    20    49  13.4   45    87

Percentage of the requests served within a certain time (ms)
```

Рисунок 3.61 – Робота команди веб-розвідки

ab -n 100 -c 10 http://192.168.0.103/test_hacking_page



```
kali@kali:~$ sudo curl http://192.168.0.103/test_hacking_page
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr/>
<address>Apache/2.4.65 (Debian) Server at 192.168.0.103 Port 80</address>
</body></html>
```

Рисунок 3.62 – Робота команди веб-розвідки

curl http://192.168.0.103/test_hacking_page

Під час атаки у вкладці Discover відображено понад 3300 подій, корельованих із логів Suricata та Apache. Події класифіковані як підозрілі та мають підвищений рівень серйозності.

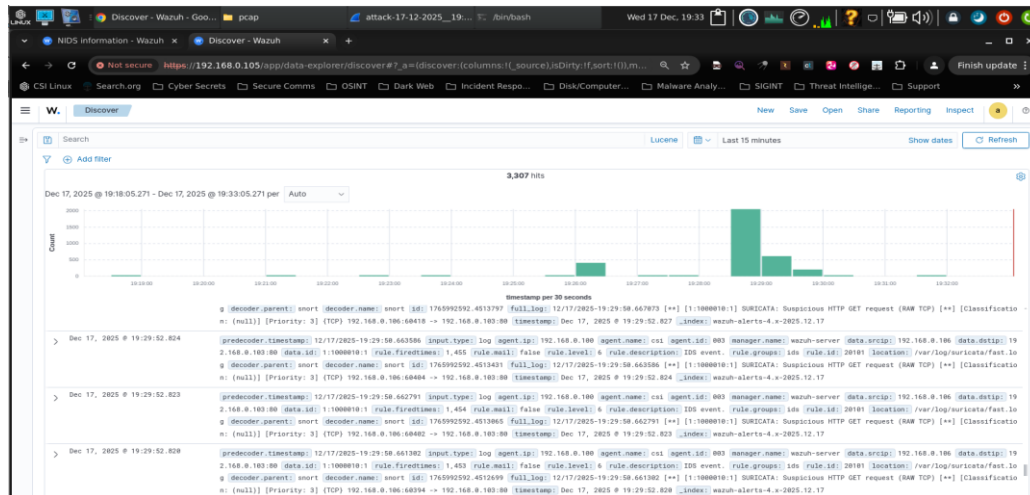


Рисунок 3.63 – Перегляд подій у режимі реального часу Discover

На дашборді Suricata було зафіксовано 1688 IDS-алертів за короткий проміжок часу. Графік демонструє різкий сплеск активності приблизно о 19:29, що відповідає моменту запуску команди. Основним спрацьованим правилом Suricata є “Suspicious HTTP GET request”, що вказує на аномальну частоту однотипних HTTP-запитів.

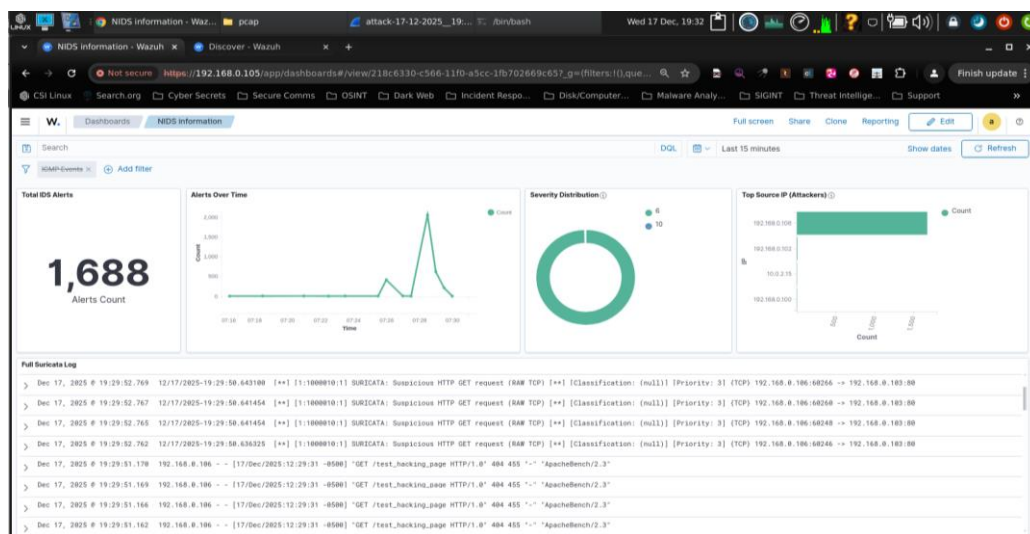


Рисунок 3.64 – Перегляд корельованих подій на панелі NIDS information

У захопленому трафіку також спостерігається велика кількість однотипних HTTP-запитів виду *GET /test_hacking_page*, що надходили за короткий проміжок часу. Повторюваність запитів та висока частота є характерною ознакою

автоматизованого навантаження. Часті TCP-прапорці SYN, ACK та FIN вказують на створення й негайне завершення великої кількості з'єднань.^[24]

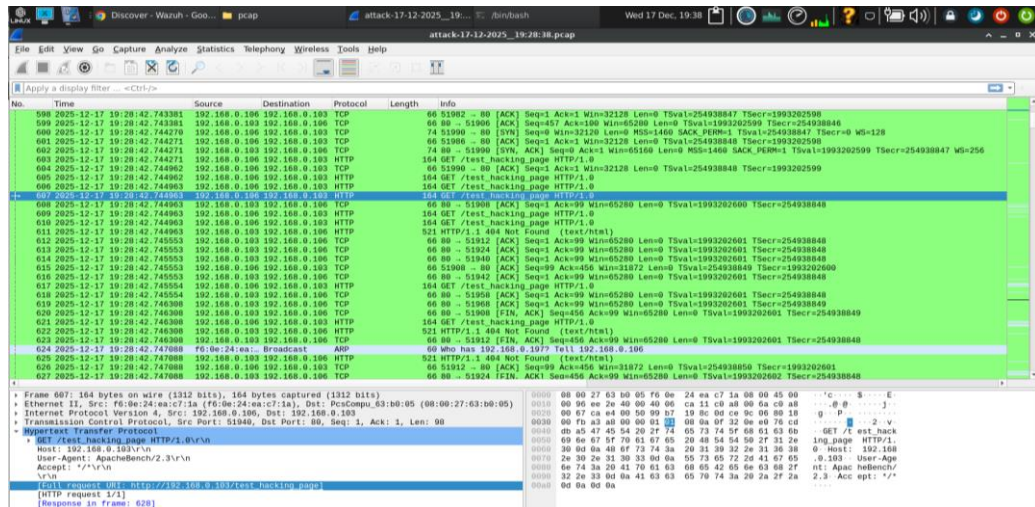


Рисунок 3.65 – Перегляд захопленого трафіку у rсар-файлі

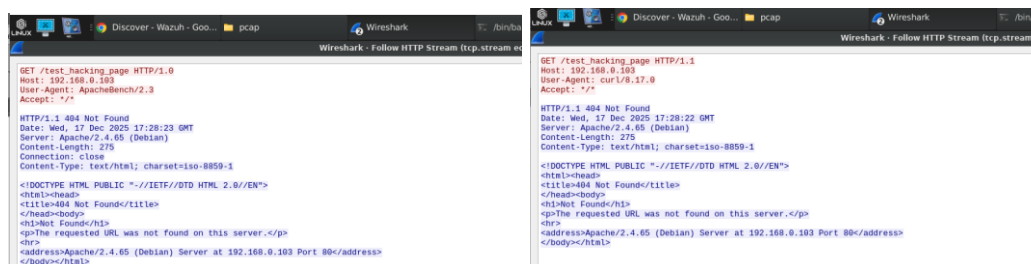
За допомогою Network Miner можна побачити деталізовану інформацію про клієнтські інструменти, зафіксовані у трафіку. Виявлено використання двох різних утиліт: стандартного HTTP-клієнта *curl/8.17.0* та спеціалізованого інструмента навантажувального тестування *ApacheBench/2.3*. Сервер у відповідь повертає код *404 Not Found*, що свідчить про звернення до неіснуючого ресурсу */test_hacking_page*.

File	Tools	Help
Hosts (12)	Files (1303)	Images
Filter keyword: curl	Messages	Credentials
Parameter name	Parameter value	Frame number
GET	/test_hacking_page	295
Host	192.168.0.103	295
User-Agent	curl/8.17.0	295
HTTP Response Status Code	404 Not Found	297
Date	Wed, 17 Dec 2025 17:28:23 GMT	297
Server	Apache/2.4.65 (Debian)	297
Content-Length	275	297
Content-Type	text/html; charset=iso-8859-1	297
NetBIOS Query	*c00*	335
NetBIOS Query	*c00*	337
NetBIOS Query	*c00*	338
NetBIOS Query	*c00*	342
NetBIOS Query	*c00*	345
NetBIOS Query	*c00*	349
GET	/test_hacking_page	567
Host	192.168.0.103	567
User-Agent	ApacheBench/2.3	567
HTTP Response Status Code	404 Not Found	569
Date	Wed, 17 Dec 2025 17:28:23 GMT	569
Server	Apache/2.4.65 (Debian)	569
Content-Length	275	569
Content-Type	text/html; charset=iso-8859-1	569
GET	/test_hacking_page	603

Рисунок 3.66 – Перелік витягнутих HTTP-запитів з одночасним використанням запитів

При перегляді потоків Follow HTTP Stream у Wireshark відображено повний діалог між клієнтом і сервером у текстовому вигляді. Сервер ідентифікує

себе як *Apache/2.4.65 (Debian)* та повертає повідомлення про відсутність запитаного ресурсу.



А

Б

Рисунок 3.67 – Реконструкція HTTP-потіку для А – ApacheBench, Б – curl

На прикладі окремого пакета розглянемо структуру HTTP-запиту. У заголовках чітко видно адресу сервера призначення 192.168.0.103 та ідентифікатор клієнта *User-Agent: curl*, що дозволяє однозначно атрибутувати джерело активності та використаний інструмент.

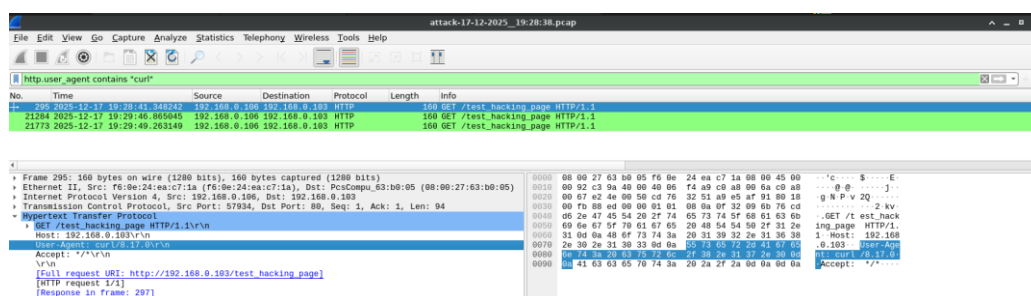


Рисунок 3.68 – Аналіз структури окремого HTTP-пакета

Таку ж саму фільтрацію можна виконати за допомогою інструмента Network Miner за ключовими словами. Результати аналізу підтверджують, що HTTP-запити надсилалися з хоста 192.168.0.106 (Kali Linux) на веб-сервер 192.168.0.103. Наявність параметрів *User-Agent: curl/8.17.0* та *ApacheBench/2.3* однозначно вказує на використання утиліт для формування HTTP-запитів, що свідчить про автоматизований характер звернень.

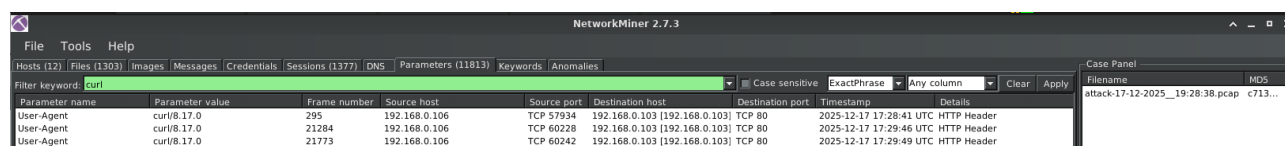


Рисунок 3.69 – Фільтрація за ключовим словом *curl*

Parameter name	Parameter value	Frame number	Source host	Source port	Destination host	Destination port	Timestamp	Details
User-Agent	ApacheBench/2.3	567	192.168.0.106	TCP 51906	192.168.0.103 [192.168.0.103]	TCP 80	2025-12-17 17:28:42 UTC	HTTP Header
User-Agent	ApacheBench/2.3	603	192.168.0.106	TCP 51908	192.168.0.103 [192.168.0.103]	TCP 80	2025-12-17 17:28:42 UTC	HTTP Header
User-Agent	ApacheBench/2.3	605	192.168.0.106	TCP 51912	192.168.0.103 [192.168.0.103]	TCP 80	2025-12-17 17:28:42 UTC	HTTP Header
User-Agent	ApacheBench/2.3	606	192.168.0.106	TCP 51924	192.168.0.103 [192.168.0.103]	TCP 80	2025-12-17 17:28:42 UTC	HTTP Header
User-Agent	ApacheBench/2.3	607	192.168.0.106	TCP 51940	192.168.0.103 [192.168.0.103]	TCP 80	2025-12-17 17:28:42 UTC	HTTP Header
User-Agent	ApacheBench/2.3	609	192.168.0.106	TCP 51942	192.168.0.103 [192.168.0.103]	TCP 80	2025-12-17 17:28:42 UTC	HTTP Header
User-Agent	ApacheBench/2.3	610	192.168.0.106	TCP 51958	192.168.0.103 [192.168.0.103]	TCP 80	2025-12-17 17:28:42 UTC	HTTP Header

Рисунок 3.70 – Фільтрація за ключовим словом *bench*

Панель MITRE ATT&CK узагальнює результати аналізу та відносить зафіксовану активність до тактик Reconnaissance і техніки T1595.002 – Vulnerability Scanning. Рівень критичності підвищено до 10, що обумовлено великою кількістю автоматизованих запитів до неіснуючих ресурсів. Це свідчить про ефективну кореляцію подій та правильну інтерпретацію атаки системою моніторингу. [22]

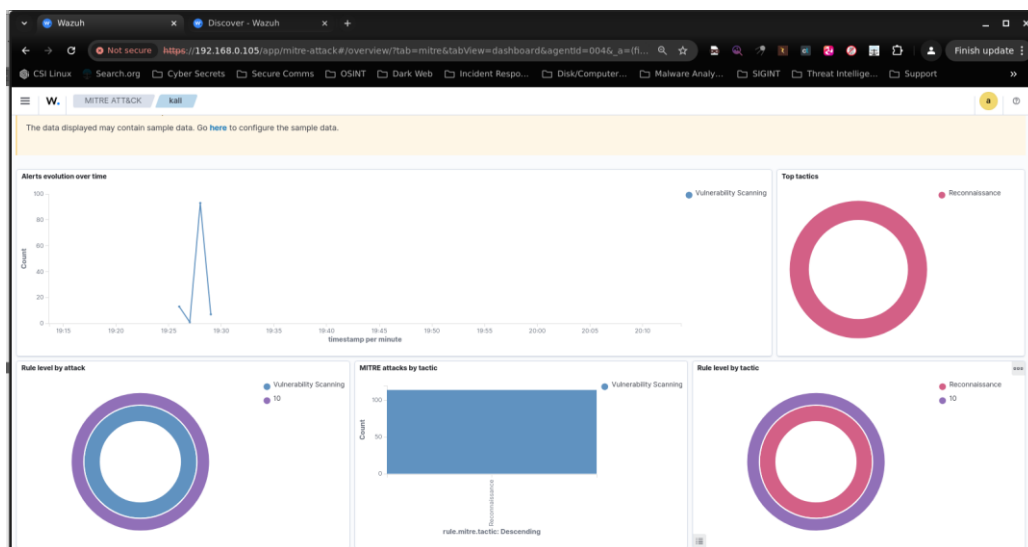


Рисунок 3.71 – Візуалізація тактик за фреймворком MITRE ATT&CK

3. Hydra SSH brute-force

Для моделювання інциденту використаємо атаку типу brute-force на сервіс SSH за допомогою інструмента Hydra. Попередньо було сформовано власний список імен потенційних користувачів, характерних для цільової системи, який збережено у файлі `/usr/share/wordlists/user_list.txt`. На основі цього файлу, разом зі словником паролів `rockyou.txt`, утворено комбінації логін-пароль, що використовувалися під час автоматизованої спроби підбору облікових даних.

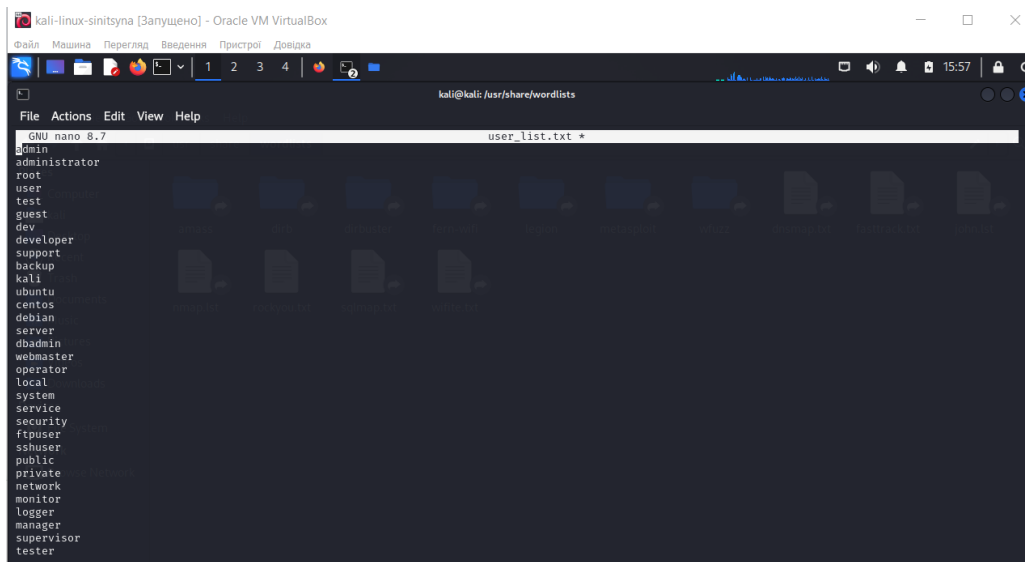


Рисунок 3.72 – Список користувачів для атаки `/usr/share/wordlists/user_list.txt`

Команда для проведення атаки:

```
hydra -L /usr/share/wordlists/user_list.txt -P /usr/share/wordlists/rockyou.txt
192.168.0.106 ssh -t 4[16]
```

У ході атаки Hydra здійснює автоматизований перебір облікових даних, комбінуючи список потенційних імен користувачів (-L) зі словником паролівних комбінацій (-P). Протокол SSH обрано як ціль через його високу критичність і поширеність, а параметр -t 4 забезпечує багатопотоковість, збільшуючи інтенсивність та швидкість підбору.

Подібна активність характерна для етапу спроби несанкціонованого доступу, під час якого зловмисник намагається знайти валідну пару логін–пароль для подальшого проникнення в систему.

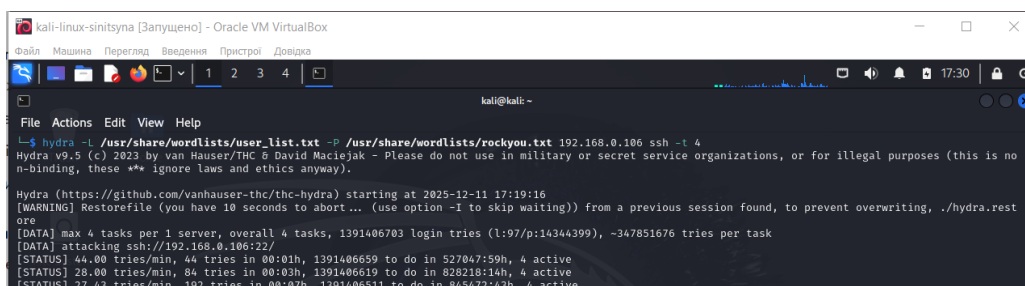


Рисунок 3.73 – Робота команди перебору облікових даних

Скріншот із платформи Wazuh демонструє активну підозрілу діяльність, пов'язану зі спробами входу на цільовий хост через SSH. Події зафіксовані приблизно між 18:30 та 18:55, а загальна кількість отриманих тригерів за останні 35 хвилин становить 144, що вказує на дуже високу інтенсивність спроб.

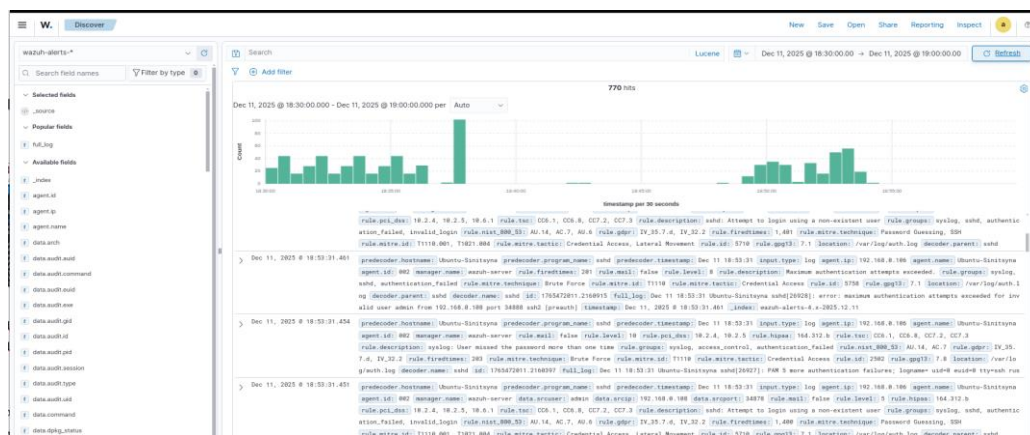


Рисунок 3.74 – Відображення зафіксованих попереджень

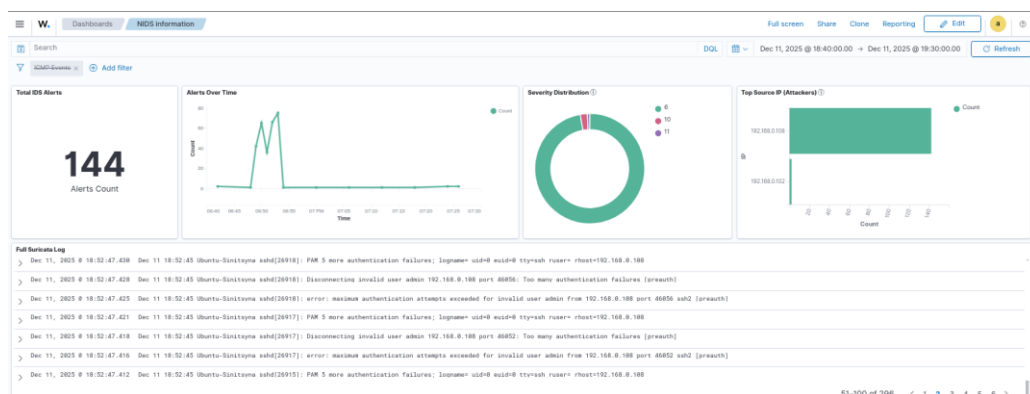


Рисунок 3.75 – Узагальнення попереджень на дашбордах

Додатково можна зазначити, що спершу система класифікувала активність як *Password Guessing, SSH*, що відповідає початковим поодиноким спробам підбору пароля. Однак із наростанням інтенсивності та частоти запитів Wazuh автоматично підвищив рівень загрози й пере класифікував подію як *Brute Force*, що чітко вказує на перехід до масового автоматизованого перебору облікових даних.

MITRE technique	Alert Name	Source IP	Source P...	Agent Name	Fired Times	Category	Count
SSH	sshd: Attempt to login using a non-existent user	192.168.0.108	36344	Ubuntu-SinitSYna	1,146	authentication_failed	1
SSH	sshd: Attempt to login using a non-existent user	192.168.0.108	36344	Ubuntu-SinitSYna	1,146	invalid_login	1
SSH	sshd: Attempt to login using a non-existent user	192.168.0.108	36344	Ubuntu-SinitSYna	1,146	sshd	1
SSH	sshd: Attempt to login using a non-existent user	192.168.0.108	36344	Ubuntu-SinitSYna	1,146	syslog	1

Рисунок 3.76 – Узагальнення попереджень на дашбордах по техніці SSH

MITRE technique	Alert Name	Source IP	Source P...	Agent Name	Fired Times	Category	Count
Password Guessing	sshd: Attempt to login using a non-existent user	192.168.0.108	33222	Ubuntu-Sinitsyna	1,232	authentication_failed	1
Password Guessing	sshd: Attempt to login using a non-existent user	192.168.0.108	33222	Ubuntu-Sinitsyna	1,232	invalid_login	1
Password Guessing	sshd: Attempt to login using a non-existent user	192.168.0.108	33222	Ubuntu-Sinitsyna	1,232	sshd	1
Password Guessing	sshd: Attempt to login using a non-existent user	192.168.0.108	33222	Ubuntu-Sinitsyna	1,232	syslog	1
Password Guessing	sshd: Attempt to login using a non-existent user	192.168.0.108	33222	Ubuntu-Sinitsyna	1,257	authentication_failed	1

Рисунок 3.77 – Узагальнення попереджень на дашбордах по техніці *Password Guessing*

MITRE technique	Alert Name	Source IP	Source P...	Agent Name	Fired Times	Category	Count
Brute Force	sshd: brute force trying to get at	192.168.0.108	36366	Ubuntu-Sinitsyna	165	authentication_failures	1
Brute Force	sshd: brute force trying to get at	192.168.0.108	36366	Ubuntu-Sinitsyna	165	sshd	1
Brute Force	sshd: brute force trying to get at	192.168.0.108	36366	Ubuntu-Sinitsyna	165	syslog	1
Brute Force	sshd: brute force trying to get at	192.168.0.108	50362	Ubuntu-Sinitsyna	177	authentication_failures	1
Brute Force	sshd: brute force trying to get at	192.168.0.108	50362	Ubuntu-Sinitsyna	177	sshd	1
Brute Force	sshd: brute force trying to get at	192.168.0.108	50362	Ubuntu-Sinitsyna	177	syslog	1

Рисунок 3.78 – Узагальнення попереджень на дашбордах по техніці *Brute Force*

З метою підтвердження інциденту переглянемо статистику протоколів у середовищі Wireshark. Як видно зі статистики протоколів, основну частку мережного трафіку становлять TCP-з'єднання з використанням протоколу SSH, що свідчить про інтенсивну взаємодію між відповідними вузлами мережі. Такий розподіл трафіку підтверджує цілеспрямований характер зафіксованої активності та узгоджується з інформацією про SSH-атаку, отриманою на попередньому етапі аналізу журналів безпеки. [24]

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	212	100.0	28871	13 k	0	0	0
Ethernet	100.0	212	10.3	2968	1437	0	0	0
Internet Protocol Version 4	100.0	212	14.7	4240	2053	0	0	0
User Datagram Protocol	1.9	4	0.1	32	15	0	0	0
NetBIOS Name Service	1.9	4	0.7	200	96	4	200	96
Transmission Control Protocol	84.9	180	67.9	19595	9488	86	2752	1332
SSH Protocol	17.0	36	8.3	2408	1166	36	2408	1166
Data	27.4	58	39.6	11427	5533	58	11427	5533
Internet Control Message Protocol	13.2	28	6.4	1836	889	26	1664	805
NetBIOS Name Service	0.9	2	0.3	100	48	2	100	48

Рисунок 3.79 – Статистика протоколів захопленого трафіку

Після фільтрації трафіку за протоколом SSH у часовому аналізі захоплених даних спостерігається значна кількість послідовних SSH-пакетів, що передаються від вузла 192.168.0.106 до 192.168.0.108 з мінімальними міжпакетними інтервалами, які вимірюються мілісекундами. Така поведінка є характерною ознакою автоматизованого перебору облікових даних. Зафіксовані

пакети містять зашифровані повідомлення типу *Client/Server: Encrypted packet*, що відповідає етапу автентифікації протоколу SSH.

У проміжку часу 18:53:21–18:53:25 спостерігається різке зростання інтенсивності SSH-трафіку, що вказує на серію повторюваних спроб авторизації. Виявлений шаблон мережної активності відповідає сценарію багатопотокової brute-force атаки та підтверджує використання автоматизованого інструмента підбору облікових даних.

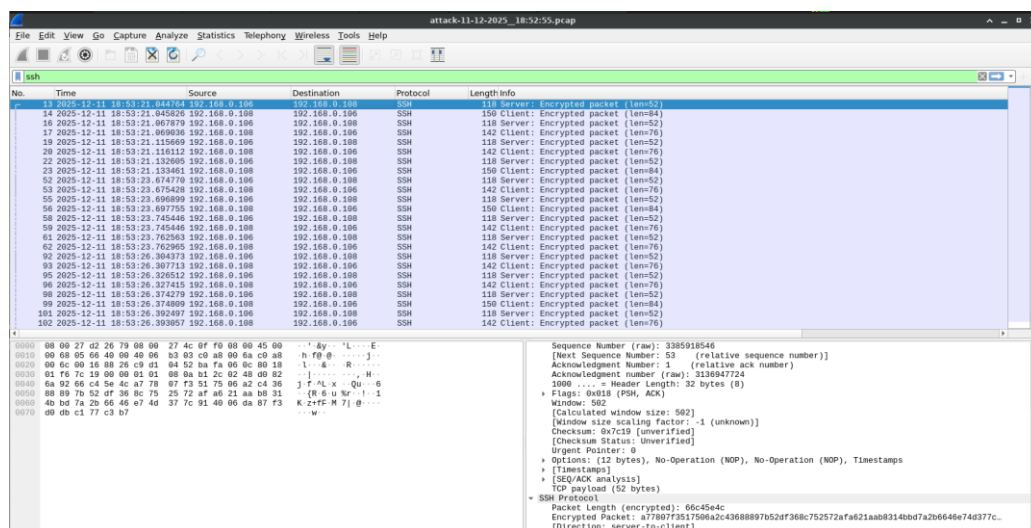


Рисунок 3.80 – Захоплений трафік за допомогою скрипта (уринок)

Під час аналізу статистики MITRE ATT&CK також можна побачити узагальнено підвищену активність, пов'язану зі спробами несанкціонованого доступу до сервісу SSH та підбору облікових даних. [22]

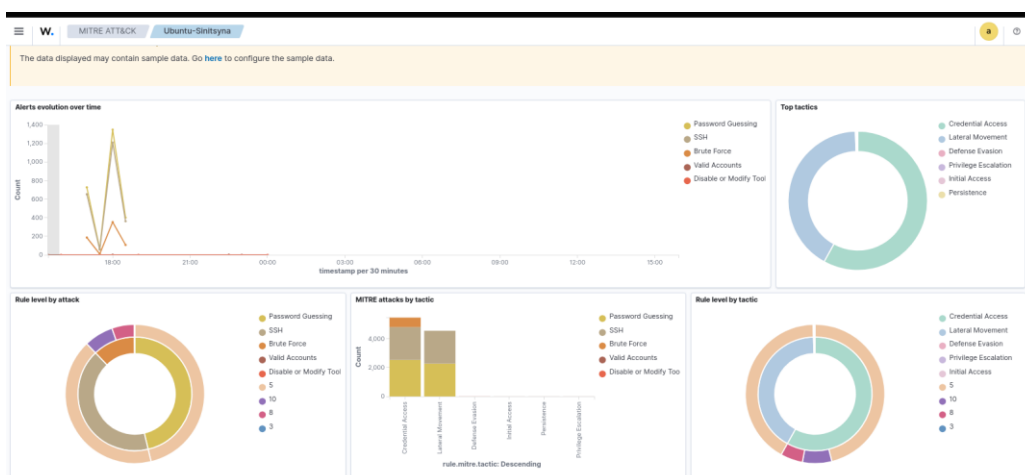


Рисунок 3.81 – Перегляд статистики MITRE ATT&CK

4. SYN Flood attack (DoS)

Для моделювання інциденту використано атаку типу SYN Flood, яка є формою відмови в обслуговуванні (Denial of Service). Ця атака спрямована на виснаження ресурсів цільового сервера шляхом переповнення його черги напіввідкритих (half-open) з'єднань TCP.

Команда: `sudo hping3 -S --flood -p 80 192.168.0.105`^[19]

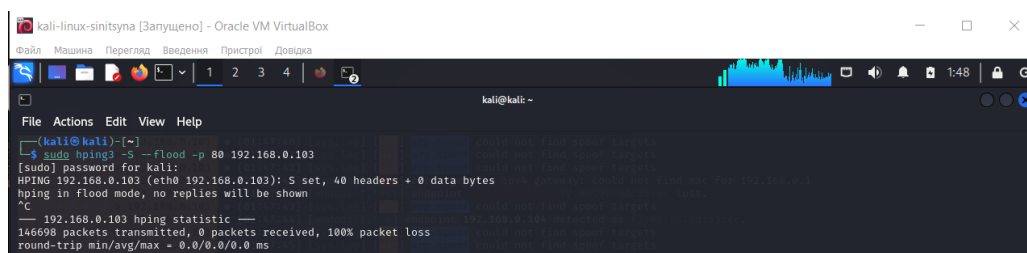


Рисунок 3.82 – Робота команди hping3 у режимі SYN Flood

Експериментальний стенд успішно зафіксував інцидент та згенерував численні попередження про аномальну мережеву активність. На дашборді спостерігається різкий сплеск кількості подій із високим рівнем серйозності, а журнали безпеки містять записи з класифікацією *Possible SYN Flood (DoS attempt)* та повідомлення про некоректні TCP-сесії. Отримані результати підтверджують здатність системи своєчасно виявляти SYN Flood-атаки та корелювати мережні події з ознаками відмови в обслуговуванні.

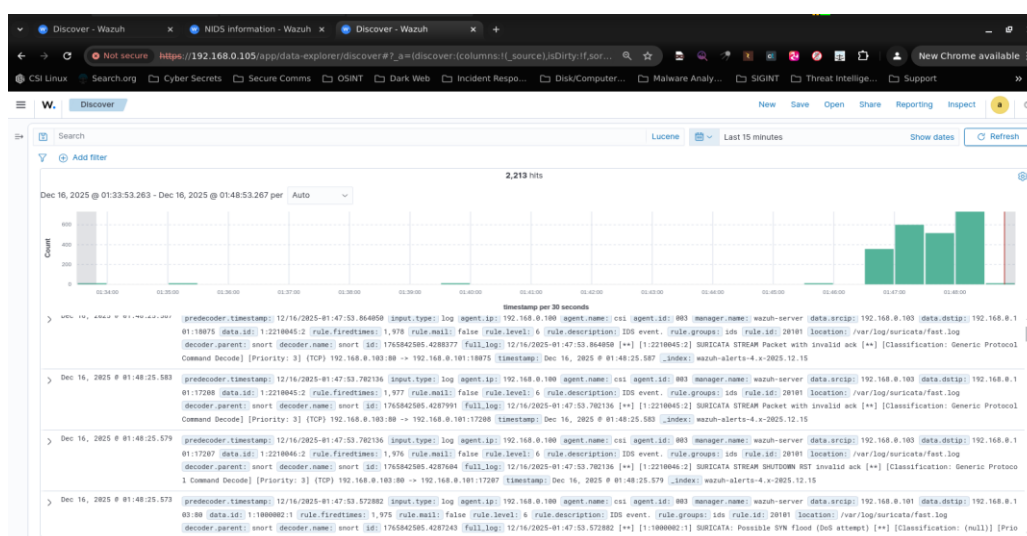


Рисунок 3.83 – Масова кількість сповіщень "Possible SYN flood" та всплеск попереджень у режимі реального часу

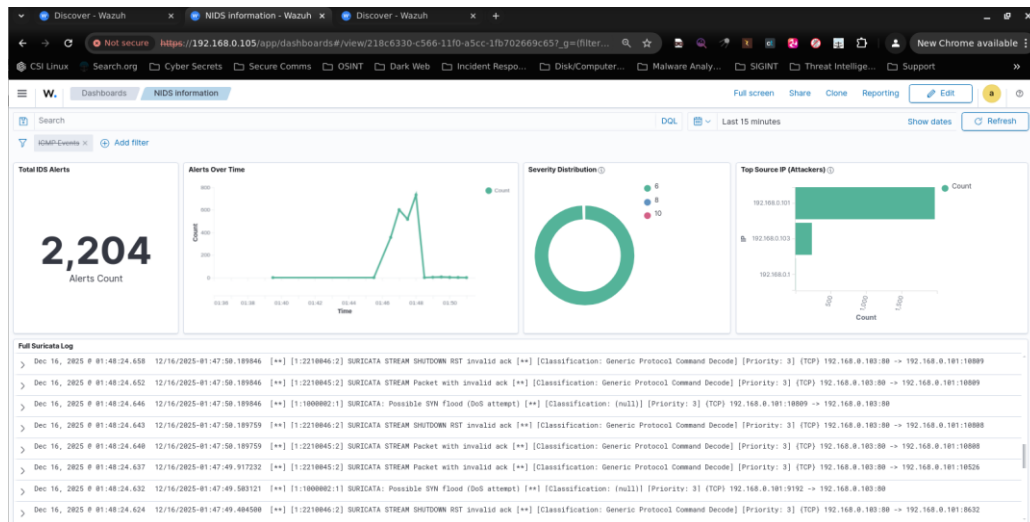


Рисунок 3.84 – Перегляд кореляції логів на дашбордах

Аналіз захопленого мережного трафіку у Wireshark підтвердив наявність характерних ознак SYN Flood-атаки. Статистика ієрархії протоколів демонструє, що 95.5% від усіх пакетів становлять пакети TCP.

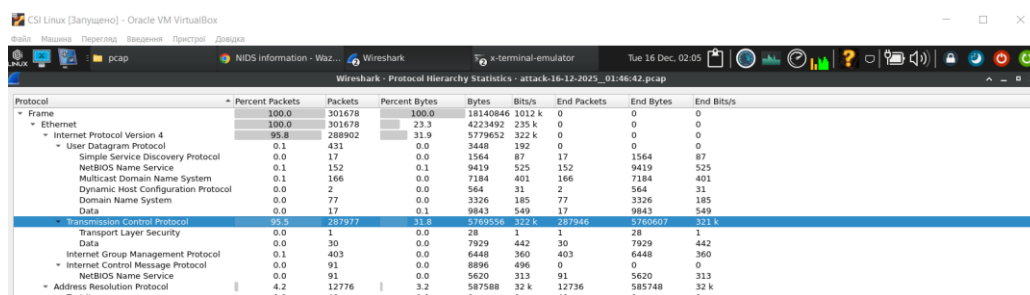


Рисунок 3.85 – Статистика протоколів захопленого трафіку у Wireshark

Перегляд таблиці TCP Conversations у Wireshark також підтверджує високу інтенсивність спроб встановлення зв'язку. Кожна розмова складається з лише 2 пакетів (1 від А до В і 1 від В до А, тобто SYN та SYN,ACK), що свідчить про незавершене з'єднання.

Ethernet - 18	IPv4 - 16	IPv6	TCP - 144152	UDP - 346
Address A	Port A	Address B	Port B	Packets
192.168.0.101	5397	192.168.0.103	80	2
192.168.0.101	5398	192.168.0.103	80	2
192.168.0.101	5399	192.168.0.103	80	2
192.168.0.101	5400	192.168.0.103	80	2
192.168.0.101	5401	192.168.0.103	80	2
192.168.0.101	5402	192.168.0.103	80	2
192.168.0.101	5403	192.168.0.103	80	2
192.168.0.101	5404	192.168.0.103	80	2
192.168.0.101	5405	192.168.0.103	80	2
192.168.0.101	5406	192.168.0.103	80	2
192.168.0.101	5407	192.168.0.103	80	2
192.168.0.101	5408	192.168.0.103	80	2
192.168.0.101	5409	192.168.0.103	80	2
192.168.0.101	5410	192.168.0.103	80	2
192.168.0.101	5411	192.168.0.103	80	2
192.168.0.101	5412	192.168.0.103	80	2
192.168.0.101	5413	192.168.0.103	80	2
192.168.0.101	5414	192.168.0.103	80	2
192.168.0.101	5415	192.168.0.103	80	2
192.168.0.101	5416	192.168.0.103	80	2
192.168.0.101	5417	192.168.0.103	80	2
192.168.0.101	5418	192.168.0.103	80	2
192.168.0.101	5419	192.168.0.103	80	2
192.168.0.101	5420	192.168.0.103	80	2
192.168.0.101	5421	192.168.0.103	80	2
192.168.0.101	5422	192.168.0.103	80	2
192.168.0.101	5423	192.168.0.103	80	2
192.168.0.101	5424	192.168.0.103	80	2
192.168.0.101	5425	192.168.0.103	80	2

Рисунок 3.86 – Таблиця TCP Conversations

Детальний перегляд пакетів показує значну кількість TCP-пакетів із прапором SYN, а також пакети RST та ACK, що виникають у відповідь на некоректні або незавершені з'єднання. Відсутність завершеного TCP-рукоштовування та велика кількість повторюваних SYN-запитів свідчать про штучно згенерований трафік, спрямований на вичерпання системних ресурсів сервера. [24]

No.	Time	Source	Destination	Protocol	Length	Info
100	2025-12-16 01:46:44.151742	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
101	2025-12-16 01:46:44.151742	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
108	2025-12-16 01:46:44.352283	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
109	2025-12-16 01:46:44.352283	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
110	2025-12-16 01:46:44.353091	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
111	2025-12-16 01:46:44.353092	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
112	2025-12-16 01:46:44.353092	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
113	2025-12-16 01:46:44.355259	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
114	2025-12-16 01:46:44.355259	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
115	2025-12-16 01:46:44.355260	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
116	2025-12-16 01:46:44.355260	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
117	2025-12-16 01:46:44.355260	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
118	2025-12-16 01:46:44.355260	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
119	2025-12-16 01:46:44.355260	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
120	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
121	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
122	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
123	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
124	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
125	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
126	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
127	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
128	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
129	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
130	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
131	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
132	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
133	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0
134	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [SYN] Seq=0, Win=0, Len=0
135	2025-12-16 01:46:44.357956	192.168.0.101	192.168.0.103	TCP	60	80 → 80 [RST, ACK] Seq=0, Win=0, Len=0

Рисунок 3.87 – Перегляд захоплених пакетів

5. MAC Flooding / DHCP Starvation

Для моделювання інциденту використано атаку DHCP Flood, яка спрямована на створення великої кількості фіктивних DHCP-запитів з метою

перевантаження DHCP-сервера, виснаження пулу IP-адрес або порушення доступності мережних сервісів.

Команда: *sudo macof -i eth0*

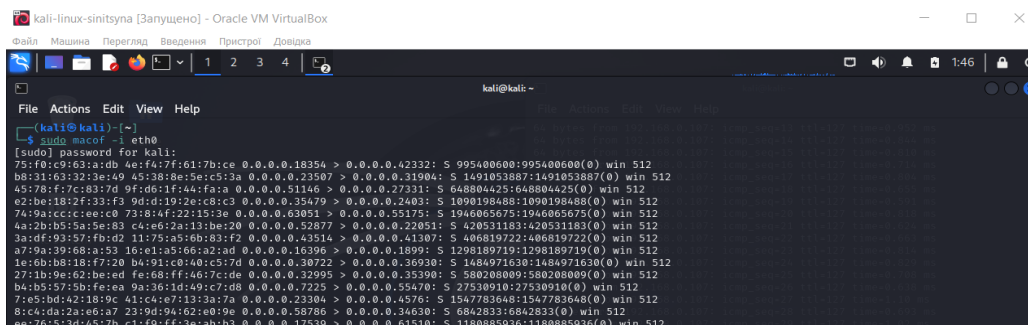


Рисунок 3.88 – Робота команди підміни MAC-адрес (уривок)

Система моніторингу Wazuh зафіксувала різке зростання кількості подій, що відображено на графіку *Discover* та свідчить про інтенсивну аномальну мережеву активність. Аналіз сповіщень Suricata підтвердив наявність великої кількості TCP/IP-пакетів, характерних для флуд-атаки. У журналах зафіксовано повідомлення типу «*TCP packet too small*» та «*Generic Protocol Command Decode*», які є типовими для трафіку, згенерованого інструментом *macof*. Такі події виникають унаслідок формування мінімальних або некоректних пакетів, що створюють значне навантаження на мережеву інфраструктуру та засоби виявлення атак.

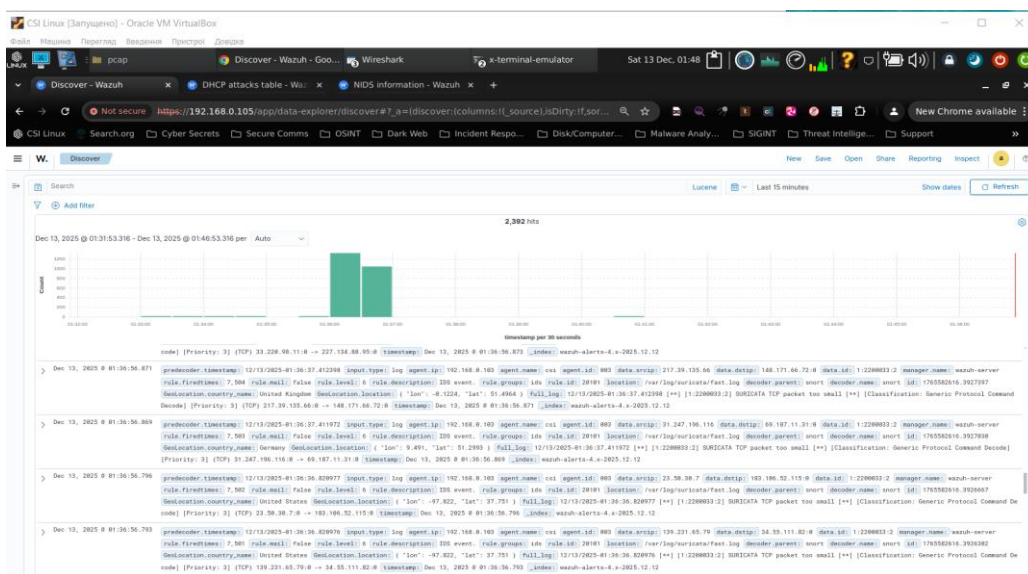


Рисунок 3.89 – Відображення зафіксованих попереджень

Також зростання подій чітко видно на панелі дашбордів *NIDS information*. Загальна кількість алертів перевищує кілька тисяч за дуже короткий проміжок часу, а графік *Alerts Over Time* демонструє різкий сплеск активності, типовий для flood-атак.

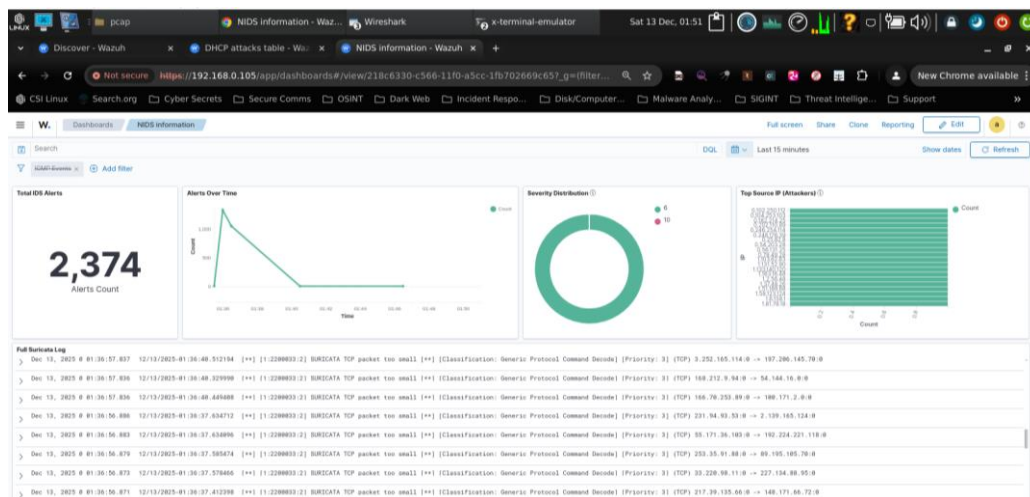


Рисунок 3.90 – Відображення кореляції зафіксованих попереджень

W.

Dashboards

NIDS information

Full screen

Share

Clone

Reporting

Edit

Alerts Table

Timestamp	Host	Agent IP	Source IP	Decoder Name	Rule ID	Rule Level	Rule Groups	Rule Fired Times	Journal Location	GeoIP Country	Count
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.103.62.83	snort	20101	6	ids	6,529	/var/log/wuricata/fast.log	South Korea	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.131.52.90	snort	20101	6	ids	728	/var/log/wuricata/fast.log	Australia	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.133.140.120	snort	20101	6	ids	6,824	/var/log/wuricata/fast.log	Australia	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.163.16.48	snort	20101	6	ids	6,173	/var/log/wuricata/fast.log	Taiwan	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.256.48	snort	20101	6	ids	809	/var/log/wuricata/fast.log	China	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.37.48.26	snort	20101	6	ids	6,237	/var/log/wuricata/fast.log	Philippines	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.51.189.68	snort	20101	6	ids	6,136	/var/log/wuricata/fast.log	China	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.58.123.124	snort	20101	6	ids	7,157	/var/log/wuricata/fast.log	China	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2 csi		192.168.0.103	1.6.158.1	snort	20101	6	ids	6,463	/var/log/wuricata/fast.log	India	1

1

Рисунок 3.91 – Відображення зафіксованих попереджень

Time	Source IP	Destinatio...	Source Co...	Rule Descri...	rule.level: ...	Rule Fired ...	Agent Name	Count
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.103.62.83	54.7.215.4:0	South Korea	IDS event.	6	6,529	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.131.52.90	36.121.45.36:0	Australia	IDS event.	6	728	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.133.140.120	218.8.229.43:0	Australia	IDS event.	6	6,824	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.163.16.48	187.192.191.78	Taiwan	IDS event.	6	6,173	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.256.48	219.98.26.87:0	China	IDS event.	6	809	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.37.48.26	73.39.50.73:0	Philippines	IDS event.	6	6,237	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.51.189.68	18.99.82.24:0	China	IDS event.	6	6,136	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.58.123.124	202.249.104.12	China	IDS event.	6	7,157	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.6.158.1	253.109.39.42:0	India	IDS event.	6	6,463	csi	1
Dec 1, 2025 @ 00:00:00.000 to Dec 13, 2025 @ 01:50:12.176	1.81.78.18	8.124.191.39:0	China	IDS event.	6	594	csi	1

Рисунок 3.92 – Відображення зафіксованих попереджень

На мапі Wazuh Alert Map відображається значна концентрація алертів (червоні точки), що, ймовірно, вказує на місця походження або цілі виявлених

аномальних подій. Це демонструє здатність системи геолокувати підозрілу активність і візуалізувати її розподіл. Наприклад, один з зафіксованих алертів, пов'язаний з Німеччиною ("lon": 9.491, "lat": 51.2993).

Table	JSON
r_geoLocation.country_name	Germany
r_geoLocation.location	{ "lon": 9.491, "lat": 51.2993 }
r_index	Wazuh-alerts-4.x-2020.12.12
r_agent.id	880
r_agent.ip	192.168.0.180
r_agent.name	csi
r_data.destip	69.187.11.31:8
r_data.id	1320803.2
r_data.srcip	31.247.196.116
r_decoder.name	snort
r_decoder.parent	snort
r_full_log	12/12/2020-01:26:37.41970 [**] [1-320803.2] SURicata TCP packet too small [**] [Classification: Generic Protocol Command Decode] [Priority: 3] (TCP) 31.247.196.116:8 -> 69.187.11.31:8

Рисунок 3.93 – Попередження, що містить координати атакуючого

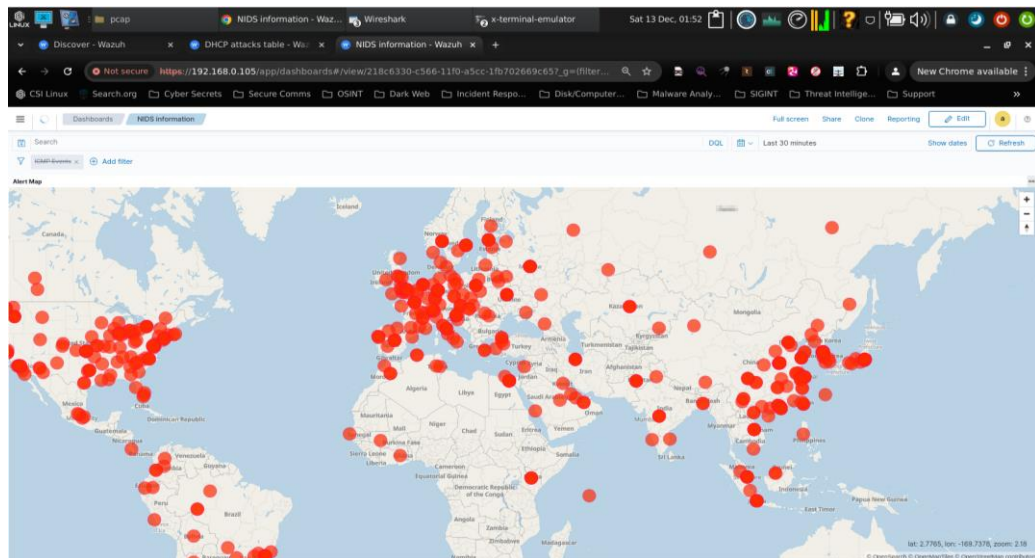


Рисунок 3.94 – Узагальнення попереджень по координатах на карті

Аналіз PCAP файлу у Wireshark підтверджує наявність захопленого мережного трафіку з великою кількістю IPv4-пакетів мінімальної довжини 60 байт. Така кількість коротких пакетів не є характерною для легітимного мережного обміну, так як прикладні та службові протоколи зазвичай формують пакети більшого розміру. [24]

No.	Time	Source	Destination	Protocol	Length	Info
1462	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1464	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1466	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1468	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1470	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1472	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1474	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1476	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1478	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1480	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1482	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1484	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1486	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1
1488	2025-12-13 01:36:03.878200	96.205.127.74	94.187.111.103	IPv4	60	192.168.1.100 → 192.168.1.1

Рисунок 3.95 – Захоплений трафік за допомогою скрипта (уривок)

Статистика Conversations на рівні Ethernet (MAC-адрес) відображає значну кількість короткотривалих взаємодій, які складаються з одного пакета обсягом 60 байт, переданого в одному напрямку. Відсутність відповідних зворотних пакетів і велика кількість унікальних MAC-адрес свідчать про широкомовний або службовий флуд, характерний для атак типу DHCP Flood.

Address A	Address B	Packets	Bytes	Duration
0.0.19.15	127.95.221.45	1	60	0.0000
0.0.40.5	38.254.231.7	1	60	0.0000
0.0.85.61	92.128.108.109	1	60	0.0000
0.3.39.125	245.175.4.50	1	60	0.0000
0.4.76.113	183.196.237.99	1	60	0.0000
0.4.125.55	172.143.41.2	1	60	0.0000
0.4.216.36	224.153.217.119	1	60	0.0000
0.4.239.115	98.158.112.65	1	60	0.0000
0.5.74.24	132.83.227.54	1	60	0.0000
0.6.1.12	56.11.174.78	1	60	0.0000
0.6.78.49	20.170.118.17	1	60	0.0000
0.6.175.106	130.119.242.114	1	60	0.0000
0.6.251.31	230.144.38.53	1	60	0.0000
0.7.137.7	94.132.40.32	1	60	0.0000
0.11.178.24	115.174.93.111	1	60	0.0000
0.12.105.6	222.186.9.102	1	60	0.0000
0.12.154.66	105.104.182.29	1	60	0.0000
0.12.238.82	161.18.228.11	1	60	0.0000
0.14.53.32	54.144.133.95	1	60	0.0000
0.14.97.102	229.227.0.31	1	60	0.0000
0.14.176.38	9.240.143.83	1	60	0.0000
0.14.189.98	75.10.40.109	1	60	0.0000
0.15.188.76	136.165.80.107	1	60	0.0000
0.16.34.45	134.180.175.43	1	60	0.0000
0.16.60.103	172.153.40.21	1	60	0.0000

Рисунок 3.96 – Статистика Conversations на рівні Ethernet

Перегляд статистики на рівні IPv4 також підтверджує наявність великої кількості однопакетних обмінів між різними IP-адресами. Така поведінка є типовою ознакою штучно згенерованого трафіку з можливим використанням підроблених адрес джерела та вказує на флуд-атаку, спрямовану на перевантаження мережної інфраструктури та служб обробки запитів.

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
00:00:77:7c:0e:ad	11:46:0d:7a:74:3e	1	60	1	60	0	0	33.614860	0.0000	—	—
00:02:f3:06:a5:9a	e5:b4:6d:79:e1:71	1	60	1	60	0	0	8.458466	0.0000	—	—
00:03:1f:3f:06:bc	be:79:74:44:3c:bd	1	60	1	60	0	0	9.060018	0.0000	—	—
00:05:41:6b:9d:3f	56:a6:5d:3e:7b:da	1	60	1	60	0	0	26.246994	0.0000	—	—
00:08:44:66:28:4c	81:e9:3a:6c:fe:ea	1	60	1	60	0	0	43.400426	0.0000	—	—
00:08:ba:45:37:8c	27:d1:32:48:c3:8b	1	60	1	60	0	0	32.932575	0.0000	—	—
00:09:ed:6f:94:2d	03:e1:5e:78:30:86	1	60	1	60	0	0	8.085965	0.0000	—	—
00:0a:0c:3a:f6:11	e2:97:c8:4a:9d:57	1	60	1	60	0	0	16.337979	0.0000	—	—
00:0a:50:2e:85:51	9e:4d:0e:25:48:2a	1	60	1	60	0	0	39.482676	0.0000	—	—
00:0c:cd:0e:01:e2	76:1e:96:03:36:f8	1	60	1	60	0	0	30.612756	0.0000	—	—
00:0d:6a:00:b9:f0	d1:82:a5:4a:ad:48	1	60	1	60	0	0	11.239328	0.0000	—	—
00:0d:a7:49:70:39	ac:73:2e:41:fa:32	1	60	1	60	0	0	28.393903	0.0000	—	—
00:0e:a0:38:07:3e	34:f8:ce:2b:45:c4	1	60	1	60	0	0	43.710969	0.0000	—	—
00:0e:f6:00:00:1b	7e:eb:8a:5a:a7:aa	1	60	1	60	0	0	28.488398	0.0000	—	—
00:0f:9d:72:d1:86	80:a8:01:4c:f4:68	1	60	1	60	0	0	8.163082	0.0000	—	—
00:10:de:58:c8:06	b0:c8:14:2e:7b:d1	1	60	1	60	0	0	8.385475	0.0000	—	—
00:12:8e:66:1e:ec	c3:b6:7c:3f:60:8d	1	60	1	60	0	0	14.166480	0.0000	—	—
00:13:12:58:42:53	f7:b7:2d:50:b5:cd	1	60	1	60	0	0	41.678737	0.0000	—	—
00:15:6f:3c:71:73	3b:9d:9f:2f:4b:a8	1	60	1	60	0	0	38.302220	0.0000	—	—
00:15:b9:6e:89:48	6a:78:95:7f:b4:a8	1	60	1	60	0	0	39.621374	0.0000	—	—
00:17:06:70:df:0d	c4:8c:98:28:17:6e	1	60	1	60	0	0	7.733297	0.0000	—	—

Рисунок 3.97 – Статистика Conversations на рівні IPv4

Так як файл завеликий, збережемо невелику частину в окремий файл для аналізу у Network Miner.

Команда:

```
sudo editcap -r /var/log/suricata/pcap/attack-13-12-2025__01:35:58.pcap
/var/log/suricata/pcap/attack-13-12-2025__01:35:58_section.pcap 1400-2000
```

NetworkMiner успішно ідентифікував велику кількість IP-адрес, які були активними під час захоплення трафіку. Велика кількість унікальних хостів у короткому проміжку часу теж підтверджує атаку DHCP Flood.

Hosts (11153)	Files	Images	Messages	Credentials	Sessions	DNS	Parameters	Keywords	Anomalies
0.95.175.29									
0.106.7.76									
0.130.228.75									
0.244.253.1									
1.23.191.38									
1.41.140.82									
1.153.34.7									
2.53.42.51									
2.142.119.50									
2.175.242.21									
2.185.56.9									
2.214.68.100									
2.222.63.93									
3.187.112									
3.86.155.41									
3.190.60.98									
3.199.181.7									
3.221.83.20									
4.100.34.115									
4.124.60.45									
4.134.5.115									
4.136.199.35									
4.160.98.1									
4.255.139.51									
5.19.102.98									
5.53.116.38									
5.105.61.119									
5.130.133.77									
5.168.74.21									
5.172.135.30									
5.240.217.108									
7.6.223.77									

Рисунок 3.98 – Огляд трафіку у Network Miner

3.5. Рекомендації щодо вдосконалення процесів проведення мережної криміналістики та експериментального стенду.

За результатами проведених експериментальних досліджень та практичного тестування розробленого стенду мережної цифрової

криміналістики доцільно визначити такі напрями його подальшого вдосконалення:

- Розширення джерел подій та артефактів
- Розширення сценаріїв тестових атак
- Впровадження елементів реагування
- Удосконалення аналітики

Корисним рішенням буде підключення серверу моніторингу на базі Zeek для збору високорівневих логів (conn.log, http.log, dns.log) і порівняння їх із алертами Suricata. Це забезпечить комплексний аналіз трафіку та підвищить доказову базу.

Рекомендується моделювати атаки на включаючи атаки з використанням експлойтів відомих вразливостей, спроби віддаленого виконання коду та складні багатоступеневі сценарії розвитку інцидентів для перевірки універсальності стенду та оцінки багатоступеневих інцидентів.

Стенд варто доповнити підсистемою IPS для автоматичного блокування IP, обмеження доступу та додаткового збору доказів при критичних подіях.

Також рекомендується впровадити автоматизовані аналітичні звіти для прискорення аналізу та підвищення ефективності роботи аналітика.

Загалом, для процесів проведення мережної криміналістики гарним рішенням буде:

- Впровадження хмарного серверного середовища
- Реплікація та централізоване зберігання мережних логів
- Регулярне оновлення правил, сигнатур і баз знань IDS/IPS
- Забезпечення цілісності та юридичної безпеки цифрових доказів

Імплементація Wazuh у Microsoft Azure дозволяє централізовано збирати та аналізувати логи, забезпечуючи масштабованість і високу доступність системи моніторингу безпеки.

Централізоване зберігання журналів запобігає втраті доказів у разі компрометації окремих вузлів і забезпечує довготривале зберігання логів відповідно до вимог цифрової криміналістики. Доцільно для цього буде використати хмарне середовище.

Постійне оновлення дозволяє підтримувати актуальність виявлення сучасних кіберзагроз і підвищує ефективність системи виявлення вторгнень.

Контроль доступу, журналювання дій аналітиків та використання криптографічних методів підтвердження незмінності даних гарантують юридичну значущість і надійність зібраних доказів.

Висновки з розділу 3:

У третьому розділі реалізовано практичну частину роботи: розгорнуто та налаштовано експериментальний стенд мережної цифрової криміналістики для Linux. Забезпечено взаємодію компонентів та централізований збір подій безпеки.

Впроваджено моніторинг мережного трафіку на базі Suricata з інтеграцією в Wazuh, що дозволяє корелювати мережні й хостові події. Тестові атаки (SSH brute-force, мережеве сканування, DHCP starvation та інші) підтвердили здатність стенду виявляти підозрілу активність. Аналіз PCAP-файлів через Wireshark і NetworkMiner відтворив перебіг інцидентів і підтвердив узгодженість мережних артефактів із журналами безпеки, засвідчивши ефективність запропонованої методики. На базі проведеного дослідження було розроблено рекомендації для процесів мережевої криміналістики та покращення стенду.

ВИСНОВКИ

У першому розділі розглянуто теоретичні засади мережної цифрової криміналістики, визначено її основні принципи, етапи та методи аналізу мережного трафіку. Окреслено типові загрози для Linux-систем і роль мережних засобів виявлення атак у процесі криміналістичного розслідування.

У другому розділі розроблено методику мережної цифрової криміналістики для Linux-середовищ та обґрунтовано вибір інструментальних засобів. Запропоновано архітектуру стенду з автоматизованим збором мережних артефактів і кореляцією подій безпеки.

У третьому розділі реалізовано та протестовано експериментальний стенд. Результати моделювання мережних атак і аналізу PCAP-файлів підтвердили ефективність запропонованої методики та її придатність для практичного використання. На основі тестування було надано рекомендації щодо процесів мережної криміналістики та покращення стенду.

У кваліфікаційній роботі досягнуто поставленої мети та виконано основні завдання, а саме: проаналізовано сучасні методи й підходи мережної цифрової криміналістики; досліджено можливості Linux-засобів для виявлення та аналізу мережних інцидентів; вивчено інструменти збору та обробки мережних артефактів, зокрема PCAP-файлів і журналів подій; розроблено та налаштовано експериментальний стенд для фіксації й аналізу мережних подій; проведено тестування методів в умовах моделювання мережних атак; оцінено ефективність застосованих засобів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мережна криміналістика у кібербезпеці [Електронний ресурс]. – Режим доступу: <https://www.netwitness.com/blog/network-forensics-in-cybersecurity/>
2. Перелік категорій кіберінцидентів [Електронний ресурс]. – Режим доступу: <https://www.cip.gov.ua/ua/news/perelik-kategorii-kiberincidentiv>
3. 7 типів систем виявлення вторгнень [Електронний ресурс]. – Режим доступу: <https://tuxcare.com/blog/types-of-ids/>
4. Рекомендації NIST [Електронний ресурс]. – Режим доступу: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-86.pdf>
5. Поняття Chain of custody [Електронний ресурс]. – Режим доступу: <https://www.thoughtco.com/chain-of-custody-4589132>
6. Офіційна документація Wazuh [Електронний ресурс]. – Режим доступу: <https://documentation.wazuh.com/current/index.html>
7. Мережний моніторинг з документація Wazuh [Електронний ресурс]. – Режим доступу: <https://wazuh.com/blog/network-security-monitoring-with-wazuh-and-zeek/>
8. Моніторинг і детектування у Wazuh [Електронний ресурс]. – Режим доступу: <https://medium.com/@Darkhue/wazuh-monitor-everything-8ba679d0ab7b>
9. Створення правил Suricata [Електронний ресурс]. – Режим доступу: <https://medium.com/@shahriar.atik/suricata-d98a7ec11d96>
10. Дистрибутив CSI Linux [Електронний ресурс]. – Режим доступу: <https://itedu.center/ua/blog/articles/csi-linux-dystrybutyv-linux-dlia-kiberrozsliduvannia/?srsltid=AfmBOoo5Yi9vu68VSTXvdkv9dMQGddvktHSnMIn-lvKUMCq9vJd8wIfa>
11. Офіційний сайт для завантаження CSI Linux [Електронний ресурс]. – Режим доступу: <https://csilinux.com/downloads.html>
12. Офіційний сайт для завантаження Suricata [Електронний ресурс]. – Режим доступу: <https://suricata.io/>

13. Документація з використання Suricata [Електронний ресурс]. – Режим доступу: <https://docs.suricata.io/en/suricata-8.0.2/>
14. Порівняння функціональних можливостей Snort та Suricata [Електронний ресурс]. – Режим доступу: <https://www.stamus-networks.com/suricata-vs-snort>
15. ТОП-30 інструментів безпеки для Linux [Електронний ресурс]. – Режим доступу: <https://introserv.com/ua/blog/top-30-instrumentiv-bezpeky-dlya-linux/>
16. SSH Bruteforce атаки [Електронний ресурс]. – Режим доступу: <https://www.geeksforgeeks.org/linux-unix/how-to-use-hydra-to-brute-force-ssh-connections/>
17. Використання утиліти ApacheBench для навантаження веб-серверу [Електронний ресурс]. – Режим доступу: <https://www.baeldung.com/linux/ab-apachebench-load-testing>
18. Написання мережних bash-скриптів [Електронний ресурс]. – Режим доступу: <https://www.fullstackfoundations.com/blog/bash-network-commands>
19. Використання утиліти Hping для DoS-атак [Електронний ресурс]. – Режим доступу: <https://www.baeldung.com/linux/ab-apachebench-load-testing>
20. Використання утиліти curl [Електронний ресурс]. – Режим доступу: <https://www.geeksforgeeks.org/linux-unix/curl-command-in-linux-with-examples/>
21. ISO/IEC 27042:2015 [Електронний ресурс]. – Режим доступу: <https://cdn.standards.iteh.ai/samples/44406/c986892bfdca440fa33c5eda1e57e23a/ISO-IEC-27042-2015.pdf>
22. Глобальна база кібератак MITRE ATT&CK [Електронний ресурс]. – Режим доступу: <https://attack.mitre.org/>
23. Gerard Johansen «Digital Forensics and Incident Response» [Електронний ресурс]. – Режим доступу: https://duikt.edu.ua/uploads/l_1786_50871291.pdf
24. Сініцина О.В., Бродський Ю.Б., Єфіменко А.А.. Алгоритм аналізу PCAP-файлу як елемента мережної криміналістики // Тези доповідей VII Всеукраїнської науково-технічної конференції «Комп'ютерні технології:

інновації, проблеми, рішення», м. Житомир, 02-03 грудня 2024 р. – Житомир: Державний університет «Житомирська політехніка», 2024.

25. Сініцина О.В., Бродський Ю.Б., Єфіменко А.А.. Аналіз найпоширеніших загроз для Linux-систем у 2024 році // Тези доповідей XV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», м. Житомир, 28-29 березня 2025 року – Житомир: Державний університет «Житомирська політехніка», 2025.

ДОДАТКИ

Порівняльна таблиця систем виявлення вторгнень та кореляції подій

Система Критерій	Wazuh	OSSEC	Security Onion	Splunk (Free/Trial)	ELK Stack + Beats / Elastic Security
Тип	HIDS / SIEM / XDR	HIDS	IDS / SIEM / NSM	SIEM / Log Analysis	SIEM / XDR
Ліцензія	Open-source	Open- source	Open-source	Комерційна (безкоштовна версія обмежена)	Open-source / Elastic License
Архітектура	Агент-сервер, багатопоточна	Агент- сервер	Включає Suricata, Zeek, Elasticsearch, Kibana	Централізован а, агенти для збору логів	Централізован а, агенти для збору логів (Beats)
Збір логів	Так, централізова но	Так	Так	Так	Так
Кореляція подій	Так, правила Wazuh	Обмежено	Так, через Elastic Stack / Zeek	Так, власні правила	Так, Elastic Security Rules
Моніторинг мережі	Через інтеграції (Suricata, Zeek)	Ні	Так	Так, через додаткові модулі	Так
Простота використання	Середня	Складна для початківців	Складна, потребує налаштуван ня	Залежить від ліцензії	Середня, потребує Elastic Stack знань
Найкраще підходить для	HIDS + SIEM + XDR у Linux	Локальний HIDS	Повний NSM / SIEM стек для мережі	Корпоративні середовища, великі дані	SIEM / XDR, аналіз логів і подій, візуалізація

Таблиця інструментів, встановлених на CSI Linux

Категорія	Приклади інструментів	Призначення
SIEM + Виявлення інцидентів	Autopsy, Kibana, Elasticsearch, Logstash, Filebeat, Packetbeat	Збір, індексація та аналіз логів, виявлення вторгнень, побудова дашбордів і таймлайнів подій.
Інструменти CSI Linux	CSI Updater, CSI Configurator, Tor Router, ProxyChains	Налаштування системи, управління VPN і трафіком, анонімізація запитів.
Безпечний зв'язок	Signal, Session, Ricochet, Tox	Захищене спілкування та обмін даними між дослідниками.
Шифрування та паролі	VeraCrypt, KeePassXC, John the Ripper, Hashcat, GnuPG	Зберігання паролів, шифрування файлів, розшифрування хешів і перевірка цілісності.
OSINT	Maltego, SpiderFoot, theHarvester, Recon-ng, Sherlock, Social Analyzer	Збір інформації з відкритих джерел, соціальних мереж, доменів і поштових адрес.
Dark Web	Tor Browser, OnionScan, Ahmia Search, Tails Tools	Безпечна робота з анонімними мережами, дослідження ресурсів Darknet.
Реагування на інциденти	Wireshark, Suricata, Zeek (Bro), Volatility, Chkrootkit, Lynis	Аналіз мережного трафіку, виявлення вторгнень, дослідження пам'яті та аудит безпеки системи.
Комп'ютерна криміналістика	Sleuth Kit, FTK Imager, X-Ways Forensics, Bulk Extractor	Створення бітових образів дисків, аналіз файлових систем і цифрових артефактів.
Мобільна криміналістика	MobSF, Andriller, Oxygen Forensics, Cellebrite UFED	Аналіз даних мобільних пристроїв, відновлення інформації з телефонів.
Судова експертиза транспортних засобів	CANalyze, BusMaster, SavvyCAN	Аналіз даних транспортних систем через протокол CAN.
Аналіз зловмисного ПЗ	Ghidra, Radare2, Cuckoo Sandbox, REMnux Tools, PESTudio	Зворотне проєктування, динамічний і статичний аналіз шкідливих програм.
SIGINT (радіорозвідка)	GNURadio, GQRX, SDR#	Аналіз радіосигналів, перехоплення даних, дослідження ефірного трафіку.
Розвідка загроз (Threat Intelligence)	MISP, Yeti, OpenCTI, AbuseIPDB, VirusTotal Tools	Моніторинг кіберзагроз, обмін ІОС-індикаторами, перевірка хешів і IP-адрес.

Порівняльна таблиця IDS Suricata та Snort

Критерій	Suricata	Snort
<i>Тип системи</i>	IDS / IPS / NSM	IDS / IPS
<i>Ліцензія</i>	Open-source (GPL)	Open-source
<i>Архітектура</i>	Багатопотокова (multi-threaded)	Однопотокова (класичний режим)
<i>Продуктивність</i>	Висока, підходить для великих обсягів трафіку	Нижча на високих навантаженнях
<i>Аналіз трафіку</i>	Глибокий аналіз пакетів (DPI)	Глибокий аналіз пакетів
<i>Підтримка протоколів</i>	Широка (HTTP, TLS, DNS, SMB, FTP тощо)	Обмеженіша
<i>Генерація NSM-даних</i>	Так (EVE JSON, flow, metadata)	Ні
<i>Запис PCAP</i>	Умовний (за правилами/подіями)	Повний або ручний
<i>Сумісність правил</i>	Підтримує правила Snort + власні	Власні правила
<i>Гнучкість правил</i>	Висока, підтримка IOCs	Висока, але менш гнучка
<i>IPS-режим</i>	Так (inline, NFQUEUE, AF_PACKET)	Так
<i>Блокування трафіку</i>	Drop, reset, rate limiting	Drop, reset
<i>Споживання ресурсів</i>	Вище, але ефективне	Нижче, але менш масштабоване
<i>Масштабованість</i>	Висока (distributed deployment)	Обмежена
<i>Інтеграція з SIEM</i>	Дуже добра (JSON, API)	Обмежена
<i>Складність налаштування</i>	Висока	Середня
<i>Активність розвитку</i>	Дуже активна	Менш активна
<i>Типове застосування</i>	NSM, SOC, forensic-аналіз	Базовий IDS/IPS

Скрипти для захоплення PCAP-трафіку

Файл: */home/csi/watch_alerts.sh*

```
#!/bin/bash

EVE_FILE="/var/log/suricata/eve.json"
STATE_FILE="/tmp/current_attack.state"
TIMEOUT=10 # після скількох секунд тиші вважаємо атаку завершеною

command -v jq >/dev/null 2>&1 || { echo "jq not installed"; exit 1; }

tail -F $EVE_FILE | jq -c 'select(.event_type=="alert") | {src:.src_ip, dst:.dest_ip, time:.timestamp}' |
while read line; do
    SRC=$(echo $line | jq -r '.src')
    DST=$(echo $line | jq -r '.dst')
    RAW_TIME=$(echo $line | jq -r '.time')
    ALERT_TIME=$(date -d "$RAW_TIME" +"%Y-%m-%d_%H-%M-%S")

    # Якщо атака ще не йде → запускаємо нову
    if [ ! -f "$STATE_FILE" ]; then
        echo "$SRC $DST $ALERT_TIME" > $STATE_FILE
        /usr/local/bin/save_alert_pcap.sh $SRC $DST $ALERT_TIME &
    fi

    # Оновлюємо час останнього alert-a
    date +%s > /tmp/last_alert_time
done
```

Файл: */usr/local/bin/save_alert_pcap.sh*

```
#!/bin/bash

SRC=$1
DST=$2
ATTACK_START=$3

INTERFACE="enp0s8"
OUTPUT_DIR="/var/log/suricata/pcap"
STATE_FILE="/tmp/current_attack.state"
LAST_ALERT_FILE="/tmp/last_alert_time"
IDLE_TIMEOUT=10 # скільки секунд без alertів = атака закінчилась

mkdir -p $OUTPUT_DIR

FILENAME="${OUTPUT_DIR}/attack-${ATTACK_START}.pcap"

echo "Starting attack capture: $FILENAME"
```

```

# Запускаємо tcpdump в бекграунді
tcpdump -i $INTERFACE host $SRC or host $DST -w $FILENAME &
TCPDUMP_PID=$!

# Основний цикл – чекати поки атака триває
while true; do
    sleep 1

    LAST=$(cat $LAST_ALERT_FILE 2>/dev/null)
    NOW=$(date +%s)

    # Якщо не було alert-ів певний час → атака завершена
    if (( NOW - LAST > IDLE_TIMEOUT )); then
        echo "Attack ended. Stopping tcpdump."
        kill $TCPDUMP_PID 2>/dev/null
        rm -f $STATE_FILE
        break
    fi
done

```

Таблиця змінних у скриптах

Змінна	Де використовується	Призначення
EVE_FILE	watch_alerts.sh	Шлях до файлу Suricata eve.json.
STATE_FILE	обидва скрипти	Прапорець, що атака активна.
LAST_ALERT_FILE	обидва скрипти	Час останнього зафіксованого alert-а.
TIMEOUT / IDLE_TIMEOUT	обидва скрипти	Пороговий час (секунди), після якого атака вважається завершеною.
INTERFACE	save_alert_pcap.sh	Мережний інтерфейс для запису трафіку.
OUTPUT_DIR	save_alert_pcap.sh	Каталог, де зберігаються PCAP-файли.
TCPDUMP_PID	save_alert_pcap.sh	PID процесу tcpdump для коректної зупинки.
SRC, DST	watch_alerts.sh	IP-адреси джерела та отримувача атаки.
ALERT_TIME	watch_alerts.sh	Час початку атаки користувачем у назві файлу.

Основні команди

Перезапуск suricata та wazuh-агента

`sudo systemctl restart suricata`

`sudo systemctl restart wazuh-agent`

Команди для файлу /home/csi/watch_alerts.sh

Створення – `sudo nano ~/watch_alerts.sh`

Надання прав доступу – `sudo chmod +x ~/watch_alerts.sh`

Запуск – `sudo ~/watch_alerts.sh`

Команди для файлу /usr/local/bin/save_alert_pcap.sh

Створення – `sudo nano /usr/local/bin/save_alert_pcap.sh`

Надання прав доступу – `sudo chmod +x /usr/local/bin/save_alert_pcap.sh`

Інтерпретація рівнів rule.level

0–3 – інформаційні події (службові повідомлення, нормальна активність)

4–6 – події низького ризику (помилки автентифікації, нетипові, але не критичні дії)

7–9 – підозріла активність (brute-force, сканування, підозрілі мережні з'єднання)

10–12 – серйозні інциденти (успішні атаки, компрометація облікових записів)

13–16 – критичні загрози (root-доступ, експлойти, активна компрометація системи)

Конфігураційні файли

Файл /etc/suricata/rules/local.rules

```
##ICMP
```

```
alert icmp any any -> any any (msg:"SURICATA: ICMP Ping detected"; itype:8; sid:1000001; rev:1;)
```

```
## TCP SYN flood
```

```
alert tcp any any -> any 80 (flags:S; msg:"SURICATA: Possible SYN flood (DoS attempt)"; threshold:type threshold, track by_src, count 20, seconds 2; sid:1000002; rev:1;)
```

```
##HTTP curl
```

```
alert tcp any any -> any 80 (msg:"SURICATA: Suspicious HTTP GET request (RAW TCP)"; flow:to_server,established; content:"GET "; sid:1000010; rev:1;)
```

Файл /var/ossec/etc/ossec.conf

```
<ossec_config>
  <client>
    <server>
      <address>192.168.0.105</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
    <config-profile>ubuntu, ubuntu22, ubuntu22.04</config-profile>
    <notify_time>20</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>
  </client>

  <client_buffer>
    <!-- Agent buffer options -->
    <disabled>no</disabled>
    <queue_size>5000</queue_size>
    <events_per_second>500</events_per_second>
  </client_buffer>

  <!-- Policy monitoring -->
  <rootcheck>
    <disabled>no</disabled>
    <check_files>yes</check_files>
    <check_trojans>yes</check_trojans>
    <check_dev>yes</check_dev>
    <check_sys>yes</check_sys>
    <check_pids>yes</check_pids>
    <check_ports>yes</check_ports>
    <check_if>yes</check_if>

    <!-- Frequency that rootcheck is executed - every 12 hours -->
    <frequency>43200</frequency>
```

```

<rootkit_files>etc/shared/rootkit_files.txt</rootkit_files>
<rootkit_trojans>etc/shared/rootkit_trojans.txt</rootkit_trojans>

<skip_nfs>yes</skip_nfs>

<ignore>/var/lib/containerd</ignore>
<ignore>/var/lib/docker/overlay2</ignore>
</rootcheck>

<wodle name="cis-cat">
  <disabled>yes</disabled>
  <timeout>1800</timeout>
  <interval>1d</interval>
  <scan-on-start>yes</scan-on-start>

  <java_path>wodles/java</java_path>
  <ciscat_path>wodles/ciscat</ciscat_path>
</wodle>

<!-- Osquery integration -->
<wodle name="osquery">
  <disabled>yes</disabled>
  <run_daemon>yes</run_daemon>
  <log_path>/var/log/osquery/osqueryd.results.log</log_path>
  <config_path>/etc/osquery/osquery.conf</config_path>
  <add_labels>yes</add_labels>
</wodle>
<!-- System inventory -->
<wodle name="syscollector">
  <disabled>no</disabled>
  <interval>1h</interval>
  <scan_on_start>yes</scan_on_start>
  <hardware>yes</hardware>
  <os>yes</os>
  <network>yes</network>
  <packages>yes</packages>
  <ports all="yes">yes</ports>
  <processes>yes</processes>

  <!-- Database synchronization settings -->
  <synchronization>
    <max_eps>10</max_eps>
  </synchronization>
</wodle>

<sca>
  <enabled>yes</enabled>
  <scan_on_start>yes</scan_on_start>
  <interval>12h</interval>
  <skip_nfs>yes</skip_nfs>
</sca>

<!-- File integrity monitoring -->
<syscheck>
  <disabled>no</disabled>

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>43200</frequency>

  <scan_on_start>yes</scan_on_start>

```

```
<!-- Directories to check (perform all possible verifications) -->
<directories>/etc,/usr/bin,/usr/sbin</directories>
<directories>/bin,/sbin,/boot</directories>

<!-- Files/directories to ignore -->
<ignore>/etc/mtab</ignore>
<ignore>/etc/hosts.deny</ignore>
<ignore>/etc/mail/statistics</ignore>
<ignore>/etc/random-seed</ignore>
<ignore>/etc/random.seed</ignore>
<ignore>/etc/adtime</ignore>
<ignore>/etc/httpd/logs</ignore>
<ignore>/etc/utmpx</ignore>
<ignore>/etc/wtmpx</ignore>
<ignore>/etc/cups/certs</ignore>
<ignore>/etc/dumpdates</ignore>
<ignore>/etc/svc/volatile</ignore>

<!-- File types to ignore -->
<ignore type="sregex">.log$|.swp$</ignore>

<!-- Check the file, but never compute the diff -->
<nodiff>/etc/ssl/private.key</nodiff>

<skip_nfs>yes</skip_nfs>
<skip_dev>yes</skip_dev>
<skip_proc>yes</skip_proc>
<skip_sys>yes</skip_sys>

<!-- Nice value for Syscheck process -->
<process_priority>10</process_priority>

<!-- Maximum output throughput -->
<max_eps>50</max_eps>

<!-- Database synchronization settings -->
<synchronization>
  <enabled>yes</enabled>
  <interval>5m</interval>
  <max_eps>10</max_eps>
</synchronization>
</syscheck>

<!-- Log analysis -->
<localfile>
  <log_format>command</log_format>
  <command>df -P</command>
  <frequency>360</frequency>
</localfile>

<localfile>
  <log_format>full_command</log_format>
  <command>netstat -tulpn | sed 's/\([[[:alnum:]]\+\)\ \+[[[:digit:]]\+\ \+[[[:digit:]]\+\ \+(\(.*)\):\([[:digit:]]*\)\ \+\\(0-9)\.\.[[:digit:]]\+)\).\ +\\([[:digit:]]*\V[[[:alnum:]]\+]*\).*\/1 2 == 3 == 4 5/' | sort -k 4 -g | sed 's/ == \(.*)\)/:/1/' | sed 1,2d</command>
  <alias>netstat listening ports</alias>
  <frequency>360</frequency>
</localfile>

<localfile>
  <log_format>full command</log format>
```

```

    <command>last -n 20</command>
    <frequency>360</frequency>
</localfile>

<!-- Active response -->
<active-response>
    <disabled>no</disabled>
    <ca_store>etc/wpk_root.pem</ca_store>
    <ca_verification>yes</ca_verification>
</active-response>

<!-- Choose between "plain", "json", or "plain,json" for the format of internal logs -->
<logging>
    <log_format>plain</log_format>
</logging>

</ossec_config>

<ossec_config>
<localfile>
    <log_format>journald</log_format>
    <location>journald</location>
</localfile>

<localfile>
    <log_format>apache</log_format>
    <location>/var/log/apache2/error.log</location>
</localfile>

<localfile>
    <log_format>apache</log_format>
    <location>/var/log/apache2/access.log</location>
</localfile>

<localfile>
    <log_format>audit</log_format>
    <location>/var/log/audit/audit.log</location>
</localfile>

<localfile>
    <log_format>syslog</log_format>
    <location>/var/ossec/logs/active-responses.log</location>
</localfile>

<localfile>
    <log_format>syslog</log_format>
    <location>/var/log/dpkg.log</location>
</localfile>

<localfile>
    <log_format>json</log_format>
    <location>/var/log/suricata/eve.json</location>
</localfile>

<localfile>
    <log_format>syslog</log_format>
    <location>/var/log/suricata/fast.log</location>
</localfile>

<localfile>

```

```

    <log_format>syslog</log_format>
    <location>/var/log/syslog</location>
</localfile>
</ossec_config>

```

Файл */etc/suricata/suricata.yaml*

```

%YAML 1.1
---

vars:
  address-groups:
    HOME_NET:
"[192.168.0.0/24,10.0.0.0/8,172.16.0.0/12]"

    EXTERNAL_NET: "!$HOME_NET"

    HTTP_SERVERS: "$HOME_NET"
    SMTP_SERVERS: "$HOME_NET"
    SQL_SERVERS: "$HOME_NET"
    DNS_SERVERS: "$HOME_NET"
    TELNET_SERVERS: "$HOME_NET"
    AIM_SERVERS: "$EXTERNAL_NET"
    DC_SERVERS: "$HOME_NET"
    DNP3_SERVER: "$HOME_NET"
    DNP3_CLIENT: "$HOME_NET"
    MODBUS_CLIENT: "$HOME_NET"
    MODBUS_SERVER: "$HOME_NET"
    ENIP_CLIENT: "$HOME_NET"
    ENIP_SERVER: "$HOME_NET"

  port-groups:
    HTTP_PORTS: "80"
    SHELLCODE_PORTS: "!80"
    ORACLE_PORTS: 1521
    SSH_PORTS: 22
    DNP3_PORTS: 20000
    MODBUS_PORTS: 502
    FILE_DATA_PORTS: "[$HTTP_PORTS,110,143]"
    FTP_PORTS: 21
    GENEVE_PORTS: 6081
    VXLAN_PORTS: 4789
    TEREDO_PORTS: 3544

default-log-dir: /var/log/suricata/

stats:
  enabled: yes
  interval: 8

outputs:
  - fast:
    enabled: yes
    filename: fast.log
    append: yes
  - eve-log:
    enabled: yes
    filetype: regular
#regular|syslog|unix_dgram|unix_stream|redis
  filename: /var/log/suricata/eve.json

```

```

community-id: false
community-id-seed: 0
xff:
  mode: extra-data
  deployment: reverse
  header: X-Forwarded-For

types:
  - alert:
    enabled: yes
  - alert
  - http
  - dns
  - tls
  - ssh
  - flow
  - icmp
  - anomaly

  - http-log:
    enabled: no
    filename: http.log
    append: yes
  - tls-log:
    enabled: no # Log TLS connections.
    filename: tls.log # File to store TLS logs.
    append: yes
  - tls-store:
    enabled: no
  - pcap-log:
    enabled: no #works fow all traffic
  - alert-debug:
    enabled: no
    filename: alert-debug.log
    append: yes
  - alert-prelude:
    enabled: no
    profile: suricata
    log-packet-content: no
    log-packet-header: yes

  - alert-exec:
    enabled: yes
    filename: alert-exec.log
    command:
/var/log/suricata/scripts/attacks_capture_pcap.sh

  - stats:
    enabled: yes
    filename: stats.log
    append: yes # append to file (yes) or overwrite
it (no)

```

totals: yes # stats for all threads merged
 together
 threads: no # per thread stats

 - syslog:
 enabled: yes
 facility: local5
 level: Info, Emergency, Alert, Critical, Error,
 Warning, Notice, Debug
 - file-store:
 version: 2
 enabled: no

 xff:
 enabled: no
 mode: extra-data
 deployment: reverse
 header: X-Forwarded-For
 - tcp-data:
 enabled: yes
 type: file
 filename: tcp-data.log
 - http-body-data:
 enabled: no
 type: file
 filename: http-data.log
 - lua:
 enabled: no
 scripts:
 logging:
 default-log-level: notice
 default-output-filter:
 outputs:
 - console:
 enabled: yes
 - file:
 enabled: yes
 level: info
 filename: suricata.log
 - syslog:
 enabled: no
 facility: local5
 format: "[%i] <%d> -- "

af-packet:
 - interface: enp0s3
 threads: 1
 cluster-id: 99
 cluster-type: cluster_flow
 defrag: yes

 - interface: enp0s8
 threads: 1
 cluster-id: 98
 cluster-type: cluster_flow
 defrag: yes
 use-mmap: yes
 mmap-lock: yes
 pcap:
 enabled: yes
 filename: pcap/alert-%n.pcap

limit: 500mb
 pcap:
 - interface: enp0s8
 - interface: default

 pcap-file:
 checksum-checks: auto

 app-layer:
 protocols:
 rfb:
 enabled: yes
 detection-ports:
 dp: 5900, 5901, 5902, 5903, 5904, 5905, 5906,
 5907, 5908, 5909
 mqtt:
 krb5:
 enabled: yes
 snmp:
 enabled: yes
 ikev2:
 enabled: yes
 tls:
 enabled: yes
 detection-ports:
 dp: 443

 dcerpc:
 enabled: yes
 ftp:
 enabled: yes
 rdp:
 ssh:
 enabled: yes
 http2:
 enabled: no
 http1-rules: no
 smtp:
 enabled: yes
 raw-extraction: no
 mime:
 decode-mime: yes

 decode-base64: yes
 decode-quoted-printable: yes
 header-value-depth: 2000

 extract-urls: yes
 body-md5: no
 inspected-tracker:
 content-limit: 100000
 content-inspect-min-size: 32768
 content-inspect-window: 4096
 imap:
 enabled: detection-only
 smb:
 enabled: yes
 detection-ports:
 dp: 139, 445

nfs:	dhcp:
enabled: yes	enabled: yes
tftp:	
enabled: yes	sip:
dns:	
tcp:	asn1-max-frames: 256
enabled: yes	
detection-ports:	coredump:
dp: 53	max-dump: unlimited
udp:	
enabled: yes	host-mode: auto
detection-ports:	
dp: 53	unix-command:
http:	enabled: yes
enabled: yes	filename: /var/run/suricata-command.socket
libhttp:	
default-config:	legacy:
personality: IDS	uricontent: enabled
request-body-limit: 100kb	engine-analysis:
response-body-limit: 100kb	rules-fast-pattern: yes
	rules: yes
request-body-minimal-inspect-size: 32kb	
request-body-inspect-window: 4kb	pcrc:
response-body-minimal-inspect-size: 40kb	match-limit: 3500
response-body-inspect-window: 16kb	match-limit-recursion: 1500
response-body-decompress-layer-limit: 2	
	host-os-policy:
http-body-inline: auto	windows: [0.0.0.0/0]
	bsd: []
swf-decompression:	bsd-right: []
enabled: yes	old-linux: []
type: both	linux: []
compress-depth: 100kb	old-solaris: []
decompress-depth: 100kb	solaris: []
	hpux10: []
double-decode-path: no	hpux11: []
double-decode-query: no	irix: []
	macos: []
server-config:	vista: []
modbus:	windows2k3: []
enabled: no	
detection-ports:	defrag:
dp: 502	memcap: 32mb
stream-depth: 0	hash-size: 65536
	trackers: 65535 # number of defragmented flows to follow
dnp3:	max-frags: 65535 # number of fragments to keep (higher than trackers)
enabled: no	prealloc: yes
detection-ports:	timeout: 60
dp: 20000	
	flow:
enip:	memcap: 128mb
enabled: no	hash-size: 65536
detection-ports:	prealloc: 10000
dp: 44818	emergency-recovery: 30
sp: 44818	
	vlan:
ntp:	
enabled: yes	

```

use-for-tracking: true

flow-timeouts:

default:
  new: 30
  established: 300
  closed: 0
  bypassed: 100
  emergency-new: 10
  emergency-established: 100
  emergency-closed: 0
  emergency-bypassed: 50
tcp:
  new: 60
  established: 600
  closed: 60
  bypassed: 100
  emergency-new: 5
  emergency-established: 100
  emergency-closed: 10
  emergency-bypassed: 50
udp:
  new: 30
  established: 300
  bypassed: 100
  emergency-new: 10
  emergency-established: 100
  emergency-bypassed: 50
icmp:
  new: 30
  established: 300
  bypassed: 100
  emergency-new: 10
  emergency-established: 100
  emergency-bypassed: 50

stream:
  memcap: 64mb
  checksum-validation: yes    # reject incorrect csums
  inline: auto                # auto will use inline mode in
IPS mode, yes or no set it statically
  reassembly:
    memcap: 256mb
    depth: 1mb                # reassemble 1mb into a
stream
  toserver-chunk-size: 2560
  toclient-chunk-size: 2560
  randomize-chunk-size: yes

host:
  hash-size: 4096
  prealloc: 1000
  memcap: 32mb
decoder:
  teredo:
    enabled: true
    ports: $TEREDO_PORTS # syntax: '[3544, 1234]'
or '3533' or 'any'.

vxlan:
  enabled: true
  ports: $VXLAN_PORTS # syntax: '[8472, 4789]' or
'4789'.

vntag:
  enabled: false

geneve:
  enabled: true
  ports: $GENEVE_PORTS # syntax: '[6081, 1234]'
or '6081'.

detect:
  profile: medium
  custom-values:
    toclient-groups: 3
    toserver-groups: 25
  sgh-mpm-context: auto
  inspection-recursion-limit: 3000

prefilter:
  default: mpm

grouping:

profiling:
  grouping:
    dump-to-disk: false
    include-rules: false    # very verbose
    include-mpm-stats: false

mpm-algo: auto

spm-algo: auto

threading:
  set-cpu-affinity: no
  cpu-affinity:
    - management-cpu-set:
      cpu: [ 0 ] # include only these CPUs in affinity
settings
    - receive-cpu-set:
      cpu: [ 0 ] # include only these CPUs in affinity
settings
    - worker-cpu-set:
      cpu: [ "all" ]
      mode: "exclusive"
      prio:
        low: [ 0 ]
        medium: [ "1-2" ]
        high: [ 3 ]
        default: "medium"
      detect-thread-ratio: 1.0
luajit:
  states: 128

profiling:
  rules:
    enabled: yes
    filename: rule_perf.log

```

append: yes	- interface: eth2
	- interface: default
limit: 10	
json: yes	pfring:
	- interface: enp0s8
keywords:	threads: auto
enabled: yes	
filename: keyword_perf.log	cluster-id: 99
append: yes	cluster-type: cluster_flow
prefilter:	- interface: default
enabled: yes	
filename: prefilter_perf.log	ipfw:
append: yes	
	napatech:
rulegroups:	
enabled: yes	streams: ["0-3"]
filename: rule_group_perf.log	
append: yes	enable-stream-stats: no
packets:	auto-config: yes
enabled: yes	
filename: packet_stats.log	hardware-bypass: yes
append: yes	
csv:	inline: no
enabled: no	
locks:	ports: [0-1,2-3]
enabled: no	
filename: lock_stats.log	hashmode: hash5tuplesorted
append: yes	
nfq:	default-rule-path: /etc/suricata/rules
nflog:	rule-files:
- group: 2	- local.rules
buffer-size: 18432	- dhcp-events.rules
- group: default	- decoder-events.rules
qthreshold: 1	- stream-events.rules
qtimeout: 100	- http-events.rules
max-size: 20000	- tls-events.rules
	- files.rules
	- dns-events.rules
capture:	classification-file: /etc/suricata/classification.config
	reference-config-file: /etc/suricata/reference.config
netmap:	