Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Магістерська кваліфікаційна робота на тему:
«Кіберполігон для розгортання та виявлення бот-мереж»

Виконав: студент 2 курсу групи 1БС-24 м
спеціальності 125 Кібербезпека
_____ Владислав ПАЛАМАРЧУК

Керівник: к.т.н., доц. каф. ЗІ
_____ Олеся ВОЙТОВИЧ
« 16 » грудня 2025 р.
Рецензент: к. т. н., доцент каф. ПЗ
_____ Володимир Майданюк
« 16 » грудня 2025 р.

Допущено до захисту
В. о. зав. кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
« 16 » грудня 2025 р.

Вінниця ВНТУ – 2025 року

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека та захист інформації
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
В.о. зав. кафедри ЗІ,
д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
«24» 09 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Владиславу ПАЛАМАРЧУКУ

1. Тема роботи: «Кіберполігон для розгортання та виявлення бот-мереж» керівник роботи: Олеся ВОЙТОВИЧ, к. т. н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ від 24 вересня 2025 року №313.
2. Строк подання студентом роботи 16 грудня 2025 р.
3. Вихідні дані до роботи:
   - бот-мережі, кіберполігони;
   - реалізація кіберполігону:
   - експериментальне дослідження.
4. Зміст текстової частини: Вступ. 1. Аналіз інформаційних джерел та існуючих рішень. 2. Модель та побудова кіберполігону. 3. Реалізація та експерименти на кіберполігоні. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Схема архітектури кіберполігону. Схема моніторингу та виявлення DDoS-атак. Схема взаємодії об'єктів кіберполігону. Топологія кіберполігону в середовищі GNS3. Перевірка статусу ботів. Результат виявлення атак через Suricata. Результат логування Zeek після експериментів. Скріншот панелі моніторингу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	О. ВОЙТОВИЧ, к.т.н., доц. каф.ЗІ	25.09.25	
2	О. ВОЙТОВИЧ, к.т.н., доц. каф.ЗІ	25.09.25	
3	О. ВОЙТОВИЧ, к.т.н., доц. каф.ЗІ	25.09.25	
4	О. ЛЕСЬКО, к.е.н., зав.каф. ЕПВМ	25.09.25	

7. Дата видачі завдання 24 вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	24.09.2025 – 26.09.2025	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	27.09.2025 – 07.10.2025	
3	Науково-технічне обргрунтування	11.10.2025 – 22.10.2025	
4	Розробка моделі та побудова архітектури кіберполігону	23.10.2025 – 26.10.2025	
5	Розробка сценаріїв	27.10.2025 – 02.11.2025	
6	Розробка системи моніторингу та виявлення DDoS	03.11.2025 – 10.11.2025	
7	Розгортання кіберполігону та проведення експериментів	10.11.2025 – 17.11.2025	
8	Розробка розділу економічного обгрунтування доцільності розробки	18.11.2025 – 22.11.2025	
9	Оформлення пояснювальної записки	23.11.2025 – 29.11.2025	
10	Попередній захист та доопрацювання МКР	29.11.2025 – 11.12.2025	
11	Перевірка на наявність текстових запозичень	12.12.2025 – 15.12.2025	
12	Представлення МКР до захисту, рецензування	16.12.2025 – 19.12.2025	
13	Захист МКР	19.12.2025 – 23.12.2025	

Студент Владислав ПАЛАМАРЧУ

Керівник роботи Олеся ВОЙТОВИ

АНОТАЦІЯ

УДК 004.056

Паламарчук В. Кіберполігон для розгортання та виявлення бот-мереж. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2025. 97 с.

Укр. мовою. Бібліогр.: 37 назв; рис.: 22; табл.: 20.

Розроблено модель кіберполігону з основними компонентами та мережевою топологією. Реалізовано експериментальне середовище у GNS3, що дало змогу відтворити роботу бот-мережі під час DDoS-атаки та зібрати мережевий трафік для подальшого аналізу. Досліджено характерні ознаки атак, поведінку трафіку та підходи до його виявлення. У результатах наведено ключові показники навантаження та параметри, отримані під час експериментів.

Ілюстративна частина складається з 8 плакатів з демонстрацією результатів моделювання і проведених досліджень.

В економічному розділі оцінено витрати на розробку.

Ключові слова: кіберполігон, бот-мережа, DDoS-атака, мережевий трафік, моделювання, виявлення атак.

ABSTRACT

UDC 004.056

Palamarchuk V. Cyber range for botnet deployment and detection. Master's thesis in the field of 125 – Cybersecurity and Information Protection, educational program – Security of Information and Communication Systems.Vinnytsia: VNTU, 2025. 97 p.

In Ukrainian language. Bibliographer: 38 titles; fig.: 22; tabl.: 20.

The master's thesis focuses on designing a cyber range for modeling and analyzing DDoS attacks and botnet behavior. The work includes an overview of botnet architectures, command-and-control systems, and existing training cyber ranges. The requirements for a safe and isolated environment suitable for reproducing malicious activity are defined.

A cyber range model with its core components and network topology is proposed. An experimental environment was implemented using GNS3, enabling the simulation of a botnet performing a DDoS attack and the collection of network traffic for further analysis. The research examines traffic patterns, characteristic signs of an attack, and approaches to detecting DDoS activity. The results section provides key performance indicators and measurements obtained during the experiments.

The graphic part consists of 8 posters demonstrating the results of modeling and research.

The economic section estimates the development costs.

Keywords: cyber range, botnet, DDoS attack, network traffic, simulation, attack detection.

ЗМІСТ

ВСТУП

Зростання кількості та складності бот-мереж робить критично важливим питання їх моделювання, моніторингу та достовірного виявлення у безпечному середовищі [1]. Сучасні дослідження показують, що традиційні підходи до побудови кіберполігонів часто опираються на імперативні, важковідтворювані інструменти та орендовану інфраструктуру, що ускладнює відтворення результатів і підвищує ризики етичного та правового характеру [2]. Для України питання створення локальних, контрольованих і приватних середовищ для навчання і досліджень є особливо актуальним у зв’язку з потребою підвищення кіберстійкості критичної інфраструктури та підготовки фахівців. Аналіз літератури та звітів провідних організацій з безпеки, таких як ENISA, MITRE, CERT-UA, Claudflare і праць вчених у сфері мережевого моніторингу і виявлення аномалій підтверджує необхідність дослідження відтворюваних, декларативних підходів до побудови полігонів та інтеграції сучасних методів захисту й анонімності [3].

Робота співвідноситься з напрямами кафедральних досліджень з кібербезпеки та методів захисту інформаційних систем, а також із національними ініціативами щодо підвищення кіберзахищеності. Практична частина може слугувати базою для подальших розробок у межах наукових тем, пов’язаних із моделюванням атак і відпрацюванням засобів реагування.

Мета роботи – розробка спеціалізованого кіберполігону, що забезпечує контрольовану імітацію розгортання бот-мереж, з метою експериментальної оцінки, вдосконалення існуючих та розробки нових методів, алгоритмів і програмних засобів для виявлення бот-мереж. Для досягнення мети вирішено такі завдання:

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати існуючі архітектури бот-мереж і систем командного керування, методи зв’язку ботів та техніки DDoS-атак;

– розробити архітектуру кіберполігону з урахуванням безпечної сегментації компонентів, мережевої ізоляції та контролю експериментальних сценаріїв;

– розгорнути кіберполігон для моделювання сценаріїв та провести експерименти;

– визначити показники економічної доцільності розробки.

Об’єкт дослідження – процеси імітаційного моделювання ddos-атак із застосуванням бот-мереж (ботнетів).

Предмет дослідження – функціональна модель, алгоритми та програмно-технічні рішення для створення та експлуатації спеціалізованого кіберполігону, призначеного для контрольованого розгортання (емуляції), моніторингу, збору даних про бот-мережі та експериментальної оцінки методів і засобів їх виявлення.

У роботі застосовано методи системного аналізу для оцінки особливостей бот-мереж; методи порівняльного аналізу для вибору архітектурних підходів до побудови полігонів, експериментальне моделювання для перевірки реалізованих сценаріїв; кількісні та якісні методи оцінювання. [4].

Удосконалено методику проведення досліджень у сфері кібербезпеки за рахунок забезпечення масштабованої та детермінованої імітації загроз бот-мережами, що дозволяє моніторинг, збір даних про бот-мережі та експериментальну оцінку методів і засобів їх виявлення. Розроблена декларативна модель кіберполігону, щодозволяє імітації загроз бот-мережами та виявлення їх ознак.

АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ ТА ІСНУЮЧИХ РІШЕНЬ У СФЕРІ КІБЕРПОЛІГОНІ ТА БОТ-МЕРЕЖ

1.1 Аналіз архітектури бот-мереж та систем командного керування

Бот-мережі є складними децентралізованими або централізованими системами, створеними для координації великої кількості заражених пристроїв, що виконують команди оператора через канали командного керування (Command and Control або скорочено C2) [5]. Архітектура безпосередньо визначає стійкість до виявлення, можливість управління ботами та ефективність виконання шкідливих дій. Аналіз архітектур C2-систем є ключовим етапом у розумінні принципів функціонування бот-мереж і формуванні підходів до їх моделювання у межах навчально-дослідницького кіберполігону. Узагальнена схема взаємодії компонентів у типовій C2-мережі зображена на рисунку 1.1.

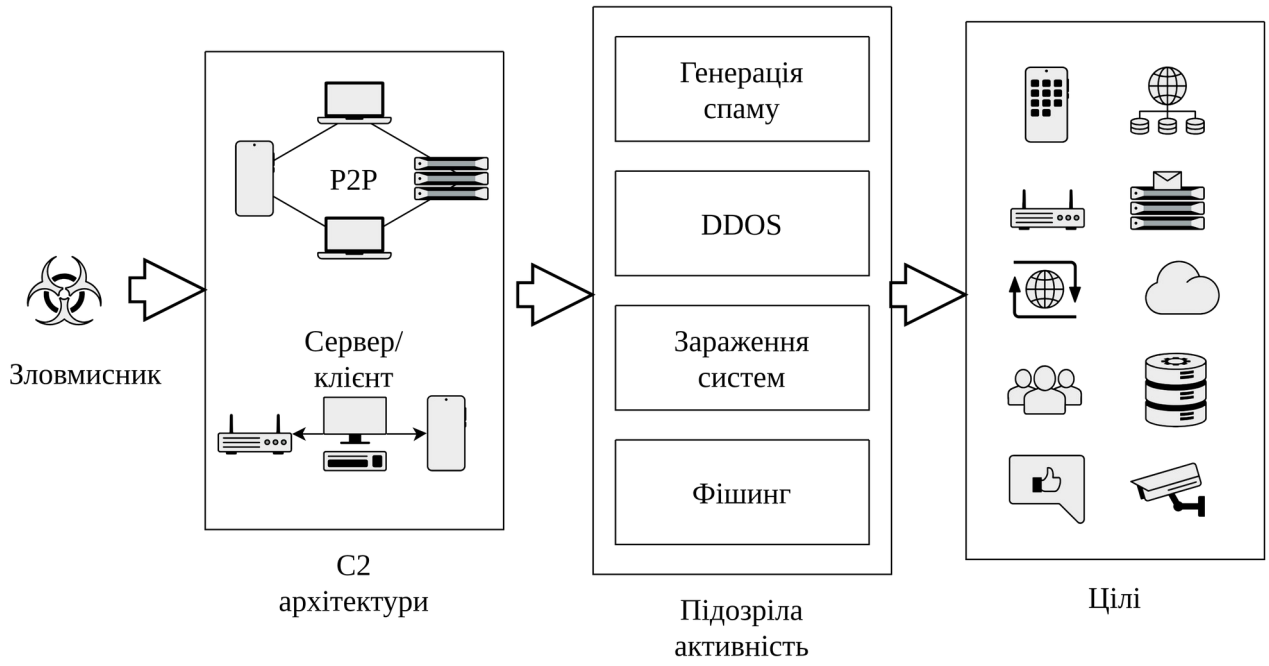


Рисунок 1.1 – Типовий вміст файлу security.txt

Централізована архітектура є найпростішою і найдавнішою формою C2-систем. У ній усі заражені вузли підключаються до одного або кількох центральних серверів, з яких отримують команди та передають результати їх

виконання. Для реалізації таких систем зазвичай використовуються протоколи HTTP(S), IRC або власні TCP-протоколи. Головною перевагою централізованих мереж є простота управління, можливість миттєвого надсилання команд усій мережі та контрольованість стану ботів. Недоліком виступає те що блокування або деавторизація основного сервера призводить до повного виведення бот-мережі з ладу. Типовими прикладами таких ботнетів були Zeus, SpyEye, BlackEnergy перших версій [7].

Peer-to-peer архітектура усуває єдину точку відмови, властиву централізованим моделям. У таких мережах кожен бот виконує роль одночасно клієнта і сервера, поширюючи команди між сусідніми вузлами. Завдяки цьому досягається висока живучість і складність виявлення. P2P-ботнети часто використовують протоколи, схожі на BitTorrent або Kad, а також власні криптографічні механізми перевірки достовірності команд [8]. Основними перевагами є автономність, стійкість до відключення окремих вузлів і відсутність фіксованого центру керування. Такі системи складні у розробці, мають обмеження на швидкість поширення команд та схильні до фрагментації у разі втрати частини вузлів. Серед відомих реалізацій можна виділити Storm, Sality, ZeroAccess [9].

Гібридна архітектура поєднує риси централізованих і P2P-систем. У ній виділяють декілька рівнів: центральні сервери керування, проміжні вузли-ретранслятори (proxy або relay) і кінцеві боти. Така ієрархічна структура забезпечує гнучкість, підвищену стійкість до ліквідації C2 і зменшує ризик швидкого виявлення. Крім того, проміжні вузли можуть виконувати функції шифрування трафіку, кешування або зміни маршрутів передачі даних. Гібридні моделі активно застосовуються у сучасних бот-мережах, зокрема в Emotet, TrickBot та сучасних версіях Necurs, які здатні автоматично переходити між централізованим і P2P-режимами при зміні мережевих умов [10].

У звітах CERT-EU, FireEye, Mandiant та інших аналітичних організацій прослідковується тенденція щодо переходу з централізованих архітектур до гібридних і P2P-систем через їхню вищу стійкість та адаптивність. Також

потрібно згадати, що сучасні C2-інфраструктури активно використовують хмарні платформи такі як AWS, Azure, Google Cloud та легітимні сервіси, наприклад GitHub, Matrix, Discord, тощо для командного обміну. Це суттєво ускладнює виявлення, оскільки трафік виглядає правдоподібним, а контрольні точки відсутні. Через це межа між звичайним користувацьким трафіком і шкідливим управлінням стає менш очевидною, що потребує нових підходів до моделювання та навчання детекторів у межах кіберполігону [11].

Сучасні архітектури бот-мереж можна оцінити за набором кількісних показників, які важливі для моделювання, виявлення та оцінки ризиків. Порівняння централізованих, P2P і гібридних архітектур C2 за основними характеристиками представлено нижче (табл. 1.1).

Таблиця 1.1 – HTTP методи

Архітектура / Метрика	Централізована	P2P	Гібридна	Пояснення метрики
Максимальна масштабованість (кількість вузлів)	10^4 – 10^5	10^4 – 10^6	10^4 – 10^6	Централізована обмежена потужністю серверів; P2P масштабується лінійно
Стійкість після відключення ключових серверів (%)	0–20 % (залежить від резервів)	60–99 %	30–90 %	% ботів, що лишаються керованими через 24 год після виведення частини інфраструктури
Час виявлення (години)	1–72 год	12–168 год	6–120 год	Середній час від початку активності до виявлення системою IDS/IPS
Латентність доставки команд (мс)	50–300 мс	100–2000 мс	50–1000 мс	Час від відправлення команди до її виконання на вузлі (медіана). Вимірюється через синтетичні повідомлення

Продовження таблиці 1.1

Архітектура / Метрика	Централізована	P2P	Гібридна	Пояснення метрики
Пропускна здатність на вузол (кбіт/с)	1–50 kbps	0.1–10 kbps	0.5–20 kbps	Середня смуга на вузол при штатному C2-трафіку (телеметрія та команди)
Частка вузлів, що зникають (% вузлів/год)	0.1–1 %/год	1–10 %/год	0.5–5 %/год	Частка вузлів, що щогодини відключаються/зникають
Частка успішного бутстрепу (%)	70–99 %	40–95 %	60–98 %	Відсоток спроб під’єднання нового вузла до C2, що завершилися успіхом
Кількість хопів до центру керування	1–3	2–10	1–6	Середня кількість мережевих ребер між ботом і джерелом команди (змінюється топологією)
Доля трафіку, що маскується під легітимний (%)	50–95 %	20–80 %	40–90 %	Частка пакетів, що використовують обфускацію/легітимні протоколи (HTTPS/DNS)
Час відновлення (RTO) після часткового виведення інфраструктури (години)	1–48 год	0.5–6 год	1–24 год	Час до відновлення керованості або досягнення попереднього рівня управління
Орієнтовні інфраструктурні витрати ($/місяць)	50–2000 $ (сервери, хостинг)	0–500 $	100–3000 $	Приблизні витрати на C2-сервера, редиректори, DDNS; сильно варіюють
Вимірювана складність виявлення (0–100)	30–70	60–95	50–90	Індекс, що комбінує TF (traffic fusion), TLS-fingerprint variety, DGA usage; чим більше тим складніше

Із зібраних метрик видно, що кожна архітектура має свої особливості: централізована C2 забезпечує низьку затримку і швидке розповсюдження

команд, але вразлива до виведення ключових серверів. P2P витримує значні втрати інфраструктури і масштабовано росте, проте має вищу затримку і складніший механізм виявлення. Гібридна архітектура знаходиться посередині, поєднуючи управлінність централізованої моделі з живучістю P2P. З практичної точки зору для навчально-дослідного полігону оптимальним є використання гібридної моделі, щоб відтворити широкий спектр сценаріїв (швидкі централізовані атаки та стійкі P2P-кампанії), а також виміряти показники, що цікавлять під час дослідження: час виявлення bootstrap rate та поведінкові патерни при різному ступені градації інфраструктури.

На основі отриманого аналізу було зроблено висновок, що для побудови дослідницького полігону доцільно використовувати гібридну архітектуру, оскільки вона поєднує керованість централізованої моделі та живучість P2P-структур, що допоможе відтворювати більш широкий спектр реалістичних сценаріїв командного управління ботами та виявлення аномальної активності.

1.2 Аналіз кіберполігонів та засобів моделювання бот-мережвебсайтів

Кіберполігон це платформа для практичного навчання, тестування систем захисту, симуляції атак і відпрацювання інцидентів без ризику для реальних мереж [12]. Головною їхньою особливістю є можливість створити контрольоване середовище з реалістичною інфраструктурою, де дослідники можуть спостерігати за поведінкою шкідливого програмного забезпечення, у тому числі бот-мереж, і перевіряти ефективність засобів виявлення (рис. 1.2).

Сучасні кіберполігони як інструменти для моделювання та дослідження шкідливої активності, включаючи роботу бот-мереж, значно відрізняються між собою за архітектурою, глибиною симуляції, рівнем інтерактивності та можливостями масштабування. Їхній розвиток зумовлений потребою організацій у регулярному тестуванні стійкості до атак, підготовці фахівців із кібербезпеки, а також необхідністю дослідження нових типів загроз, які постійно еволюціонують.

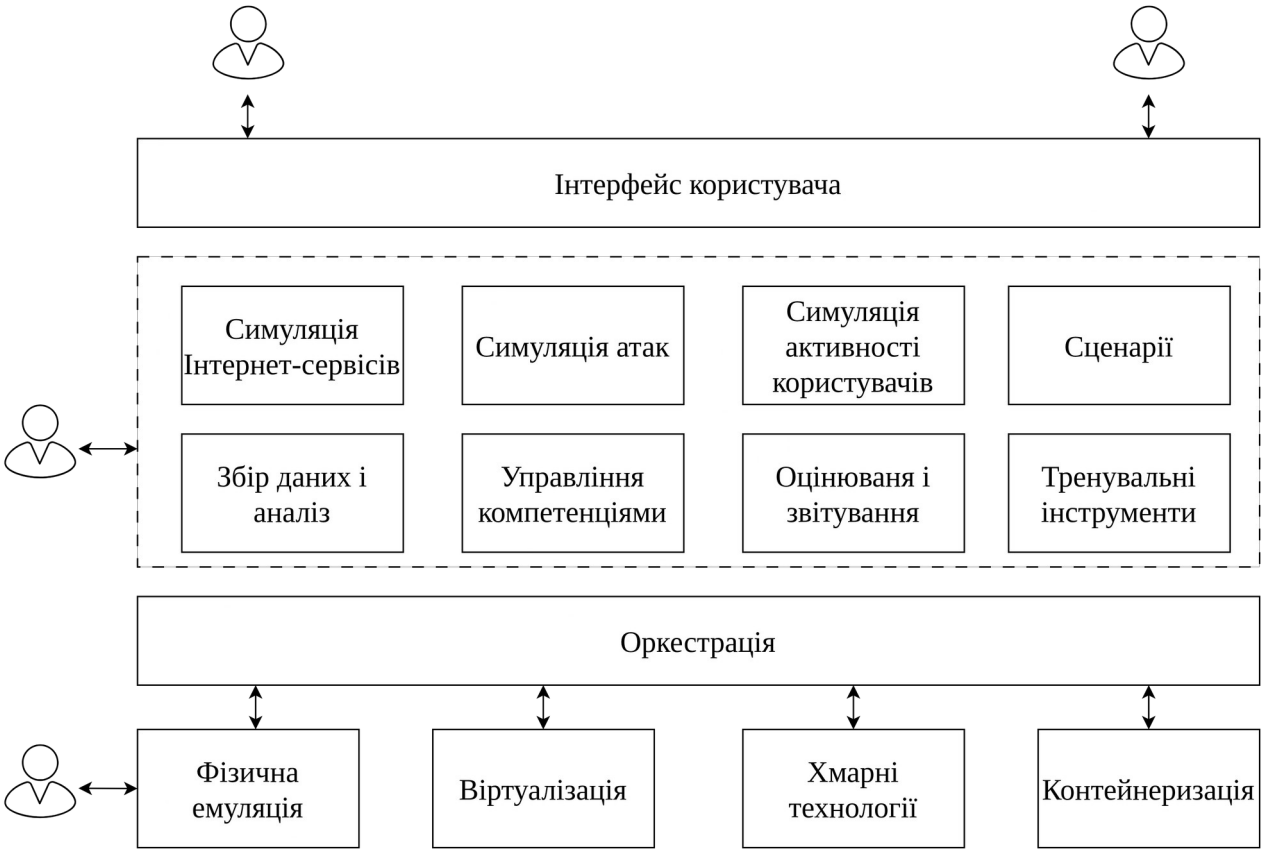


Рисунок 1.2 – Схема взаємодії клієнтської та серверної частин вебсайту

Кіберполігони можна умовно розділити на дві великі групи: комерційні комплексні рішення та відкриті або навчальні середовища, однак їх можливості у контексті моделювання бот-мереж суттєво відрізняються [13].

Комерційні кіберполігони на кшталт Cyberbit Range, IBM X-Force Command Center, Raytheon Cyber Range або корпоративних інфраструктур на основі кіберсимуляторів є орієнтованими на підприємства, урядові структури та організації з високими вимогами до інформаційної безпеки [14]. Їхні можливості включають створення складних сценаріїв атак із реалістичною поведінкою противника, симуляцію атак на рівні мережевої інфраструктури, операційних систем та корпоративних застосунків. Платформи побудовані у вигляді окремих дата-центрів або хмарних середовищ зі спеціалізованим обладнанням. Основною перевагою є їхня здатність моделювати масштабні, розподілені та стійкі до блокування ботнети з десятків тисяч вузлів, включно з реалістичними каналами командного керування, системами приховування

трафіку, розгалуженими інфраструктурними ланцюжками та складною мережею взаємодії.

Попри свою потужність, комерційні кіберполігони мають суттєві обмеження. Найвідчутнішим є їхня висока вартість, що робить такі рішення недоступними для навчальних закладів, студентських проєктів або невеликих компаній. Ще одним недоліком є закритість архітектури: користувач має доступ до інструментів, але не до внутрішньої логіки симуляцій або моделей бот-мереж, що унеможливлює глибоке кастомізоване дослідження поведінки або створення нетривіальних експериментів.

Відкриті або навчальні кіберполігони, такі як TryHackMe, Hack The Box, Metasploit Lab або FLARE VM, значною мірою орієнтовані на розвиток навичок пентестингу, аналізу вразливостей, реверс-інжинірингу та взаємодії з окремими системами [15]. На відміну від комерційних рішень, вони доступні широкій аудиторії, не вимагають складної інфраструктури та дозволяють користувачам взаємодіяти з реальними операційними системами у контрольованому середовищі. У таких платформах можна відпрацьовувати експлуатацію вразливостей, досліджувати шкідливе ПЗ або навіть розгортати мінімальні симульовані мережі. Однак їхня головна відмінність полягає у тому, що вони не моделюють складні багаторівневі кіберінфраструктури та практично не призначені для реалістичного створення бот-мереж.

Навчальні кіберполігони часто не підтримують сценарії командного керування, редирекцію трафіку, симуляцію великої кількості вузлів та моделювання життєвого циклу ботнетів. Вони більше зосереджені на завданнях типу capture-the-flag, пошуку вразливих служб або аналізу окремих зразків malware поза контекстом великих розподілених інфраструктур. У платформах Hack The Box або TryHackMe є можливість працювати в окремих ізольованих VPN-середовищах, однак обмеження масштабу та відсутність інструментів для керування мережевими сценаріями роблять їх непридатними для повноцінного вивчення динаміки DDoS-атак, поведінки ботів та реакції інфраструктури захисту [16].

На відміну від навчальних платформ, деякі вільні інструменти на зразок GNS3, EVE-NG або контейнерних середовищ можуть частково виконувати роль полігону для моделювання бот-мереж, але вони не є повноцінними кіберполігонами. Ці платформи слугують для створення складних мережевих топологій, запускати десятки або сотні вузлів, використовувати віртуальні маршрутизатори, NAT, TAP, VLAN і різні типи трафіку. Завдяки цьому дослідник може моделювати структуру ботнетів, перевіряти їхнє навантаження, створювати маршрути зі штучними затримками або обмеженнями пропускної здатності. Проте такі рішення не забезпечують автоматизованих сценаріїв атак, не мають власних систем реагування або симуляції поведінки цілі, а також вимагають значних ресурсів та досвіду [17].

Окрему категорію становлять спеціалізовані open-source рішення для дослідження шкідливого ПЗ, наприклад, Cuckoo Sandbox, CAPEv2, Remnux [18]. Вони дозволяють аналізувати ботнет-клієнти, але не масштабні мережеві інфраструктури. Для моделювання взаємодії сотень або тисяч ботів вони не підходять, хоча можуть доповнювати загальний кіберполігон, якщо використовувати їх для дослідження окремих компонентів шкідливої інфраструктури.

Критеріїв для порівняння кіберполігонів дуже багато, тому нижче наведено кілька таблиць, щоб краще орієнтуватися при створені власного полігону. Порівняльний аналіз основних комерційних та відкритих полігонів за інфраструктурою, масштабованістю, інтеграцією та кількістю сценаріїв представлено нижче (табл. 1.2, 1.3, 1.4, 1.5).

Таблиця 1.2 – Порівняння інфраструктури відомих кіберполігонів

Платформа	Тип розгортання	Підтримка віртуалізації / контейнеризації	Автоматизація
Cyberbit Range	on-prem/cloud	VMware, KVM / Docker	API
IBM X-Force Cyber Range	on-prem	VMware / немає інформації	API
Raytheon Cyber Range	on-prem	VMware, KVM / Docker	API
GIAC Cyber Range	cloud	KVM / Docker	API
Cyber Range Switzerland	Гібридний	VMware/Docker	API
TryHackMe	cloud	Немає / Docker	Так
Hack The Box	cloud	KVM / Docker	Так
Metasploitable Lab	on-prem	VirtualBox, Vmware / немає	Ні
FLARE VM	on-prem	Хост / немає	Ні
RangeForce	cloud	KVM / Docker	Так
Immersive Labs	cloud	Немає / на боці платформи	Так
OpenStack-based custom range	on-prem/cloud	KVM / Docker, Kubernetes	Так
MININET-based academic range	on-prem	Емульована віртуалізація / Docker частково	Так
Cuckoo Sandbox ranges	on-prem/cloud	KVM, VirtualBox / Docker	Так
AWS Cyber Range	cloud	KVM / Docker, ECS	Так
Azure Cyber Range (AZ Labs)	cloud	Hyper-V / Docker, AKS	Так
Google Cloud Cyber Range	cloud	KVM / Docker, Kubernetes	Так

Таблиця 1.3 – Порівняння масштабованості та мережевих можливостей кіберполігонів

Платформа	Максимальна кількість вузлів	Підтримка мережевих протоколів	Симуляція умов (lat/jit/loss	Ізоляція
Cyberbit Range	200+	TCP/UDP/ICMP/ BGP/OSPF	Так	Так
IBM X-Force Cyber Range	150+	TCP/UDP/ICMP	Часткова	Так
Raytheon Cyber Range	500+	TCP/UDP/BGP/ OSPF/MPLS	Так	Так
GIAC Cyber Range	100+	TCP/UDP/ICMP	Так	Так
Cyber Range Switzerland	250+	TCP/UDP/BGP/OSPF	Так	Так
TryHackMe	~50	TCP/UDP	Ні	Так
Hack The Box	~100	TCP/UDP	Частково	Так
Metasploitable Lab	10–20	TCP/UDP	Ні	Частково
FLARE VM	1-5	TCP/UDP	Ні	Ні
RangeForce	50-150	TCP/UDP	Так	Так
Immersive Labs	20-100	TCP/UDP	Ні	Так
OpenStack-based custom range	1000+	TCP/UDP/BGP/ OSPF/MPLS	Так	Так
MININET-based academic range	4096+	TCP/UDP	Так	Так
Cuckoo Sandbox ranges	5-50	TCP/UDP	Частково	Так
AWS Cyber Range	Необмежено	всі IP-протоколи	Так	Так
Azure Cyber Range (AZ Labs)	Необмежено	всі IP-протоколи	Так	Так
Google Cloud Cyber Range	Необмежено	всі IP-протоколи	Так	Так

Таблиця 1.4 – Порівняння телеметрії, інтеграції та безпеки кіберполігонів

Платформа	Типи журналів і даних	Інтеграція SIEM/EDR	Sandbox	Експорт даних
Cyberbit Range	Системні журнали, логи, PCAP	SIEM/EDR	Так	PCAP/JSON/ CSV
IBM X-Force Cyber Range	Системні події	SIEM	Так	PCAP/JSON
Raytheon Cyber Range	Повна телеметрія, PCAP	SIEM/EDR/ SOAR	Так	PCAP/JSON
GIAC Cyber Range	Системні журнали, логи	SIEM	Так	JSON/CSV
Cyber Range Switzerland	Системні журнали, мережеві дампи	SIEM/EDR	Так	PCAP
TryHackMe	Логи	Відсутня	Ні	Немає
Hack The Box	Системні журнали	Відсутня	Ні	Немає
Metasploitable Lab	Системні журнали	Відсутня	Ні	Локальні файли
FLARE VM	Логи	Відсутня	Так	Локальні файли
RangeForce	Повна телеметрія	SIEM	Так	JSON
Immersive Labs	Системні журнали	Частково	Ні	JSON
OpenStack-based custom range	Повна телеметрія	SIEM/EDR	Так	PCAP/JSON/ CSV
MININET-based academic range	Мережеві журнали	Відсутня	Ні	PCAP
Cuckoo Sandbox ranges	Логи	Відсутня	Так	JSON
AWS Cyber Range	CloudTrail/VPC Flow	SIEM	Так	JSON
Azure Cyber Range (AZ Labs)	CloudTrail/VPC Flow	SIEM	Так	JSON
Google Cloud Cyber Range	VPC Flow Logs	SIEM	Так	JSON

Таблиця 1.5 – Порівняння сценарії та контенту кіберполігонів

Платформа	Кількість сценаріїв	Типи сценаріїв	Власні сценарії	Наявність тренажерів
Cyberbit Range	100+	Атаки, SOC, інциденти	Так	Blue/red/purple
IBM X-Force Cyber Range	50+	Атаки, кризові симуляці	Ні	Blue
Raytheon Cyber Range	100+	Атаки APT, SOC, мережеві інциденти	Так	Blue/red
GIAC Cyber Range	30+	Сертифікаційні практикуми	Ні	Blue
Cyber Range Switzerland	40+	Інциденти, APT	Так	Blue/red
TryHackMe	500+	CTF	Ні	Red
Hack The Box	1000+	CTF	Ні	Red/purple
Metasploitable Lab	10	Експлуатація	Так	Red
FLARE VM	10+	Реверс-інженерія	Ні	Red
RangeForce	200+	SOC, аналіз шкідливого ПЗ	Так	Blue/purple
Immersive Labs	300+	CTF, SOC, криміналістика	Ні	Blue/purple
OpenStack-based custom range	~	Будь-які	Так	Blue/red/purple
MININET-based academic range	~	Мережеві симуляції	Частково	Немає
Cuckoo Sandbox ranges	10+	Аналіз шкідливого ПЗ	Частково	Blue
AWS Cyber Range	50+	Cloud security	Частково	Blue/red
Azure Cyber Range (AZ Labs)	50+	Cloud, AD атаки	Частково	Blue/red/purple
Google Cloud Cyber Range	30+	Cloud security	Частково	Blue

Порівняння інфраструктури показує, що комерційні рішення переважно використовують повноцінну віртуалізацію та контейнеризацію з керуванням через API. Вони спираються на VMware і KVM, забезпечуючи передбачуваність та стабільність. Хмарні рішення на базі AWS, Azure чи GCP вирізняються масштабованістю, але потребують окремої уваги до конфігурації та політик безпеки. Навчальні платформи та локальні лабораторії працюють спрощеною моделлю, орієнтованою на окремі завдання, а не комплексні мережеві сценарії. Найбільш універсальними залишаються кастомні полігони на OpenStack або Kubernetes, що дають повний контроль над інфраструктурою та дозволяють створювати середовища будь-якої складності [19].

Аналіз масштабованості демонструє різкий поділ між навчальними та інженерними полігонами. Комерційні системи масштабуются до сотень чи тисяч вузлів, підтримують маршрутизацію, MPLS та роботу зі складними топологіями. Хмарні полігони забезпечують практично необмежене масштабування та доступ до розширених мережевих параметрів. TryHackMe, Hack The Box та локальні стенди не підтримують такі можливості, що обмежує реалістичність складних сценаріїв [20].

За телеметрією та інтеграцією найсильніші платформи – комерційні та кастомні на OpenStack/Kubernetes. Вони підтримують збір усіх типів журналів, експорт PCAP, інтеграцію з SIEM, EDR, SOAR та роботу з повними мережевими дампами. Навчальні платформи зазвичай обмежуються базовими логами або взагалі не дають можливості отримати сирі дані, що не дозволяє повторювати аналіз інцидентів у повному циклі. Хмарні середовища надають детальну telemetria cloud-native формату, але вона менш гнучка для класичних мережевих експериментів [21].

Порівняння сценаріїв показує, що відкриті платформи переважно орієнтовані на CTF та окремі техніки, тоді як комерційні полігони на повноцінне моделювання APT, інцидентів та багатовекторних атак. OpenStack/Kubernetes дозволяють створювати власні сценарії будь-якого рівня

складності, включно з розподіленими атаками, SOC-тренуваннями та комбінованими сценаріями.

Разом з тим потрібно перерахувати найпоширеніші недоліки сучасних кіберполігонів: відсутність гарантованої відтворюваності середовища, ручне відновлення конфігурацій, складність моделювання взаємодії між великою кількістю вузлів, брак централізованого контролю над ВМ, слабка інтеграція з трафік-аналізом та затрати часу на розгортання. Через це тестування бот-мереж або складних розподілених систем стає повільним і важким для автоматизації.

Сучасні засоби віртуалізації та контейнеризації усувають більшу частину цих обмежень. Використання Docker, LXC чи Podman дозволяє запускати сотні легковагових інстансів, а Docker Compose чи Kubernetes автоматизувати їх життєвий цикл та керувати ними декларативно. Оркестрація спрощує масштабування, відтворюваність і ізоляцію, що особливо важливо для моделювання бот-мереж [22].

Окремо стоять хмарні платформи, які дають можливість створювати розподілені структури, наближені до реальних ботнетів, з різними географічними регіонами та IP-адресами. Проте використання шкідливого ПЗ у хмарі обмежується політиками провайдерів, тому такі системи здебільшого застосовуються для безпечних симуляцій або як зовнішні сегменти полігону.

Мережеві емулятори (GNS3, EVE-NG, Mininet, ns-3) дозволяють повністю контролювати мережеву топологію, симулювати затримки, втрати, перевантаження каналів, що робить їх корисними при побудові сценаріїв DDoS та аналізі поведінки великих груп вузлів [23]. Вони не підходять для виконання реального шкідливого ПЗ, але дають точний контроль над мережевою поведінкою.

Пісочниці та середовища для аналізу шкідливих програм (Cuckoo, CAPEv2, Remnux) корисні для дослідження окремих ботнет-клієнтів та їхніх протоколів, але не для моделювання цілісних мереж із сотень вузлів.

Також критичним елементом є засоби ізоляції: окремі VLAN, air-gap сегменти, внутрішні фаєрволи, policy routing та фільтри на рівні гіпервізора.

Вони забезпечують безпеку проведення експериментів зі шкідливим ПЗ та блокують будь-які потенційні витоки трафіку.

Сучасні дослідження фокусуються на поведінкових моделях ботнетів, тому потрібні гнучкі топології, підтримка різних C2-механізмів (централізованих, P2P, гібридних) та можливість змішувати реальний і синтетичний трафік. Для цього використовуються генератори трафіку, віртуальні маршрутизатори та системи збору телеметрії на базі Zeek, Elastic або Grafana [18].

На основі аналізу можна зробити висновок, що більшість полігонів орієнтовані на навчання або загальну кібербезпеку, але майже не пристосовані до дослідження бот-мереж як складних розподілених систем. Власний полігон має передбачати повну відтворюваність, автоматизацію, масштабованість до сотень вузлів, гнучку мережеву модель, централізовану телеметрію, прискорене відновлення середовища, а також модульність для швидкої зміни сценаріїв. Саме цих характеристик найчастіше бракує існуючим рішенням.

1.3 Аналіз відомих моделей бот-мереж

Розвиток сучасних бот-мереж демонструє еволюцію від вузькоспеціалізованих до багатофункціональних структур, здатних виконувати різноманітні завдання: від розповсюдження шкідливого ПЗ до розвідки, ексфільтрації даних і криптомайнінгу. Аналіз відомих моделей бот-мереж дозволяє простежити загальні закономірності розвитку інфраструктур керування, методів обходу детекції та характеру використання ботів у різних сценаріях. Знання цих моделей допоможе зробити реалістичні експерименти у навчально-дослідницькому полігоні.

Бот-мережі можна поділити на різні типи, виходячи з їхнього основного призначення, характеру інфраструктури та способів взаємодії між ботами і центрами керування. Кожен із типів має свій набір технічних ознак, метрик ефективності та унікальних сценаріїв застосування, тому їх доцільно розглядати окремо.

Бот-мережі, орієнтовані на розсилання спаму, зазвичай використовують масиви заражених пристроїв з відкритими SMTP-портами або релеями. Типовими представниками є Cutwail і Grum, які використовували централізовані C2-сервери з ротацією IP-адрес, щоб уникнути блокування. Основні показники таких бот-мереж включають обсяг вихідної пошти (до мільйонів листів за годину), кількість активних SMTP-реле та середній час життя бота до детекції.

Інший клас бот-мереж це проксі або тунельні системи, що забезпечують анонімність або обходи блокувань. Відомі приклади – Socks4/Socks5 ботнети, TrickBot, а також модулі проксі у QakBot. Ці мережі часто використовуються для нелегального рентингу трафіку, організації ботів як резидентних проксі або прихованого тунелювання C2-трафіку. Метрики, які варто досліджувати: середня затримка (latency) між вузлами, стабільність з’єднання, середній час активності бота у мережі, кількість одночасних клієнтів.

Моделі, орієнтовані на credential stuffing або brute-force атаки, використовують великі бот-флоти для автоматизованої перевірки облікових даних у вебсервісах. Представники цього класу – Muraena, Modlishka, Genesis. Вони використовують HTTP/HTTPS запити, іноді через проксі або Tor, щоб виглядати як легітимний трафік. Основні метрики: кількість запитів за секунду, частка успішних логінів, середній розмір трафіку на спробу автентифікації.

У сфері криптомайнінгу виділяються бот-мережі типу Smominru, DarkMiner, які використовують заражені сервери або IoT-пристрої для виконання обчислень. Ці ботнети не створюють великого мережевого шуму, але споживають значні ресурси процесора й енергії. Метрики включають hash rate (H/s), середнє навантаження CPU, кількість активних ботів і час життя інфекції.

Data-stealing моделі, наприклад Emotet або Azorult, орієнтовані на збір облікових даних, cookies, знімків екрану, ключових слів або конфігурацій браузера. Вони використовують гібридні C2-схеми і часто шифрують канали передачі даних. Метрики: обсяг ексфільтрованих даних, середній час реакції

C2, частота сесій передачі. Окремої уваги заслуговують мультифункціональні бот-мережі, такі як TrickBot або Emotet у пізніх версіях [24].

Після розгляду мультифункціональних і спеціалізованих типів варто окремо виділити бот-мережі, основною функцією яких є проведення DDoS-атак. DDoS-орієнтовані бот-мережі мають свої особливості керування, специфічні метрики та відмінності у поведінці порівняно з іншими класами, тому їх необхідно розглянути окремо.

DDoS-атаки класифікуються за способом впливу на ціль і за рівнем стеку, на який вони спрямовані. Для практичного моделювання у полігоні корисно виділяти три основні класи:

– Volumetric (об’ємні).
– Protocol (протокольні).
– Application-layer (прикладні).

Кожен клас характеризується відмінними цілями, метриками та очікуваною поведінкою бот-мереж і C2.

Volumetric – спрямовані на виснаження пропускної здатності каналу або ресурсів мережевої інфраструктури. Характерні показники: загальний трафік (Gbps), пік пак/с, кількість унікальних джерел. Типові приклади: UDP flood, DNS amplification у великому масштабі. У моделі volumetric-атаки важливі такі числові метрики: сумарний трафік (0.1–500+ Gbps залежно від масштабів), пік пакетів/сек (10^5–10^9 pps), частка spoofed-пакетів (%). C2 у таких кампаніях централізовано або гібридно генерує команду «включити флейту» і синхронізує хвилі трафіку.

Protocol (протокольні) – націлені на вразливості або поведінку мережевих протоколів (TCP, UDP, ICMP). Приклад: SYN flood, TCP fragmentation, ACK flood. Метрики: пакети/сек (10^4–10^8 pps), процент незавершених сесій, кількість напіввідкритих з’єднань (наприклад SYN backlog). C2 у таких сценаріях зазвичай координує параметри та маневрує джерелами для уникнення фільтрації [19].

Application-layer – спрямовані на виснаження логіки сервісу або ресурсів прикладного шару (HTTP, HTTPS, DNS-queries на рекурсивні інфраструктури, API-запити). Метрики: запити/сек на кінцеву точку (RPS), середній час обробки запиту (мс), відсоток помилок сервісу. Приклад: HTTP flood, Slowloris. Ці атаки складніше виявляти сигнатурно, бо трафік може імітувати легітимні запити; тому важлива поведінкова аналітика і модельні підходи [25].

Нижче в таблиці 1.6 представлено порівняння різних технік для проведення атак.

Таблиця 1.6 – Порівняння відомих технік DDoS-атак

Характеристика / Техніка	HTTP Flood	DNS Amplification	SYN Flood	UDP Flood	Memcached Amplification
Рівень OSI	L7	L3/L4	L4	L3/L4	L3/L4
Тип трафіку	HTTP/HTTPS	DNS UDP	TCP SYN	UDP	Memcached UDP
Основна мета атаки	Виснаження логіки сервісу / CPU / бекенду	Перевантаження каналу	Переповнення SYN backlog	Виснаження каналу	Масштабне перевантаження каналу
Пікове навантаження	10k-1M+ RPS	10-300+ Gbps	1-100+ Mpps	1–500+ Gbps	50-1500+ Gbps
Середня потреба у ботах	Середня (1k-50k)	Низька (100-3000)	Середня	Низька	Дуже низька (10-100)
Amplification factor	N/A	1:10-1:70	N/A	N/A	1:10k-1:51k
Частка підроблених IP (spoofing)	Низька - майже не застосовується	Висока	Висока	Висока	Висока
Вектор обходу детекції	Поведінкові аномалії, TLS-fingerprinting	IP-спуфінг, ротація резолверів	Спуфінг, високий PPS	Спуфінг та маскування під бродкаст	Спуфінг, неконтрольовані відкриті сервера
Ключові метрики	RPS, time-to-respond, error rate	pps, Gbps, AF	pps, SYN backlog utilization	pps, Gbps	AF, Gbps, кількість відкритих Memcached вузлів

Продовження таблиці 1.6

Необхідність C2-координації	Висока	Середня	Середня	Низька	Низька
Імітація легітимності	Висока	Низька	Низька	Дуже низька	Низька
Тип систем захисту, що реагують	WAF, L7 AI/ML anomaly detection	Anti-DDoS L3/L4, rate limiting	Stateful firewall / SYN cookies	L3/L4 anti-DDoS	Спеціалізовані edge-фільтри провайдерів

Порівняння різних технік DDoS-атак демонструє, що їх ефективність залежить не лише від обсягу трафіку, але й від здатності обходити сучасні засоби фільтрації. Базові методи, такі як SYN flood або UDP flood, незважаючи на високу пікову інтенсивність, сьогодні часто виявляються недостатньо результативними через широке впровадження L3/L4-фільтрації, stateful-захисту та автоматичних rate-limit механізмів на рівні провайдерів. У той самий час техніки, орієнтовані на L7 або на використання неконтрольованих сервісів з високим amplification factor, здатні створювати критично небезпечні сценарії навіть за умов незначної кількості ботів. Це чітко видно на прикладі Memcached amplification, де коефіцієнт підсилення може перевищувати десятки тисяч, а також на прикладі DNS amplification, який залишився одним із найстабільніших векторів за останнє десятиліття.

Для побудови реалістичного кіберполігону важливо розуміти, які техніки мають практичну цінність для моделювання сучасних загроз. Одним з найпоказовіших факторів стає зміщення акценту з простих volumetric-векторів до комбінованих атак. У багатьох реальних бот-мережах спостерігається тенденція до гібридизації. Mirai в класичній формі використовувала SYN flood, ACK flood та UDP flood як основні механізми, але її численні модифікації додали GRE flood, L7 HTTP flood та зміни поведінки для адаптації під захист. Mēris підняв рівень атак на порядок, використовуючи нестандартно велике значення RPS під час L7-навантаження через вразливі MikroTik-пристрої.

Реальні кейси Mēris показали, що правильно сформований HTTP flood може перевантажити навіть системи з високопродуктивними балансувальниками, якщо атака достатньо варіативна і використовує складні шаблони, включно з динамічними URL, псевдовипадковими заголовками, TLS-характеристиками та варіативністю HTTP-методів.

Ще одна важлива категорія це protocol abuse. Прийоми типу DNS або NTP amplification залишаються домінуючими у volumetric-сегменті, оскільки вони дозволяють атакувальному трафіку в десятки разів перевищувати реальний вихідний обсяг, а механізм IP-спуфінгу дає змогу працювати майже без участі самих ботів. Memcached-атаки, хоча й стали рідкіснішими після масового закриття відкритих серверів, залишаються найяскравішим прикладом того, що невелика кількість вузлів може сформувати трафік у сотні гігабіт і більше.

Однак найбільш перспективними сьогодні є гібридні та інтелектуальні техніки, які дозволяють моделювати реальні тенденції розвитку бот-мереж. Використання зашифрованого трафіку сприяв появі сценаріїв, де L7-атаки виглядають як легітимна взаємодія клієнта з сервером. Наприклад, складні HTTP/2-flood-сценарії, які активно використовують multiplexing, rapid reset та інші особливості сучасних протоколів, вони майже не виявляються традиційними методами rate-limiting, вимагають поведінкової аналітики, а тому є надзвичайно важливими для тестування оборонних систем у кіберполігоні. Ще одна тенденція це поява бот-мереж, що не покладаються на численні IoT-пристрої. Деякі сучасні активні екосистеми використовують хмарні ресурси, орендовані VPS, контейнерні середовища або навіть зламані Kubernetes-кластери [20].

У кінці можна сказати, що найкращий результат дає поєднання кількох технік. Обов’язковими до включення є хоча б один високорівневий вектор (наприклад, HTTP flood із можливістю налаштовувати поведінку ботів), один класичний volumetric-вектор (DNS amplification або UDP flood) і один протокол-орієнтований вектор, на кшталт SYN flood або TCP connection

exhaustion. Додатково можна включити множину нестандартних сучасних методів: HTTP/2 rapid reset, адаптивний L7 flood з TLS-фінгерпринт-імітацією, а також сценарії з використанням IP-спуфінгу для імітації бот-мереж типу Mirai або характерних volumetric-атаки рівня провайдерів.

1.4 Постановка задачі

Стрімке зростання складності бот-мереж та автоматизованих систем координації атак створює потребу у контрольованих середовищах, здатних точно відтворювати поведінку шкідливих екосистем. Реальні мережі не підходять для таких досліджень: навіть мінімальний витік трафіку чи помилка ізоляції може спричинити небажані взаємодії з інтернет-інфраструктурою. Тому виникає необхідність у власному полігоні, де дослідник контролює кожен елемент від мережевих сегментів до моделей поведінки вузлів.

Потреба у створенні кіберполігону випливає з низки об’єктивних обмежень існуючих рішень [1]. Комерційні платформи орієнтовані насамперед на навчальні сценарії і містять заздалегідь підготовлені вправи без можливості повноцінної реконструкції шкідливих екосистем. Їх внутрішня архітектура недоступна, а поведінка змодельованих ботнетів часто спрощена, що не дозволяє включати власні експериментальні моделі або перевіряти роботу рідкісних чи кастомних C2-протоколів. Навчальні ресурси на кшталт TryHackMe чи HackTheBox забезпечують симуляцію вразливостей, але не створюють умов для повного мережевого експерименту з вузлами, що генерують реалістичний DDoS-трафік, використовують спуфінг IP-адрес або функціонують у режимі підвищеного навантаження. Відсутність повного контролю над мережевими просторами, гіпервізором, ядром та механізмами ізоляції виключає можливість моделювання бот-мереж на рівні системних викликів, стеків протоколів та міжвузлової маршрутизації.

Отже проєкт вимагає створення власного полігону, де дослідник контролює кожний компонент: від мережевого шлюзу до правил файлової системи.

У межах задачі враховуються технічні, часові та ресурсні обмеження, а також вимоги повної ізоляції, відтворюваності конфігурацій та безпечності експериментів. Результати мають бути придатними для подальшого аналізу й розширення моделі.

1.5 Висновки до першого розділу

У першому розділі роботи було здійснено аналіз сучасних підходів до побудови бот-мереж та механізмів реалізації DDoS-атак. Показано, що еволюція шкідливих екосистем відбувається у напрямку децентралізації, активного використання шифрування, прихованих або нетрадиційних каналів керування, а також адаптивної поведінки ботів у відповідь на захисні дії. Це призводить до ускладнення як виявлення атак, так і їх експериментального дослідження.

Результати аналізу свідчать, що ефективний кіберполігон для дослідження DDoS-атак повинен підтримувати динамічні топології, різні моделі взаємодії між вузлами та можливість швидкої зміни сценаріїв. Статичні або заздалегідь визначені конфігурації не дозволяють адекватно відобразити реальні умови функціонування сучасних бот-мереж.

Огляд існуючих комерційних і навчальних платформ показав, що більшість з них орієнтовані на типові сценарії пентесту, реагування на інциденти або навчальні завдання. Їх спільним недоліком є обмежений рівень контролю над середовищем, відсутність доступу до внутрішньої архітектури та неможливість моделювання комплексної поведінки шкідливих компонентів у повністю ізольованій мережі.

2 МОДЕЛЬ ТА ПОБУДОВА КІБЕРПОЛІГОНУ

2.1 Формування вимог до кіберполігону

Аналіз архітектур і моделей бот-мереж показав, що дані системи характеризуються складною структурою командного керування, великою кількістю взаємодіючих вузлів і потенційно небезпечним трафіком, тому дослідження їх поведінки має здійснюватися виключно у контрольованому середовищі. Основна мета створення кіберполігону полягає в тому, щоб забезпечити можливість експериментів зі шкідливими компонентами без ризику для зовнішньої інфраструктури та гарантувати стабільну роботу навіть за умов навмисного навантаження системи. Його архітектура повинна базуватися на принципах конфіденційності, цілісності та доступності (CIA-тріада), а також з урахуванням специфіки дослідження шкідливих мережевих структур [26].

Конфіденційність гарантує, що навіть за умови роботи зі шкідливими компонентами або створення трафіку підвищеного ризику, жодна інформація не вийде за межі експериментальної зони. На рисунку ці елементи відображені як ключові складові, що створюють замкнене середовище з контрольованим доступом.

Критерій цілісності визначає здатність кіберполігону зберігати послідовність свого стану незалежно від кількості проведених експериментів. Він охоплює можливість відтворення попередніх конфігурацій, перевірку узгодженості системних параметрів і фіксацію поточних станів, щоб уникнути накопичення похибок або неконтрольованих змін.

Доступність гарантує, що системи керування, експериментальні вузли та інструменти спостереження залишатимуться активними, незалежно від того, яке навантаження моделюється. На рисунку цей критерій відображено як основу, що підтримує працездатність усіх елементів полігону.

Варто також згадати про критерій спостережності, який забезпечує прозорість процесів, що відбуваються всередині полігону. Він включає

механізми збору та аналізу подій, що генеруються мережевими та системними компонентами, а також структуровану сегментацію мережі, яка дозволяє відокремлювати різні домени активності та простежувати взаємодію між ними.

У сукупності ці критерії формують цілісну модель вимог до кіберполігону, що дозволяє відтворювати складні сценарії поведінки бот-мереж у контрольованих і безпечних умовах. Задіяні принципи охоплюють не лише класичні аспекти захисту інформації, але й додаткові характеристики, необхідні для роботи зі шкідливими компонентами, симуляцією масових атак та дослідженням внутрішніх механізмів сучасних ботнетів. Наведена на рисунку 2.1 схема узагальнює цю структуру.

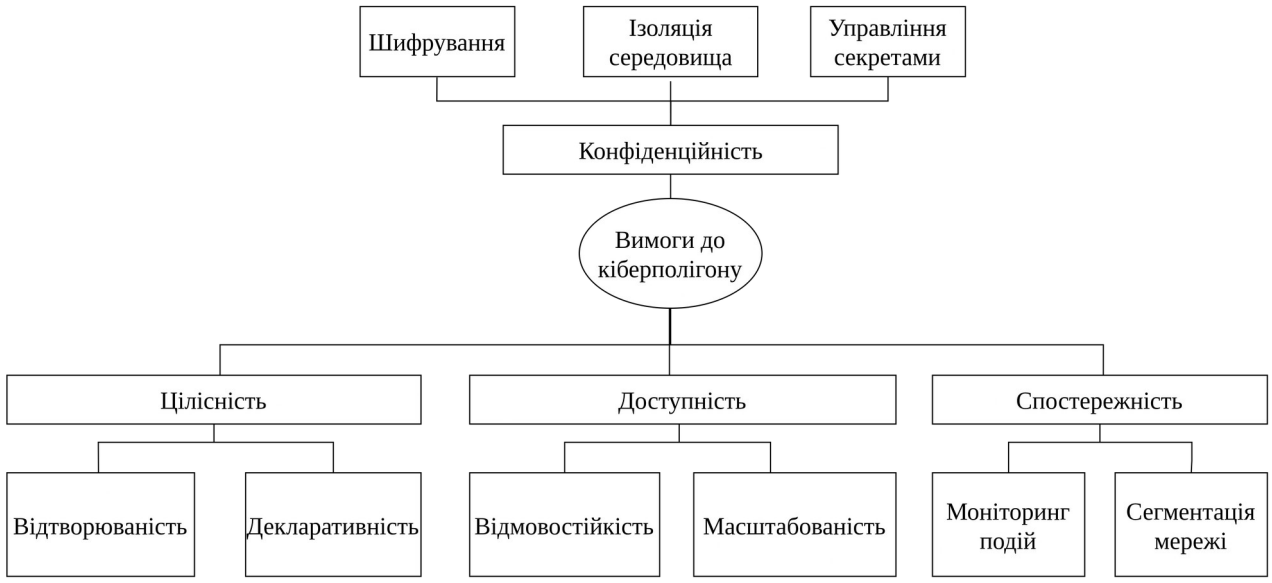


Рисунок 2.1 – Схема концептуальних вимог до кіберполігону

Шифрування є ключовим механізмом захисту даних у межах кіберполігону та спрямоване на те, щоб унеможливити доступ до інформації сторонніми суб’єктами навіть у разі фізичного компрометування середовища. Воно гарантує, що всі артефакти експериментів, журнали, конфігурації та моделі бот-мереж залишаються конфіденційними та не можуть бути відтворені поза межами контрольованої інфраструктури. Шифрування також забезпечує стабільність результатів дослідження: будь-які дані, що містять потенційно

небезпечні компоненти, не можуть бути зчитані або видозмінені за межами полігону, що мінімізує ризик витоку або несанкціонованої модифікації.

Ізоляція середовища визначає межі, у яких функціонує кіберполігон, і забезпечує повне відокремлення тестового простору від зовнішніх ресурсів. Вона гарантує, що моделювання шкідливої активності, поведінки бот-мереж чи навмисного мережевого навантаження не може вплинути на продуктивні системи. Ізоляція охоплює розмежування мережевих маршрутів, блокування невизначених каналів зв’язку та створення автономних доменів взаємодії між компонентами полігону. Завдяки цьому дослідник отримує абсолютний контроль над комунікаціями та може проводити експерименти з будь-якими сценаріями без ризику виходу трафіку за межі середовища.

Управління секретами відповідає за захист конфіденційних параметрів, що використовуються під час роботи полігону: ключів, токенів, конфігураційних значень та внутрішніх ідентифікаторів. Основна мета цього критерію це виключити можливість витоку критичних даних під час проведення експериментів або під час доступу до результатів дослідження. Правильна система управління секретами забезпечує контроль доступу, аудит операцій із чутливою інформацією та обмеження її розповсюдження, що є фундаментальним для безпечного функціонування полігону.

Відтворюваність визначає здатність полігону повертатися до попереднього безпечного стану та забезпечувати відтворення тих самих умов під час повторних експериментів. Цей критерій особливо важливий у дослідженнях бот-мереж і шкідливих компонентів, де зміни стану системи можуть бути непередбачуваними. Відтворюваність гарантує, що кожен експеримент можна виконати у точно таких самих умовах, а також що будь-яке пошкодження середовища або некоректна поведінка шкідливого програмного забезпечення не призведе до тривалих наслідків [27].

Декларативність означає опис конфігураційної логіки системи у вигляді формальних правил, що дозволяє мінімізувати помилки, пов’язані з ручними змінами. У контексті кіберполігону це означає, що вся інфраструктура від

мережевої топології до параметрів системного середовища визначається у вигляді структурованих специфікацій, які можна відтворювати у будь-який момент. [28]

Відмовостійкість забезпечує стабільність роботи полігону навіть за умов навмисного навантаження або внутрішніх збоїв. Під час моделювання бот-мереж часто створюються ситуації зі значним споживанням ресурсів, що може призвести до деградації продуктивності. Відмовостійкість передбачає наявність механізмів, які гарантують, що окремі помилки не вплинуть на загальну працездатність системи.

Масштабованість визначає здатність полігону розширювати свою інфраструктуру шляхом додавання нових вузлів, розширення мережевих сегментів або переходу до складніших моделей бот-мереж. Цей критерій є важливим, оскільки сучасні ботнети можуть включати сотні або тисячі пристроїв, і дослідницьке середовище повинно мати змогу пропорційно збільшувати обчислювальні ресурси, мережеві межі та обсяг телеметрії.

Моніторинг подій забезпечує повний контроль над поведінкою компонентів полігону та фіксує всі процеси, що відбуваються під час експериментів. Цей критерій включає збирання системних журналів, відстеження мережевих взаємодій, аналіз аномалій і контроль за внутрішніми подіями. Наявність детального моніторингу дозволяє досліднику реконструювати перебіг експерименту, ідентифікувати причини відхилень та оцінювати ефективність моделюваних захисних механізмів.

Сегментація мережі відповідає за структуризацію внутрішніх зв’язків між компонентами полігону. У контексті дослідження бот-мереж сегментація дає змогу розділяти ботів, командні центри, жертви та захисні вузли, що покращує точність експериментів і дає змогу моделювати реалистичні сценарії з розподіленою інфраструктурою.

Сукупність наведених критеріїв формує цілісну основу, на якій будується безпечний, передбачуваний і придатний до досліджень кіберполігон,

який дозволяє точно відтворювати складні сценарії поведінки бот-мереж без ризику для зовнішньої інфраструктури.

2.2 Розробка архітектури кіберполігону

Незалежно від обраних технологій або конкретних засобів віртуалізації, базова архітектура полігону будується по принципу розмежування зон відповідальності та чіткого поділу компонентів на сегменти, що виконують різні функції. Можна побачити нижче загальну логічну схему, де кожен сегмент відокремлений від інших, але зберігає можливість взаємодії відповідно до сценарію експерименту (рис. 2.1). В даній архітектурі можна сформувати передбачувані траєкторії трафіку, контролювати навантаження та запобігати випадковому виходу небажаної активності за межі тестового середовища.

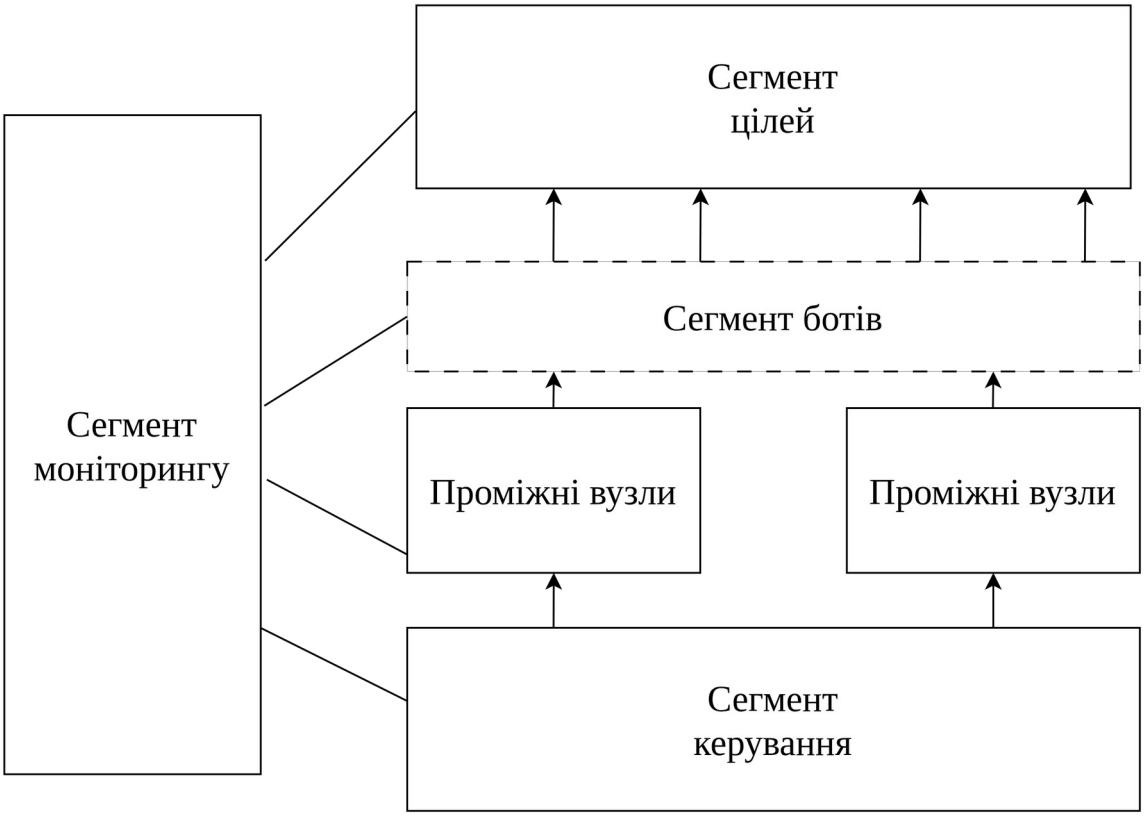


Рисунок 2.2 – Схема архітектури кіберполігону

З рисунку вище можна побачити, що схема включає кілька ключових зон: сегмент керування, сегмент проміжних вузлів, сегмент ботів, сегмент цілей та окремий сегмент моніторингу. Усі ці зони взаємодіють між собою за

принципом сходинкової моделі, де сигнал передається від елементу керування до проміжних вузлів, після чого потрапляє до ботів, а згодом у бік цілі. У зворотному напрямку рухаються телеметричні дані, журнали мережевої активності та результати аналізу. На полігоні одночасно поєднано симуляцію загрози й інструменти для оцінки її впливу, тому можна використовувати полігон як для експериментів так і для навчання.

Основа полігону це сегмент керування, тут розміщено сервер командного центру. Він виступає центральною точкою координації експерименту, тобто надсилає команди ботам, отримує інформацію про їхній стан і задає параметри атаки. У загальній схемі цей сегмент перебуває в окремій ізольованій мережі, яка не має прямого підключення до інших частин полігону, окрім ретельно визначених каналів через проміжні вузли. Використовуючи ізоляцію мережі, зберігається контрольованість експерименту та запобігаються небажані випадкові взаємодії між командним центром і іншими елементами мережі. Оператори підключаються до цього сегменту через заздалегідь визначений канал керування, і саме ця точка є стартовою у всіх сценаріях. За рахунок чого можна статично контролювати доступ і забезпечувати прозорість усіх дій, спрямованих на ініціацію атаки.

Другим елементом є сегмент проміжних вузлів, які виконують роль буферу між командним центром та ботами. Вони призначені для створення природних шарів абстракції, тому що у реальних умовах бот-мережі рідко спілкуються з командним центром напряму. Тому прийнято рішення, що буде доцільно відтворити цю поведінку і в межах полігону. У загальній схемі ці вузли можуть розташовуватися у власній ізольованій мережі, мати обмежений набір відкритих портів і взаємодіяти з командним сегментом та з сегментом ботів. Проміжні вузли потрібні для забезпечення структурованості маршруту та відслідковують цікаві при навчанні параметри: фільтрація трафіку, властивості маршрутизації та вплив мережевого шуму.

Сегмент ботів являє собою найбільш чисельну групу вузлів, оскільки саме вони генерують трафік атаки та імітують поведінку розподіленої мережі

заражених пристроїв. У простішому варіанті бот може бути окремою віртуальною машиною або контейнером, який виконує заздалегідь визначені сценарії в ідеальному варіанті це має бути фізичний пристрій. Боти розташовані у певному логічному сегменті, куди мають доступ лише проміжні вузли. Для експериментів боти ізольовані один від одного та напряму не впливають на зовнішні вузли, окрім визначених цілей, що задовільняє вимоги захищеного полігону.

Сегмент цільових систем представляє вузли, які отримують навантаження під час DDoS-атаки, що дає змогу досліджувати, як змінюються їхні характеристики під впливом різних типів трафіку. У загальній схемі вони відокремлені від сегмента ботів за допомогою проміжних мережевих пристроїв, які дозволяють контролювати інтенсивність трафіку, маршрути та методи доставки пакетів.

П’ятий ключовий компонент загальної схеми полігону – сегмент моніторингу. У цьому середовищі розміщується система аналізу трафіку, журналів та поведінкових аномалій, яка отримує дані з інших сегментів полігону. Основна мета цього сегменту полягає в тому, щоб зібрати повну картину подій під час експерименту. Сюди надсилається трафік з мережевого дзеркала або портів, призначених для спостереження. Цей сегмент ізольований, але з’єднаний з іншими частинами полігону таким чином, щоб отримувати повні метадані, не впливаючи при цьому на сам експеримент, тобто інші компоненти ніяк не повинні знати про існування цього сегменту. За допомогою цього сегменту буде проводится оцінювання потужності DDoS та ефективність виявлення.

Логічна структура полігону побудована так, щоб відтворити повну картину під час проведення DDoS-атаки, яку можна аналізувати з будь-якого боку. Тому на схемі зазвичай виділяють три основні домени: домен атакувальної інфраструктури, домен проміжних вузлів і домен цільових ресурсів. Перший домен включає C2, операторські робочі станції та допоміжні засоби для управління сценаріями атаки. Другий охоплює редиректори, які

виступають посередниками між центром керування і ботами або між ботами та ціллю. Третій домен представлений серверами, що мають реалістичні сервіси, які використовуються в якості жертви та піддаються навантаженню під час експерименту. Кожен домен має чітко визначені маршрути та правила фільтрації, які забезпечують контрольований перебіг експерименту та можливість його відтворення.

У загальній схемі полігону критично важливою є роль плаваючих маршрутів та симульованих ISP-мереж, які дозволяють імітувати реалістичний Інтернет-ланцюг. Замість того, щоб під’єднувати реальні зовнішні канали, у полігоні використовується внутрішня маршрутизація між сегментами, де кожна підмережа має унікальні характеристики. Наприклад, мережа ботів може бути розташована у різних підмережах, що мають різну затримку або пропускну здатність, може бути використано для того, щоб подивитися, як змінюється результат атаки залежно від умов мережі. Схема маршрутизації включає статичні або динамічні маршрути, залежно від того, наскільки експеримент націлений на симуляцію поведінки провайдерів [29].

Гнучкість схеми одна з головних властивостей кіберполігону. Вона має дозволяти швидко змінювати маршрутизацію, перепризначати ролі вузлів, перемикати ботів між різними серверами, обирати різні типи редирекції або моделювати деградацію ботнету (втрата частини вузлів, зміна пропускної здатності, збільшення затримок). Для цього мережеві зв’язки між сегментами побудовано так, щоб їх можна було перепідключати за кілька хвилин, не руйнуючи загальну інфраструктуру. Топологія не статична. Вона має поводитися як лабораторний аналог реальної мережі оператора або хмарного середовища, де зміни маршрутизації, фільтрації або QoS є природним явищем.

Динамічність платформи також дозволяє моделювати поведінку реальних бот-мереж: від структурних змін командному центрі до перерозподілу навантаження між ботами. Наприклад, одні боти можуть працювати в умовах штучно збільшеної затримки, інші в умовах обмеженої пропускної здатності

або періодичної втрати пакетів. C2 може змінювати частоту команд, переходити між протоколами або імітувати приховані канали.

Коли структура та логічні взаємозв’язки полігону зрозумілі, можна перейти до розробки сценаріїв атак та їх виявлення.

2.3 Сценарії DDoS-атак

На кіберполігоні DDoS-атака відтворюється як керований процес: C2 запускає сценарій, редиректори розподіляють навантаження, а боти формують інтенсивний трафік до цілі. Загальний алгоритм подано на рисунку 2.3.

Рисунок 2.3 – Схема алгоритму розгортання DDoS-атаки

Використовуючи порівняння технік DDoS-атак з попереднього розділу, було обрано 3 найкращі для створення власних сценаріїв. Система моделювання DDoS-навантажень побудована на основі трьох окремих сценаріїв, кожен із яких відтворює різний тип взаємодії:

Перший сценарій, зображений на рисунку 2.4, базується на найпростішій послідовності операцій, у якій акцент зроблено на моделювання інтенсивного трафіку на транспортному рівні [30].

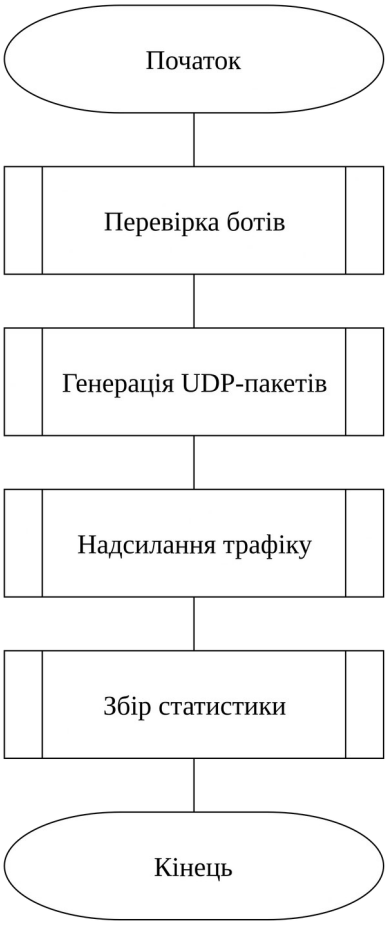


Рисунок 2.4 – Схема алгоритму L4-атаки

Початковий етап алгоритму передбачає запит до всіх активних ботів із метою перевірки їх доступності та фіксації стану. Якщо бот підтверджує готовність, він додається до актуального списку учасників сценарію. Далі всі боти отримують однакові параметри: адресу цілі, набір необхідних числових значень для регулювання інтенсивності, тривалість виконання та інші керувальні показники. Головна операція полягає у генерації великої кількості

транспортних пакетів, що надсилаються без складних варіацій або динамічної зміни структури.

Другий сценарій (рис. 2.5) відтворює складнішу модель, у якій навантаження формується на основі запитів, що змінюють свою структуру під час виконання.

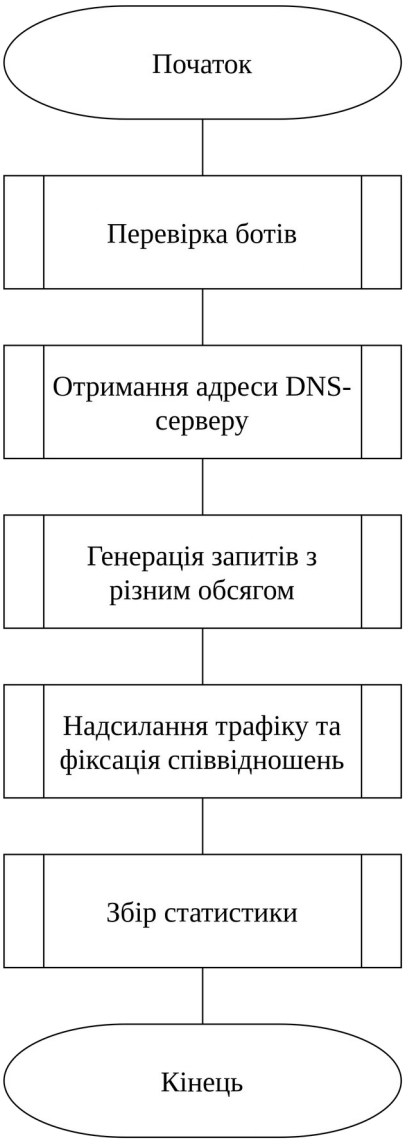


Рисунок 2.5 – Схема алгоритму DNS-amplification атаки

Алгоритм починається з аналогічної перевірки ботів, далі йде послідовне надсилання запитів різного обсягу та характеру, що дозволяє моделювати непередбачуваний клієнтський трафік. Боти формують запити, у яких змінюються як розмір, так і частота, а також фіксують співвідношення успішних та невдалих відповідей. Завершальний блок алгоритму збирає

статистику щодо реакції сервера на змінні параметри, що дає змогу оцінити його стійкість до нерівномірних потоків та навантаження, яке постійно змінюється.

Третій сценарій, відображений на рисунку 2.6, реалізує найбільш адаптивну модель. Основна частина алгоритму полягає в циклічній взаємодії, у якій боти встановлюють з’єднання, генерують змінні запити та фіксують реакцію цілі на повторне відкриття або раптове закриття потоків [30].

Рисунок 2.6 – Схема алгоритму HTTP/2 Rapid Reset атаки

Щодо керування ботами, схема підключення ботів реалізована як окремий керований алгоритм ініціалізації, який виконується до запуску будь-якого сценарію атаки, можна побачити на рисунку 2.7. Основна ідея полягає у виявленні нових вузлів без постійного агентського каналу та без підтримання

довготривалих з’єднань. Процес запускається з боку C2, який ініціює перевірку через проміжні вузли. Редиректори розсилають короткі контрольні запити у відповідні сегменти, не зберігаючи станів і не підтримуючи сесій. Бот, отримавши такий запит, формує мінімальну відповідь, що містить унікальну ознаку, закладену на етапі створення образу. Відповідь передається назад тим самим шляхом без додаткової обробки.

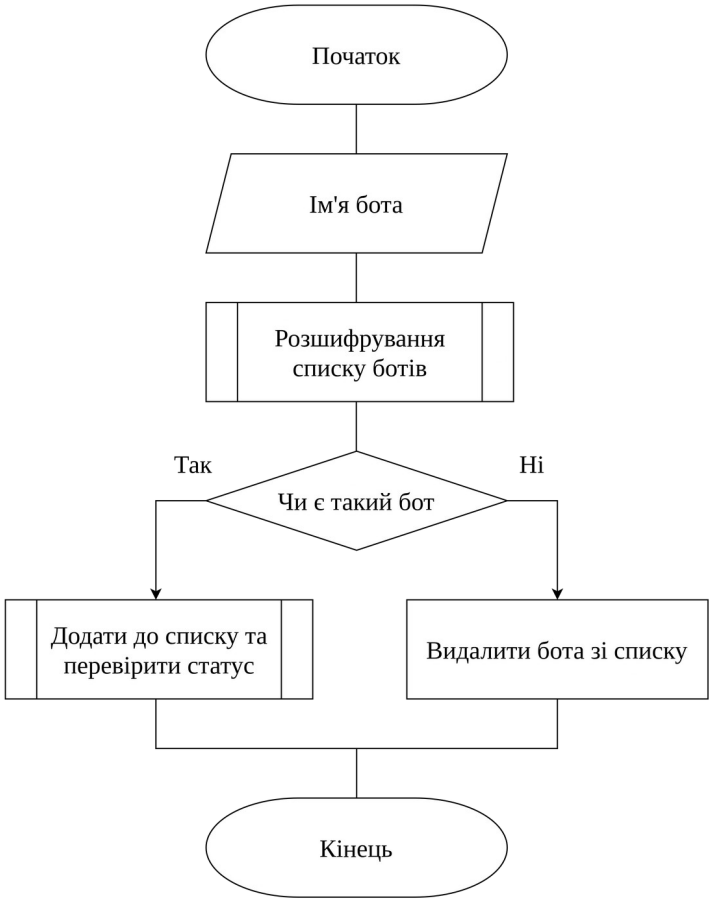


Рисунок 2.7 – Схема алгоритму підключення ботів

На стороні C2 відповіді обробляються послідовно: кожен маркер зіставляється з внутрішнім реєстром активних вузлів. Якщо бот з таким ідентифікатором уже присутній у списку, інформація про нього видаляється. Якщо маркер не знайдено, вузол додається як новий учасник полігону з початковим статусом. Весь процес є одноразовим для кожного запуску та не потребує постійної присутності бота в керуючому каналі.

Процес ініціюється C2 шляхом запуску сценарію перевірки, який послідовно звертається до проміжних вузлів (рис. 2.8). Редиректори передають тестові запити ботам та очікують відповіді у межах заданого проміжку часу. Кожен бот відповідає коротким сигналом, що підтверджує готовність до участі у сценарії.

Рисунок 2.8 – Схема алгоритму перевірки ботів

На цьому етапі завершено формування логіки підключення та перевірки ботів, а також описано їх місце у загальній структурі сценаріїв DDoS-атак. Наступний розділ присвячено системам моніторингу, аналізу трафіку та виявленню подій, що виникають у результаті виконання розроблених сценаріїв.

2.4 Моніторинг та виявлення DDoS на кіберполігоні

Процес моніторингу та виявлення DDoS-атак у межах кіберполігону був організований як послідовна багаторівнева аналітична система, що поєднує детальне спостереження за трафіком, аналіз поведінкових характеристик та виявлення. Загальна структура системи була відображена на схемі моніторингу та виявлення атак, представленій на рисунку 2.9.

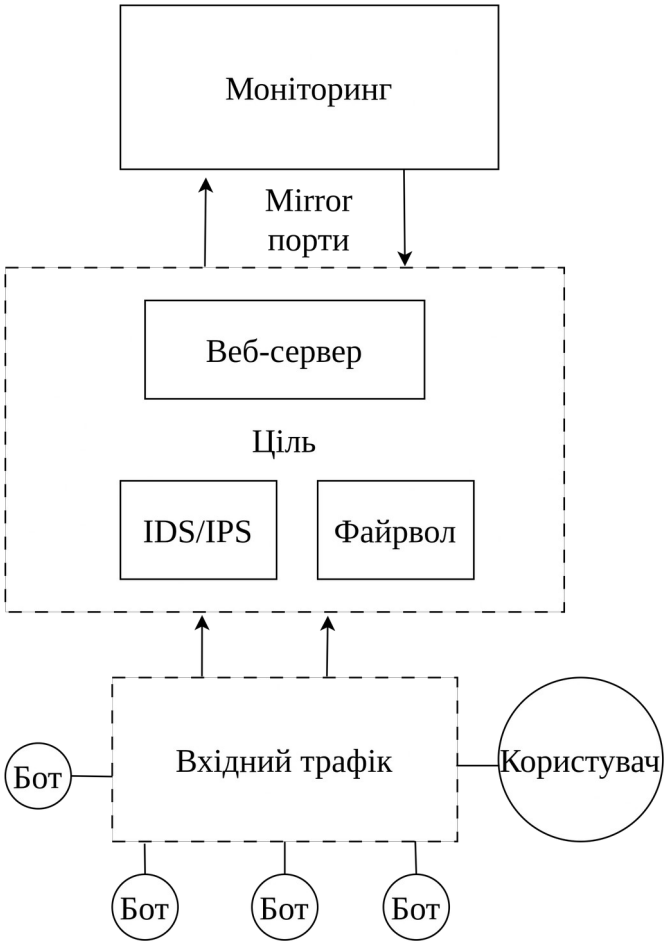


Рисунок 2.9 – Схема моніторингу та виявлення атак

У центрі моніторингової підсистеми знаходився механізм збору мережевих подій, що приймає дані з вузлів, редиректорів та серверів-цілей. Кожен мережевий елемент передає потоки пакетів до аналізатора, де відбувається нормалізація параметрів, побудова векторів характеристик і фіксація часових маркерів. У межах цього етапу система формує структурований опис навантаження: інтенсивність, варіативність між пакетами,

співвідношення ключових протокольних прапорів, повторюваність заголовків, стабільність TCP або UDP-полів. Таке попереднє опрацювання є фундаментальним, оскільки дозволяє зменшити шум і створити однаковий формат даних для подальших детекторів.

Подальший аналіз ґрунтувався на поєднанні сигнатурних та поведінкових методів. Сигнатурний рівень був відповідальним за виявлення простих шаблонних атак, таких як спроби масового встановлення неповних TCP-з’єднань, характерні послідовності пакетів для SYN flood, повторювані UDP-запити фіксованого формату або типові структури для DNS та NTP amplification. У цих випадках система порівнювала структуру заголовків із відомими патернами, а виявлені збіги позначалися як події підвищеного ризику. Важливою властивістю цього рівня було швидке спрацьовування: сигнатури дозволяли зафіксувати очевидні атаки без складних обчислень [31].

Другий рівень, поведінковий, був ключовим у межах полігону, оскільки забезпечував виявлення сценаріїв, що не відповідали жодним відомим шаблонам. Алгоритм оцінював динаміку зміни параметрів у часі, а саме темп появи нових з’єднань, збільшення частоти запитів, нехарактерні коефіцієнти повторення, різкий спад або зростання ентропії портів, стабільність значень у структурі HTTP-запитів та ознаки машинної генерації трафіку. Наприклад, надмірна рівномірність пауз між пакетами розцінювалася як ознака синхронізованої атаки від групи ботів, а не як природна активність реальних користувачів. Поведінковий аналіз також враховував зміну time-to-first-byte, інтенсивність L7-запитів, повторюваність ідентичних заголовків у коротких часових вікнах і дисбаланс між вихідним та вхідним трафіком. Саме ця частина логіки дозволяла виявляти атаки, приховані за редиректорами або NAT, де велика кількість ботів могла виходити в мережу під єдиною або невеликою кількістю IP-адрес.

На рисунку 2.10 представлено загальну структуру алгоритму виявлення атак, де чітко видно, як кореляційні механізми поєднують дані з різних джерел. Йдеться про аналіз синхронності сплесків активності між кількома вузлами,

схожості структури пакетів, одночасної появи аномалій на кількох рівнях протоколів та відповідності часових залежностей. У реальному ботнеті велике навантаження формується за рахунок сукупної активності багатьох малих джерел, тому саме кореляція дозволяла розпізнати ознаки координації навіть тоді, коли кожен бот окремо генерував незначний трафік.

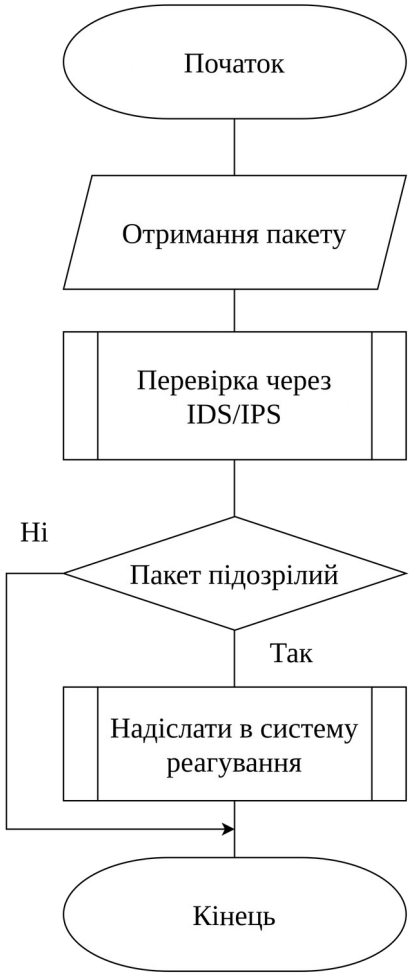


Рисунок 2.10 – Схема алгоритму виявлення атак

Кожному рівню відповідали різні допустимі механізми впливу. Наприклад, на ранніх етапах система застосовувала мінімально інвазивні заходи: підвищення деталізації логування, додатковий збір метрик, короткочасний перерахунок характеристик трафіку та повторне оцінювання на відповідність порогам. Метою цього етапу було впевнитися, що спостережувана аномалія не є одиничним відхиленням або результатом легітимного піку активності.

На останньому рівні рівні ескалації застосовувалися найбільш рішучі заходи: блокування цілих підмереж, зміна маршрутів для розвантаження окремих сегментів, відключення підозрілих каналів або примусове переведення трафіку через додаткові контролюючі вузли (рис. 2.11). Такі дії допускалися лише після підтвердження атак багаторівневою кореляцією та тоді, коли менш радикальні заходи не дозволяли стабілізувати систему. Таким чином, логіка реагування працювала не як статичний набір правил, а як динамічна модель, що адаптується до складності атаки та мінімізує ризик хибнопозитивних блокувань.

Рисунок 2.11 – Схема алгоритму реагування

Окрему роль у системі відігравав алгоритм перевірки швидкості, наведений на рисунку 2.12. Його завданням було формування обґрунтованих обмежувальних параметрів, необхідних для стабілізації навантаження під час атаки, коли різні потоки створювали нерівномірний або поступово зростаючий тиск на ресурси сервера. Алгоритм аналізував інтенсивність трафіку, часові інтервали між пакетами, частку повторюваних заголовків, ритм запитів та співвідношення між легітимними і підозрілими операціями. На основі цих даних він визначав, з якою швидкістю слід знижувати пропускну здатність для окремих потоків, не завдаючи шкоди коректним користувачам.

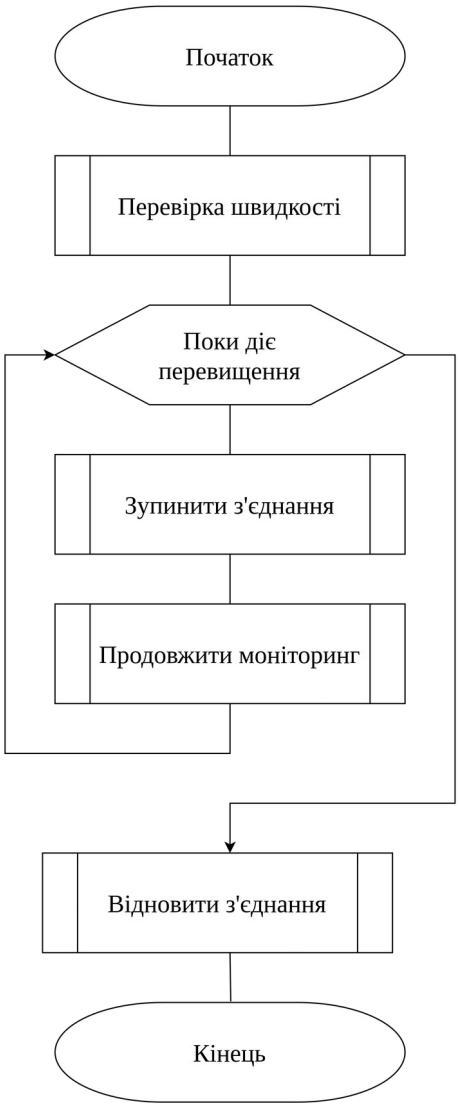


Рисунок 2.12 – Схема алгоритму перевірки швидкості

У ситуаціях, коли трафік не демонстрував різкого пік, але показував поступове накопичення навантаження, модуль перевірки швидкості виявляв приховані ризики виснаження ресурсів. Алгоритм у таких випадках дозволяв коригувати межі повторних запитів залежно від їхнього профілю, а також визначав, чи слід додавати додаткові обмеження на рівні портів або протоколів [32]. Це забезпечувало плавне зниження навантаження без різких обмежень, які могли б негативно вплинути на якість обслуговування легітимного сегмента.

Взаємодія між системою виявлення і системою реагування працювала як замкнений контур безперервного моніторингу, що забезпечував оперативне прийняття рішень без втручання оператора. Кожен зафіксований відхил був перевірений на кількох рівнях: сигнатурному, поведінковому і кореляційному. Лише після підтвердження аномалії алгоритм реагування ініціював відповідні заходи, а модуль перевірки швидкості оптимізував їх інтенсивність. Такий механізм дозволив уникнути надмірного блокування, зберегти стабільність полігону та забезпечити коректне функціонування інфраструктури навіть під час інтенсивних атак.

2.5 Висновки до другого розділу

Другий розділ присвячено побудові кіберполігону та обґрунтуванню його архітектури, функціональних компонентів і дослідницьких можливостей. На основі аналізу вимог і доступних технічних ресурсів було визначено оптимальну модель, що поєднує реалістичність, безпеку, масштабованість та можливість багаторазового відтворення експериментів. Розроблена архітектура кіберполігону забезпечує логічну сегментацію інфраструктури, ізоляцію ключових ролей та контрольоване керування міжмережевими взаємодіями.

3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТИ НА КІБЕРПОЛІГОНІ

3.1 Компоненти кіберполігону та їх роль

Відповідно до побудованої архітектури у кожному сегменті кіберполігону знаходяться налаштовані пристрої, які є компонентами кіберполігону. Далі буде подано детальний аналіз елементів полігону, їх технічних ролей, логічних зв’язків, трохи покрито тему їх налаштування та технічних характеристик, а також призначення у сценарії моделювання DDoS-атаки. Те як взаємодіють об’єкти на кіберполігоні можна побачити нижче на рисунку 3.1.

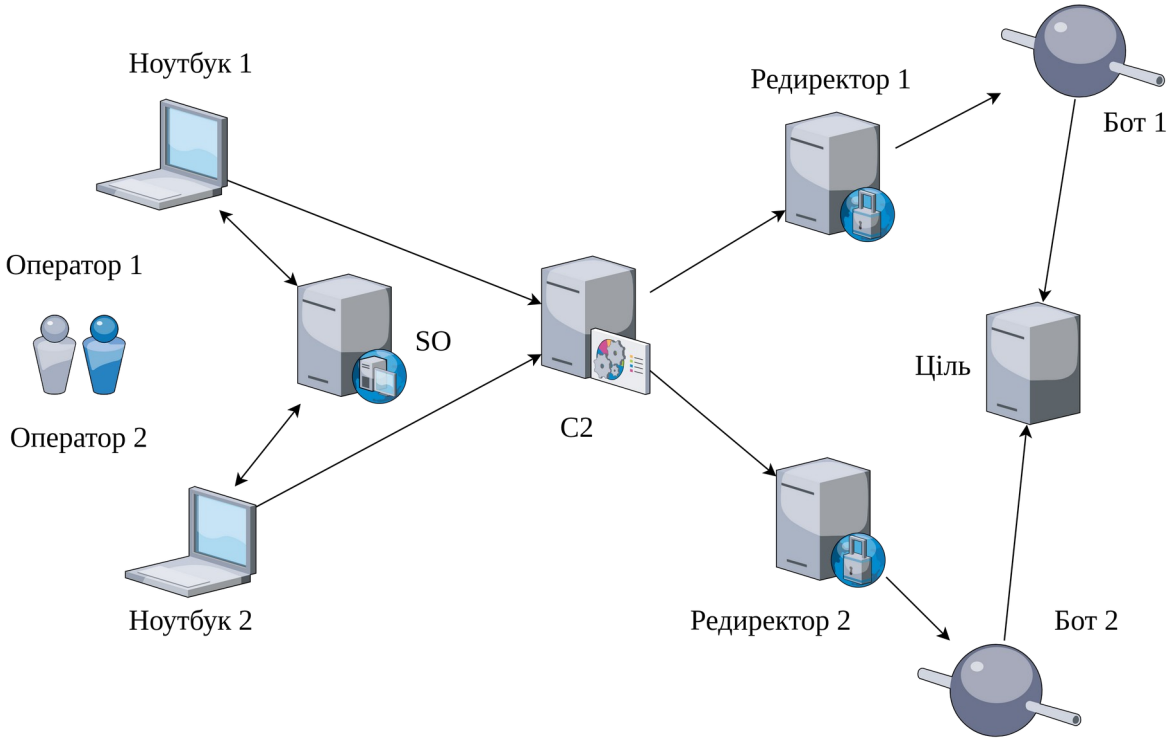


Рисунок 3.1 – Архітектурна засобу аналізу безпеки вебсайтів

З аналізом архітектури завершено, тому можна оприділитися з платформою для реалізації кіберполігону. Було обрано Proxmox як головний сервер гіпервізора, на якому розгорнуто всю інфраструктуру, він виступає точкою узгодженого управління ресурсами [33]. Вибір зумовлений тим, що це рішення забезпечує цілісну інтеграцію контейнеризації та повноцінної віртуалізації KVM та пропонує стабільний інтерфейс керування, зручну

систему бекапів, підтримку кластеризації та сегментації мереж на рівні віртуальних мостів. З технічної точки зору Proxmox надає найбільш контрольований та гнучкий спосіб організації навчального середовища: він працює на Debian, підтримує апаратну віртуалізацію VT-x, рознесення IOMMU, а також надійну інтеграцію з ZFS та Btrfs. Файлову систему btrfs також було обрано тому що у лабораторному середовищі важливо мати швидкі знімки, дешеві в плані дискового простору та придатні для швидкого відкату станів системи, наприклад під час тестування поведінки інфікованих хостів або моделювання підвищеного навантаження.

Для наочності та кращого представлення, що представляє із себе кожен компонент кіберполігону було побудовано таблицю 3.1.

Таблиця 3.1 – Компоненти кіберполігону

Компонент	Технічні характеристики	Тип середовища	Операційна система
Платформа кіберполігону	2× Xeon Silver 4210, 128 ГБ ОЗП, 4× SSD NVMe 1 ТБ (RAID10), 4× HDD 4 ТБ (RAID5), 10GbE NIC	Bare-metal сервер	Proxmox VE
C2	8 ГБ ОЗП, 2 ядра, 128 ГБ SSD	VM	NixOS
Редиректор 1	4 ГБ ОЗП, 2 ядра, 40 ГБ SSD	VM	Alpine Linux
Редиректор 2	4 ГБ ОЗП, 2 ядра, 40 ГБ SSD	VM	Alpine Linux
Група ботів 1	8 ботів: кожен 2 ГБ ОЗП, 1 ядро, 20 ГБ диску	Віртуальні машини	Windows / Linux
Група ботів 2	8 контейнерів: 256–512 МБ ОЗП, 1 vCPU, образ 200–400 МБ	Docker	Linux
Група ботів 3	8 контейнерів: 512 МБ ОЗП, 1 ядро, 2–4 ГБ дискового простору	LXC	Linux
Група ботів 4	8 пристроїв IoT: ARM CPU, 256–1024 МБ ОЗП, 4–16 ГБ eMMC	Фізичні пристрої	Windows / Linux

Продовження таблиці 3.1

Компонент	Технічні характеристики	Тип середовища	Операційна система
Моніторинг	16 ГБ ОЗП, 4 ядра, 200 ГБ SSD	VM	Security Onion
Оператор 1	8 ГБ ОЗП, 4 потоки, 256 ГБ SSD	Фізичний пристрій	GNU/Linux або Windows
Оператор 2	8 ГБ ОЗП, 4 потоки, 256 ГБ SSD	Фізичний пристрій	GNU/Linux або Windows
Ціль	8 ГБ ОЗП, 4 ядра, 100 ГБ SSD	VM або bare-metal	Rocky Linux

Командний центр C2 реалізовано як окрему віртуальну машину з операційною системою NixOS [34]. Вибір NixOS зумовлений декларативною моделлю конфігурації, яка забезпечує відтворюваність середовища, контроль змін і можливість швидкого повернення до визначеного стану. Ресурси C2 обмежені двома ядрами та 8 ГБ оперативної пам’яті, оскільки на ньому не виконуються обчислювально інтенсивні задачі. Основне навантаження формується за рахунок запуску сценаріїв, роботи з таблицями станів ботів, логування та керування з’єднаннями. Дисковий простір виділений з запасом для зберігання журналів виконання сценаріїв, проміжних результатів експериментів та службових артефактів.

Редиректори розгорнуті як окремі легкі віртуальні машини на базі будь-якого стандартного Linux-дистрибутиву, наприклад Alpine. Вимоги до операційної системи мінімальні, оскільки на вузлах працюють лише базові мережеві інструменти, реверс-проксі та фільтрація трафіку. Виділення по 4 ГБ оперативної пам’яті та двох ядер забезпечує стабільну обробку потоків без утворення вузьких місць під час пікових навантажень. Наявність двох незалежних редиректорів використовується для розподілу трафіку, ускладнення топології та моделювання багатоточкової взаємодії між ботами і ціллю. Кожен редиректор працює автономно, без обміну станами з іншим.

Група ботів першого типу представлена повноцінними віртуальними машинами. Така конфігурація використовується для моделювання вузлів із повноцінним мережевим стеком, окремим ядром та власною файловою системою. Виділення 2 ГБ оперативної пам’яті і одного ядра на кожен бот забезпечує коректну роботу мережевих інструментів під тривалим навантаженням. Ця група застосовується для сценаріїв, де важлива поведінка повноцінної операційної системи, включно з управлінням процесами, буферами та системними лімітами.

Друга група ботів реалізована у вигляді Docker-контейнерів [15]. Контейнерна модель використовується для імітації великої кількості однотипних вузлів з мінімальними накладними витратами. Обмежений обсяг оперативної пам’яті та компактні образи відповідають типовим умовам хмарних або масових клієнтських середовищ. Контейнери швидко запускаються, легко масштабуються та підходять для сценаріїв, у яких основний акцент зроблено на генерацію прикладного трафіку.

Третя група ботів побудована на базі LXC-контейнерів [16]. Вони займають проміжне положення між повноцінними віртуальними машинами та Docker-контейнерами. LXC використовується для моделювання середовищ з більш тісною інтеграцією з хостом і ближчою до класичної Linux-системи ізоляцією. Виділення окремого файлового простору дозволяє зберігати локальні журнали та тестувати сценарії з тривалішим життєвим циклом ботів.

Четверта група ботів представлена фізичними IoT-пристроями на базі ARM або x86 архітектури. Ці вузли застосовуються для моделювання нестабільної поведінки, характерної для вбудованих систем: обмежені ресурси, нестійка мережева взаємодія, затримки обробки. Наявність фізичних пристроїв у полігоні дозволяє включати у сценарії фактори, які неможливо коректно відтворити лише у віртуальному середовищі.

Система моніторингу реалізована у вигляді окремої віртуальної машини з розгорнутим комплексом Security Onion. Виділені ресурси орієнтовані на зберігання великих обсягів мережевих даних, обробку PCAP та виконання

кореляційного аналізу. Security Onion використовується як центральний вузол збору телеметрії, журналів IDS та мережевих метаданих без участі у керуванні полігоном.

Операторські вузли представлені фізичними машинами з універсальними операційними системами. Вони використовуються для підключення до C2, запуску сценаріїв, аналізу результатів та збору експериментальних даних. Розділення на два операторські вузли дозволяє моделювати різні ролі користувачів або паралельну роботу з полігоном.

Ціль реалізована як окремий сервер на базі Rocky Linux. Вибір дистрибутиву зумовлений його стабільністю та типовістю для серверних середовищ. Цей вузол виступає об’єктом атак, приймає навантаження та генерує журнали прикладного і системного рівнів, що використовуються для подальшого аналізу.

Програмне забезпечення кіберполігону підбиралося з орієнтацією на контрольованість середовища, прозорість роботи компонентів та мінімальну кількість прихованих механізмів. Кожен вузол використовує лише ті технології, які безпосередньо задіяні у виконанні його функцій, без розгортання фонових агентів або сервісів загального призначення.

Архітектура полігону свідомо уникає клієнт-серверних систем керування ботами, централізованих агентів або постійно активних демонів. Усі перевірки доступності, ініціалізація ботів і передача керувальних команд виконуються через базові мережеві утиліти операційної системи.

Вибір файлових систем і механізмів захисту дисків підпорядкований вимогам цілісності даних, відтворюваності стану та мінімізації побічних артефактів. Для керівних і критичних вузлів використовується повне шифрування накопичувачів, що унеможливлює доступ до конфігурацій і журналів поза межами середовища експерименту. На вузлах із високою динамікою стану застосовуються файлові системи, орієнтовані на знімки та ізоляцію змін, що спрощує повернення до початкового стану між експериментами. Список технологій та інструментів подано нижче (табл. 3.2).

Таблиця 3.2 – Програмне забезпечення компонентів кіберполігону

Компонент	Безпека диску / Файлова система	Додаткові технології	ПЗ / Інструменти
Платформа кіберполігону	Шифрування накопичувачів, ZFS/Btrfs	Кластеризація, резервне копіювання, ізоляція ВМ	Віртуалізація KVM/QEMU, LXC, керування мережами
C2	Повне шифрування, Btrfs subvolumes	Декларативне керування, зберігання секретів, незмінність системи	Інструменти керування ботами, журнали, мережеві утиліти
Редиректор 1	Шифрування диску, Btrfs	Реверс-проксі, ізольовані мережеві простори	Nftables, TCP/UDP forwarder, логуючі утиліти
Редиректор 2	Шифрування диску, XFS/Btrfs	Реверс-проксі, ізольовані мережеві простори	Проксі-інструменти, аналізатор трафіку
Група ботів 1	Шифрування образів, Ext4/Btrfs	Віртуальні мережі, автоматизоване розгортання	Bash-скрипти генерації трафіку, базові мережеві утиліти
Група ботів 2	Overlay/GraphFS	Контейнери, мережеві неймспейси	Скрипти в контейнерах, легковагові утиліти
Група ботів 3	LUKS/ZFS (host-level)	Легковагова віртуалізація, профілі ізоляції	CLI-скрипти, інструменти навантаження
Група ботів 4	Вбудоване шифрування	TPM, апаратна ізоляція	Утиліти для мережевих тестів, прості генератори трафіку
Моніторинг	Шифрування логів, XFS/Btrfs	Інтегрований стек моніторингу та аналізу	Suricata, Zeek, Wazuh, Elastic stack
Оператор 1	Шифрування диску	Контрольований доступ, безпечний користувацький профіль	SSH-клієнт, інструменти аналітики
Оператор 2	Шифрування диску	Ізольовані робочі середовища	Мережеві інструменти, панелі керування полігоном
Ціль	Шифрування диску	Декларативне керування, зберігання секретів, незмінність системи	Веб-сервер Apache, IDS/IPS, файрвол

На рівні платформи кіберполігону використовується повне шифрування накопичувачів та сучасні файлові системи з підтримкою знімків і контролю цілісності. ZFS або Btrfs застосовуються для ізоляції даних віртуальних машин, резервного копіювання та відновлення станів. Віртуалізація на базі KVM/QEMU і LXC забезпечує розділення середовищ без змішування мережевих і дискових контекстів. Керування мережами інтегровано безпосередньо у платформу, без використання зовнішніх SDN-контролерів.

Командний центр C2 розгорнутий на NixOS з повним шифруванням диска та використанням Btrfs subvolume для поділу системних і прикладних даних. Декларативна модель керування системою зменшує кількість неявних станів і спрощує відтворення конфігурації. Для керування секретами використовується sops-nix, оскільки він інтегрується безпосередньо в механізм збірки системи та працює з зашифрованими файлами конфігурації без окремого шару абстракції. У порівнянні з agenix, sops-nix не вимагає жорсткої прив’язки до ключів на рівні хоста та краще підходить для сценаріїв з частими змінами конфігурації.

Механізм impermanence використовується для відокремлення стану системи від змінних даних, що критично для експериментального середовища. Системний стан відновлюється при кожному перезапуску, тоді як журнали та артефакти експериментів зберігаються окремо. Flakes обрані як основний спосіб керування конфігураціями через чітку фіксацію залежностей і версій. Home-manager інтегровано для керування користувацьким середовищем операторів на C2 без змішування системних і користувацьких налаштувань.

Підключення операторів до C2 здійснюється через OpenVPN у поєднанні з SSH. OpenVPN обраний замість WireGuard через його стабільну роботу в складних мережевих умовах, підтримку TCP-режиму та можливість використання єдиного каналу поверх обмежених або фільтрованих мереж. Для операторського доступу важлива передбачувана поведінка з’єднання, а не

мінімальні затримки. SSH використовується як основний інструмент керування без додаткових проксі або агентів.

Редиректори мають повне шифрування дисків і використовують Btrfs або XFS для стабільної роботи з журналами. На цих вузлах розгорнуті реверс-проксі, які виконують роль проміжної точки між ботами та ціллю. Реверс-проксі використовується для керування потоками, агрегації трафіку та розділення логіки маршрутизації і генерації навантаження. Nftables застосовується як основний механізм фільтрації та обробки пакетів через єдину підсистему ядра без застарілих компонентів. Для переспрямування трафіку використовуються прості TCP/UDP forwarder’и та утиліти логування без складних фреймворків.

Групи ботів усіх типів не використовують спеціалізовані клієнти, агентів або демонів. Уся взаємодія будується на базових інструментах операційної системи: netcat, socat, стандартні утиліти генерації трафіку та shell-скрипти. Такий підхід усуває приховану логіку, зменшує кількість залежностей і робить поведінку ботів повністю спостережуваною. У віртуальних машинах і контейнерах застосовуються стандартні файлові системи з шифруванням на рівні образів або хоста. Для фізичних IoT-пристроїв використовується вбудоване шифрування та апаратна ізоляція, включно з TPM, за наявності підтримки.

Система моніторингу побудована на інтегрованому стеку Security Onion із шифруванням логів та використанням XFS або Btrfs для зберігання великих обсягів даних. Suricata використовується як IDS/IPS на основі сигнатур і потокового аналізу. Zeek застосовується для побудови протокольної та поведінкової моделі трафіку. Wazuh доповнює стек подіями з хостів, а Elastic Stack використовується для зберігання, індексації та візуалізації. Кожен компонент працює незалежно, без спільного агента керування.

Операторські вузли використовують повне шифрування дисків і ізольовані робочі середовища. На них встановлено лише клієнтські інструменти

доступу, аналізу та перегляду даних, без участі у генерації трафіку або маршрутизації.

Сервер цілі розгорнутий з повним шифруванням диска та мінімальним набором сервісів. Як веб-сервер обрано Apache через його зрілу модель обробки з’єднань, детальні журнали та широке покриття сигнатурами IDS. Apache часто використовується у корпоративних середовищах, що підвищує репрезентативність результатів. Nftables використовується як основний файрвол через прозору інтеграцію з ядром. Suricata та Zeek на цілі застосовуються для фіксації подій без активного втручання у трафік.

Загальна архітектура програмного забезпечення кіберполігону орієнтована на мінімалізм, відсутність прихованих агентів та використання стандартних інструментів операційної системи.

3.2 Мережеве середовище для запуску експериментів

Налаштування мережевого середовища починається з визначення базової IPv6-архітектури кіберполігону. Вибір IPv6 є усвідомленим і принциповим, оскільки сучасні дослідження мережевих атак усе частіше стикаються з обмеженнями, закладеними в IPv4-моделі. Попри тривалий глобальний перехід на IPv6, більшість навчальних і практичних матеріалів досі зосереджуються виключно на IPv4, що формує розрив між реальними умовами експлуатації та лабораторними сценаріями. Окремим аргументом на користь IPv6 є спрощення моделі з’єднання великої кількості вузлів, а також найбільш ефективний спосіб передачі пакетів на сьогоднішній день через мультикаст.

Усі сегменти ізольовано через окремі VLAN, кожен з яких прив’язаний до власного Linux bridge на Proxmox. Для платформи кіберполігону створено mgmt-br0 з тегом VLAN 10 та infra-br1 з тегом VLAN 20, на які призначено префікси fd00:10::/64 та fd00:20::/64 відповідно, що забезпечує окремі канали керування та службових операцій. Маршрутизацію між сегментами вимкнено на рівні хоста, а передача трафіку здійснюється лише через визначені router-VM у GNS3. таблиці 3.3.

Таблиця 3.3 Інформаційні потоки та маршрутизація на кіберполігоні

Компонент	Інформаційні потоки	VLAN / Мережевий інтерфейс	Проміжки IP-адрес (IPv6)
Платформа кіберполігону	Керування гіпервізором, трафік управління, логування хостів	mgmt-br0 (VLAN 10), infra-br1 (VLAN 20)	fd00:10::/64, fd00:20::/64
C2	Команди керування ботами, журнали сценаріїв, телеметрія від ботів	c2-net0 (VLAN 30)	fd00:30::/64
Редиректор 1	Вхідний трафік від ботів, вихідний до цілі, журнали переспрямування	redir1-net0 (VLAN 40)	fd00:40::/64
Редиректор 2	Вхідний трафік від ботів, вихідний до цілі, статистика потоків	redir2-net0 (VLAN 41)	fd00:41::/64
Група ботів 1	Локальна телеметрія, вихідний транспортний/прикладний трафік	bots-vm-net (VLAN 50)	fd00:50::/64
Група ботів 2	Вихідні HTTP/UDP запити, контейнерні логи	bots-dock-net (VLAN 51)	fd00:51::/64
Група ботів 3	Поведінкові шаблони трафіку, контрольні відповіді	bots-lxc-net (VLAN 52)	fd00:52::/64
Група ботів 4	Фізичні апаратні відхилення, нестабільний трафік	bots-phy-net (VLAN 53)	fd00:53::/64
Моніторинг	Повні PCAP, логи IDS/IPS, кореляційні дані, статистика потоків	so-net0 (VLAN 60), so-mirror (VLAN trunk)	fd00:60::/64
Оператор 1	Панель керування, збір результатів, SSH-доступ	operator1-net (VLAN 70)	fd00:70::/64
Оператор 2	Звітність, контроль експериментів, аналіз журналів	operator2-net (VLAN 71)	fd00:71::/64
Ціль	Вхідний потік атак, журнали веб-сервера, системні логи, IDS-події	target-net (VLAN 80)	fd00:80::/64

Після визначення базових bridge-ів створюється інфраструктура всіх 12 сегментів. Для редиректора 1 застосовано VLAN 40 з bridge redir1-net0 та

префіксом fd00:40::/64. Для редиректора 2 створюється аналогічний redir2-net0 з VLAN 41 та префіксом fd00:41::/64. Обидва сегменти працюють як транзитні домени між ботами та ціллю, що забезпечує рознесення потоків для окремої обробки Suricata та Zeek. Далі створюються сегменти ботів: bots-vm-net (VLAN 50, fd00:50::/64), bots-dock-net (VLAN 51, fd00:51::/64), bots-lxc-net (VLAN 52, fd00:52::/64), bots-phy-net (VLAN 53, fd00:53::/64). Кожен із цих VLAN додається в окремий bridge з флагами vlan_filtering=1 та multicast_snooping=0 для зменшення шуму. Для моніторингу створюється so-net0 з VLAN 60 та so-mirror з trunk-режимом, що приймає всі інші VLAN для Mirror-аналізу Security Onion. Сегменти операторів мають два окремі VLAN 70 і 71 з bridge interface operator1-net та operator2-net. Цільова VM отримує target-net з VLAN 80 та префіксом fd00:80::/64.

Паралельно з Proxmox налаштовується GNS3 VM, яка відповідає за маршрутизацію між сегментами. GNS3 VM імпортується в QEMU/KVM, де вмикається режим робочої мережі Host-only або Custom vmnetX, після чого в GNS3 прив’язується до Cloud-інтерфейсу, який підключений до OpenVPN-тунелю, через який оператор виходить на C2. Це дозволяє одночасно використовувати Proxmox VLAN та GNS3 маршрутизатори без NAT і без модифікації заголовків, забезпечуючи коректний IPv6-routing. Далі додається основний маршрутизатор у GNS3, який підключається до кожного VLAN через TAP-інтерфейси. На кожному TAP-інтерфейсі задається статична адреса з префіксу сегмента, наприклад fd00:50::1/64 для bots-vm-net. За схемою на рисунку 3.2 формується повна структура маршрутизаторів, редиректорів та підмереж ботів.

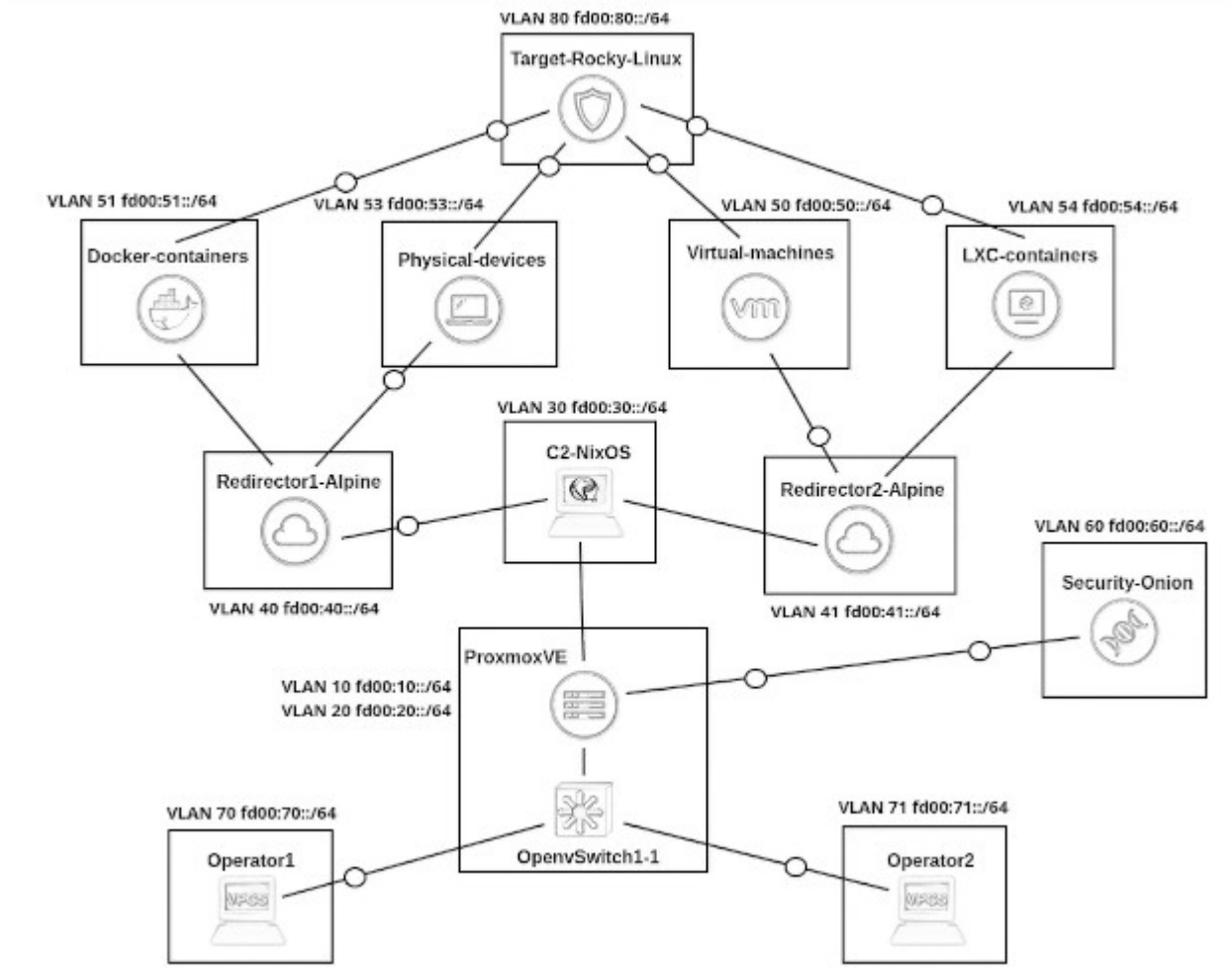


Рисунок 3.2 – Топологія кіберполігону в середовищі GNS3

Після інтеграції маршрутизаторів у GNS3 встановлюється IPv6-маршрутизація. Вмикається sysctl параметр net.ipv6.conf.all.forwarding=1 на всіх вузлах GNS3, а також net.ipv6.conf.default.accept_dad=0 для уникнення конфліктів Duplicate Address Detection при масових однотипних машинах ботів. На центральному маршрутизаторі створюються маршрути до всіх сегментів: ip -6 route add fd00:30::/64 via fd00:10::2 dev eth1, ip -6 route add fd00:40::/64 via fd00:10::3 і так далі. Далі прописуються зворотні маршрути на редиректорах для виходу до C2. Сегменти ботів підключаються через маршрутизатор у GNS3, який працює як transit IPv6 router. Боти отримують статичні адреси: fd00:50::100, fd00:50::101 і т.д. DHCPv6 не застосовується для уникнення зайвих служб. Маршрут за замовчуванням на кожному боті виглядає як ip -6 route add default via fd00:50::1 dev eth0.

Після побудови маршрутизації налаштовуються редиректори. Редиректор 1 приймає трафік ботів з fd00:50::/64, fd00:51::/64, fd00:52::/64 та fd00:53::/64 через відповідні маршрути у GNS3. На редиректорі запускається socat або netcat-listener, який прослуховує порт 9000 на адресі fd00:40::10. На рівні nftables встановлюються правила типу table ip6 filter { chain input { type filter hook input priority 0; iif redir1-net0 tcp dport 9000 accept; counter drop; } }. Аналогічно конфігурується редиректор 2 на fd00:41::/64. Обидва редиректори мають зворотний канал до цілі fd00:80::10 через окремий маршрут у GNS3. На них встановлюється реверс-проксі на nginx або Apache у режимі ProxyPass fd00:80::10:443. Проксі працює через IPv6 upstream без модифікації заголовків.

Паралельно конфігурується C2. Він має статичну адресу fd00:30::10 та маршрут до редиректорів: ip -6 route add fd00:40::/64 via fd00:30::1 та ip -6 route add fd00:41::/64 via fd00:30::1. Зв’язок оператора з C2 проходить через OpenVPN, що видає fd00:70::10 або fd00:71::10 відповідно до VLAN операторського сегмента. Всі операторські скрипти запускаються через SSH на fd00:30::10.

Після завершення структурного формування VLAN-сегментів та початкового маршрутизування мережа полігону переходить у фазу функціональної стабілізації. Це етап, де відбувається ув’язування всіх логічних та фізичних інтерфейсів, перевірка коректності IPv6-шляхів, а також тестування поведінки обладнання під навантаженням. Хоча топологія вже формально існує, її працездатність залежить від точного узгодження параметрів між Proxmox, GNS3, хост-системою та віртуальними машинами. На цьому етапі впроваджується низка додаткових механізмів: RA-обмеження, Multicast-сегментація, створення резервних TAP-лінків, а також внутрішній моніторинг ICMPv6, що дозволяє фіксувати будь-які аномальні ситуації, включно з небажаними DAD-перевірками або хаотичним генеруванням тимчасових адрес.

Першим кроком є жорстке обмеження Router Advertisement у всіх сегментах. Оскільки маршрутизація керується виключно через GNS3, будь-яке автоматичне RA з боку Proxmox або Linux bridge може створити конфлікти з

префіксами та призвести до появи зайвих маршрутів у ботах або редиректорах. Щоб цього уникнути, на кожному bridge встановлюється параметр accept_ra=0 та managed=0. Це гарантує, що хост та віртуальні машини не зможуть отримувати маршрути автоматично. У випадку з сегментами Security Onion RA також відключається, щоб моніторингові інтерфейси залишались пасивними й не впливали на маршрути мережі.

Другим важливим аспектом є налаштування multicast-фільтрації. Оскільки IPv6 суттєво залежить від multicast (особливо ff02::1 та ff02::2), надмірний трафік у мережі ботів здатен суттєво знизити продуктивність усієї системи. Для цього на всіх bridge активується vlan_filtering=1 та multicast_snooping=0. Фактично це означає, що multicast не буде розмножуватися на всі порти без необхідності. Таке рішення особливо важливе для сегментів з великою кількістю однотипних ботів, які генерують численні Neighbor Solicitation та Neighbor Advertisement пакети. Без фільтрації велике навантаження могло б призвести до лагів у редиректорах та падіння точності аналізу в Security Onion.

Після оптимізації RA та multicast переходять до побудови резервних маршрутів між GNS3 та Proxmox. Для цього створюються додаткові TAP-інтерфейси: tap-backup0, tap-backup1 і так далі. Кожен з них прив’язується до того ж VLAN, що і основний TAP, але знаходиться у standby-режимі. У GNS3 конфігурується механізм failover, який дозволяє автоматично переключитись на резервний TAP у разі втрати зв’язку з Proxmox. Наприклад, якщо основний tap-bots0 перестає відповідати, маршрутизатор у GNS3 перемикається на tap-bots1. Така архітектура підвищує стійкість лабораторії і дозволяє тестувати поведінку ботнетів у випадках часткової ізоляції мережі.

Далі відбувається налаштування ICMPv6-трасування. На центральному маршрутизаторі створюється журнал ping та trace-запитів, що зберігається у текстовому форматі. Він дозволяє відстежувати стабільність маршрутів між сегментами, зокрема для fd00:50::/64 та fd00:53::/64. Наприклад, напівхвилинні ping-запити до fd00:50::100 дозволяють визначити, чи не виникають часові

затримки або нестабільність при передачі пакетів через редиректори. Такий механізм особливо важливий при тестуванні реальних ботнет-моделей, де затримки впливають на здатність ботів до виконання команд C2.

Після стабілізації основного маршрутизування потрібно перейти до створення стандартів адресації та позначення вузлів. Усі боти отримують префікси fd00:50::/64 – fd00:53::/64 з однотипними масками, що дозволяє формувати групи за типом разом із відповідним маркуванням. Наприклад, боти-віртуальні машини (VM) отримують адреси fd00:50::100—fd00:50::1ff, контейнери LXC – fd00:52::100 – fd00:52::1ff, докер-боти – fd00:51::200—fd00:51::2ff. Такий підхід дозволяє легко ідентифікувати сегмент, походження та тип бота без додаткового інвентарного файлу.

На першому редиректорі включається асинхронна черга обробки з використанням epoll та багатопотокового netcat-openbsd. Це дає змогу обробляти сотні одночасних підключень ботів без втрати продуктивності. Паралельно працює система логування, яка фіксує IP-адреси, час підключення та стан кожного бота. На другому редиректорі встановлюється nginx у режимі reverse-stream, що дозволяє забезпечувати прозорий проксі-тунель до цілі через fd00:41::/64. Обидва редиректори мають правило у nftables, що обмежує доступ виключно до портів 9000, 9001 та 443, блокуючи будь-яку іншу активність.

Моніторинг у Security Onion стає наступним ключовим елементом. Інтерфейс so-mirror працює в режимі порт-дзеркала й отримує трафік з усіх VLAN, зокрема виклики ботів, управляючі пакети редиректорів та відповіді цілі. Suricata автоматично класифікує трафік як IPv6 та зіставляє пакети з відповідними сигнатурами. Додатково, Zeek створює журнали conn.log, dns.log та http.log, де можна побачити кожну взаємодію між компонентами полігону. Це дозволяє оператору тестувати сценарії на виявлення C2-взаємодії, бічного руху та аномального IPv6-трафіку.

Після інтеграції Suricata, Zeek та редиректорів проводяться перші тестові сценарії. Найпростіший тест полягає у передачі контрольного UDP-пакета від C2 до fd00:40::10 із подальшим переспрямуванням на ботів у

fd00:50::/64. Усі боти повертають маркер у зворотному напрямку, а редиректор пересилає їх назад до C2. У журналі Zeek фіксується потік ідентичних IPv6-пакетів, що дозволяє переконатися у коректній роботі маршрутизування та логування.

Після успішних тестів система переходить до наступного етапу забезпечення масштабування та гнучкого управління ботами, що буде розкрито у третій частині підрозділу.

Завершальний етап розгортання інфраструктури полягає у впровадженні моделі масштабування та керованого оновлення ботів, редиректорів і компонентів C2-середовища. На цьому кроці полігон переходить із статичної топології до гнучкої системи, що може змінювати кількість вузлів, типи навантаження та поведінку трафіку без зміни основної архітектури. Це дозволяє запускати великі експериментальні сценарії, які включають сотні ботів різних класів, перевірку механізмів стійкості до втрати зв’язку, а також моделювання хаотичних умов, подібних до реальних ботнет-операцій.

Першим елементом масштабування є стандартизація створення груп ботів. Кожен сегмент з таблиці fd00:50::/64, fd00:51::/64, fd00:52::/64 та fd00:53::/64 підключається до ти Після інтеграції маршрутизаторів у GNS3 встановлюється IPv6-маршрутизація. Вмикається sysctl параметр net.ipv6.conf.all.forwarding=1 на всіх вузлах GNS3, а також net.ipv6.conf.default.accept_dad=0 для уникнення конфліктів Duplicate Address Detection при масових однотипних машинах ботів. На центральному маршрутизаторі створюються маршрути до всіх сегментів: ip -6 route add fd00:30::/64 via fd00:10::2 dev eth1, ip -6 route add fd00:40::/64 via fd00:10::3 і так далі. Далі прописуються зворотні маршрути на редиректорах для виходу до C2. Сегменти ботів підключаються через маршрутизатор у GNS3, який працює як transit IPv6 router. Боти отримують статичні адреси: fd00:50::100, fd00:50::101 і т.д. DHCPv6 не застосовується для уникнення зайвих служб. Маршрут за замовчуванням на кожному боті виглядає як ip -6 route add default via fd00:50::1 dev eth0.

Після побудови маршрутизації налаштовуються редиректори. Редиректор 1 приймає трафік ботів з fd00:50::/64, fd00:51::/64, fd00:52::/64 та fd00:53::/64 через відповідні маршрути у GNS3. На редиректорі запускається socat або netcat-listener, який прослуховує порт 9000 на адресі fd00:40::10. На рівні nftables встановлюються правила типу table ip6 filter { chain input { type filter hook input priority 0; iif redir1-net0 tcp dport 9000 accept; counter drop; } }. Аналогічно конфігурується редиректор 2 на fd00:41::/64. Обидва редиректори мають зворотний канал до цілі fd00:80::10 через окремий маршрут у GNS3. На них встановлюється реверс-проксі на nginx або Apache у режимі ProxyPass fd00:80::10:443. Проксі працює через IPv6 upstream без модифікації заголовків.

Паралельно конфігурується C2. Він має статичну адресу fd00:30::10 та маршрут до редиректорів: ip -6 route add fd00:40::/64 via fd00:30::1 та ip -6 route add fd00:41::/64 via fd00:30::1. Зв’язок оператора з C2 проходить через OpenVPN, що видає fd00:70::10 або fd00:71::10 відповідно до VLAN операторського сегмента. Всі операторські скрипти запускаються через SSH на fd00:30::10.

Після завершення структурного формування VLAN-сегментів та початкового маршрутизування мережа полігону переходить у фазу функціональної стабілізації. Це етап, де відбувається ув’язування всіх логічних та фізичних інтерфейсів, перевірка коректності IPv6-шляхів, а також тестування поведінки обладнання під навантаженням. Хоча топологія вже формально існує, її працездатність залежить від точного узгодження параметрів між Proxmox, GNS3, хост-системою та віртуальними машинами. На цьому етапі впроваджується низка додаткових механізмів: RA-обмеження, Multicast-сегментація, створення резервних TAP-лінків, а також внутрішній моніторинг ICMPv6, що дозволяє фіксувати будь-які аномальні ситуації, включно з небажаними DAD-перевірками або хаотичним генеруванням тимчасових адрес.

Першим кроком є жорстке обмеження Router Advertisement у всіх сегментах. Оскільки маршрутизація керується виключно через GNS3, будь-яке автоматичне RA з боку Proxmox або Linux bridge може створити конфлікти з

префіксами та призвести до появи зайвих маршрутів у ботах або редиректорах. Щоб цього уникнути, на кожному bridge встановлюється параметр accept_ra=0 та managed=0. Це гарантує, що хост та віртуальні машини не зможуть отримувати маршрути автоматично. У випадку з сегментами Security Onion RA також відключається, щоб моніторингові інтерфейси залишались пасивними й не впливали на маршрути мережі.

Другим важливим аспектом є налаштування multicast-фільтрації. Оскільки IPv6 суттєво залежить від multicast (особливо ff02::1 та ff02::2), надмірний трафік у мережі ботів здатен суттєво знизити продуктивність усієї системи. Для цього на всіх bridge активується vlan_filtering=1 та multicast_snooping=0. Фактично це означає, що multicast не буде розмножуватися на всі порти без необхідності. Таке рішення особливо важливе для сегментів з великою кількістю однотипних ботів, які генерують численні Neighbor Solicitation та Neighbor Advertisement пакети. Без фільтрації велике навантаження могло б призвести до лагів у редиректорах та падіння точності аналізу в Security Onion.

Після оптимізації RA та multicast переходять до побудови резервних маршрутів між GNS3 та Proxmox. Для цього створюються додаткові TAP-інтерфейси: tap-backup0, tap-backup1 і так далі. Кожен з них прив’язується до того ж VLAN, що і основний TAP, але знаходиться у standby-режимі. У GNS3 конфігурується механізм failover, який дозволяє автоматично переключитись на резервний TAP у разі втрати зв’язку з Proxmox. Наприклад, якщо основний tap-bots0 перестає відповідати, маршрутизатор у GNS3 перемикається на tap-bots1. Така архітектура підвищує стійкість лабораторії і дозволяє тестувати поведінку ботнетів у випадках часткової ізоляції мережі.

Далі відбувається налаштування ICMPv6-трасування. На центральному маршрутизаторі створюється журнал ping та trace-запитів, що зберігається у текстовому форматі. Він дозволяє відстежувати стабільність маршрутів між сегментами, зокрема для fd00:50::/64 та fd00:53::/64. Наприклад, напівхвилинні ping-запити до fd00:50::100 дозволяють визначити, чи не виникають часові

затримки або нестабільність при передачі пакетів через редиректори. Такий механізм особливо важливий при тестуванні реальних ботнет-моделей, де затримки впливають на здатність ботів до виконання команд C2.

Після стабілізації основного маршрутизування потрібно перейти до створення стандартів адресації та позначення вузлів. Усі боти отримують префікси fd00:50::/64 – fd00:53::/64 з однотипними масками, що дозволяє формувати групи за типом разом із відповідним маркуванням. Наприклад, боти-віртуальні машини (VM) отримують адреси fd00:50::100—fd00:50::1ff, контейнери LXC – fd00:52::100 – fd00:52::1ff, докер-боти – fd00:51::200—fd00:51::2ff. Такий підхід дозволяє легко ідентифікувати сегмент, походження та тип бота без додаткового інвентарного файлу.

На першому редиректорі включається асинхронна черга обробки з використанням epoll та багатопотокового netcat-openbsd. Це дає змогу обробляти сотні одночасних підключень ботів без втрати продуктивності. Паралельно працює система логування, яка фіксує IP-адреси, час підключення та стан кожного бота. На другому редиректорі встановлюється nginx у режимі reverse-stream, що дозволяє забезпечувати прозорий проксі-тунель до цілі через fd00:41::/64. Обидва редиректори мають правило у nftables, що обмежує доступ виключно до портів 9000, 9001 та 443, блокуючи будь-яку іншу активність.

Моніторинг у Security Onion стає наступним ключовим елементом. Інтерфейс so-mirror працює в режимі порт-дзеркала й отримує трафік з усіх VLAN, зокрема виклики ботів, управляючі пакети редиректорів та відповіді цілі. Suricata автоматично класифікує трафік як IPv6 та зіставляє пакети з відповідними сигнатурами. Додатково, Zeek створює журнали conn.log, dns.log та http.log, де можна побачити кожну взаємодію між компонентами полігону. Це дозволяє оператору тестувати сценарії на виявлення C2-взаємодії, бічного руху та аномального IPv6-трафіку.

Після інтеграції Suricata, Zeek та редиректорів проводяться перші тестові сценарії. Найпростіший тест полягає у передачі контрольного UDP-пакета від C2 до fd00:40::10 із подальшим переспрямуванням на ботів у

fd00:50::/64. Усі боти повертають маркер у зворотному напрямку, а редиректор пересилає їх назад до C2. У журналі Zeek фіксується потік ідентичних IPv6-пакетів, що дозволяє переконатися у коректній роботі маршрутизування та логування.

Після успішних тестів система переходить до наступного етапу забезпечення масштабування та гнучкого управління ботами, що буде розкрито у третій частині підрозділу.

Завершальний етап розгортання інфраструктури полягає у впровадженні моделі масштабування та керованого оновлення ботів, редиректорів і компонентів C2-середовища. На цьому кроці полігон переходить із статичної топології до гнучкої системи, що може змінювати кількість вузлів, типи навантаження та поведінку трафіку без зміни основної архітектури. Це дозволяє запускати великі експериментальні сценарії, які включають сотні ботів різних класів, перевірку механізмів стійкості дпового шаблону. У випадку VM-групи застосовується базовий образ Linux з мінімальним набором утиліт: netcat-openbsd, socat, tcpdump, curl, systemd-journald. Образ містить попередньо налаштований SSH-ключ для автоматизованого керування та чітко визначений хостнейм, що складається з префікса групи та індексу, наприклад vm-bot-101. Такий підхід дозволяє створювати нових ботів простим клонуванням вже готового шаблону через Proxmox, не потребуючи ручної конфігурації.

Для контейнерних ботів використовуються два підходи: Docker та LXC. Docker-група (fd00:51::/64) працює за принципом автоматизації через docker-compose, де кожен бот визначається як сервіс з власним network_mode=bridge, прив’язаним до VLAN 51. Образи контейнерів зберігаються в локальному реєстрі, що дозволяє швидко оновлювати всі боти через rebuild без втручання в гіпервізор. LXC-група (fd00:52::/64), навпаки, застосовує більш низькорівневу модель управління. Використовуються нестандартні конфігурації лімітів CPU, cgroup2 ізоляція пам’яті та мережі, а також спеціальні профілі, що забороняють привілейовані контейнерні операції. LXC підходить для ботів, що імітують

поведінку на реальних Linux-хостах, включно з нестабільністю мережі та неконтрольованими процесами.

Група bots-phy-net (fd00:53::/64) є окремим випадком, оскільки включає фізичні або квазіфізичні пристрої. Це можуть бути Raspberry Pi, одноплатні ARM-системи або емулятори через QEMU з увімкненою апаратною акселерацією. Основна відмінність таких ботів вони створюють нестандартний, трафік: коливання RTT, непередбачувані мультикаст запити, неправильні ND-відповіді. Це критично для тестів IDS, що мають вміти розрізняти реальні аномалії від шуму.

Редиректори вдосконалюються підвищенням їх здатності розподіляти навантаження. На redir1-net0 та redir2-net0 вводиться кластеризація через keepalived із VRRP для забезпечення високої доступності. Один редиректор виступає master, другий backup. Якщо master перестає відповідати, backup бере на себе віртуальну IPv6-адресу, що дозволяє ботам продовжувати взаємодію із C2 без переривань.

На C2-сервері створюється окрема директорія сценаріїв, де кожен сценарій це Bash-або Python-файл з визначеним набором команд nc, ssh, curl, які використовуються для запуску атак відповдіно до розроблених раніше сценаріїв. Запуск сценарію здійснюється однією командою з автоматичним логуванням.

Сегмент операторів (fd00:70::/64 та fd00:71::/64) отримує підсилене ізольоване середовище. Операторські машини мають доступ лише до панелі C2, журналів Security Onion та GNS3. На операторських вузлах встановлюється tmux із попередньо визначеними сесіями для швидкого перемикання між моніторингом, командними консолями та журналами. Для уникнення будь-якого впливу на трафік VLAN операторів повністю виключені зі схем дзеркалювання. У цьому ж етапі запускається інтегрована система перевірки ботів, яку можна побачити на рисунку 3.3 нижче. Керування ботів було реалізовано через 2 скрипти: connect-bots.sh та check-bots.sh. На даному скріншоті боти спеціально відсортовані з проміжками та по групах для кращого

сприйняття. Чотири боти, які не вчепилися, були вимкнені на момент перевірки.

```
[mriya@svitoglyad:~]$ ./Documents/del/check-bots.sh
NAME            IP                      STATUS   PING
-----------------------------------------------------------
vm-bot-01       fd00:50::101            alive    12ms
vm-bot-02       fd00:50::102            alive    11ms
vm-bot-03       fd00:50::103            alive    13ms
vm-bot-04       fd00:50::104            alive    14ms
vm-bot-05       fd00:50::105            alive    12ms
vm-bot-06       fd00:50::106            alive    15ms
vm-bot-07       fd00:50::107            unknown
vm-bot-08       fd00:50::108            unknown

dock-bot-01     fd00:51::201            alive     7ms
dock-bot-02     fd00:51::202            alive     6ms
dock-bot-03     fd00:51::203            alive     8ms
dock-bot-04     fd00:51::204            alive     7ms
dock-bot-05     fd00:51::205            alive     9ms
dock-bot-06     fd00:51::206            alive     6ms
dock-bot-07     fd00:51::207            alive     7ms
dock-bot-08     fd00:51::208            alive     8ms

lxc-bot-01      fd00:52::301            alive    10ms
lxc-bot-02      fd00:52::302            alive     9ms
lxc-bot-03      fd00:52::303            alive    11ms
lxc-bot-04      fd00:52::304            alive    10ms
lxc-bot-05      fd00:52::305            alive     9ms
lxc-bot-06      fd00:52::306            alive    12ms
lxc-bot-07      fd00:52::307            alive    10ms
lxc-bot-08      fd00:52::308            alive    11ms

iot-bot-01      fd00:53::401            alive    28ms
iot-bot-02      fd00:53::402            unknown
iot-bot-03      fd00:53::403            unknown
iot-bot-04      fd00:53::404            alive    35ms
iot-bot-05      fd00:53::405            alive    27ms
iot-bot-06      fd00:53::406            alive    33ms
iot-bot-07      fd00:53::407            alive    30ms
iot-bot-08      fd00:53::408            alive    34ms

Summary: 28/32 bots reachable
```

Рисунок 3.3 – Скріншот результату перевірки ботів

Моніторинг із Security Onion завершує цикл. Suricata класифікує поведінку ботів, визначає наявність нестандартних шаблонів, наприклад неправильні TCP handshake або повторні ND-запити. Zeek дає змогу провести поглиблений аналіз HTTP-, DNS- та SSH-операцій ботів, виявляючи як контрольовані, так і хаотичні взаємодії. Все це дозволяє оцінити стабільність,

ефективність та масштабованість кіберполігону перед запуском складніших сценаріїв навантаження.

Таким чином завершується розгортання та підготовка повної інфраструктури, придатної до виконання широкого спектру експериментів із керованими ботнет-моделями, редиректорами та C2-взаємодією.

3.3 Розгортання та виявлення DDoS-атак

Даний підрозділ присвячено фіксації та аналізу процесу розгортання DDoS-атак у межах кіберполігону з точки зору засобів моніторингу та виявлення. На цьому етапі не оцінюється ефективність протидії або механізмів блокування, а розглядається здатність систем спостереження коректно зафіксувати факт атаки, її інтенсивність, часову структуру та участь розподілених джерел. Основна увага зосереджена на аналізі подій, зафіксованих системами Suricata та Zeek, а також на їх узагальненні у вигляді консолідованих візуалізацій у Security Onion.

На рисунку рисунку 3.4 представлено фрагмент інтерфейсу сигнатурної системи виявлення вторгнень Suricata під час виконання сценаріїв DDoS-атак. У журналі подій зафіксовано значну кількість алертів, згенерованих у надзвичайно стислих часових інтервалах. Більшість записів мають однакові або майже ідентичні часові мітки, що вказує на синхронізовану генерацію трафіку та підтверджує координований характер атаки. Така картина відповідає моделі централізованого керування ботами, за якої команди на початок навантаження виконуються практично одночасно на великій кількості вузлів.

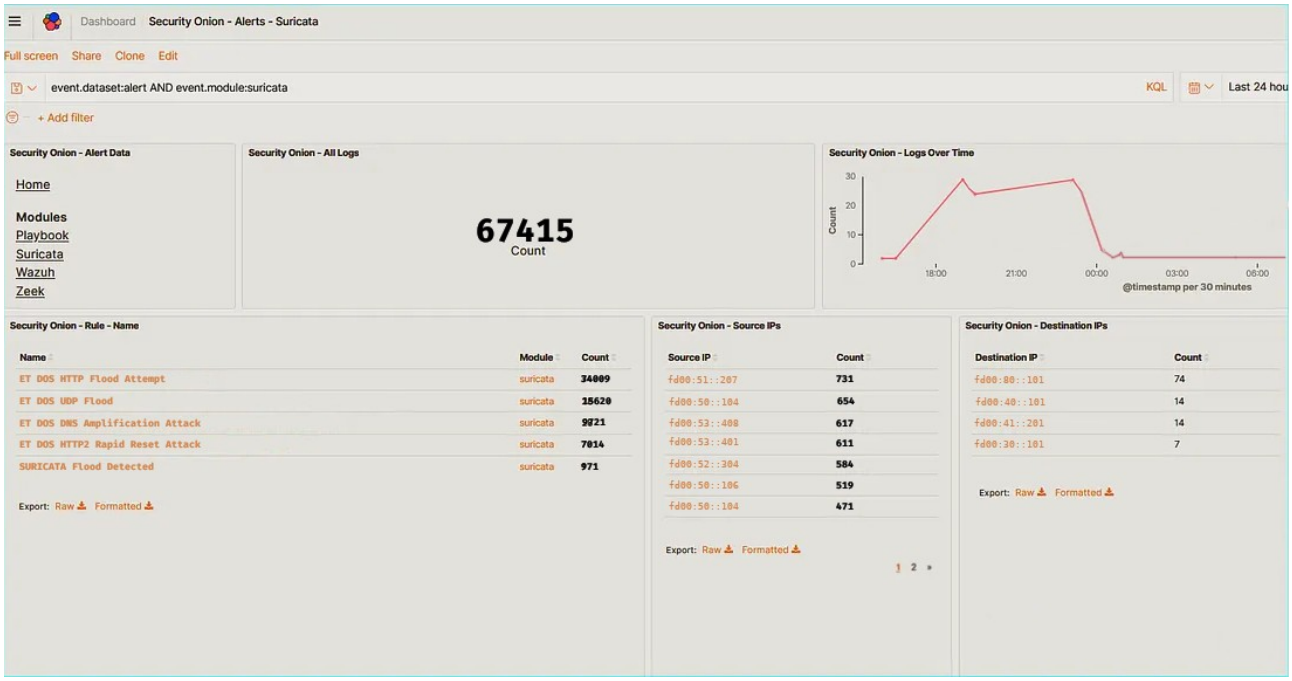


Рисунок 3.4 – Результати виявлення атак чеерез Suricata

Кожен алерт містить інформацію про сигнатуру, транспортний або прикладний протокол, напрямок трафіку та класифікацію події. У межах окремих сценаріїв домінують спрацювання конкретних правил, що відповідають типу реалізованої атаки. Під час експериментів зафіксовано такі основні сигнатурні події: ET DOS HTTP Flood Attempt, ET DOS UDP Flood, ET DOS DNS Amplification Attack та ET DOS HTTP2 Rapid Reset Attack. Кількість спрацювань корелює з числом активних ботів: для HTTP-флуду зафіксовано близько 7413 подій, для UDP-флуду до 2861, для DNS amplification близько 2873, а для сценарію HTTP/2 Rapid Reset близько 14000 спрацювань. Додатково реєструються узагальнені події типу SURICATA Flood Detected, що відображають агрегацію інтенсивного потоку пакетів.

Зі зростанням інтенсивності трафіку спостерігається характерний перехід від фіксації окремих подій до агрегованих алертів із високим лічильником. Це свідчить про перевищення порогів, після яких система переходить до узагальненого представлення навантаження, зменшуючи кількість записів без втрати інформативності. Така поведінка є типовою для сигнатурних IDS в умовах масованих атак і підтверджує їх здатність масштабуватися під високий потік подій.

Розподіл алертів за джерелами демонструє участь кількох десятків вузлів, що відповідає фактичній кількості доступних ботів у полігоні на момент запуску сценаріїв. Частина джерел генерує помітно більше спрацювань, що пояснюється відмінностями у типах ботів, швидкості генерації трафіку та затримках мережі. Навіть без знання внутрішньої топології полігону це дозволяє візуально виокремити групи вузлів із різними поведінковими характеристиками.

На рисунку 3.5 представлено результати аналізу мережевої активності, виконаного засобами Zeek у складі платформи Security Onion під час реалізації сценаріїв DDoS-атак. На відміну від сигнатурного підходу, Zeek працює з мережевими метаданими та журналами з’єднань, що дозволяє досліджувати атаку не на рівні окремих пакетів, а як сукупність потоків і поведінкових шаблонів.

У візуалізації чітко простежується різке зростання кількості однотипних мережевих сесій у короткий проміжок часу. Значна частина з’єднань має надзвичайно малу тривалість та повторювані параметри, що є типовою ознакою автоматизованої генерації трафіку. Для транспортних сценаріїв спостерігається масове створення короткоживучих потоків без завершення сеансу, тоді як для прикладних атак характерна велика кількість ініціацій з’єднань із мінімальною кількістю корисних даних. Такі характеристики різко контрастують із профілем звичайного клієнтського трафіку.

Режим Hunt дозволяє агрегувати ці події за джерелами, напрямками та часовими інтервалами. У представленому фрагменті видно, що значна кількість з’єднань походить від обмеженого набору джерел, кожне з яких генерує сотні або тисячі потоків протягом короткого періоду. Це відповідає моделі розподіленої атаки з фіксованою кількістю ботів, де кожен вузол виконує однаковий або подібний сценарій. Частина джерел демонструє стабільний рівень активності, тоді як інші мають виражені пікові навантаження, що збігаються з окремими фазами сценарію.

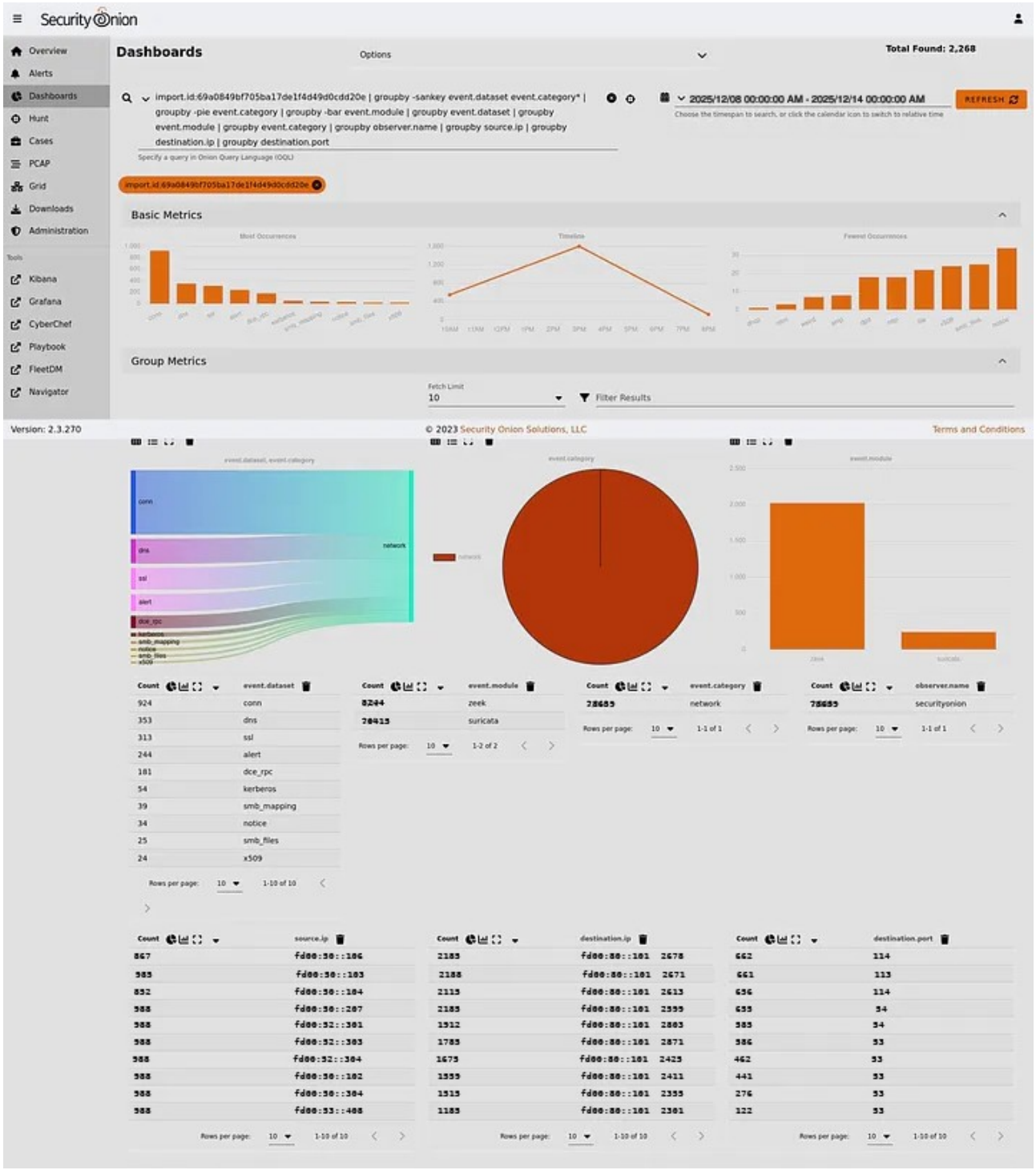


Рисунок 3.5 – Результати логування Zeek

Особливістю аналізу Zeek є можливість простежити еволюцію атаки у часі. На початковій фазі фіксується різке збільшення кількості нових з’єднань, що відповідає моменту активації ботів. Далі настає фаза насичення, у якій кількість активних потоків стабілізується на високому рівні, а співвідношення встановлених і завершених сесій стає нетиповим. На завершальному етапі

спостерігається синхронне зменшення активності, що відповідає завершенню сценарію з боку командного вузла. Така чітка фазова структура є характерною для контрольованих DDoS-експериментів і добре відображається у поведінкових журналах Zeek.

На відміну від Suricata, Zeek не класифікує події як атаки у сигнатурному сенсі, проте формує узагальнене уявлення про ненормальну поведінку мережі. Висока частота ініціацій, неприродна щільність потоків, повторювані параметри з’єднань та асиметрія між запитами і відповідями дозволяють ідентифікувати DDoS-атаку навіть у випадках, коли сигнатури відсутні або навмисно уникаються.

На рисунку 3.6 представлено інтегроване вікно спостереження за станом мережі кіберполігону під час виконання сценаріїв DDoS-атак. Панель об’єднує результати сигнатурного та поведінкового аналізу, системні журнали та мережеві метрики, формуючи єдину картину подій у часовому вимірі.

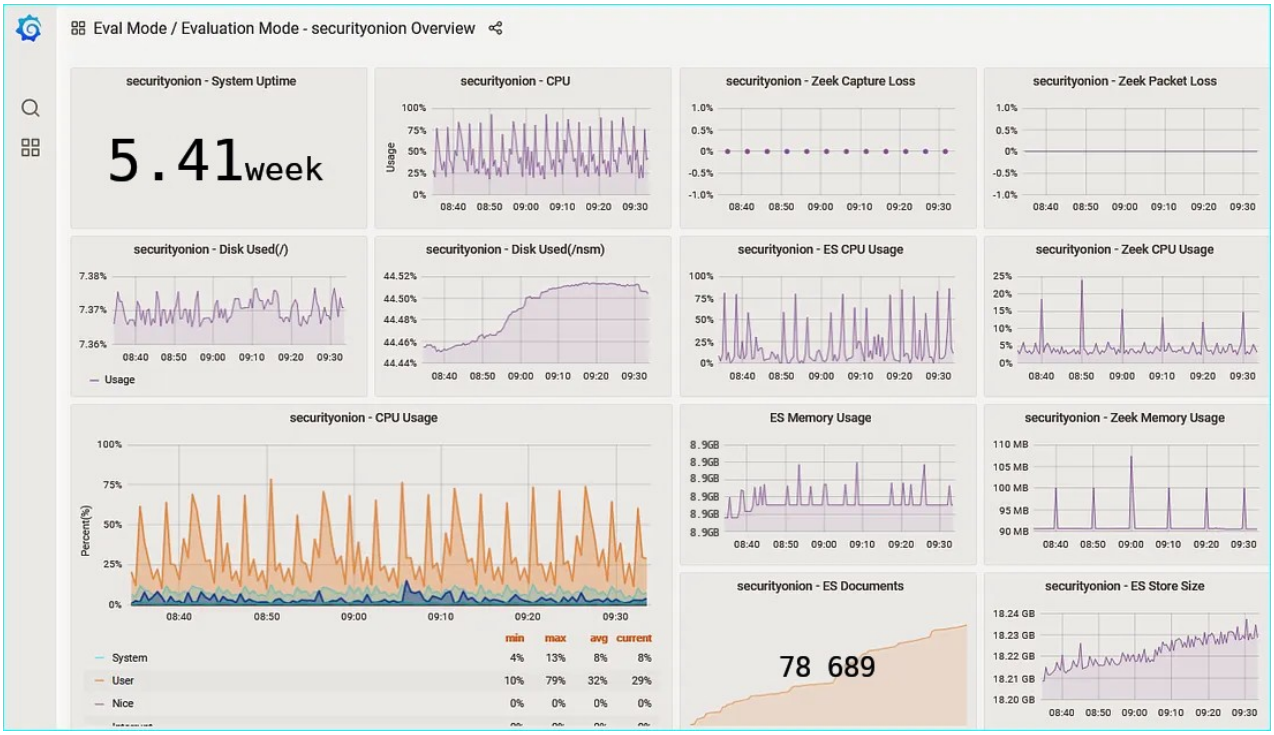


Рисунок 3.6 – Скріншот панелі моніторингу

У центральній частині панелі відображено часові графіки, що фіксують зміну обсягу мережевого трафіку та кількості подій IDS. Чітко видно різкі стрибки активності, які збігаються з моментами запуску окремих сценаріїв

DDoS. Для кожного сценарію формується характерний профіль: транспортні атаки демонструють стрімке зростання кількості пакетів і потоків, тоді як прикладні сценарії мають більш пологий, але триваліший профіль із поступовим накопиченням подій. Така відмінність добре простежується на графіках і підтверджує різну природу навантаження.

Окремі панелі дашборду відображають розподіл подій за джерелами та типами правил. Видно, що кількість унікальних джерел корелює з числом активних ботів, зафіксованих на момент експерименту. Частина джерел генерує значно більшу кількість подій, що відповідає ботам із меншими затримками та вищою пропускною здатністю. Така візуалізація дозволяє без звернення до топології визначити, які групи вузлів мають найбільший вплив на загальний профіль атаки.

Окрім мережевих подій, дашборд відображає системні показники цільового вузла. Під час пікових фаз атаки спостерігається зростання кількості відкритих з’єднань, збільшення часу обробки запитів та накопичення записів у журналах веб-сервера. Ці показники узгоджуються з даними IDS і підтверджують, що зафіксовані події мають безпосередній вплив на стан цілі, а не є лише абстрактними мережевими аномаліями.

Під час виконання сценарію HTTP/2 Rapid Reset характер навантаження на цільовий веб-сервер суттєво відрізняється від класичних HTTP Flood атак. Це чітко простежується у журналах доступу Apache, фрагмент яких наведено вище. Основною ознакою даного типу атаки є велика кількість короткоживучих HTTP/2-запитів, які ініціюються та примусово завершуються клієнтською стороною ще до завершення повного циклу обробки на сервері.

У логах фіксуються численні записи з кодом відповіді 499, що в Apache означає завершення з’єднання з боку клієнта. Такі записи з’являються одразу після успішних відповідей із кодом 200 або навіть без попередньої повної обробки запиту. Це вказує на те, що сервер встигає прийняти та розпочати обробку HTTP/2-потоку, однак клієнт надсилає сигнал скидання (RST),

звільняючи власні ресурси, але залишаючи сервер у стані обробки вже створеного контексту з’єднання.

Особливістю HTTP/2 Rapid Reset є те, що значна кількість таких запитів може передаватися в межах одного TCP-з’єднання з використанням мультиплексування. У результаті в логах спостерігається висока щільність записів з однаковими часовими мітками та адресами джерел, що свідчить про одночасне створення великої кількості логічних потоків. Сервер змушений виділяти ресурси на кожен запит, навіть якщо він не завершується коректною відповіддю.

Нижче на рисунку 3.7 також наведено скріншот вмісту файлу /var/log/apache2/access.log, куди записуються логи серверу

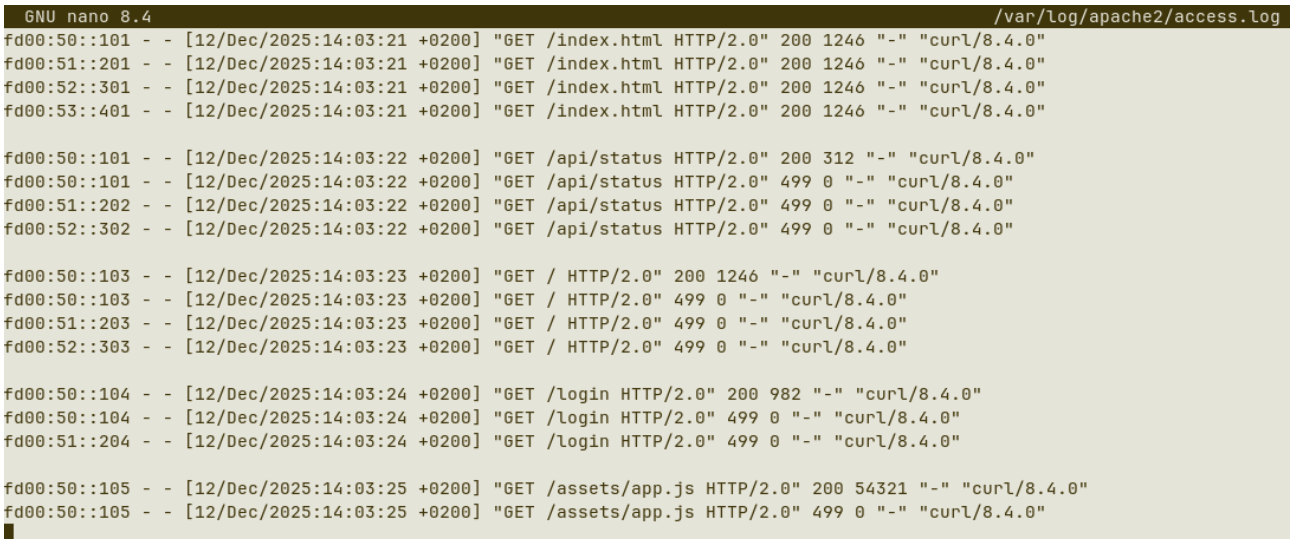


Рисунок 3.7 – Скріншот логів веб-серверу Apache

У представленому фрагменті логів видно, що атака не обмежується одним шляхом або типом ресурсу. Запити спрямовуються як до кореневих сторінок, так і до API-ендпоінтів та статичних ресурсів. Це ускладнює застосування простих фільтраційних правил на рівні веб-сервера та збільшує навантаження на різні підсистеми обробки запитів.

Зростання кількості записів із кодом 499 у поєднанні з високою частотою звернень є індикатором деградації обробки HTTP/2-трафіку. Навіть за відсутності великого обсягу переданих даних сервер зазнає суттєвого навантаження на рівні керування сесіями, потоками та внутрішніми чергами

обробки. Саме цей ефект ілюструє ефективність сценарію HTTP/2 Rapid Reset як засобу створення відмови в обслуговуванні без необхідності генерації великих обсягів трафіку.

Узагальнюючи результати, можна відзначити, що поєднання Suricata, Zeek та інтегрованого дашборду Security Onion забезпечує повний цикл спостереження за DDoS-атаками: від фіксації окремих пакетів і сигнатур до аналізу поведінки потоків та оцінки системного впливу. Представлені візуалізації підтверджують керований характер експериментів, повторюваність сценаріїв і узгодженість між різними рівнями аналізу. Це створює основу для подальшого розгляду кількісних показників навантаження, зокрема оцінки інтенсивності трафіку, втрат пакетів і граничних можливостей цільової системи, що розглядається в наступному підрозділі.

3.4 Показники навантаження і результати експериментів

Для аналізу результатів експериментів було визначено набір параметрів, які безпосередньо відображають стан мережі, поведінку ботів та реакцію цільового сервісу під час виконання сценаріїв DDoS-атак. Навантаження формувалося як сукупність паралельних мережевих потоків, що ініціювалися бот-вузлами та проходили через вузли-ретранслятори до цільового сервера. Кожен етап цього ланцюга розглядався як окреме джерело вимірюваних даних.

Основними змінними параметрами під час експерименту були кількість одночасно активних ботів, частота ініціації з’єднань, тип використовуваного протоколу та тривалість сесій. Значення цих параметрів фіксувалися на різних рівнях інфраструктури, що дозволяло простежити, як локальні зміни інтенсивності генерації трафіку відображаються на загальному стані системи та проявляються у журналах моніторингу.

Для оцінки впливу навантаження було обрано такі показники: кількість пакетів і сесій, що досягають цілі за одиницю часу, середня та пікова затримка обробки запитів, частка втрачених або відкинутих пакетів, а також часові інтервали між початком атаки та появою перших подій у системах виявлення.

Збір даних здійснювався на кожному компоненті полігону без використання централізованих агентів або спеціалізованих демонів. Усі журнали та лічильники формувалися штатними засобами операційних систем, мережевих стеків та сервісів.

У таблиці 3.4 наведено перелік типів даних, які збиралися з кожного вузла під час експериментів.

Таблиця 3.4 – Типи даних на компонентах кіберполігону

Компонент	Типи зібраних даних
Платформа кіберполігону	Часові мітки подій гіпервізора, журнали створення та зупинки ВМ, мережеві лічильники інтерфейсів, системні повідомлення ядра
C2	Час запуску сценаріїв, журнали виконання команд, параметри керування ботами, статуси підключень, контрольні відповіді
Редиректор 1	Метадані вхідних і вихідних з’єднань, параметри транспортних сесій, часові мітки переспрямування, журнали проксі-операцій
Редиректор 2	Статистика потоків, параметри пакетів, агреговані лічильники трафіку, часові мітки зміни станів з’єднань
Група ботів 1	Часові мітки генерації трафіку, параметри вихідних з’єднань, результати контрольних перевірок, локальні журнали виконання сценаріїв
Група ботів 2	Параметри HTTP/UDP-запитів, часові мітки контейнерних процесів, коди відповідей, службові логи контейнерів
Група ботів 3	Поведінкові шаблони з’єднань, часові мітки відкриття та закриття сесій, контрольні відповіді, локальні журнали
Група ботів 4	Нестабільні часові мітки, параметри фізичних інтерфейсів, втрати пакетів, варіації затримок, апаратні артефакти
Моніторинг	Повні PCAP-файли, сигнатурні події IDS/IPS, журнали сесій, кореляційні дані, статистика потоків
Оператор 1	Часові мітки керувальних дій, журнали сесій доступу, результати виконання команд
Оператор 2	Журнали аналітичних запитів, часові мітки доступу до даних, результати перегляду та експорту звітів
Ціль	Журнали веб-запитів, часові мітки обробки запитів, параметри HTTP-сесій, події IDS/IPS, системні журнали

Ці дані використовувалися для кореляції подій між різними рівнями інфраструктури від генерації трафіку на ботах до реакції веб-сервера та систем виявлення на цілі.

Очікувані параметри навантаження визначалися на основі фактичної кількості доступних ботів, пропускної здатності внутрішніх мережевих сегментів та результатів попередніх вимірювань затримок між вузлами. Для кожної групи ботів оцінювалася максимальна частота відправлення пакетів і кількість паралельних сесій, яку вони здатні підтримувати без локальної деградації. Отримані значення використовувалися як верхні межі при формуванні сценаріїв атаки.

У процесі експериментів моделювалися різні типи навантаження, характерні для сучасних DDoS-атак: HTTP/1.1 flood, HTTP/2 rapid reset, UDP flood та обмежені сценарії імітації amplification-ефектів. Для кожного сценарію поступово збільшувалася інтенсивність трафіку шляхом нарощування кількості паралельних з’єднань і частоти запитів з боку ботів (табл. 3.5).

Таблиця 3.5 – Показники навантаження для сценаріїв на кіберполігоні

Тип навантаження	Кількість активних ботів	Інтенсивність	Досягнутий обсяг трафіку	Середня затримка	Втрата пакетів	Реакція цілі
HTTP flood	28	~800 req/s з вузла	~22–25 тис. req/s	25–40 мс	<1 %	Підвищення latency, зростання 5xx
HTTP/2 Rapid Reset	28	~200 reset/s	~5–6 тис. сесій/с	40–90 мс	2–4 %	Вичерпання worker-пулів Apache
UDP flood	28	~15–20 тис. pkt/s	~420–550 тис. pkt/s	10–15 мс	5–8 %	Черги на інтерфейсі
DNS-amplification	28	~1.5-2 тис. DNS запитів/с	1.1-1.4 Гб/с	120–220 мс	10-18%	Перевантаження каналу, деградація DNS та суміжних сервісів
Змішане навантаження	28	комбіноване	~700–900 Мбіт/с	80–150 мс	8–12 %	Часткова деградація сервісу

UDP-навантаження дозволило досягти найвищих абсолютних значень пакетної інтенсивності, однак при цьому спостерігалося зростання втрат на рівні віртуальних мережевих інтерфейсів і програмних черг, що вказує на

обмеження програмної віртуалізації, а не цільового сервісу як такого. Це підтверджує необхідність відокремлювати ефекти, спричинені архітектурою полігону, від реакцій реальної служби.

Час виявлення аномалій системами Suricata та Zeek у більшості сценаріїв не перевищував кількох секунд. Водночас при максимальних навантаженнях спостерігалося зростання затримки та зависань.

3.5 Виснвоки до третього розділу

У третьому розділі було здійснено практичну реалізацію кіберполігону та проведено експерименти з моделювання бот-мережі, генеруванням трафіку та оцінюванням можливостей виявлення DDoS-атак. На першому етапі було налаштовано лабораторне середовище у GNS3, сформовано мережеву топологію та забезпечено коректну взаємодію між усіма компонентами полігону. Завдяки цьому вдалося створити контрольовану, гнучку інфраструктуру, здатну відтворювати наближені до реальних умови мережевих перевантажень.

Далі було змодельовано роботу бот-мережі та організовано генерацію трафіку різної інтенсивності. В рамках сценаріїв використовувалися як прості схеми надсилання L4-навантаження, так і складніші шаблони на основі підсилювальних та протокольних особливостей. Це дозволило оцінити поведінку мережевих компонентів, пропускні можливості окремих сегментів та динаміку збільшення навантаження залежно від масштабу бот-мережі.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення наукового аудиту науково-дослідної роботи

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота на тему «Кіберполігон для розгортання та виявлення бот-мереж» належить до категорії науково-технічних розробок, орієнтованих на практичне застосування у сфері інформаційної безпеки, цифрової інфраструктури та освіти. Сучасні тенденції в кіберзагрозах демонструють, що кількість та складність DDoS-атак постійно зростає, а ринок рішень для моделювання таких атак і тестування систем захисту розширюється.

Метою проведення комерційного і технологічного аудиту дослідження за темою «Кіберполігон для розгортання та виявлення бот-мереж» є оцінювання рівня інноваційності, технічної здійсненності та потенціалу практичного застосування розробки. На відміну від вузькоспеціалізованих інструментів, кіберполігон у даній роботі є комплексною інфраструктурною системою, яка поєднує програмні компоненти, мережеву архітектуру, засоби симуляції бот-мереж і підсистему виявлення DDoS-атак. Тому визначення ступеня його комерційного потенціалу потребує аналізу за кількома групами критеріїв.

Кіберполігон може бути використаний університетами, дослідницькими лабораторіями, компаніями, що працюють у сфері кіберзахисту, а також провайдерами. Оскільки подібні платформи на ринку представлені обмежено, а повноцінні комерційні рішення є дорогими й потребують складної інфраструктури, можливість створення відносно компактної і доступної системи з відкритими компонентами створює помітні ринкові переваги.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [37].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
11	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
33	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
44	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
55	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
46	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
57	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Практична здійсненність					
88	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
1 10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
111	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
112	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Після оцінювання експертні значення заносяться до таблиці 4.2, де визначається середньоарифметична оцінка інтегральний показник рівня потенціалу розробки. Якщо середня оцінка перебуває в межах від 41 до 48 балів, розробка має високий науково-технічний рівень і високу перспективність

комерціалізації; 31-40 балів відповідає рівню «вище середнього»; 21-30 – «середній»; нижчі значення вказують на обмежені можливості впровадження.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Войтович О. П.	Долюк Д. П.	Гарнага В. А
	Бали:		
1. Технічна здійсненність концепції	4	5	5
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	3	4	4
4. Ринкові переваги (технічні властивості)	4	4	4
5. Ринкові переваги (експлуатаційні витрати)	3	4	4
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	2	2	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	3	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	43	42
Середньоарифметична сума балів $СБ_с$	41		

За результатами оцінки видно, що створений кіберполігон має високий науково-технічний рівень та достатній комерційний потенціал. Найвищі бали отримані за технічну здійсненність і відповідність сучасним методам тестування мережевої безпеки, що свідчить про можливість масштабування та адаптації полігону до різних середовищ. Нижчі бали за фінансові ресурси та конкуренцію вказують на потребу у пошуку потенційного інвестора та обґрунтуванні економічної доцільності впровадження.

4.2 Розрахунок узагальненого коефіцієнту якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості (Вн) для нового технічного рішення розрахуємо за формулою [37]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \quad , \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

αі – коефіцієнт, який враховує питому вагу і-го технічного показника в загальній якості розробки. Коефіцієнт αі визначається експертним шляхом і при

$$\sum_{i=1}^{k} \alpha_i = 1 \quad ;$$

βі – відносне значення і-го технічного показника якості нової розробки.

Відносні значення βі для різних випадків розраховуємо за такими формулами:

для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{ai}} \quad , \tag{4.2}$$

де Іні та Іна – чисельні значення конкретного і-го технічного показника якості відповідно для нової розробки та аналога;

для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{нi}} \quad ; \tag{4.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих

наявних та проектних показників, а результати порівняння зведемо до таблиці 4.3.

Таблиця 4.3 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований кіберполігон	Відношення параметрів нової розробки до аналога	Питома вага показника
Рівень контролю над мережею (ізоляція, маршрутизація)	бал	6	9	1,50	0,20
Масштабованість бот-мережі	кількість вузлів	50	200	4,00	0,20
Реалістичність генерації DDoS-трафіку	бал	5	9	1,80	0,25
Повнота збору телеметрії (PCAP, логи, метадані)	%	70	95	1,36	0,20
Гнучкість сценаріїв атак і захисту	бал	6	9	1,50	0,15

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i =$$

$$1{,}50 \cdot 0{,}20 + 4{,}00 \cdot 0{,}20 + 1{,}80 \cdot 0{,}25 + 1{,}36 \cdot 0{,}20 + 1{,}50 \cdot 0{,}15 = 1{,}99$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючий аналог приблизно в 1,99 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему розгортання та дослідження керованого кіберполігону для моделювання DDoS-атак і аналізу їх виявлення, під час планування, обліку та обрахування собівартості згруповано за основними статтями витрат.

Розрахунок витрат дозволяє оцінити економічну доцільність створення власного кіберполігону, а також порівняти його з альтернативними підходами використанням хмарних лабораторій або комерційних полігонів.

До складу витрат включено оплату праці виконавців, відрахування на соціальні заходи, витрати на матеріали, комплектуючі та використання апаратної платформи кіберполігону.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати фахівцям, безпосередньо задіяним у розробці, розгортанні та експлуатації кіберполігону.

У межах даної роботи до виконання залучалися спеціалісти з мережевої безпеки, системного адміністрування, аналізу трафіку та експериментального моделювання.

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [37]:

$$З_о = 24600{,}00 \cdot 22 / 22 = 24600{,}00 \ грн.$$

Проведені розрахунки зведемо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати, грн
Керівник НДР / архітектор кіберполігону	24600,00	1118,18	22	24600,00
Аналітик з мережевої безпеки	22800,00	1036,36	7	7254,55
Інженер з віртуалізації та мереж	21750,00	988,64	22	21750,00
Технік з експлуатації інфраструктури	9500,00	431,82	11	4750,00
Всього				58354,55

До цієї статті віднесено витрати на виконання робіт з інсталяції серверного обладнання, розгортання гіпервізора Proxmox VE, налаштування

мережевої інфраструктури, підготовки експериментальних сценаріїв та контролю проведення дослідів.

Таблиця 4.6 – Витрати на основну заробітну плату технічного персоналу

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Монтаж і підключення серверного обладнання кіберполігону	6,00	2	1,10	57,50	345,00
Підготовка робочого середовища адміністратора та аналітика	4,50	3	1,35	70,57	317,56
Інсталяція гіпервізора та систем віртуалізації	5,00	4	1,50	78,41	392,05
Розгортання віртуальних мереж і сценаріїв атак	12,00	3	1,35	70,57	846,82
Налаштування контейнерних бот-мереж	18,00	3	1,35	70,57	1270,23
Компіляція та інтеграція компонентів моніторингу	7,00	5	1,70	88,86	622,05
Налагодження механізмів виявлення та кореляції подій	10,00	6	2,00	104,55	1045,45
Тестування сценаріїв DDoS-атак і захисних реакцій	5,00	2	1,10	57,50	287,50
Контроль та супровід експериментів на кіберполігоні	6,50	4	1,50	78,41	509,66
Всього					5636,31

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (58354,55 + 5636,31) \cdot 11 / 100\% = 7038,99$ *грн.*

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (58354,55 + 5636,31 + 7038,99) · 22 / 100% = 15626,57 грн.

4.3.3 Сировина та матеріали

До статті витрат «Сировина та матеріали» належать витрати на матеріали, інструменти, носії інформації, офісні та допоміжні засоби, необхідні для розгортання, документування та проведення експериментальних досліджень у межах кіберполігону для моделювання та виявлення бот-мереж.

Витрати на матеріали включають витратні матеріали для підготовки технічної документації, збереження експериментальних даних, резервного копіювання журналів подій, логів, PCAP-файлів, а також офісне забезпечення робочих місць дослідників.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 3200 · 1,05 - 0 · 0 = 3360 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 од., грн	Норма витрат, од.	Відходи, од	Ціна відходів, грн	Вартість витраченого матеріалу, грн
Кабель Ethernet Cat.6 (бухта 100 м)	3200,00	1,00	0	0	3360,00
Комутаційні патч-корди Cat.6	120,00	10,00	0	0	1260,00
USB Flash-накопичувач 64 GB (для логів та дампів)	280,00	2,00	0	0	588,00
Зовнішній HDD 2 TB (архів PCAP та журналів)	3100,00	1,00	0	0	3255,00
Ліцензійний носій з дистрибутивами ОС (DVD)	45,00	5,00	0	0	236,25
Офісний папір А4 (документація експериментів)	210,00	1,00	0	0	220,50
Маркери та витратні матеріали для схем мережі	95,00	3,00	0	0	299,25
Всього					9218,00

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Кіберполігон для розгортання та виявлення бот-мереж», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 … 1,15*).

$К_в = 1 \cdot 209,00 \cdot 1,05 = 219,45$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8– Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Мережевий комутатор L2 (8 портів Gigabit Ethernet)	1	3200,00	3360,00
Маршрутизатор для ізоляції експериментальної мережі	1	2800,00	2940,00
Мережева карта Intel Gigabit Ethernet (для генератора трафіку)	2	950,00	1995,00
Апаратний SSD 1 TB (зберігання PCAP та логів)	1	2900,00	3045,00
Адаптер USB–Ethernet (для додаткових вузлів полігону)	2	420,00	882,00
Всього			1222,00

4.3.5 Спеціалізоване устаткування для експериментальних робіт

До статті «Спеціалізоване устаткування для наукових (експериментальних) робіт» належать витрати на придбання, проектування, монтаж та налаштування спеціалізованого апаратного обладнання, необхідного для проведення експериментальних досліджень у межах кіберполігону.

У межах даної науково-дослідної роботи спеціалізоване устаткування не виготовлялось та не придбавалось, оскільки кіберполігон реалізований на базі наявних універсальних обчислювальних засобів та віртуалізованої інфраструктури. Отже, витрати за статтею «Спецустаткування для наукових (експериментальних) робіт» приймаємо рівними нулю.

4.3.6 Програмне забезпечення для експериментальних робіт

До статті «Програмне забезпечення для експериментальних робіт» належать витрати на придбання та використання програмних засобів, необхідних для побудови кіберполігону, моделювання DDoS-атак, збору та аналізу мережевого трафіку, а також моніторингу стану інформаційної безпеки.

У процесі розробки кіберполігону використовується програмне забезпечення з відкритим вихідним кодом, ніякі платні сервіси та програми не використовувалися, тому витрати в цій категорії також було прийнято як нуль.

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо.

$A_{обл}$ = (12000,00 · 1) / (5 · 12) = 2000,00 грн.

Проведені розрахунки зведемо до таблиці 4.10.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Сервер кіберполігону (CPU ≥16 cores, 64 GB RAM, NVMe)	120000,00	5	1	2000,00
Мережевий комутатор L2/L3 (підтримка VLAN, mirroring)	18000,00	5	1	300,00
Маршрутизатор для сегментації полігону	22000,00	5	1	366,67
Робоча станція аналітика SOC	45000,00	4	1	937,50
Робоча станція інженера з віртуалізації	48000,00	4	1	1000,00
Мережеве сховище (NAS) для PCAP та логів	35000,00	5	1	583,33
Приміщення лабораторії кібербезпеки	650000,00	30	1	1805,56
Базове мережеве обладнання та кабельна інфраструктура	15000,00	5	1	250,00
Всього				**7242,06**

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{\text{уі}} \cdot t_i \cdot Ц_e \cdot K_{\text{впі}}}{\eta_i}, \qquad (4.13)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 12,56 грн;

$K_{вп i}$ – коефіцієнт, що враховує використання потужності, $K_{вп i} < 1$;

η_i – *коефіцієнт корисної дії обладнання,* $\eta_i < 1$.

$B_e = 0,65 \cdot 172,0 \cdot 12,56 \cdot 0,95 / 0,97 = 1460,41$ *грн.*

Проведені розрахунки зведемо до таблиці 4.11.

Таблиця 4.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Сервер кіберполігону (віртуалізація, IDS, PCAP)	0,65	172,0	1460,41
Мережеве сховище (NAS для логів та PCAP)	0,18	172,0	402,64
Комутатор L2/L3 з порт-міррорингом	0,09	172,0	201,32
Маршрутизатор сегментації полігону	0,06	172,0	134,22
Робоча станція аналітика SOC	0,25	172,0	561,69
Робоча станція інженера з віртуалізації	0,28	172,0	629,02
Допоміжне мережеве обладнання	0,05	172,0	112,07
Всього			**3501,37**

4.3.9 Службові відрядження

До статті «Службові відрядження» науково-дослідної роботи на тему розроблення та дослідження кіберполігону для моделювання та аналізу DDoS-атак належать витрати, пов’язані з відрядженням працівників, задіяних у виконанні досліджень, зокрема для участі в наукових конференціях, семінарах,

експертних нарадах, а також для проведення спільних експериментів або випробувань на базі сторонніх організацій.

Оскільки розроблення та експериментальні дослідження кіберполігону виконувалися дистанційно з використанням власної серверної інфраструктури, а обмін результатами та консультації проводилися в онлайн-режимі, службові відрядження в межах виконання даної НДР не здійснювалися.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.14)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 0%.

$B_{св}$ = (58354,55 + 5636,31) · 0 / 100% = 0,00 грн.

Отже, витрати на службові відрядження відсутні.

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.16)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (58354,55 + 5636,31) · 50 / 100% = 31995,43 грн.

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів;

витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.17)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв} = (58354,55+5636,31)\cdot 120\% = 63990,86\cdot 1,2 = 76789,03$ *грн*

Витрати на проведення науково-дослідної роботи на тему «Кіберполігон для розгортання та виявлення бот-мереж» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.18)$$

$B_{заг}$ = *58354,55 +5636,31 +7038,99 +15626,57 +7780,50 +1455,30 +0,00 + 13186,95 + 5392,73 +1322,82 +0,00 +19197,26 +31995,43 +63990,85 = 250217,24 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.19)$$

4.4 Розрахунок економічної ефективності науково-технічної розробки від її впровадження безпосередньо замовником

В умовах ринкової економіки узагальнюючим позитивним результатом, який може отримати потенційний інвестор від впровадження результатів науково-технічної розробки, є зростання величини чистого прибутку, отриманого внаслідок комерційного використання інноваційного продукту.

Результати досліджень, проведених за темою «Кіберполігон для розгортання та виявлення бот-мереж», передбачають можливість комерціалізації протягом 4 років шляхом надання доступу до кіберполігону у вигляді:

навчально-дослідної платформи для підготовки фахівців з кібербезпеки;

інструменту тестування захищеності мережевої інфраструктури;

сервісу для проведення контрольованих експериментів з моделювання атак типу DDoS.

Розробка належить до класу програмно-інфраструктурних рішень, орієнтованих на корпоративних замовників, освітні заклади та організації, що здійснюють дослідження у сфері кібербезпеки.

У цьому випадку майбутній економічний ефект формується на основі таких даних:

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	4000	8000	5000	2000

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 210,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 67,81 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [37]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 40\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 =$(67,81·200000,00+277,81·4000)·0,83·0,4·(1-0,18/100%)=3994642,86 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 =$(67,81·200000,00+277,81·12000)·0,83·0,4·(1-0,18/100%)=4599690,81 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 =$(67,81·200000,00+277,81·17000)·0,83·0,4·(1-0,18/100%)=4977845,78 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 =$(67,81·200000,00+277,81·19000)·0,83·0,4·(1-0,18/100%)=5129107,77 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,12;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =399132,50/(1+0,12)[1]+4398851,42/(1+0,12)[2]+4664207,78/(1+0,12)[3]+
+4792891,65/(1+0,12)[4]=3566645,41+3666845,35+3543132,31+3259640,72=
=13021425,31 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.22)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=1,5;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 256642,49 грн.

$PV = k_{інв} \cdot ЗВ$ = 1,5 · 256642,49 = 384963,74 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (4.23)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 14036263,79 грн;

PV – теперішня вартість початкових інвестицій, 384963,74 грн.

$E_{абс} = ПП - PV$ = 14036263,79 - 384963,74 = 13651300,04 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.24)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 13651300,04 грн;

PV – теперішня вартість початкових інвестицій, 384963,74 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+13651300,04/384963,74)^{1/4} = 1,73.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.25)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,3.

$\tau_{мін}$ = 0,11+0,3 = 0,41 < 1,73 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Кфберполігон для розгортання та виявлення бот-мереж» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (4.26)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,73 = 0,57 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновки до розділу

Відповідно до проведених досліджень рівень комерційного потенціалу розробки за темою «Кіберполігон для розгортання та виявлення бот-мереж» становить 41 бал, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,99 рази.

Також термін окупності становить 0,57 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Кіберполігон для розгортання та виявлення бот-мереж».

ВИСНОВОК

У першому розділі проведено аналіз сучасних інформаційних джерел і наукових публікацій у сфері функціонування бот-мереж та атак типу DDoS. Розглянуто класифікацію бот-мереж, основні підходи до їх побудови та керування, а також існуючі методи виявлення та протидії. Проаналізовано поняття кіберполігону як інструменту навчання і досліджень у галузі кібербезпеки. На основі проведеного аналізу сформульовано мету роботи та визначено основні задачі, необхідні для її досягнення.

У другому розділі розроблено модель кіберполігону та сформовано вимоги до його функціонування. Запропоновано архітектуру кіберполігону, що включає компоненти для розгортання бот-мереж, генерації мережевого трафіку та моніторингу атак. Описано типові сценарії використання кіберполігону, зокрема для моделювання атак типу DDoS, а також визначено підходи до виявлення аномальної активності та шкідливого трафіку.

У третьому розділі здійснено практичну реалізацію запропонованого кіберполігону. Проаналізовано та обґрунтовано вибір програмних і мережевих компонентів, розгорнуто тестове середовище та описано його конфігурацію. Проведено експерименти з моделювання атак, продемонстровано процес моніторингу та виявлення бот-мережевої активності, а також отримано основні метрики продуктивності й навантаження системи.

В економічній частині виконано розрахунок витрат на розробку та впровадження кіберполігону, а також обґрунтовано економічну доцільність його можливого використання та комерціалізації.

Отримані результати підтверджують ефективність запропонованого підходу та можливість практичного використання розробленого кіберполігону для навчальних, дослідницьких і прикладних завдань у сфері кібербезпеки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ENISA Threat Landscape 2025. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2025 (дата звернення: 04.09.2025).

2. Hyper-volumetric DDoS attacks skyrocket: Cloudflare’s 2025 Q2 DDoS threat report. URL: https://blog.cloudflare.com/ddos-threat-report-for-2025-q2 (дата звернення: 04.09.2025).

3. Evaluation of vulnerability reproducibility in container-based Cyber Range URL: https://arxiv.org/abs/2010.16024 (дата звернення 04.09.2025).

4. Tarman et al., «Comparing reproduced cyber experimentation studies» URL: https://cset21.isi.edu/papers/cset21-7.pdf (дата звернення 04.09.2025)

5. Scalable evaluation of blue team posture (Bianchi et al.) URL: https://arxiv.org/abs/2312.17221 (дата звернення 04.09.2025)

6. MITRE ATT&CK – Command and Control URL: https://attack.mitre.org/tactics/TA0011 (дата звернення 22.09.2025).

7. Alto Networks. What is botnet?. Palo Alto Networks Cyberpedia. URL: https://www.paloaltonetworks.com/cyberpedia/what-is-botnet (дата звернення: 22.09.2025).

8. Md Abu Sayed1, Asif Rahman, Christopher Kiekintveld, Sebastian García. Fine-tuning Large Language Models for DGA and DNS Exfiltration Detectionю. University of Texas at El Paso, El Paso, Texas. URL: https://arxiv.org/html/2410.21723v1 (дата звернення: 22.09.2025).

9. JIS – Eurasip Journals article (2024) URL: https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-024-00169-0 (дата звернення: 22.09.2025).

10. Spamhaus. Botnet Threat Update (Jul–Dec 2024). URL: https://www.spamhaus.org/resource-hub/botnet-c-c/botnet-threat-update-jul-to-dec-2024 (дата звернення: 22.09.2025).

11. How accessible is cybersecurity training? A survey on the accessibility and technology stack of Cyber Ranges», Christoph Steininger et al., 2025. URL:

https://d197for5662m48.cloudfront.net/documents/publicationstatus/253322/preprint_pdf/660171fc4c27bf8901fc6f3f83510f12.pdf (дата звернення: 01.10.2025).

12. Cyber Range Features Checklist – List of European Providers (2025) URL: https://ecs-org.eu/ecso-uploads/2025/02/Cyber_Range_Features_Checklist__List_of_European_Providers_2025.pdf (дата звернення 01.10.2025).

13. FLARE-VM : A Comprehensive Guide To Establishing A Reverse Engineering Lab On Windows (2025) URL: https://kalilinuxtutorials.com/flare-vm/ (дата звернення 01.10.2025).

14. Codio «AI Lab Environments | Seamless Integration for Real-World URL: https://www.codio.com/virtual-labs/cyber-range-environments (дата звернення 01.10.2025).

15. How virtual labs and cyber range simulations are different 2020 URL: https://www.infosecinstitute.com/resources/cyber-range/how-virtual-labs-and-cyber-range-simulations-are-different (дата звернення 01.10.2025).

16. Dockerized Android: a container-based platform to build mobile Android scenarios for Cyber Ranges URL: https://arxiv.org/abs/2205.09493 (дата звернення 01.10.2025).

17. Cyber Range Technical Requirements URL: https://kb.uscyberrange.org/faq/technical-requirements.html (дата звернення 01.10.2025).

18. THE CYBER RANGE A GUIDE URL: https://www.nist.gov/system/files/documents/2023/09/29/The%20Cyber%20Range_A%20Guide.pdf (дата звернення 01.10.2025).

19. Design and setup of C2 traffic redirectors. Medium. URL: https://ditrizna.medium.com/design-and-setup-of-c2-traffic-redirectors-ec3c11bd227d (дата звернення: 0.0.2025).

20. Bluscreenofjeff. Red-Team-Infrastructure-Wiki. GitHub. URL: https://github.com/bluscreenofjeff/Red-Team-Infrastructure-Wiki (дата звернення 10.10.2025).

21. NixOS with flakes – Modularize the configuration. URL: https://nixos-and-flakes.thiscute.world/nixos-with-flakes/ modularize-the-configuration (дата звернення: 10.10.2025).

22. DDoS-attack detection using artificial neural networks in Matlab Leonid M. Kupershtein, Tatiana B. Martyniuk, Olesia P. Voitovych, Bohdan V. Kulchytskyi, Andrii V. Kozhemiako, Daniel Sawicki, Mashat Kalimoldayev Proceedings Volume 11176, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2019; 111761S (2019) URL: https://doi.org/10.1117/12.2536478 (дата звернення: 20.10.2025).

23. Якімов О. П., Войтович О. П. Elastic search як оптимальне рішення пошуку та аналізу подій кібербезпеки у режимі реального часу. Матеріали LII Науково-технічної конференції підрозділів ВНТУ, Вінниця, 21-23 червня 2023 р. (дата звернення: 20.10.2025).

24. MITRE Caldera Automated Adversary Emulation Platform. URL: https://caldera.mitre.org (дата звернення: 12.10.2025).

25. Hack The Box Cybersecurity Training Platform. URL: https://www.hackthebox.com (дата звернення: 12.10.2025).

26. Proxmox Virtual Environment Documentation 2025. URL: https://pve.proxmox.com/wiki/Documentation (дата звернення: 01.11.2025).

27. Docker Documentation. URL: https://docs.docker.com (дата звернення: 01.11.2025).

28. LXC / LXD Linux Containers Documentation. URL: https://linuxcontainers.org (дата звернення: 12.10.2025).

29. Security Onion Documentation. URL: https://docs.securityonion.net (дата звернення: 12.10.2025).

30. Elastic Stack Documentation (Elasticsearch, Logstash, Kibana). URL: https://www.elastic.co/guide (дата звернення: 12.10.2025).

31. Suricata User Guide. URL: https://suricata.readthedocs.io (дата звернення: 12.10.2025).

32. 32. Zeek Network Security Monitor Documentation. URL: https://docs.zeek.org (дата звернення: 12.10.2025).

33. 33. OWASP WebGoat Project. URL: https://owasp.org/www-project-webgoat (дата звернення: 12.10.2025).

34. Open vSwitch Documentation. URL: https://www.openvswitch.org/support/dist-docs (дата звернення: 12.10.2025).

35. NixOS Manual Declarative System Configuration. URL: https://nixos.org/manual/nixos/stable (дата звернення: 12.10.2025).

36. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с (дата звернення: 20.10.2025).

37. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с. (дата звернення: 20.10.2025).

ДОДАТКИ

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Кіберполігон для розгортання та виявлення бот-мереж

Автор роботи: Паламарчук Владислав Олегович

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ, група 1 БС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **0,08** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

В. о. зав. кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних т.. комунікаційних систем» к.т.н., доцент _____ Олеся ВОЙТОВИЧ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Олеся ВОЙТОВИЧ

Здобувач _____ Владислав ПАЛАМАРЧУК

ІЛЮСТРАТИВНА ЧАСТИНА

КІБЕРПОЛІГОН ДЛЯ РОЗГОРТАННЯ ТА ВИЯВЛЕННЯ БОТ-МЕРЕЖ

Виконав: студент 2 курсу групи 1БС-24 м спеціальності 125 Кібербезпека та захист інформації

_____ Владислав ПАЛАМАРЧУК

Керівник: к. т. н., доц., доцент каф. ЗІ

_____ Олеся ВОЙТОВИЧ

«16» 12 2025 р.

АРХІТЕКТУРА КІБЕРПОЛІГОНУ

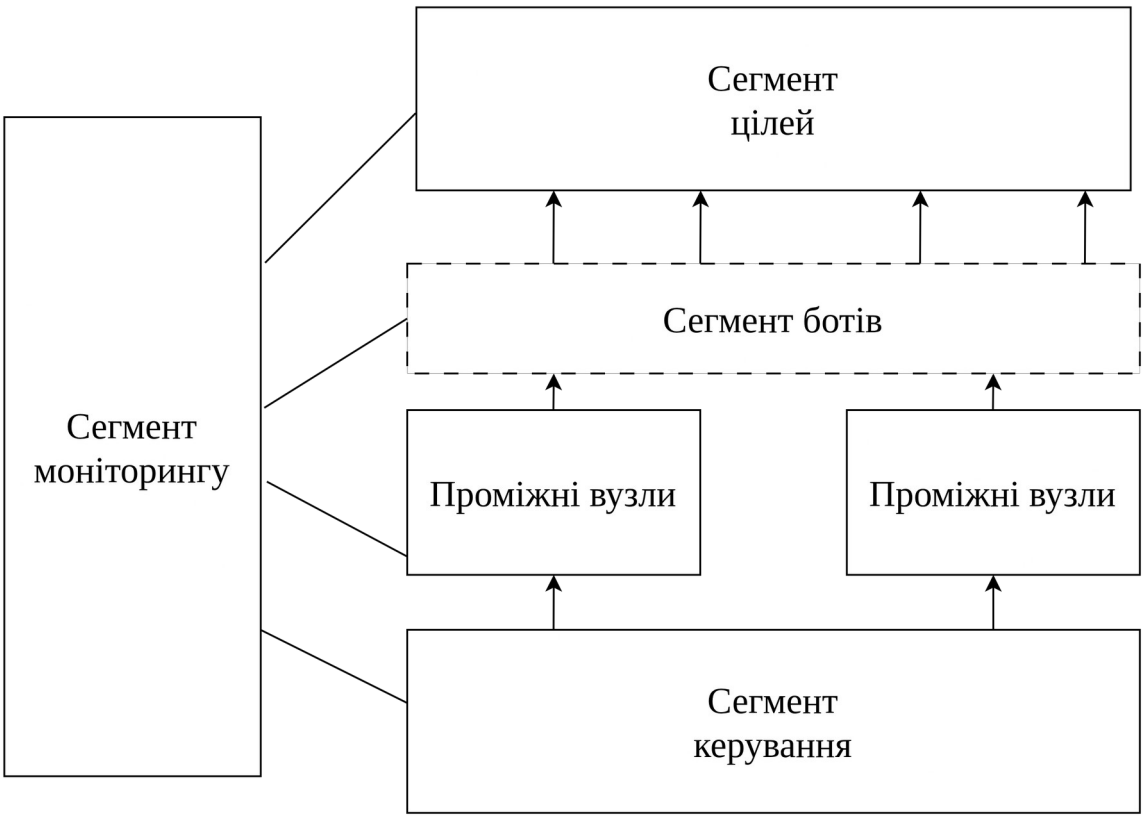


СХЕМА МОНІТОРИНГУ ТА ВИЯВЛЕННЯ АТАК

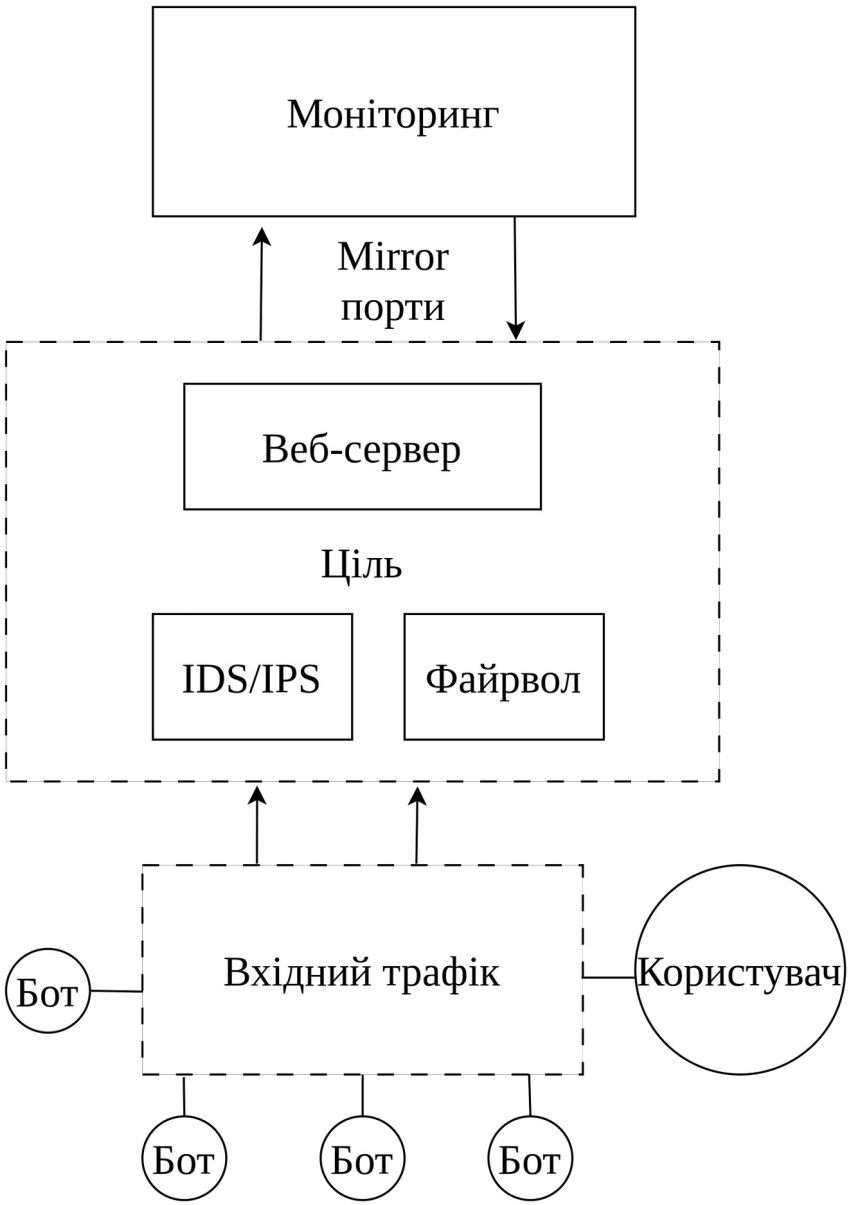
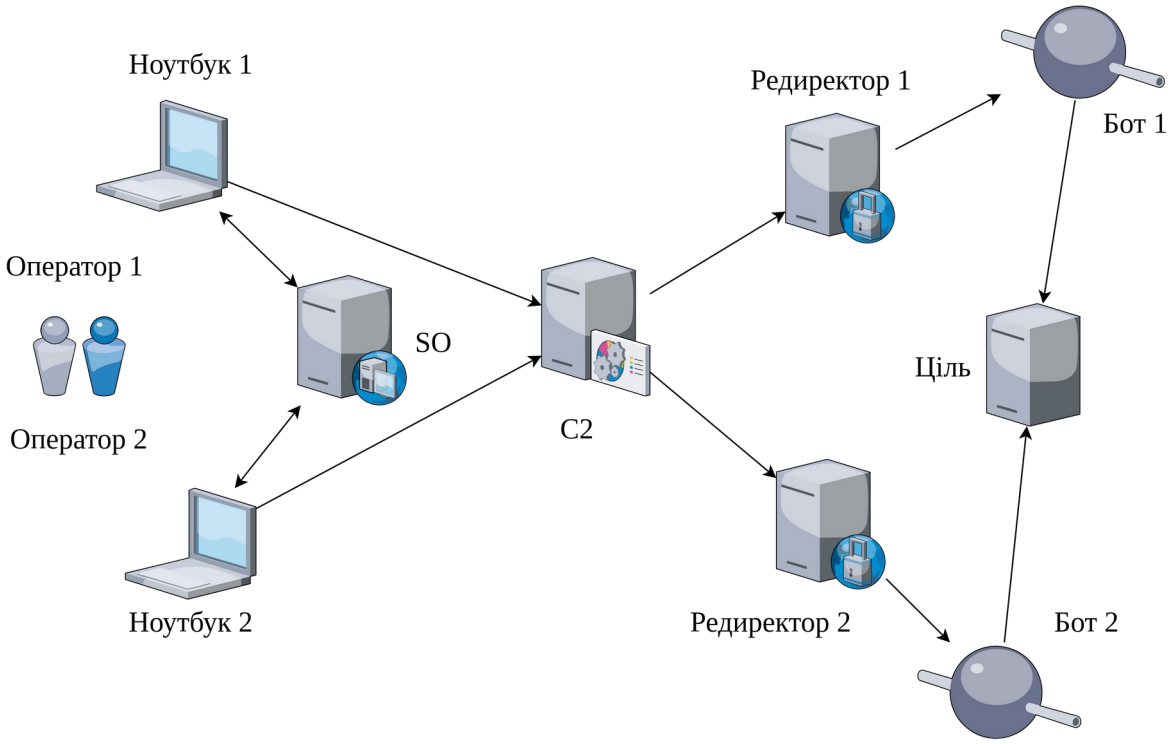
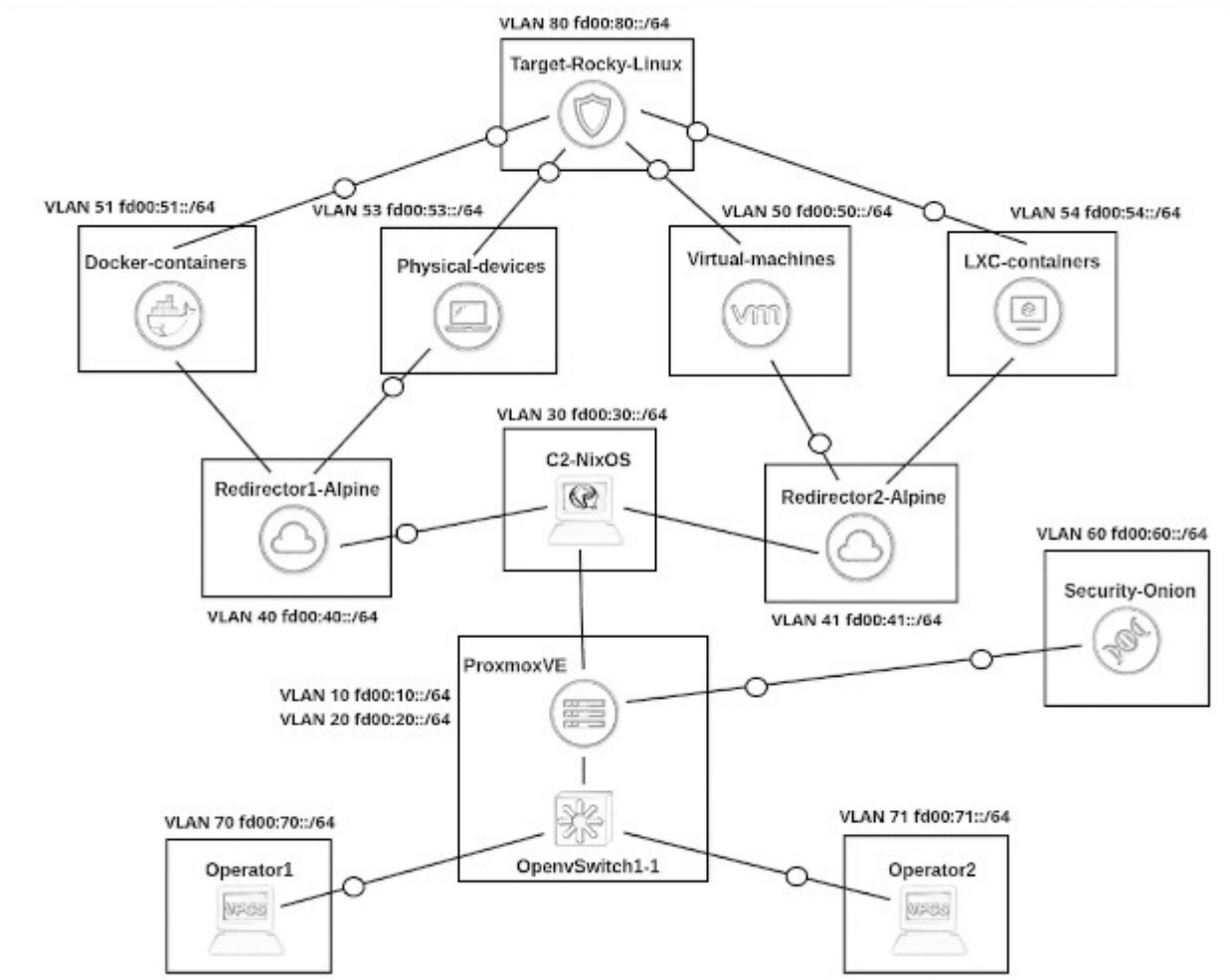


СХЕМА ВЗАЄМОДІЇ КОМПОНЕНТІВ КІБЕРПОЛІГОНУ



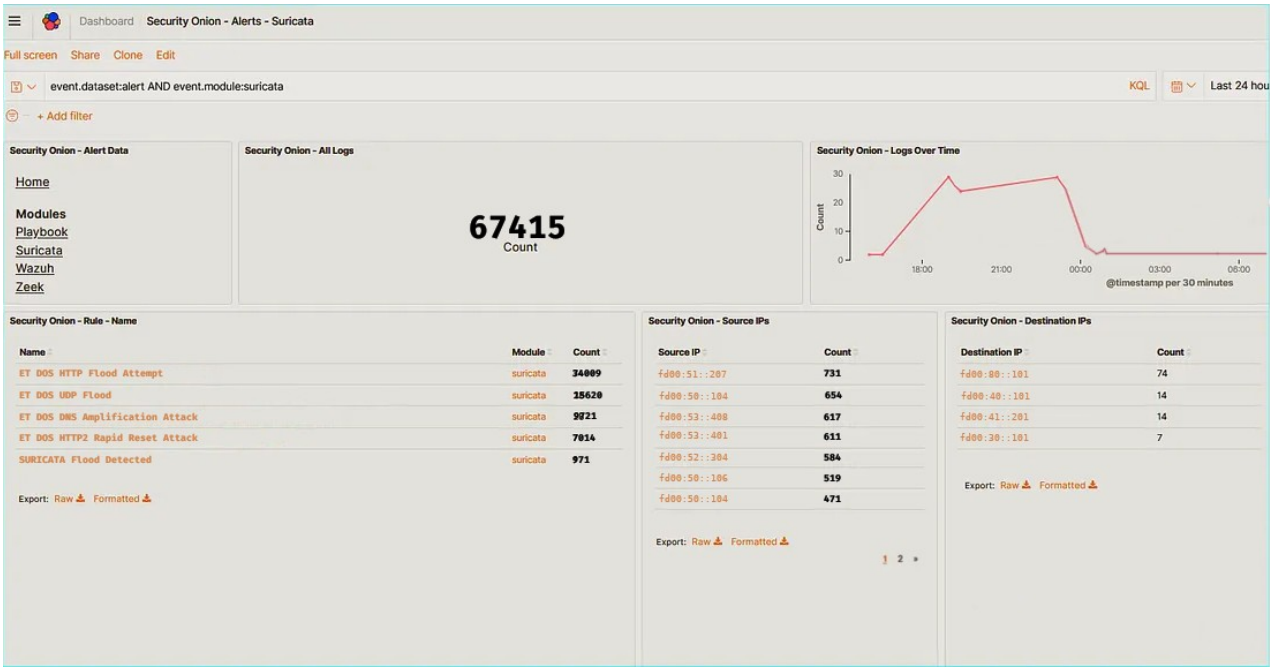
ТОПОЛОГІЯ КІБЕРПОЛІГОНУ В СЕРЕДОВИЩІ GNS3



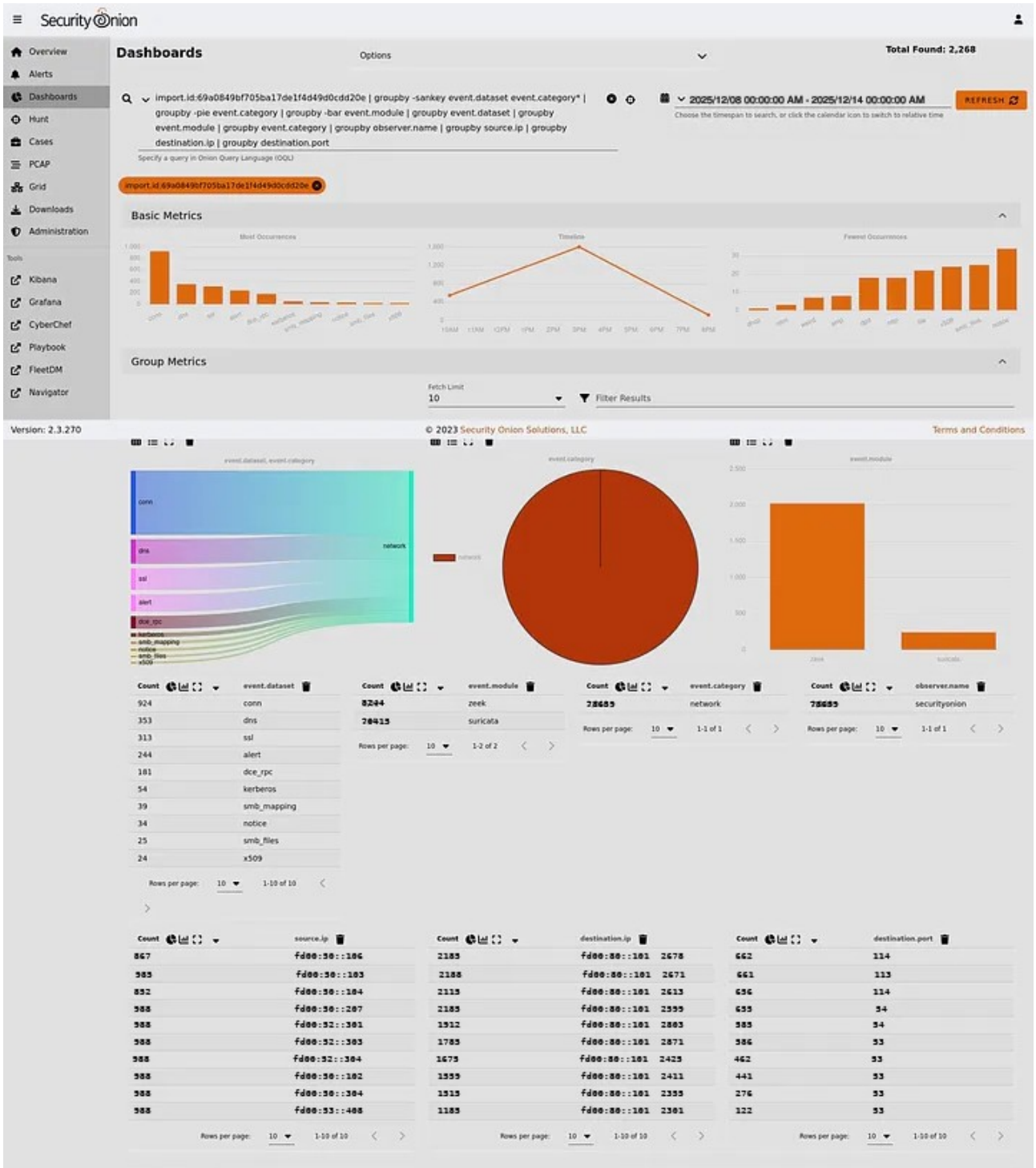
ПЕРЕВІРКА СТАТУСУ БОТІВ

```
[mriya@svitoglyad:~]$ ./Documents/del/check-bots.sh
NAME            IP                      STATUS   PING
-----------------------------------------------------------
vm-bot-01       fd00:50::101            alive    12ms
vm-bot-02       fd00:50::102            alive    11ms
vm-bot-03       fd00:50::103            alive    13ms
vm-bot-04       fd00:50::104            alive    14ms
vm-bot-05       fd00:50::105            alive    12ms
vm-bot-06       fd00:50::106            alive    15ms
vm-bot-07       fd00:50::107            unknown
vm-bot-08       fd00:50::108            unknown

dock-bot-01     fd00:51::201            alive     7ms
dock-bot-02     fd00:51::202            alive     6ms
dock-bot-03     fd00:51::203            alive     8ms
dock-bot-04     fd00:51::204            alive     7ms
dock-bot-05     fd00:51::205            alive     9ms
dock-bot-06     fd00:51::206            alive     6ms
dock-bot-07     fd00:51::207            alive     7ms
dock-bot-08     fd00:51::208            alive     8ms

lxc-bot-01      fd00:52::301            alive    10ms
lxc-bot-02      fd00:52::302            alive     9ms
lxc-bot-03      fd00:52::303            alive    11ms
lxc-bot-04      fd00:52::304            alive    10ms
lxc-bot-05      fd00:52::305            alive     9ms
lxc-bot-06      fd00:52::306            alive    12ms
lxc-bot-07      fd00:52::307            alive    10ms
lxc-bot-08      fd00:52::308            alive    11ms

iot-bot-01      fd00:53::401            alive    28ms
iot-bot-02      fd00:53::402            unknown
iot-bot-03      fd00:53::403            unknown
iot-bot-04      fd00:53::404            alive    35ms
iot-bot-05      fd00:53::405            alive    27ms
iot-bot-06      fd00:53::406            alive    33ms
iot-bot-07      fd00:53::407            alive    30ms
iot-bot-08      fd00:53::408            alive    34ms

Summary: 28/32 bots reachable
```

РЕЗУЛЬТАТ ВИЯВЛЕННЯ АТАК ЧЕРЕЗ SURICATA



РЕЗУЛЬТАТ ЛОГУВАННЯ ZEEK



СКРІНШОТ ПАНЕЛІ МОНІТОРИНГУ