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

# ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему: «Проєкт Deception-підсистеми для мережної інфраструктури організації»

Виконав: студент 4-го курсу групи КБ-22-3
спеціальності 125 «Кіберберзпека»
_____ Артем СУХОВЕЦЬКИЙ
Керівник: старший викладач кафедри КІ та КБ
_____ Ігор ФАЛЬКОВСЬКИЙ
Рецензент: професор кафедри КІ та КБ, д.т.н., доцент
_____ Володимир ВОРОТНІКОВ

Житомир – 2026

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

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
Андрій Єфіменко
2 лютого 2026 р.

**ЗАВДАННЯ**
на випускну кваліфікаційну роботу бакалавра

Студент: Суховецький Артем Олександрович
Тема роботи: «Проєкт Deception-підсистеми для мережної інфраструктури організації»
затверджена Наказом університету від 2 лютого 2026 р. № 58/с
Термін здачі студентом закінченої роботи 08 червня 2026 р.

Вихідні дані роботи:
Мережна інфраструктура умовної організації; лабораторний віртуалізований стенд з Windows-хостом, Linux-хостом, Honeypot VM та Attack VM; сценарії контрольованого сканування, спроб автентифікації та дослідження журналів подій.

Обладнання та програмне забезпечення:
- Персональний комп'ютер на базі процесора AMD Ryzen 5 5600, 16 ГБ оперативної пам'яті, відеокарта NVIDIA GeForce RTX 4060;
- Гіпервізор Oracle VM VirtualBox 7.2 для розгортання ізольованого стенду проєкту;
- Віртуальна машина Attack VM: Kali Linux 2026.1 (64-bit), 2 vCPU, 4096 МБ RAM, 80 ГБ віртуального диска;
- Віртуальна машина Serv-1-SAO: Ubuntu Server 22.04 LTS (64-bit), 2 vCPU, 4096 МБ RAM, 25 ГБ віртуального диска;
- Віртуальна машина WS-1-SAO: Windows 10 Pro (64-bit), 2 vCPU, 2048 МБ RAM, 20 ГБ віртуального диска;
- Віртуальна машина HoneyPot-SAO: Ubuntu Server 22.04 LTS (64-bit), 2 vCPU, 4096 МБ RAM, 16 ГБ віртуального диска;
- SSH/Telnet honeypot Cowrie 3.0.1;
- Вбудовані засоби Linux для аналізу мережевої активності та журналів подій (OpenSSH Server, systemd-journald, ss, tail, cat);
- Віртуальна мережа SAO-NAT (172.22.11.0/24) для взаємодії вузлів стенду проєкту.

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

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| 1 | Андрій ЄФІМЕНКО | 2026 | 02.02.26 |

Студент _____ Артем СУХОВЕЦЬКИЙ

Керівник _____ Ігор ФАЛЬКОВСЬКИЙ

**Календарний план**

| № з/п | Назви етапів випускної кваліфікаційної роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Постановка завдання, уточнення теми, об'єкта і предмета дослідження | 18.05.2026 | Виконано |
| 2 | Добір джерел, аналіз нормативної та технічної бази | 19.05.2026 | Виконано |
| 3 | Розроблення та погодження змісту роботи | 20.05.2026 | Виконано |
| 4 | Підготовка вступу та першого теоретичного розділу | 21.05.2026 | Виконано |
| 5 | Проєктування лабораторного стенду і схеми мережі | 25.05.2026 | Виконано |
| 6 | Підбір програмних засобів і опис архітектури Deception-підсистеми | 28.05.2026 | Виконано |
| 7 | Розгортання віртуальних машин і сервісів-приманок | 02.06.2026 | Виконано |
| 8 | Налаштування honeypot та збір журналів тестових подій | 04.06.2026 | Виконано |
| 9 | Імітація атак у контрольованому середовищі та аналіз логів | 06.06.2026 | Виконано |
| 10 | Формування висновків, додатків, презентації та доповіді | 07.06.2026 | Виконано |
| 11 | Нормоконтроль, перевірка оформлення та підготовка до захисту | 08.06.2026 | Виконано |

Студент _____ Артем СУХОВЕЦЬКИЙ

Керівник _____ Ігор ФАЛЬКОВСЬКИЙ

# АНОТАЦІЯ

Кваліфікаційна робота бакалавра присвячена розробленню проєкту Deception-підсистеми для мережної інфраструктури організації. У роботі розглядається підхід, за якого в межах контрольованого віртуалізованого середовища створюються приманки для фіксації несанкціонованої активності, а подальший аналіз журналів дає змогу оцінити типові дії порушника, його зацікавленість окремими сервісами та практичну цінність honeypot для завдань кібербезпеки.

Метою кваліфікаційної роботи є розроблення проєкту Deception-підсистеми для мережної інфраструктури організації та дослідження її роботи в умовах контрольованої імітації атак на базі віртуального лабораторного середовища. Для досягнення мети передбачено аналіз сучасних підходів до Deception, порівняння honeypot-рішень, формування моделі порушника, проєктування архітектури стенду, розгортання Windows- та Linux-хостів, налаштування honeypot, збирання логів і класифікацію дій порушника.

Очікуваним результатом роботи є проєкт віртуалізованої Deception-підсистеми, яка включає робочі хости, honeypot-вузол, Attack VM, схему адресації, логіку безпечної ізоляції та методику аналізу журналів атак. Практична цінність полягає у можливості використання результатів як навчального та дослідного стенду для демонстрації принципів виявлення небажаної активності, накопичення індикаторів поведінки порушника та підготовки рекомендацій щодо підвищення видимості подій у мережі організації.

Основний вміст пояснювальна записки до випускної кваліфікаційної роботи містить 86 сторінки, 22 таблиць та 25 рисунків. Список використаних джерел містить 25 найменувань і займає 2 сторінки. Невід'ємною частиною робота є 3 додатки.

Ключові слова: DECEPTION-ПІДСИСТЕМА, HONEYPOT, ЖУРНАЛЮВАННЯ, КІБЕРБЕЗПЕКА, МЕРЕЖНА ІНФРАСТРУКТУРА.

# ANNOTATION

The bachelor qualification work is devoted to the development of a Deception subsystem project for an organization network infrastructure. The work considers an approach in which controlled decoy assets are created inside a virtualized laboratory environment in order to record unauthorized activity, study attacker behavior, identify attractive services and evaluate the practical value of a honeypot for cybersecurity tasks.

The purpose of the qualification work is to design a Deception subsystem for an organization network infrastructure and to study its operation under controlled attack simulation in a virtual laboratory environment. The work includes the analysis of Deception technologies, comparison of honeypot tools, formation of an attacker model, design of the laboratory architecture, deployment of Windows and Linux hosts, honeypot configuration, log collection and classification of attacker actions.

The expected result is a virtualized Deception subsystem project that includes production-like hosts, a honeypot node, an Attack VM, an addressing scheme, secure isolation logic and a methodology for analyzing attack logs. The practical value of the work lies in the possibility of using the proposed solution as an educational and research stand for demonstrating detection principles, collecting behavioral indicators and preparing recommendations for improving network security visibility.

The main body of the explanatory note to the final qualification project consists of 86 pages, 22 tables, and 25 figures. The list of references contains 25 entries and spans 2 pages. The project includes 3 appendices.

Keywords: CYBERSECURITY, DECEPTION SUBSYSTEM, HONEYPOT, LOGGING, NETWORK INFRASTRUCTURE.

ЗМІСТ

ВСТУП 8

**РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ЗАСТОСУВАННЯ DECEPTION-ПІДСИСТЕМ У КІБЕРБЕЗПЕЦІ ОРГАНІЗАЦІЇ** 11

**1.1 Аналіз проблеми виявлення атак у мережній інфраструктурі організації** ..... 11

**1.2 Сутність технологій Deception, honeypot і honeynet у системах кіберзахисту** ..... 15

1.3 Порівняльний аналіз інструментів побудови honeypot-середовища ..... 19

1.4 Модель порушника, сценарії атак та технічні вимоги до Deception-підсистеми ..... 23

**Висновки до розділу 1** ..... 28

**РОЗДІЛ 2 ПРОЄКТУВАННЯ DECEPTION-ПІДСИСТЕМИ НА ОСНОВІ ВІРТУАЛІЗОВАНОГО ЛАБОРАТОРНОГО СЕРЕДОВИЩА** 29

**2.1 Формування технічного завдання та вимог до експериментального стенду** ..... 29

**2.2 Архітектура мережі, схема адресації та ролі віртуальних машин** ..... 32

**2.3 Обґрунтування вибору операційних систем, сервісів і програмних засобів** ..... 36

**2.4 Проєктування логіки маскування сервісів і перенаправлення уваги порушника** ..... 39

**2.5 Проєктування підсистеми журналювання, ізоляції та безпечного проведення експерименту** 43

**Висновки до розділу 2** ..... 47

**РОЗДІЛ 3 РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ DECEPTION-ПІДСИСТЕМИ** 49

**3.1 Порядок розгортання та перевірки віртуального стенду** ..... 49

**3.2 Налаштування honeypot та підготовка журналювання подій** ..... 57

**3.3 Сценарії контрольованої імітації атак** ..... 63

**3.4 Аналіз журналів та класифікація дій порушника** ..... 70

**3.5 Аналіз кількісних показників роботи Deception-підсистеми** ..... 76

**3.6 Оцінка ефективності Deception-підсистеми та практичні рекомендації** ..... 77

**Висновки до розділу 3** ..... 81

**ВИСНОВКИ** 82

**СПИСОК ВИКОРИСТАНИХ** 84

ДОДАТКИ 87

Додаток А. Таблиця адресації вузлів експериментального середовища ..... 87

Додаток В. Основні параметри конфігурації Cowrie ..... 88

Додаток С. Фрагмент журналу подій Cowrie ..... 89

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

APT - Advanced Persistent Threat - складна цільова кібератака або група дій, орієнтована на тривале приховане перебування в системі.

HTTP - HyperText Transfer Protocol - протокол передавання гіпертексту.

IDS - Intrusion Detection System - система виявлення вторгнень.

IP - Internet Protocol - протокол мережного рівня для адресації та маршрутизації пакетів.

NAT - Network Address Translation - трансляція мережних адрес.

RDP - Remote Desktop Protocol - протокол віддаленого доступу до робочого столу Windows.

SIEM - Security Information and Event Management - система централізованого збирання, кореляції та аналізу подій безпеки.

SOC - Security Operations Center - центр моніторингу та реагування на події кібербезпеки.

SSH - Secure Shell - захищений протокол віддаленого адміністрування.

TCP - Transmission Control Protocol - транспортний протокол із встановленням з'єднання.

VM - Virtual Machine - віртуальна машина.

Honeypot - контрольований вузол-приманка, який імітує привабливий сервіс або систему та фіксує взаємодію з порушником.

Honeynet - група взаємопов'язаних honeypot-вузлів, що імітує фрагмент мережної інфраструктури.

Deception-підсистема - комплекс приманок, контрольованих сервісів, журналювання та процедур аналізу, призначений для виявлення і дослідження небажаної активності.

# ВСТУП

Сучасна мережна інфраструктура організації майже завжди поєднує робочі станції, сервери, віддалені служби, веб-ресурси, облікові записи користувачів і засоби адміністрування. Навіть невеликий набір доступних сервісів створює для порушника простір для розвідки: можна перевіряти активні вузли, відкриті порти, типи служб, реакцію систем на спроби входу та можливість використання помилково налаштованих облікових даних. Через це завдання кіберзахисту вже не зводиться лише до блокування трафіку на периметрі. Не менш важливо своєчасно побачити небажану активність, зібрати дані про її характер і зрозуміти, як саме порушник поводиться після виявлення привабливої цілі.

У традиційній моделі захисту основний акцент робиться на запобіганні: фільтрації трафіку, антивірусному контролі, правилах доступу, оновленні програмного забезпечення та політиках автентифікації. Такі механізми залишаються необхідними, однак вони не завжди дають повну картину поведінки порушника. Частина атак не завершується прямим зламом, але залишає корисні сліди у вигляді сканувань, повторюваних спроб входу, звернень до типових портів, перевірки банерів сервісів або спроб виконання команд після успішної взаємодії з фальшивим середовищем. Саме такі дії доцільно не тільки блокувати, а й досліджувати.

Deception-підхід вирішує це завдання через створення контрольованих об’єктів, які виглядають для порушника як справжні ресурси організації, але фактично не містять критичних даних і не виконують виробничих функцій. Honeypot у такій архітектурі є не «зайвим сервером», а спеціально підготовленою точкою спостереження. Коли порушник взаємодіє з приманкою, система може зафіксувати джерело підключення, час, використаний протокол, введені логіни і паролі, типові команди, повторюваність дій та інші ознаки, які потім використовуються для аналізу інциденту.

Особливу практичну цінність має лабораторне дослідження Deception-підсистеми на базі віртуальних машин. Такий формат дозволяє побудувати схему мережі без ризику для реальних ресурсів організації, розгорнути Windows-хост, Linux-хост, honeypot і окрему Attack VM, перевірити сценарії взаємодії та отримати

наочні журнали атак. Для дипломної роботи це важливо, тому що результат не обмежується описом технології: можна показати принцип роботи мережної приманки, структуру стенду, логіку відкритих сервісів, приклади подій у журналах і висновки щодо поведінки порушника.

Нормативний контекст теми пов'язаний із загальною вимогою забезпечення кібербезпеки та захисту інформаційних ресурсів. Закон України «Про основні засади забезпечення кібербезпеки України» визначає організаційні та правові основи захисту[1] життєво важливих інтересів людини, суспільства і держави у кіберпросторі. У технічному вимірі тема також узгоджується з підходом NIST Cybersecurity Framework 2.0[2], де поряд із функціями захисту підкреслено значення виявлення, реагування та відновлення. Deception-підсистема безпосередньо підтримує блок «DETECT», оскільки створює додаткові точки фіксації підозрілої активності.

Метою кваліфікаційної роботи є розроблення проєкту Deception-підсистеми для мережної інфраструктури організації та дослідження її роботи в умовах контрольованої імітації атак на базі віртуального лабораторного середовища.

Для досягнення визначеної мети у роботі потрібно розв'язати такі завдання:

- проаналізувати проблему виявлення атак у мережній інфраструктурі організації та визначити місце Deception-підсистеми серед інших засобів кіберзахисту;
- розкрити сутність понять Deception, honeypot, honeynet, сервіс-приманка, журналювання подій та поведінковий аналіз дій порушника;
- порівняти доступні інструменти побудови honeypot-середовища, зокрема Cowrie, Dionaea, OpenCanary і спрощений фейковий SSH із логуванням;
- сформувати модель порушника та визначити контрольовані сценарії імітації атак у межах лабораторного стенду;
- спроєктувати мережну схему з використанням Windows-хоста, Linux-хоста, Honeypot VM та Attack VM;
- визначити вимоги до журналювання, ізоляції, безпеки експерименту та подальшої класифікації дій порушника;
- підготувати основу для подальшого розгортання, тестування і демонстрації результатів на захисті.

Об’єктом дослідження є процес виявлення та дослідження несанкціонованої активності в мережній інфраструктурі організації.

Предметом дослідження є методи, засоби та сценарії побудови Deception-підсистеми на основі honeypot, віртуальних машин, журналювання подій та аналізу поведінки порушника.

Методи дослідження. У роботі застосовуються методи системного аналізу для визначення місця Deception-підсистеми в архітектурі кіберзахисту, порівняльний аналіз для вибору honeypot-рішення, моделювання для побудови лабораторного стенду, експериментальний метод для контрольованої імітації атак, а також методи класифікації подій для групування дій порушника за типами активності.

Практичне значення одержаних результатів полягає у можливості використання запропонованого проєкту як навчального та дослідного стенду для демонстрації принципів Deception-захисту, збору журналів підозрілої активності, аналізу спроб автентифікації та підготовки рекомендацій щодо підвищення видимості подій у мережі організації. Окремо цінним є те, що стенд не потребує використання реальної інфраструктури і може бути розгорнутий у локальному ізольованому середовищі.

Публікації за результатами роботи відсутні.

Структура та обсяг роботи. Випускна кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків. У першому розділі розглядаються теоретичні основи Deception-підходу, аналізується проблема виявлення атак, порівнюються honeypot-рішення та формуються вимоги до підсистеми. У другому розділі буде подано проєктування лабораторного середовища, схему мережі, адресацію, ролі віртуальних машин і логіку сервісів-приманок. У третьому розділі буде описано розгортання, налаштування, контрольоване тестування, аналіз журналів і класифікацію дій порушника.

# РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ЗАСТОСУВАННЯ DECEPTION-ПІДСИСТЕМ У КІБЕРБЕЗПЕЦІ ОРГАНІЗАЦІЇ

## 1.1 Аналіз проблеми виявлення атак у мережній інфраструктурі організації

Мережна інфраструктура організації є сукупністю технічних, програмних та організаційних компонентів, через які працівники отримують доступ до даних, сервісів і прикладних систем. У практичному вигляді вона може містити робочі станції, сервери, мережне обладнання, служби віддаленого доступу, внутрішні веб-ресурси, файлові сховища, бази даних, засоби моніторингу та облікові записи користувачів. Кожен із цих елементів створює окрему поверхню атаки. Якщо хоча б один сервіс має слабку конфігурацію, застарілу версію або передбачувані облікові дані, порушник може використати його як початкову точку для подальших дій.

У більшості випадків атака на мережну інфраструктуру не починається з прямого проникнення. Спочатку порушник збирає відомості: визначає активні вузли, перевіряє відкриті порти, аналізує банери сервісів, пробує встановити тип операційної системи, шукає віддалені служби або веб-інтерфейси. Такі дії можуть виглядати як звичайний мережний шум, особливо якщо вони виконуються повільно, з різних джерел або з використанням типових пакетів. Через це традиційні журнали не завжди одразу привертають увагу адміністратора.

Проблема виявлення атак посилюється тим, що не всі небажані дії мають очевидний наслідок. Наприклад, одинична невдала спроба входу через SSH або RDP може бути наслідком помилки користувача. Водночас сотні схожих спроб із різними логінами та паролями вже вказують на підбір облікових даних. Якщо ж порушник використовує справжній або випадково підібраний обліковий запис, його активність може маскуватися під звичайну віддалену роботу. MITRE ATT&CK описує brute force[7] як техніку повторюваного вгадування пароля або іншого секрету, а valid accounts[9] - як використання легітимних облікових записів для доступу, закріплення або обходу захисних механізмів.

Особливо показовими для цієї роботи є віддалені служби SSH, RDP і HTTP. SSH часто використовується для адміністрування Linux-серверів і мережного обладнання, RDP - для віддаленого доступу до Windows-систем, а HTTP або HTTPS - для веб-інтерфейсів адміністрування, внутрішніх порталів або тестових застосунків. Відповідно, ці сервіси природно привертають увагу порушника. У MITRE ATT&CK окремо описано використання SSH як підвиду техніки Remote Services[8], коли порушник після отримання облікових даних може виконувати дії у віддаленій системі як авторизований користувач.

З позиції адміністратора складність полягає не тільки у тому, щоб заборонити небезпечні підключення. Потрібно також розуміти, що саме відбувається: чи була це автоматична перевірка портів, чи спроба підбору паролів, чи ручне дослідження системи після входу. Якщо система безпеки фіксує лише факт підключення, але не дає змістовної картини подальших дій, її користь для аналітика обмежена. Саме тому у сучасному захисті важливе місце займають журнали подій, кореляція, поведінковий аналіз і контрольовані середовища спостереження.

Класичні засоби виявлення вторгнень поділяються на мережні, хостові, поведінкові та комбіновані. NIST SP 800-94[5] підкреслює, що IDPS-рішення допомагають організаціям розуміти, проєктувати, налаштовувати, захищати, моніторити і підтримувати системи виявлення та запобігання вторгненням [3]. Однак навіть якісний IDPS не завжди пояснює наміри порушника. Він може сформувати попередження про підозрілий трафік, але не завжди дає змогу безпечно «подивитися», що порушник робив би після того, як сервіс здався йому доступним.

Ще одна практична проблема - хибні спрацювання. У реальній мережі є багато легітимних подій: користувачі забувають паролі, адміністратори тестують підключення, автоматизовані системи перевіряють доступність сервісів, засоби інвентаризації надсилають службові запити. На цьому тлі небажана активність може загубитися. Deception-підхід частково змінює логіку спостереження: якщо ресурс не використовується у нормальному бізнес-процесі, будь-яка взаємодія з ним стає підозрілою за замовчуванням. Це суттєво спрощує інтерпретацію сигналу.

Для дипломного проєкту важливо показати не тільки загальну проблему, а й її відтворення в контрольованому середовищі. Умовна організація може мати Windows-хост і Linux-хост як звичайні вузли, а також окремий honeypot-вузол, який імітує привабливі сервіси. Порушник, представлений Attack VM, бачить у мережі декілька потенційних цілей і взаємодіє з ними. Якщо він обирає honeypot, усі його дії потрапляють до журналів, після чого можуть бути класифіковані.

У межах цієї роботи Deception-підсистема розглядається як додатковий рівень видимості, а не як заміна фаєрволу, антивірусу, IDS або політик доступу. Вона не повинна надавати порушнику доступ до реальних даних. Її завдання - створити контрольований об’єкт інтересу, відокремлений від критичних систем, і надати досліднику інформацію про послідовність дій. У цьому сенсі honeypot виконує роль датчика, який активується не штатним користувачем, а саме підозрілою стороною.

У звітах про загрози останніх років неодноразово підкреслюється значення експлуатації відкритих сервісів, використання облікових даних та автоматизованих спроб проникнення. ENISA Threat Landscape 2025 серед ключових спостережень описує експлуатацію публічних сервісів, використання типових облікових даних та credential brute-forcing як помітні елементи активності порушників [14]. Для обраної теми це означає, що стенд із SSH, RDP, HTTP і honeypot є не довільною лабораторною конструкцією, а наближеним до реальності спрощеним відображенням поширених векторів атак.

Отже, проблема виявлення атак у мережній інфраструктурі полягає у нестачі контексту, складності відрізнення легітимної активності від небажаної, великій кількості подій та потребі у безпечному дослідженні поведінки порушника. Deception-підсистема дозволяє перевести частину цієї проблеми у контрольовану площину: порушник отримує привабливу ціль, а захисник - події, журнали, сценарії та матеріал для аналізу.

Таблиця 1.1

Типові проблеми виявлення атак та роль Deception-підсистеми

| **Проблема** | **Прояв у мережі організації** | **Як допомагає Deception-підсистема** |
|---|---|---|
| Надмірна кількість подій | У журналах одночасно фіксуються легітимні входи, помилки користувачів, автоматичні перевірки та підозрілі запити. | Приманка не має штатних користувачів, тому звернення до неї одразу розглядається як подія інтересу. |
| Недостатній контекст атаки | Звичайний журнал часто показує лише факт підключення або помилку автентифікації. | Honeypot дозволяє фіксувати не тільки спробу входу, а й подальші команди, логіни, паролі та послідовність дій. |
| Складність аналізу намірів | Порушник може діяти повільно або маскуватися під звичайну активність. | Взаємодія з не виробничим ресурсом вказує на зацікавленість у несанкціонованому доступі. |
| Ризик для реальних систем | Експериментальні перевірки на виробничих вузлах можуть створювати небезпеку. | Лабораторний honeypot ізольований і не містить критичних даних. |
| Потреба у матеріалі для навчання | Абстрактні описи атак не завжди демонструють реальну послідовність подій. | Журнали honeypot надають приклади для класифікації дій і підготовки висновків. |

Як видно з табл. 1.1, Deception-підхід не усуває потреби у базових засобах захисту, але додає окремий шар спостереження. Його перевага полягає у тому, що взаємодія з приманкою має високий аналітичний пріоритет і може бути використана для раннього виявлення, навчання персоналу та підготовки правил реагування.




Рисунок 1.1 - Узагальнене місце Deception-підсистеми в мережному захисті організації

На рис. 1.1 показано, що Deception-підсистема не є центральним виробничим сервером. Вона розміщується як окремий контрольований сегмент, пов’язаний із журналюванням та аналітичним контуром. Для дипломної роботи саме така логіка є зручною, тому що дозволяє відокремити реальні вузли від вузла-приманки та пояснити, чому взаємодія з honeypot має дослідницьке значення.

## 1.2 Сутність технологій Deception, honeypot і honeynet у системах кіберзахисту

Термін Deception[4] у кібербезпеці позначає групу підходів, які створюють для порушника контрольовану хибну картину інфраструктури. Її мета не у тому, щоб бездумно «замаскувати» реальні ресурси, а в тому, щоб направити небажану активність до безпечних об’єктів, де вона може бути помічена, записана й про-

аналізована. У такій системі використовуються приманки, фальшиві сервіси, контрольовані облікові записи, вигадані файли, мережні вузли, банери служб і журнали подій.

Honeypot[10] є одним із найпоширеніших практичних інструментів Deception-підходу. У спрощеному вигляді це вузол або сервіс, який виглядає привабливо для порушника, але фактично призначений не для обслуговування користувачів, а для спостереження. Якщо до honeypot звертається хтось із мережі, така подія вже є підозрілою, адже штатний бізнес-процес не передбачає використання цього ресурсу. Саме через цю властивість honeypot може давати відносно чисті сигнали порівняно з багатьма іншими журналами.

Honeynet є розвитком ідеї honeypot. Якщо окремий honeypot імітує один вузол або набір служб, то honeynet[10] створює цілий фрагмент мережі-приманки. Він може містити кілька вузлів із різними ролями: фальшивий веб-сервер, умовний файловий ресурс, імітацію бази даних, SSH-сервіс, RDP-сервіс або інші об’єкти. Для дипломної роботи повноцінна honeynet може бути надмірною, але її принцип корисний для розуміння майбутнього розвитку: лабораторний стенд із кількома VM фактично наближається до мінімальної моделі мережної приманки.

За рівнем взаємодії honeypot поділяють на низькоінтерактивні, середньоінтерактивні та високоінтерактивні. Низькоінтерактивні рішення лише імітують базову відповідь сервісу і збирають прості події. Вони безпечніші та простіші, але дають менше інформації про дії порушника. Середньоінтерактивні рішення можуть імітувати оболонку, приймати команди, зберігати сесії та формувати більш змістовні логи. Високоінтерактивні середовища наближаються до реальних систем і дають багатий матеріал, однак потребують значно суворішої ізоляції.

Для обраної теми найбільш доречним є середньоінтерактивний підхід. Він дозволяє показати не тільки факт підключення, а й поведінку після входу: що порушник намагався переглянути, які команди вводив, як реагував на імітоване файлове середовище, чи шукав конфігурації, користувачів або мережні параметри. При цьому такий стенд усе ще може залишатися безпечним, якщо він розгортається в

ізольованому віртуальному сегменті і не має доступу до реальної мережі організації.

Deception-підсистема має кілька функцій. Перша - раннє виявлення. Якщо порушник взаємодіє з приманкою, це створює подію з високим пріоритетом. Друга - збирання інформації. Система фіксує IP-адресу, час, сервіс, спроби автентифікації, використані рядки, команди та інші параметри. Третя - відволікання. Поки порушник взаємодіє з приманкою, він не отримує доступу до справжніх ресурсів. Четверта - навчальна та дослідна функція, оскільки журнали honeypot можна використовувати для аналізу сценаріїв атак.

Важливо розуміти обмеження Deception-підходу. Honeypot не повинен бути єдиним захисним механізмом і не гарантує блокування атаки. Якщо його неправильно розмістити або надати йому зайві права, він сам може перетворитися на ризик. Якщо він занадто очевидний, досвідчений порушник швидко визначить приманку і припинить взаємодію. Якщо він не журналює події або не передає їх до аналітичного контуру, його практична цінність знижується. Тому у проєкті потрібно одночасно враховувати правдоподібність, безпеку і повноту журналів.

У межах дипломного проєкту Deception-підсистема повинна виконувати дослідну, демонстраційну та аналітичну функції. Вона не призначена для реального затримання порушника або активної протидії. Її завдання - відтворити типовий фрагмент мережної інфраструктури, показати принцип роботи honeypot, отримати журнали контрольованої атаки і зробити висновки про те, які сервіси виглядають найбільш привабливими.

З технічного погляду Deception-підсистема складається з кількох рівнів. Перший рівень - мережний, де визначаються адресація, сегменти, маршрути і правила доступу. Другий рівень - сервісний, де налаштовуються SSH, RDP, HTTP або їх імітації. Третій рівень - журналювання, де зберігаються події, сесії і команди. Четвертий рівень - аналітичний, де дослідник класифікує події за типами дій порушника. Саме така багаторівнева структура і буде використана у подальшому проєктуванні.

Окремо слід звернути увагу на співвідношення реальних і фальшивих сервісів. У завданні передбачено, що Windows- та Linux-хости можуть мати назовні винесені змінені порти служб, а honeypot - реальні порти сервісів-приманок. Це створює для порушника правдоподібну картину: він бачить відкритий SSH або інший сервіс на звичному порту і починає взаємодію. Водночас реальні вузли залишаються менш очевидними, а дослідник отримує журнал подій саме на honeypot.

Таким чином, сутність Deception-підсистеми полягає в активному створенні контрольованої видимості для порушника. На відміну від пасивного очікування інциденту, Deception формує спеціальне середовище, яке саме провокує небажану сторону на взаємодію з безпечним об'єктом. Це робить honeypot корисним не тільки для виявлення, а й для пояснення логіки атаки.

Таблиця 1.2

Основні поняття Deception-підходу в межах дипломного проєкту

| **Поняття** | **Зміст у роботі** | **Практичне значення** |
|---|---|---|
| Deception-підсистема | Комплекс контрольованих приманок, сервісів і процедур журналювання у віртуалізованому стенді. | Дозволяє показати додатковий рівень виявлення та дослідження небажаної активності. |
| Honeypot | Окремий вузол або сервіс, що імітує привабливу ціль для порушника. | Фіксує підключення, спроби входу, команди та інші дії. |
| Honeynet | Група взаємопов'язаних вузлів-приманок, що імітує фрагмент мережі. | Може бути напрямом подальшого розвитку лабораторного стенду. |
| Сервіс-приманка | SSH, HTTP або інший сервіс, доступний для взаємодії порушника. | Створює точку входу для контрольованого запису подій. |
| Журналювання | Збереження подій, облікових даних, сесій, команд і часових міток. | Дає матеріал для класифікації дій та формування висновків. |

З табл. 1.2 видно, що Deception-підсистема не зводиться до одного програмного пакета. Вона включає логіку розміщення, налаштування сервісів, правила ізоляції, збирання подій і подальшу аналітичну інтерпретацію. У дипломному проєкті ці компоненти потрібно розглядати разом, інакше honeypot залишиться лише технічним сервісом без дослідної цінності.

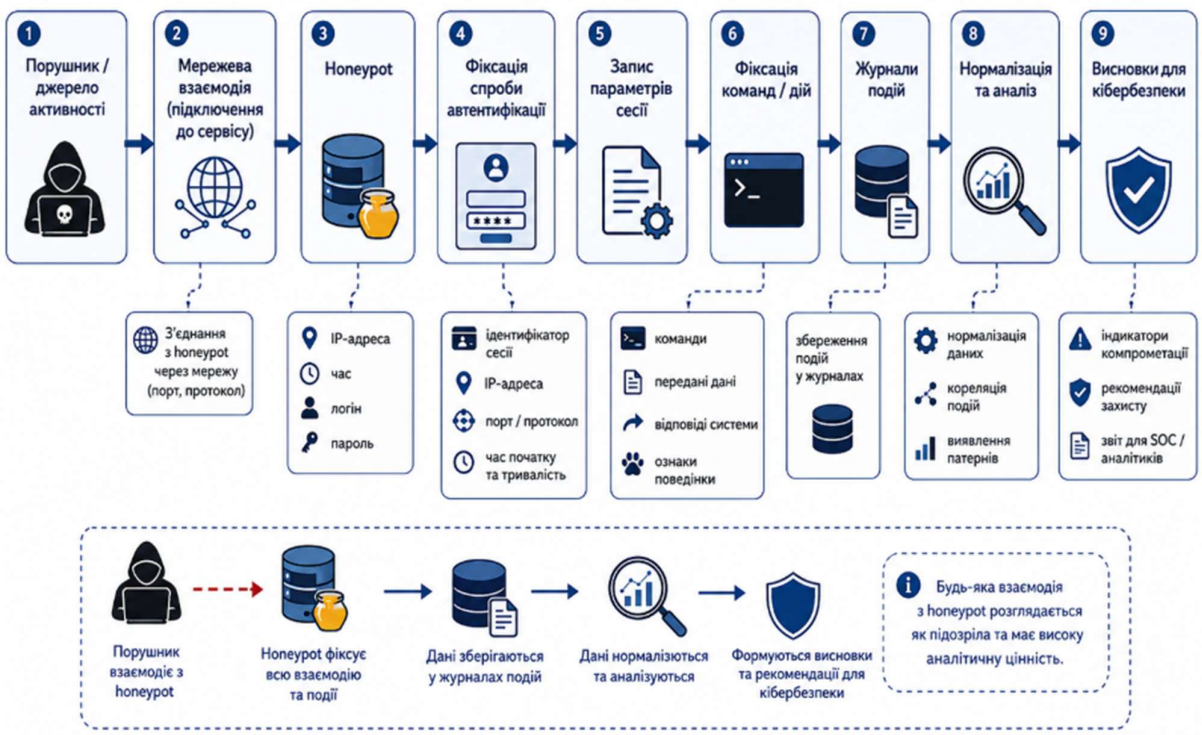



Рисунок 1.2 - Логіка фіксації взаємодії порушника з honeypot

На рис. 1.2 узагальнено послідовність, яка є основою подальшого тестування. Порушник спочатку шукає доступні вузли і сервіси, потім взаємодіє з приманкою, після чого honeypot формує записи про подію. Для захисту організації важливий не тільки сам факт підключення, а й те, які саме дії виконувалися під час взаємодії.

## 1.3 Порівняльний аналіз інструментів побудови honeypot-середовища

Вибір інструменту для побудови honeypot-середовища залежить від мети роботи. Якщо потрібно фіксувати спроби входу через SSH та команди після автентифікації, найбільш зручними є рішення, орієнтовані на SSH/Telnet[16, 17]. Якщо метою є збирання шкідливих файлів або взаємодія з різними мережними протоколами, доцільно розглядати багатопротокольні honeypot. Якщо потрібна проста внутрішня «канарка» для сигналізації про небажані звернення, можна використовувати легкі системи, які імітують декілька служб і надсилають попередження.

У технічному завданні до цієї дипломної роботи прямо зазначено, що ключова особливість проєкту полягає в імітації та дослідженні атак. Тобто рішення повинно не просто створити відкритий порт, а дати матеріал для аналізу: спроби

логіна, підібрані пари логін-пароль, сесії, команди, часову послідовність подій. Саме тому головним критерієм вибору є якість журналювання взаємодії, а не кількість підтримуваних протоколів.

Cowrie є одним із найзручніших варіантів[10, 11] для такого завдання. Офіційна документація визначає Cowrie як середньо- або високоінтерактивний SSH/Telnet[16, 17] honeypot, призначений для журналювання brute force атак та shell-взаємодії порушника [9]. У середньоінтерактивному режимі він імітує UNIX-подібну систему, а в проксі-режимі може працювати як посередник для спостереження за взаємодією з іншою системою. Для даної роботи достатньо середньоінтерактивного режиму, оскільки потрібно показати спроби входу і команди після потрапляння до приманки.

Перевагою Cowrie є те, що він добре відповідає сценарію brute-force SSH. Порушник бачить SSH-сервіс, пробує різні облікові дані, а система зберігає спроби автентифікації. Якщо налаштування дозволяє контрольований вхід до імітованої оболонки, можна фіксувати команди, які вводяться після входу. Це особливо зручно для дипломної роботи, тому що на захисті можна показати не абстрактні висновки, а конкретні фрагменти журналів: час, логін, пароль, тип події, виконані команди.

Dionaea має інший акцент[13]. Репозиторій проєкту описує його як honeypot, що підтримує низку протоколів, зокрема FTP, HTTP, SMB, MSSQL, MySQL, SIP, TFTP та інші, а також різні механізми журналювання. Такий інструмент корисний для збору шкідливих підключень і спроб експлуатації мережних сервісів. Однак для дипломної роботи, де головним сценарієм є SSH-підбір і аналіз команд, Dionaea може бути допоміжним або альтернативним засобом, але не основним.

OpenCanary орієнтований на іншу практичну модель[12]. Документація визначає його як daemon, який запускає canary services і формує попередження, коли ці сервіси використовуються або зловживаються. Репозиторій Thinkst підкреслює, що OpenCanary є багатопротокольним honeypot з низькими ресурсними вимогами, а основний сценарій його застосування - виявлення порушників після

потрапляння у непублічну мережу. Для організації це корисна концепція, але для глибокого аналізу команд у SSH-сесії Cowrie дає більш прямий результат.

Окремо можна розглянути спрощений фейковий SSH із логуванням. Його перевага - простота: можна створити сервіс, який приймає підключення, записує введені дані та завершує взаємодію. Однак такий варіант має слабку правдоподібність, обмежену функціональність і невелику дослідну цінність. Він може бути корисним як демонстраційний елемент, але для дипломної роботи краще використовувати готове honeypot-рішення з усталеною логікою запису подій.

Порівняння інструментів варто виконувати не лише за технічними характеристиками, а й за відповідністю меті роботи. Якщо обрати надто складний продукт, більша частина дипломної роботи перетвориться на опис його встановлення. Якщо обрати надто простий продукт, не буде достатньо матеріалу для аналізу поведінки порушника.

З урахуванням цих критеріїв базовим інструментом для подальшого проєктування доцільно обрати Cowrie. Він безпосередньо відповідає сценарію SSH/Telnet honeypot, фіксує brute-force активність і shell-взаємодію, має зрозумілу структуру журналів, добре підходить для ізольованого лабораторного середовища та дозволяє отримати матеріал для аналітичного розділу. Dionaea і OpenCanary можуть бути подані як альтернативи або можливості розширення, але основна лінія дипломної роботи має залишатися навколо Cowrie.

Важливо також врахувати, що інструмент honeypot не є самодостатнім без правильної мережної архітектури. Навіть найкраще рішення не дасть якісного результату, якщо воно розміщене в хаотичній мережі, має неконтрольований доступ до Інтернету або не ізольоване від справжніх даних. Тому вибір Cowrie є лише першим кроком. Далі потрібно визначити адресацію, ролі VM, правила взаємодії, порядок збору журналів і сценарії тестування.

Для захисту дипломної роботи вибір Cowrie має ще одну перевагу - просту демонстраційну логіку. Можна пояснити, що Windows- і Linux-хости імітують звичайні вузли інфраструктури, Attack VM створює контрольовану небажану активність, а Cowrie фіксує взаємодію з SSH/Telnet-приманкою. Така схема зрозуміла

комісії, не потребує демонстрації складного програмного коду і водночас дає достатньо технічної глибини.

Таблиця 1.3

Порівняльна характеристика інструментів для побудови honeypot-середовища

| Інструмент | Основне призначення | Переваги для роботи | Обмеження для роботи | Оцінка доречності |
|---|---|---|---|---|
| Cowrie | SSH/Telnet honeypot для фіксації brute-force і shell-взаємодії. | Добре підходить для сценаріїв підбору SSH, записує логіни, паролі, сесії та команди. | Переважно орієнтований на SSH/Telnet, інші сервіси потрібно закривати додатковими засобами. | Основне рішення. |
| Dionaea | Багатопротокольний honeypot для фіксації шкідливих підключень і взаємодії з різними сервісами. | Підтримує багато протоколів, може розширити дослідження за межі SSH. | Менш зручний для демонстрації shell-команд після входу через SSH. | Альтернатива або розширення. |
| OpenCanary | Модульний canary honeypot для внутрішніх мереж і сигналізації про небажані звернення. | Низькі вимоги до ресурсів, багато сервісів-приманок, зручний для загальної мережної видимості. | Для глибокого аналізу SSH-сесій поступається спеціалізованому Cowrie. | Допоміжний варіант. |
| Фейковий SSH із логуванням | Спрощена імітація сервісу, що записує спроби підключення. | Найпростіший для демонстрації базового принципу. | Низька правдоподібність, слабка аналітична цінність, мало даних про поведінку. | Резервний демонстраційний варіант. |

За результатами порівняння у табл. 1.3 основним рішенням для подальшого проєктування обрано Cowrie. Такий вибір обумовлений не популярністю інструменту самою по собі, а відповідністю конкретним завданням роботи: імітації SSH-сервісу, фіксації спроб автентифікації, запису команд та подальшій класифікації дій порушника.

## 1.4 Модель порушника, сценарії атак та технічні вимоги до Deception-підсистеми

Модель порушника у даній роботі потрібна для того, щоб чітко обмежити межі експерименту. Без неї практична частина може перетворитися на набір випадкових дій. У межах дипломного проєкту порушник не розглядається як абстрактний «хакер» із необмеженими можливостями. Його роль виконує Attack VM, яка працює тільки в ізольованому лабораторному сегменті і використовується для контрольованого створення подій, що потім аналізуються у журналах honeypot.

Базова модель передбачає, що порушник має мережний доступ до тестового сегмента, але не має легітимних облікових даних до реальних вузлів. Він може виконувати розвідку доступних адрес, перевіряти порти, намагатися встановити типи сервісів, виконувати спроби автентифікації та взаємодіяти з сервісом, який сприймає як справжній. Усі ці дії повинні відбуватися в межах дозволеного стенду і не спрямовуватися на сторонні адреси або реальні системи.

У межах дипломної роботи доцільно виділити три рівні активності порушника. Перший рівень - автоматизована розвідка, коли Attack VM виявляє активні вузли та відкриті сервіси. Другий рівень - спроби підбору доступу, коли перевіряються типові логіни та паролі для SSH або іншого сервісу-приманки. Третій рівень - первинна взаємодія після входу, коли порушник вводить команди, намагається зрозуміти, де опинився, переглядає доступні дані, перевіряє користувача, систему та мережні параметри.

З погляду класифікації ці дії можна пов’язати з відомими техніками MITRE ATT&CK. Підбір облікових даних відповідає техніці Brute Force T1110. Використання віддаленого доступу через SSH відповідає підходу Remote Services: SSH T1021.004. Якщо порушник використовує підібрані або отримані облікові дані як спосіб доступу, така поведінка співвідноситься з Valid Accounts T1078. У роботі ці зв’язки потрібні не для формального цитування, а для того, щоб класифікація подій не була довільною.

Сценарії атак у цій роботі мають бути контрольованими і безпечними. Їх не потрібно описувати як інструкцію для несанкціонованого втручання. Навпаки,

вони повинні подаватися як методика перевірки власного лабораторного стенду. Наприклад, дослідник може сформувати події сканування, спроб автентифікації та обмеженої взаємодії з honeypot, після чого порівняти, як ці дії відображаються у журналах. Такий підхід відповідає загальній логіці технічного тестування, описаній у NIST SP 800-115, де наголошено на плануванні, проведенні перевірок, аналізі результатів і розробленні рекомендацій.

Технічні вимоги до Deception-підсистеми доцільно поділити на функціональні, нефункціональні та вимоги безпеки. Функціональні вимоги описують, що система має робити: приймати підключення на сервіс-приманку, записувати спроби входу, зберігати сесії, фіксувати команди, надавати журнали для аналізу. Нефункціональні вимоги стосуються стабільності, простоти розгортання, повторюваності експерименту та зручності перегляду логів. Вимоги безпеки визначають ізоляцію, заборону взаємодії з реальними ресурсами та відсутність критичних даних у приманці.

Однією з ключових вимог є журналювання. CIS Critical Security Control 8 описує управління аудит-журналами як збирання, сповіщення, перегляд і зберігання подій, які допомагають виявити, зрозуміти або відновитися після атаки [5]. У контексті цієї роботи журналювання повинно бути достатньо детальним, щоб за ним можна було відтворити ланцюжок дій порушника. Мінімально потрібні часові мітки, IP-адреса джерела, тип події, введені облікові дані, результат автентифікації, команди у сесії та службові повідомлення honeypot.

Другою важливою вимогою є правдоподібність. Якщо honeypot виглядає занадто штучно, порушник не буде взаємодіяти з ним достатньо довго, а журнали виявляться бідними. Правдоподібність досягається через використання типових портів, зрозумілих назв вузлів, схожої структури файлового середовища, реалістичних відповідей оболонки та логічного розміщення у мережі.

Третя вимога - відтворюваність. Дипломна робота має бути побудована так, щоб схему можна було повторити: створити такі самі VM, призначити адреси, перевірити доступність, запустити honeypot, сформувати контрольовані події та отримати порівнювані журнали.

Четверта вимога - аналітичність результатів. Події потрібно не лише показати, а й класифікувати. Для цього у третьому розділі має бути сформована таблиця, де кожен фрагмент журналу співвідноситься з типом дії: розвідка, спроба підбору, успішна взаємодія з приманкою, збір інформації про систему, перевірка мережі або спроба виконання дій після входу.

П’ята вимога - контроль меж експерименту. Будь-які дії Attack VM повинні залишатися всередині лабораторної мережі. У роботі не потрібно використовувати зовнішні ресурси, реальні домени, чужі IP-адреси або виробничі системи. Для демонстрації достатньо локального сегмента, де всі вузли належать досліднику. Це не тільки безпечніше, а й методично правильніше: у такому випадку результати можна пояснити та відторити без правових або етичних ризиків.

У підсумку модель порушника і технічні вимоги створюють міст між теоретичною і проєктною частинами. Перший розділ пояснює, чому Deception-підхід доречний і який інструмент варто обрати. Другий розділ на основі цих вимог має показати, як саме буде побудована схема мережі. Третій розділ перевірить, чи справді запропонована підсистема збирає потрібні події і дає достатній матеріал для аналізу.

Таблиця 1.4

Сценарії контрольованої активності порушника та очікувані події

| **Етап** | **Сутність контрольованої дії** | **Очікувані сліди у журналах** | **Відповідність MITRE ATT&CK** |
|---|---|---|---|
| Розвідка | Виявлення активних вузлів і відкритих сервісів у лабораторному сегменті. | Факти звернення до портів, часові мітки, IP-адреса Attack VM. | Попередня активність перед використанням Remote Services. |
| Пошук віддаленого сервісу | Визначення, що на honeypot доступний сервіс SSH або інша приманка. | Запити до сервісу, банер або спроба ініціалізації сесії. | T1021.004 Remote Services: SSH. |
| Підбір облікових даних | Повторювані спроби входу з різними логінами або паролями. | Логіни, паролі, результат автентифікації, кількість спроб. | T1110 Brute Force. |
| Використання отриманого доступу | Взаємодія з імітованою оболонкою після контрольованого входу. | Початок сесії, введені команди, тривалість взаємодії. | T1078 Valid Accounts у частині використання облікових даних. |

| | | | |
|---|---|---|---|
| Первинне дослідження системи | Спроби зрозуміти середовище, користувача, файли та мережні параметри. | Команди у shell-сесії, помилки, повторюваність дій. | Дії після отримання доступу, що підлягають класифікації у роботі. |

Таблиця 1.5

Технічні вимоги до Deception-підсистеми

| **Група вимог** | **Вимога** | **Пояснення для проєкту** |
|---|---|---|
| Функціональні | Приймати підключення до сервісу-приманки | Honeypot має бути доступним у лабораторному сегменті на портах, визначених у схемі мережі. |
| Функціональні | Фіксувати спроби автентифікації | Потрібно зберігати логіни, паролі, час, джерело та результат взаємодії. |
| Функціональні | Записувати shell-взаємодію | Для Cowrie важливим є журнал команд, введених після входу до імітованого середовища. |
| Нефункціональні | Бути простою для повторного розгортання | Конфігурацію треба описати так, щоб стенд можна було відтворити у подальших розділах. |
| Нефункціональні | Надавати зручні журнали для аналізу | Події повинні бути придатними для ручної класифікації та вставлення фрагментів у роботу. |
| Безпека | Працювати в ізольованому сегменті | Attack VM і honeypot не повинні мати доступу до чужих або виробничих систем. |
| Безпека | Не містити реальних секретів | У приманці не можна розміщувати справжні ключі, паролі, персональні дані або конфіденційні файли. |
| Аналітика | Підтримувати класифікацію дій порушника | Журнали мають дозволяти виділити розвідку, підбір, вхід, команди і спроби збору інформації. |

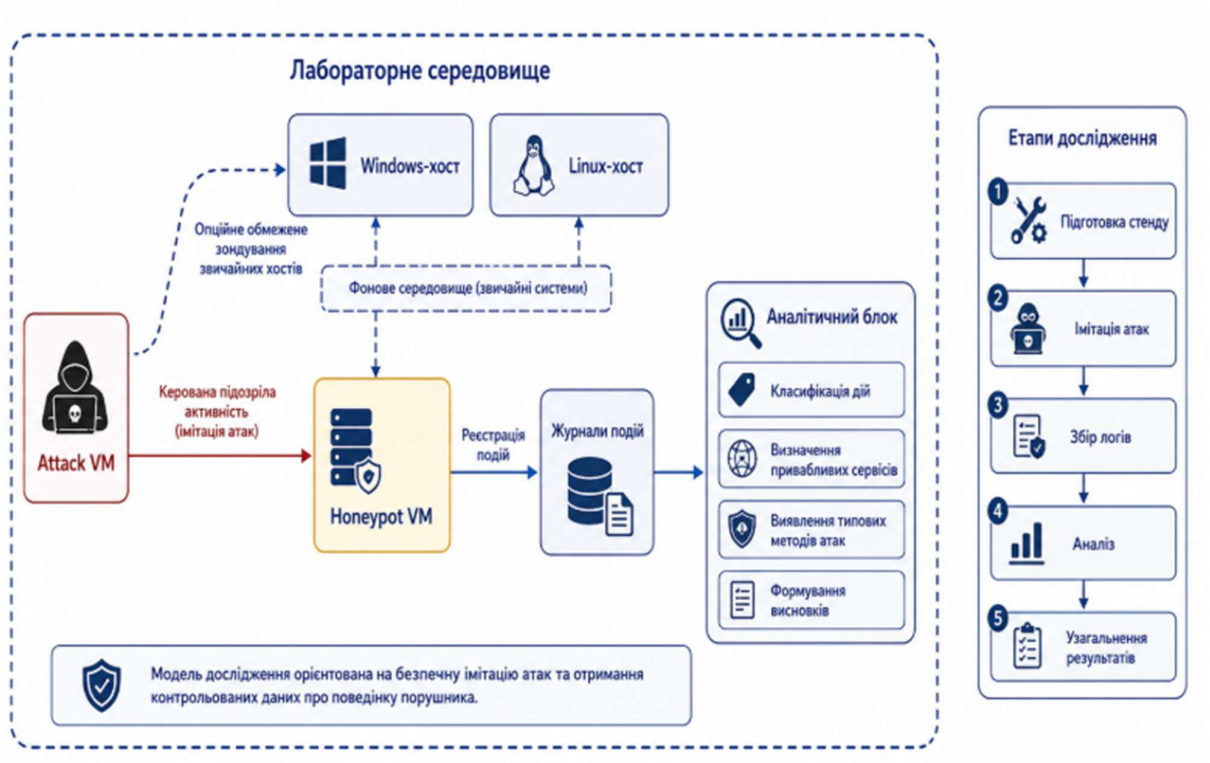



Рисунок 1.3 - Модель дослідження дій порушника у лабораторному середовищі

На рис. 1.3 показано логіку, яку доцільно використати у подальшій практичній частині. Спочатку формується контрольована активність, потім honeypot записує події, після чого дослідник класифікує їх і робить висновки. Така модель допомагає не відходити від теми дипломної роботи: головною цінністю є не сам факт встановлення honeypot, а аналіз дій, які він дозволяє зафіксувати.

## Висновки до розділу 1

У першому розділі було розглянуто теоретичні основи застосування Deception-підсистем у кібербезпеці організації. Встановлено, що проблема виявлення атак у мережній інфраструктурі пов'язана не тільки з наявністю відкритих сервісів, а й з обмеженою видимістю дій порушника, складністю інтерпретації великої кількості подій та потребою у безпечному дослідженні поведінки після взаємодії з віддаленим сервісом.

Показано, що Deception-підхід дозволяє створити контрольовані об'єкти інтересу, які не виконують виробничих функцій, але фіксують небажану активність. Honeypot у такій системі розглядається як спеціальний вузол-приманка, а не як звичайний сервер. Його цінність полягає у можливості зберігати спроби підключення, параметри автентифікації, сесії, команди та інші події, придатні для подальшого аналізу.

Порівняльний аналіз інструментів Cowrie, Dionaea, OpenCanary і спрощеного фейкового SSH показав, що для цілей даної роботи найбільш доречним є Cowrie. Це рішення безпосередньо відповідає сценаріям SSH/Telnet[16, 17] honeypot, дозволяє журналювати brute-force активність і shell-взаємодію, а також формує достатньо наочний матеріал для демонстрації на захисті.

Сформовано модель порушника, який діє лише в межах ізольованого лабораторного сегмента через Attack VM. Основними сценаріями визначено розвідку, пошук відкритих сервісів, спроби автентифікації, взаємодію з honeypot після входу та первинне дослідження середовища. Для цих сценаріїв визначено очікувані журнальні події та зв'язок із техніками MITRE ATT&CK.

Також сформульовано технічні вимоги до Deception-підсистеми. Вона повинна приймати підключення до сервісу-приманки, фіксувати спроби автентифікації, записувати взаємодію у сесії, бути відтворюваною, працювати в ізольованому середовищі, не містити реальних секретів і надавати журнали, придатні для класифікації дій порушника. Отримані результати є основою для другого розділу, де буде спроєктовано структуру віртуалізованого лабораторного стенду.

# РОЗДІЛ 2 ПРОЄКТУВАННЯ DECEPTION-ПІДСИСТЕМИ НА ОСНОВІ ВІРТУАЛІЗОВАНОГО ЛАБОРАТОРНОГО СЕРЕДОВИЩА

## 2.1 Формування технічного завдання та вимог до експериментального стенду

Другий розділ має проєктний характер, тому в ньому Deception-підсистема розглядається не як абстрактна технологія, а як конкретний лабораторний стенд, який можна відтворити, показати на захисті та використати для збору контрольованих журналів атак. На відміну від теоретичного аналізу, що був виконаний у першому розділі, тут увага переноситься на склад вузлів, схему мережної взаємодії, межі експерименту, адресацію, правила доступу, логіку роботи honeypot і очікувані результати. Такий підхід відповідає змісту теми, оскільки головна цінність роботи полягає не лише у поясненні поняття Deception, а й у створенні умов, де можна дослідити поведінку порушника на основі журналів підключень, спроби автентифікації та подальшу взаємодію з приманкою.

Експериментальний стенд проєктується як ізольована модель мережної інфраструктури умовної організації. У межах такої моделі не ставиться завдання відтворити велику корпоративну мережу з десятками серверів, доменною інфраструктурою та складними засобами моніторингу. Навпаки, для дипломного проєкту доцільною є компактна структура, у якій кожний вузол має чітко визначену роль і безпосередньо допомагає перевірити роботу підсистеми. До складу стенду включаються Attack VM, Windows-хост, Linux-хост, Honeypot VM і віртуальний мережний сегмент, який відділяє дослідницьке середовище від реальних ресурсів. Завдяки цьому можна показати повний цикл: від появи підозрілої активності до отримання даних, придатних для аналізу.

Технічне завдання для стенду формулюється з урахуванням кількох обмежень. По-перше, робота не повинна передбачати взаємодію зі сторонніми системами або реальними інформаційними ресурсами. По-друге, усі атаки мають бути імітаційними та контрольованими, тобто виконуватися лише в межах локального лабораторного сегмента. По-третє, вузол-приманка не повинен містити реальних

облікових записів, службових документів, ключів доступу або інших даних, які можуть мати цінність за межами навчального експерименту. По-четверте, результати мають бути оформлені так, щоб їх можна було використати у пояснювальній записці: через таблиці журналів, схему адресації, опис сервісів, скриншоти й узагальнені висновки.

Основною метою технічного завдання є створення умов для спостереження за типовою активністю порушника, яка пов'язана з відкритими віддаленими сервісами. У роботі доцільно обмежитися сценаріями, що добре візуалізуються і не потребують розроблення власного програмного коду: виявлення активних вузлів, пошук відкритих портів, спроби автентифікації до SSH або Telnet, фіксація введених імен користувачів і паролів, а також запис послідовності дій після входу до імітованого середовища. Такий набір подій прямо пов'язаний з техніками brute-force, використанням віддалених служб і зловживанням обліковими записами у термінології MITRE ATT&CK.

Важливо, що підсистема не проєктується як засіб активної атаки або інструмент прихованого впливу на зовнішні системи. Її функція полягає у контрольованому прийманні небажаної взаємодії на спеціально підготовленому вузлі, де така взаємодія не шкодить робочим сервісам. Саме тому ключовим принципом є розділення навчальних ролей: Attack VM використовується тільки як джерело тестових дій, Windows- і Linux-хости створюють фон звичайної мережі, Honeypot VM приймає інтерес порушника, а журнали забезпечують матеріал для подальшої класифікації. У такій структурі навіть простий за обсягом стенд дозволяє показати логіку кіберзахисту значно краще, ніж суто описова модель.

Під час формування вимог враховується також NIST Cybersecurity Framework 2.0, у якому функція Detect пов'язана з виявленням подій кібербезпеки, а Respond - з подальшою обробкою та реагуванням на них. У межах дипломного проєкту повноцінна побудова SOC або SIEM не виконується, однак сама логіка роботи стенду підтримує ці функції: подія фіксується, зберігається, перевіряється, а потім класифікується за типом дії порушника. Через це honeypot у роботі розглядається як

джерело даних для виявлення, а не як єдиний самодостатній механізм захисту. Така позиція є більш коректною з погляду практичної кібербезпеки.

Окрема група вимог стосується відтворюваності експерименту. Куратор або члени комісії мають розуміти, з яких компонентів складається стенд, які адреси призначені вузлам, які служби відкриті, які порти використовуються для штатних хостів і які порти віддано під honeypot. У роботі бажано уникати ситуації, коли результат залежить від випадкового налаштування конкретної машини. Тому всі параметри потрібно занести до таблиць, а логіку доступу описати у вигляді схеми. Якщо під час захисту немає можливості демонструвати робочі віртуальні машини, така документація все одно дозволить переконливо пояснити, як було організовано дослідження.

З погляду безпеки експерименту найважливішою вимогою є ізоляція. Для дипломної роботи достатньо застосувати внутрішній або host-only сегмент гіпервізора, у якому віртуальні машини бачать одна одну, але не мають прямого доступу до продуктивної мережі організації чи до сторонніх адрес.

Якщо для завантаження оновлень або пакетів тимчасово потрібен доступ до Інтернету, він має використовуватися до початку тестування, після чого експериментальний сегмент повертається в ізольований режим. Це дозволяє зберегти навчальний характер роботи і не створювати ризик неконтрольованого трафіку.

Для забезпечення логічної повноти стенду в технічному завданні визначаються не лише вузли, а й критерії успішності. Проєкт можна вважати реалізованим на рівні другого розділу, якщо сформовано мережну архітектуру, визначено таблицю адресації, обрано програмні компоненти, описано призначення сервісів, спроєктовано схему журналювання та наведено правила безпечного проведення експерименту. На наступному етапі, у третьому розділі, ці рішення мають бути перетворені у конкретні налаштування, скриншоти, тестові події та аналіз логів.

Таблиця 2.1

Вимоги до експериментального стенду Deception-підсистеми

| Група вимог | Зміст вимоги | Очікуваний результат |
|---|---|---|
| Функціональні | Стенд має містити Attack VM, Windows-хост, Linux-хост, Honeypot VM і внутрішній мережний сегмент. | Можливість відтворити взаємодію порушника з реалістичною, але контрольованою мережею. |
| Мережні | Вузли повинні мати статичні адреси, зрозумілу схему доступу та відокремлення від зовнішніх ресурсів. | Зручність демонстрації, відсутність випадкових змін адресації та контроль трафіку. |
| Безпекові | Attack VM не повинна використовуватися для взаємодії зі сторонніми системами; honeypot не містить реальних секретів. | Зниження ризиків під час тестування та збереження навчального характеру експерименту. |
| Журнальні | Підсистема повинна зберігати події підключення, спроби автентифікації, параметри сесій і дії після входу. | Формування матеріалу для класифікації дій порушника та висновків з кібербезпеки. |
| Документальні | Усі проєктні рішення мають бути подані у вигляді таблиць, схем і текстового обґрунтування. | Можливість включити результати до пояснювальної записки й презентації до захисту. |

Узагальнення вимог у табл. 2.1 показує, що проєктування Deception-підсистеми не зводиться до вибору одного honeypot-рішення. Потрібно одночасно вирішити питання топології, ролей вузлів, відмежування тестового трафіку, сценаріїв фіксації подій і форми подання результатів. Саме тому в роботі обрано не окрему програму, а цілісну модель лабораторної інфраструктури, де програмний засіб Cowrie є центральним, але не єдиним елементом.

### 2.2 Архітектура мережі, схема адресації та ролі віртуальних машин

Архітектура лабораторної мережі повинна одночасно бути достатньо простою для відтворення та достатньо змістовною для демонстрації принципу Deception. Якщо в мережі буде лише Attack VM і honeypot, то експеримент перетвориться на пряму перевірку одного сервісу, а не на моделювання мережної інфраструктури організації. Тому у стенді передбачаються додаткові штатні вузли - Windows-хост і Linux-хост. Вони створюють фон, який нагадує невеликий сегмент організації, де одночасно існують робочі станції, серверні служби та спеціальний вузол-приманка. Це дозволяє показати, як порушник може обирати ціль не випадково, а за ознаками відкритих і звичних портів.

У запропонованій моделі використовується один ізольований сегмент з приватною адресацією. Для прикладу доцільно взяти мережу 172.22.11.0/24, яка часто використовується у host-only середовищах гіпервізорів. Такий вибір не є принциповим, але він зручний для пояснення, тому що адреси легко читати, а діапазон не перетинається з більшістю домашніх або навчальних мереж. Вузлам призначаються статичні адреси, щоб у логах було зрозуміло, від кого й до кого йшов трафік. Для Attack VM виділяється адреса 172.22.11.10, для Windows-хоста - 172.22.11.11, для Linux-хоста - 172.22.11.12, для Honeypot VM - 172.22.11.13, а для віртуального шлюзу або керуючого інтерфейсу - 172.22.11.1.

Роль Attack VM полягає у створенні контрольованої активності. Вона не розглядається як реальний інструмент нападу, а використовується як лабораторне джерело подій, яке дозволяє перевірити, чи бачить honeypot спроби підключення, чи зберігає параметри автентифікації та чи формує зрозумілі журнали. У тексті дипломної роботи не потрібно наводити небезпечні інструкції або повні сценарії атаки.

Windows-хост у стенді виконує роль типової робочої станції або сервера адміністративного доступу. Його присутність потрібна не для того, щоб проводити атаки на Windows, а для створення реалістичного оточення. У багатьох організаціях віддалений доступ до Windows-систем асоціюється з RDP, тому у проєкті доцільно показати, що така служба може існувати, але її порт змінено або обмежено в межах лабораторного сегмента. Наприклад, замість типового порту 3389 можна використати 3390 як позначення того, що штатний вузол відокремлено від приманки. Це допомагає зробити honeypot більш помітним для сценарію дослідження, але не створює ілюзії, що зміна порту сама по собі є достатнім захистом.

Linux-хост виконує роль умовного сервера організації, на якому можуть бути присутні служби SSH або HTTP, але вони також розміщуються на змінених портах. Наприклад, SSH для цього вузла може бути доступний на 2222, а навчальний HTTP-сервіс - на 8080. Це рішення має методичне значення: воно дозволяє порівняти поведінку штатного вузла та honeypot. Якщо порушник або тестовий сценарій шукає типові порти, то першим об’єктом інтересу може стати саме Honeypot VM, яка

слухає реальні порти 22 і 23. Якщо ж тестування передбачає ширшу розвідку, то в журналах можна побачити різницю між реакцією на змінені та типові порти.

Honeypot VM є центральним вузлом підсистеми. У межах проєкту її доцільно розгорнути на Linux, оскільки Cowrie орієнтований на імітацію SSH/Telnet середовища і добре підходить для фіксації brute-force та shell-взаємодії. На відміну від штатних вузлів, honeypot не повинен виконувати виробничих функцій. Його завдання - приймати підключення, створювати правдоподібну відповідь служби, записувати дані про сесію та забезпечувати дослідника матеріалом для аналізу. Такий вузол не потрібно маскувати як повністю продуктивний сервер із реальними даними.

Віртуальний шлюз у даній схемі не обов'язково має бути окремою машиною. У простому варіанті його роль може виконувати вбудований механізм гіпервізора, який створює host-only мережу і забезпечує базову зв'язність між віртуальними машинами. Якщо робота виконується у більш розгорнутому форматі, можна використати окремий маршрутизуючий вузол, який дозволяє чіткіше описати правила ізоляції. Проте для дипломного проєкту важливо не ускладнювати стенд зайвими компонентами.

На рис. 2.1 наведено узагальнену схему віртуалізованого середовища. Вона показує, що Attack VM перебуває в одному сегменті з іншими вузлами, але вся взаємодія обмежена лабораторним контуром. Windows- і Linux-хости формують фон мережної інфраструктури, Honeypot VM приймає підозрілу активність, а журнали й аналітичний контур забезпечують матеріал для третього розділу. Така схема є достатньо наочною для захисту, бо комісія одразу бачить, де знаходиться порушник, де розміщено приманку і як результати переходять у висновки.

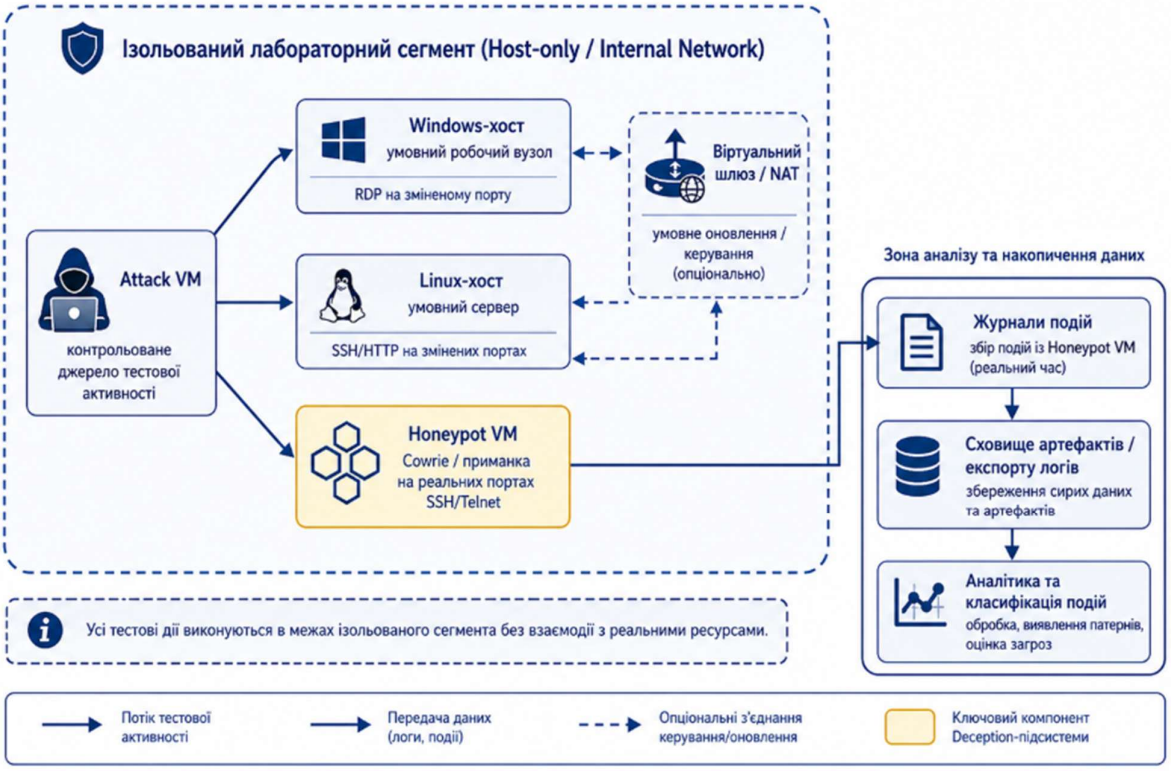



Рисунок 2.1 - Проєктна схема віртуалізованого лабораторного середовища Deception-підсистеми

Таблиця 2.2

Схема адресації та ролі віртуальних машин

| Вузол | IP-адреса | Основна роль | Особливості доступу |
|---|---|---|---|
| Attack VM | 172.22.11.10 | Джерело контрольованої тестової активності. | Працює тільки в ізольованому сегменті; не використовується для зовнішніх цілей. |
| Windows-хост | 172.22.11.11 | Умовний робочий вузол або сервер адміністративного доступу. | RDP переноситься на змінений порт або обмежується правилами доступу. |
| Linux-хост | 172.22.11.12 | Умовний сервер організації з базовими мережними службами. | SSH/HTTP можуть бути доступні на змінених портах для відокремлення від honeypot. |
| Honeypot VM | 172.22.11.13 | Вузол-приманка для фіксації спроб підключення та взаємодії. | Приймає підключення на типових портах SSH/Telnet і зберігає журнали. |
| Віртуальний шлюз | 172.22.11.1 | Керуюча точка лабораторної мережі або інтерфейс гіпервізора. | Використовується для зв'язності всередині стенду; вихід назовні обмежується. |

Таблиця 2.2 фіксує базову адресацію, яка використовується у подальшому описі. Такий формат зручний тим, що під час аналізу логів можна одразу відрізнити

джерело події від вузла-приймача. Якщо у журналі honeypot буде зазначено джерело 172.22.11.10 і призначення 172.22.11.113, то така подія однозначно належить до контрольованої активності з Attack VM. Це спрощує обробку результатів і дозволяє не змішувати навчальні події з випадковим фоновим трафіком.

### 2.3 Обґрунтування вибору операційних систем, сервісів і програмних засобів

Вибір програмних засобів для стенду визначається не лише їх популярністю, а й відповідністю темі дипломної роботи. Оскільки проєкт присвячений Deception-підсистемі, головним критерієм є можливість показати обманну взаємодію, журналювання і подальше дослідження дій порушника. Для цього не потрібні складні комерційні платформи або розроблення власного коду. Більш доречним є використання відкритих інструментів, які мають зрозумілу документацію, підтримують поширені сценарії і дозволяють швидко отримати демонстраційні результати. Такий підхід робить роботу реалістичною для бакалаврського рівня та дає змогу зосередитися на архітектурі, методиці й аналізі.

Базовим середовищем для розгортання стенду є гіпервізор. У роботі можна використовувати VirtualBox[14], VMware Workstation або інше подібне рішення, яке підтримує створення кількох віртуальних машин і внутрішніх мережних сегментів. У пояснювальній записці доцільно не прив’язуватися жорстко до конкретного продукту, якщо це не вимагає кафедра. Головне - описати, що гіпервізор забезпечує ізоляцію, керування мережними адаптерами, створення знімків стану, відновлення після помилок і відокремлення лабораторного трафіку від основної системи. Саме ці функції важливі для теми, а не конкретна назва інструмента.

Windows-хост обирається тому, що в типовій організації значна частина робочих станцій або адміністративних серверів працює під керуванням Windows. Для цілей дипломного проєкту не має принципового значення, чи використовується Windows 10, Windows 11 або тестова серверна редакція. Важливішою є роль такого вузла у мережі: він демонструє наявність RDP або іншого сервісу, який може привертати увагу під час розвідки. При цьому Windows-хост не повинен бути головною

ціллю експерименту. Його завдання полягає у створенні контексту і порівняння з honeypot, що має спеціально підготовлені приманки.

Linux-хост потрібен для імітації серверної частини мережі. У невеликих організаціях Linux часто застосовується для веб-сервісів, файлових служб, інструментів адміністрування або внутрішніх технічних систем. У межах стенду достатньо розгорнути легку редакцію Linux, наприклад Ubuntu Server або Debian. Вона не потребує значних ресурсів, добре працює у віртуальному середовищі та дозволяє показати базові служби без перевантаження дипломної роботи другорядними налаштуваннями. Для розділу 2 важливо визначити, які служби повинні бути присутні логічно, а конкретні кроки налаштування доцільніше перенести до третього розділу.

Honeypot-рішенням у проєкті обирається Cowrie[10, 11]. Згідно з документацією, Cowrie є SSH/Telnet[16, 17] honeypot середнього або високого рівня взаємодії, який призначений для журналювання brute-force атак і shell-взаємодії порушника [7]. Саме це відповідає поставленій темі, оскільки у роботі потрібно показати не лише факт підключення, а й подальшу поведінку після входу до імітованого середовища. Cowrie також зручний для демонстрації, бо його журнали можуть містити час події, IP-адресу джерела, логін, пароль, тип події, ідентифікатор сесії та інші дані, які потім можна звести в таблиці.

Dionaea і OpenCanary можуть розглядатися як допоміжні або альтернативні рішення. Dionaea орієнтований на фіксацію шкідливих підключень і може бути корисним для інших протоколів, тоді як OpenCanary дозволяє створювати універсальні приманки для різних служб. Проте для цієї роботи їх варто використовувати обережно, щоб не розмити фокус. Якщо в дипломі головною фішкою є імітація та дослідження атак на SSH, то Cowrie є більш прямим вибором. Додаткові рішення можна згадати як перспективу розширення стенду або як резервний варіант, але не робити їх центром практичної частини.

Окреме обґрунтування потребують служби, які виносяться назовні. У проєктній логіці штатні вузли не повинні повністю конкурувати з honeypot за увагу порушника. Тому для Windows-хоста пропонується змінений RDP-порт, для Linux-

хоста - змінений SSH-порт і навчальний HTTP-порт, а для Honeypot VM - типові порти SSH і Telnet. Таке рішення створює контрольований сценарій, у якому вузол-приманка виглядає найбільш природною ціллю для перевірки віддаленого доступу. Водночас штатні вузли залишаються в схемі та підтверджують, що досліджується саме мережна інфраструктура, а не один окремий сервер.

Для журналювання важливо передбачити не тільки збереження файлів на Honeypot VM, а й подальшу зручність обробки. У простому варіанті достатньо локальних журналів Cowrie, які потім експортуються в текстовому або табличному вигляді. У розширеному варіанті можна передбачити окремий каталог для збереження артефактів, скриншотів і таблиць класифікації. CIS Control 8 підкреслює значення збору, перегляду та збереження audit logs для розуміння атак. У межах дипломної роботи ця вимога реалізується через проектування повного ланцюга: подія - журнал - нормалізація - класифікація - висновок.

Важливим принципом вибору інструментів є мінімальна достатність. Надмірна кількість програм може створити враження технічної складності, але не завжди підвищує якість роботи. Для захисту набагато переконливіше показати добре пояснену схему з одним основним honeypot, зрозумілими портами, таблицею адресації, фрагментами журналів і висновками щодо дій порушника. Тому у другому розділі Cowrie визначається як основний інструмент, а решта компонентів підпорядковуються завданню дослідження.

Таблиця 2.3

Обґрунтування вибору основних програмних компонентів стенду

| Компонент | Призначення у стенді | Обґрунтування вибору |
|---|---|---|
| Гіпервізор | Створення ізольованих віртуальних машин і внутрішньої мережі. | Забезпечує відтворюваність, знімки стану та безпечне відокремлення тестового середовища. |
| Windows-хост | Імітація робочої станції або адміністративного сервера організації. | Додає реалістичний вузол із характерною службою RDP, але не є основною ціллю досліду. |
| Linux-хост | Імітація серверного вузла із SSH/HTTP-службами. | Показує серверний сегмент організації та дозволяє порівняти змінені порти зі службами honeypot. |
| Cowrie | Основний SSH/Telnet honeypot для фіксації brute-force і shell-взаємодії. | Безпосередньо відповідає темі дослідження, формує журнали автентифікації та сесій. |
| Dionaea / OpenCanary | Додаткові або альтернативні приманки для інших сценаріїв. | Можуть бути використані для розширення, але не є обов'язковими у базовій моделі. |
| Сховище логів | Збереження журналів, скриншотів і таблиць результатів. | Дозволяє оформити практичні результати в третьому розділі та презентації. |

Зміст табл. 2.3 показує, що кожний компонент має окреме проєктне призначення. Такий підхід запобігає випадковому включенню програмного забезпечення лише заради збільшення переліку інструментів. У дипломній роботі потрібно пояснити не просто те, що певний компонент був використаний, а те, яку функцію він виконує у загальній системі й як пов'язаний із метою дослідження.

## 2.4 Проєктування логіки маскування сервісів і перенаправлення уваги порушника

Проєктування Deception-підсистеми передбачає створення такого середовища, у якому порушник або тестовий сценарій бачить переконливі об'єкти інтересу, але фактично взаємодіє з контрольованою приманкою. У цій роботі під маскуванням сервісів розуміється не приховування реальної інфраструктури будь-якою ціною, а методичне розмежування штатних служб і служб honeypot. Windows- і Linux-хости зберігають роль умовних робочих систем, але їхні віддалені сервіси

не розміщуються на типових портах або обмежуються. Натомість Honeypot VM отримує типові порти, які часто перевіряються під час первинної розвідки.

Такий підхід має дві переваги. Перша полягає в тому, що він спрощує отримання навчальних логів: якщо типовий SSH-порт належить honeypot, то тестові спроби підключення з високою ймовірністю будуть зафіксовані саме там. Друга перевага полягає в наочності для захисту. На схемі та у таблиці портів легко показати, чому вузол-приманка привертає увагу, які служби на ньому відкриті й чому штатні хости не є головними об'єктами ризику в експерименті. Водночас важливо підкреслити, що в реальній організації зміна портів не замінює автентифікацію, сегментацію, оновлення та моніторинг. Вона тут використовується лише як елемент лабораторного дизайну.

У базовому варіанті Honeypot VM приймає підключення на порт 22 як SSH-приманка і на порт 23 як Telnet-приманка. Якщо потрібно додати простий веб-сценарій, можна передбачити HTTP-приманку на порті 80, однак центральним залишається саме SSH/Telnet[16, 17], оскільки Cowrie спеціалізується на цих протоколах. Windows-хост може мати RDP на зміненому порті 3390, а Linux-хост - SSH на 2222 і навчальний HTTP на 8080. Така схема не вимагає складної маршрутизації, але дозволяє чітко відділити реальні вузли від вузла-приманки.

Проєктна логіка не повинна передбачати пряме перенаправлення трафіку зі штатних вузлів на honeypot без розуміння наслідків. У лабораторній роботі простіше і безпечніше зробити honeypot самостійним вузлом із власною IP-адресою та реальними портами приманки. Це відповідає вимозі завдання про реальні, не форвардні порти сервісів на honeypot. Якщо використовувати перенаправлення, у студента може виникнути плутанина: чи атакувався штатний сервер, чи вузол-приманка, де саме виникла подія, які правила мережі спрацювали. Самостійна адреса Honeypot VM робить результати прозорими.

Водночас у тексті можна пояснити ідею «перенаправлення уваги», яка відрізняється від технічного port forwarding. У цьому випадку увага порушника переноситься не мережним правилом, а структурою середовища: найочевидніші порти відкриті на honeypot, а штатні служби зміщені або обмежені. Для дипломного

проєкту це більш коректна модель, тому що вона не потребує складної конфігурації і прямо демонструє принцип Deception. Порушник бачить службу, яка здається звичайною, але вся взаємодія з нею зберігається і контролюється.

Для опису привабливості сервісів у проєкті можна використати три рівні. Перший рівень - службова видимість, коли порт відкритий і відповідає очікуваній ролі. Другий рівень - автентифікаційна взаємодія, коли система приймає спроби введення імен користувачів і паролів. Третій рівень - післяавтентифікаційна сесія, коли порушник виконує команди або досліджує імітоване середовище. Cowrie корисний саме тим, що дозволяє перейти за межі простої фіксації факту з'єднання і отримати дані про поведінку після входу, що важливо для дослідження.

Проєктування маскування також пов'язане з легендою вузла-приманки. У роботі можна описати, що Honeypot VM імітує технічний сервер невеликої організації, який з погляду порушника може бути цікавим через наявність віддаленого доступу. При цьому не потрібно створювати реальні файли з конфіденційною інформацією або вводити справжні облікові дані. Достатньо умовної структури, яка дає змогу побачити типові команди й поведінку в журналах. Такий варіант не порушує безпекову межу між дослідженням і створенням ризиків.

На рис. 2.2 показано логіку розміщення сервісів. Схема підкреслює, що штатні вузли мають змінені або другорядні порти, а honeypot використовує типові порти для приймання підозрілої активності. Це рішення прямо пов'язане з метою роботи: зібрати матеріал для аналізу атак і показати, які саме сервіси виявляються найбільш привабливими для порушника. У третьому розділі ця логіка має бути підтверджена фрагментами журналів і класифікацією дій.




Рисунок 2.2 - Логіка розміщення сервісів і зміщення інтересу порушника до honeypot

Таблиця 2.4

Проєктне розміщення сервісів у лабораторному сегменті

| **Вузол** | **Сервіс** | **Порт у стенді** | **Проєктне призначення** |
|---|---|---|---|
| Windows-хост | RDP | 3390 | Показати наявність типового адміністративного сервісу, але не робити його головною ціллю досліду. |
| Linux-хост | SSH | 2222 | Зберегти можливість службового доступу до Linux-вузла, відокремивши його від SSH-приманки. |
| Linux-хост | HTTP | 8080 | Імітувати навчальний веб-сервіс або інформаційну сторінку організації. |
| Honeypot VM | SSH | 22 | Приймати тестові спроби підключення та фіксувати brute-force активність. |
| Honeypot VM | Telnet | 23 | Показати застарілий або небезпечний віддалений сервіс як приманку для спостереження. |
| Honeypot VM | HTTP-приманка | 80 | Додатковий варіант для розширення стенду, якщо потрібно показати веб-взаємодію. |

Запропонована таблиця портів є проєктною і може коригуватися під час фактичного розгортання. Проте її бажано зберегти у пояснювальній записці, оскільки саме вона пояснює, чому в журналах очікуються події на конкретних вузлах. Якщо

в практичному розділі будуть наведені фрагменти логів із портом 22 на Honeypot VM, читач зможе повернутися до табл. 2.4 і зрозуміти, що це не випадковий збіг, а запланована частина архітектури.

### 2.5 Проєктування підсистеми журналювання, ізоляції та безпечного проведення експерименту

Останній підрозділ проєктної частини присвячено тим елементам, які часто недооцінюються під час опису honeypot, але фактично визначають якість дипломного дослідження. Йдеться про журналювання, збереження артефактів, ізоляцію стенду та правила безпечного проведення тестів. Якщо honeypot просто приймає підключення, але результати не систематизуються, то робота залишається технічною демонстрацією. Якщо ж події проходять шлях від фіксації до класифікації, то диплом набуває дослідницького змісту. Саме тому журналювання треба проєктувати на тому ж рівні, що й мережну схему.

У проєкті визначаються кілька типів даних, які повинні зберігатися. До першого типу належать мережні параметри події: дата, час, IP-адреса джерела, IP-адреса призначення, порт і протокол. До другого типу належать автентифікаційні параметри: введене ім’я користувача, пароль або факт невдалої спроби. До третього типу належить сесійна взаємодія: послідовність команд, запити, реакції імітованого середовища. До четвертого типу належать службові дані, потрібні для аналізу: ідентифікатор сесії, тип події, тривалість взаємодії, кількість спроб і умовна категорія ризику. Така структура дозволяє перейти від сирих логів до таблиць у дипломі.

У межах базового стенду журнали можуть зберігатися локально на Honeypot VM. Це спрощує розгортання і не потребує окремої SIEM-платформи. Проте навіть у такому варіанті потрібно передбачити порядок роботи з файлами: де вони розміщуються, як відбираються фрагменти для пояснювальної записки, як фіксується час експерименту, як відокремлюються тестові сесії одна від одної. Якщо тестування виконується кілька разів, доцільно вести окремі папки або таблиці для кожного сценарію. Це допоможе уникнути змішування результатів і підвищить переконливість висновків.

З погляду аналітики журнали мають бути перетворені в зрозумілі категорії. Для дипломної роботи достатньо виділити такі класи дій: розвідка, спроба підключення, підбір облікових даних, успішна або умовно успішна автентифікація, первинне дослідження середовища, спроба отримати інформацію про систему, спроба змінити стан середовища. Така класифікація добре узгоджується з логікою MITRE ATT&CK, але не вимагає надмірної деталізації. У третьому розділі ці класи можна використати як основу для таблиці аналізу подій.

Ізоляція стенду є обов'язковою умовою безпечного проведення експерименту. Attack VM повинна працювати тільки у внутрішньому сегменті, а список адрес для тестування має бути обмежений вузлами, зазначеними у таблиці адресації. Не допускається використання тестових дій щодо реальних сайтів, зовнішніх серверів, університетської мережі або будь-яких сторонніх адрес. Це потрібно прямо зазначити в дипломній роботі, тому що тема може бути неправильно сприйнята як виконання атаки. Насправді дослідження спрямоване на захисну сторону: спостереження, фіксацію і класифікацію поведінки порушника в безпечному середовищі.

Для додаткового контролю доцільно використовувати знімки стану віртуальних машин. Перед тестуванням створюється контрольний стан стенду, після чого проводиться імітація подій. Якщо в процесі експерименту конфігурація була пошкоджена або накопичено зайві дані, віртуальну машину можна повернути до початкового стану. Цей підхід особливо корисний для honeypot, оскільки дозволяє повторити тестові сценарії кілька разів і порівняти результати. У пояснювальній записці достатньо описати сам принцип використання знімків, без переліку конкретних технічних команд.

Окремо проєктується порядок збереження матеріалів для захисту. До нього належать скриншоти схеми мережі, таблиця параметрів віртуальних машин, фрагменти логів, таблиця класифікації дій і короткі пояснення до кожного сценарію. Саме ці матеріали потім можуть бути перенесені до презентації. Якщо розділ 2 добре підготовлений, розділ 3 буде логічним продовженням: спочатку реалізація,

потім демонстрація подій, потім аналіз. У такій структурі робота не виглядає як набір випадкових скриншотів, а сприймається як цілісний проєкт.

На рис. 2.3 наведено схему журналювання і переходу від події до висновків. Вона показує, що подія не є кінцевим результатом. Спочатку Honeypot VM фіксує мережну взаємодію, потім зберігає первинні й сесійні журнали, після чого дослідник нормалізує дані, класифікує дії і формує висновки щодо привабливості сервісів. Такий підхід відповідає рекомендаціям щодо audit log management, оскільки журнали повинні не лише накопичуватися, а й переглядатися, зберігатися та використовуватися для розуміння інцидентів.




Рисунок 2.3 - Проєктована схема журналювання та безпечного аналізу подій

Таблиця 2.5

Проєктні поля журналювання подій Honeypot VM

| Поле журналу | Зміст поля | Значення для дослідження |
|---|---|---|
| Дата і час | Момент виникнення події у лабораторному сегменті. | Дозволяє відновити послідовність дій та відокремити сценарії тестування. |
| IP-адреса джерела | Адреса вузла, з якого ініційовано підключення. | Підтверджує, що подія походить від Attack VM або іншого контрольованого вузла. |
| IP-адреса призначення | Адреса Honeypot VM або іншого вузла, до якого зверталися. | Дозволяє визначити, які об'єкти були привабливими для сценарію. |
| Порт і протокол | Служба, до якої виконувалася взаємодія. | Показує інтерес до SSH, Telnet, HTTP або іншого сервісу. |
| Логін і пароль | Введені тестові облікові дані або факт невдалої спроби. | Дозволяє оцінити характер підбору та повторюваність спроб. |
| Ідентифікатор сесії | Умовний номер або ознака конкретної взаємодії. | Допомагає об'єднати події одного підключення в один сценарій. |
| Дії після входу | Послідовність команд або запитів у сесії. | Є основою для класифікації поведінки порушника. |
| Категорія події | Розвідка, підбір, взаємодія, дослідження середовища тощо. | Перетворює технічні записи на аналітичний матеріал для висновків. |

Наведений перелік полів не означає, що всі вони обов'язково мають бути присутні у кожному рядку сирого журналу. Частина даних може формуватися під час подальшої обробки. Наприклад, категорія події часто визначається дослідником вручну на основі типу запису та контексту сесії. Проте проєктування таких полів наперед дозволяє не втратити важливі дані і підготувати основу для зрозумілої таблиці результатів у третьому розділі.

Таблиця 2.6

Правила безпечного проведення лабораторного експерименту

| Правило | Зміст правила | Пояснення |
|---|---|---|
| Ізоляція мережі | Тестові дії виконуються лише у внутрішньому сегменті. | Це відокремлює навчальний експеримент від реальних ресурсів. |
| Обмеження адрес | Для перевірки використовуються тільки IP-адреси стенду. | Дає змогу уникнути випадкового звернення до сторонніх систем. |
| Відсутність реальних секретів | На honeypot не розміщуються справжні паролі, ключі або документи. | Зменшує ризики у разі помилки конфігурації. |

Продовження таблиці 2.6

| Правило | Зміст правила | Пояснення |
|---|---|---|
| Фіксація сценаріїв | Кожний тест має короткий опис мети й очікуваних логів. | Підвищує відтворюваність і спрощує аналіз. |
| Збереження артефактів | Логи, скриншоти й таблиці зберігаються окремо для кожного сценарію. | Полегшує оформлення практичного розділу та презентації. |
| Відновлення стану | Перед тестами створюється контрольний стан віртуальних машин. | Дозволяє повторити експеримент без накопичення помилок. |

Правила, наведені у табл. 2.6, виконують не формальну, а практичну функцію. Вони захищають роботу від головної методичної помилки - змішування навчальної імітації з реальною діяльністю з проведення атак. Усі результати, наведені в третьому розділі, повинні бути отримані саме в такому контрольованому середовищі. Це дає можливість безпечно дослідити дії порушника, не виходячи за межі освітнього завдання.

## Висновки до розділу 2

У другому розділі було виконано проєктування Deception-підсистеми на основі віртуалізованого лабораторного середовища. Сформовано технічне завдання до стенду, у якому визначено склад вузлів, функціональні, мережні, безпекові, журнальні та документальні вимоги. Показано, що для дипломної роботи достатньо компактної, але логічно завершеної моделі, яка містить Attack VM, Windows-хост, Linux-хост, Honeypot VM і внутрішній мережний сегмент.

Розроблено архітектуру мережі та схему адресації. Запропоновано використовувати ізольований сегмент 172.22.11.0/24 зі статичними адресами для кожного вузла. Такий підхід забезпечує відтворюваність експерименту, полегшує аналіз журналів і дозволяє однозначно визначати джерело та приймача кожної події. Windows- і Linux-хости виконують роль умовних штатних вузлів, а Honeypot VM виступає центральним об’єктом Deception-підсистеми.

Обґрунтовано вибір програмних компонентів. Основним honeypot-рішенням визначено Cowrie, оскільки воно відповідає сценаріям SSH/Telnet, дозволяє фіксувати спроби підбору облікових даних і записувати подальшу shell-взаємодію. Dionaea та OpenCanary розглянуто як можливі додаткові або альтернативні рішення, але для базового варіанта роботи вони не є обов'язковими. Це дозволяє зберегти фокус на дослідженні атак і логів, а не на надмірному розширенні інструментальної бази.

Спроєктовано логіку розміщення сервісів. Штатні Windows- і Linux-вузли пропонується використовувати зі зміненими або обмеженими портами, тоді як Honeypot VM приймає взаємодію на типових портах SSH і Telnet. Така структура не є технічним перенаправленням трафіку, але створює перенаправлення уваги порушника до вузла-приманки. У результаті підозріла активність концентрується там, де її можна безпечно зафіксувати й дослідити.

Також визначено підсистему журналювання, поля для подальшого аналізу і правила безпечного проведення експерименту. Запропоновано зберігати дані про час події, IP-адреси, порти, протоколи, спроби автентифікації, сесії та категорії дій. Окремо сформульовано правила ізоляції, обмеження адрес, відсутності реальних секретів і збереження артефактів. Отримані проєктні рішення створюють основу для третього розділу, у якому має бути описано розгортання стенду, налаштування вузлів, імітацію атак, аналіз журналів і висновки щодо ефективності Deception-підсистеми.

# РОЗДІЛ 3 РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ DECEPTION-ПІДСИСТЕМИ

## 3.1 Порядок розгортання та перевірки віртуального стенду

Третій розділ присвячено практичній частині кваліфікаційної роботи, тобто опису реалізації спроєктованої Deception-підсистеми, перевірці її працездатності та аналізу отриманих результатів. Якщо перший розділ був спрямований на теоретичне обґрунтування технологій deception і honeypot, а другий розділ визначав архітектуру лабораторного середовища, то в цьому розділі основна увага приділяється тому, як стенд має бути підготовлений до демонстрації, які події потрібно отримати в журналах і як ці події інтерпретувати з позиції кібербезпеки. Така структура викладу дозволяє показати не лише факт наявності віртуальних машин, а й дослідницьку логіку роботи: від налаштування до висновків.

Практична реалізація стенду виконується у форматі контрольованого навчального середовища. Його склад відповідає проєктним рішенням другого розділу: Attack VM використовується як джерело тестових дій, Windows-хост і Linux-хост імітують звичайні вузли організації, Honeypot VM приймає взаємодію на типових сервісах, а мережний сегмент host-only[15] або internal network забезпечує ізоляцію. Саме ізоляція є принциповою умовою, тому що всі тестові дії повинні залишатися в межах віртуального стенду і не створювати ризиків для реальних зовнішніх систем. Такий підхід узгоджується із загальними принципами керування кіберризиками, у межах яких виявлення подій і реагування мають виконуватися контрольовано та документовано.

Перед початком налаштування необхідно підготувати робоче середовище гіпервізора. У дипломному проєкті можна використовувати Oracle VM VirtualBox або VMware Workstation, оскільки обидва варіанти підтримують ізольовані мережні режими, потрібні для лабораторних сценаріїв. У режимі host-only віртуальні машини можуть обмінюватися трафіком між собою та з хостовою системою через окремий віртуальний інтерфейс, але не обов’язково мають прямий доступ до зовнішньої мережі. Для цієї роботи такий режим є зручним, адже дає можливість

перевірити підключення між вузлами, не відкриваючи стенд для випадкової взаємодії з Інтернетом.

Початковий етап реалізації передбачає створення чотирьох віртуальних машин. Для Attack VM може бути використаний Linux-дистрибутив для навчальних задач кібербезпеки або звичайний Linux із мінімальним набором мережних утиліт. Для Windows-хоста достатньо клієнтської або серверної редакції Windows, яка імітує робочу станцію або службовий вузол. Linux-хост доцільно розглядати як умовний сервер із відкритими або перенесеними службами. Honeypot VM створюється на базі Linux, оскільки Cowrie найчастіше розгортається саме в такому середовищі й призначений для імітації SSH/Telnet-взаємодії.

Після створення віртуальних машин задаються статичні адреси у межах одного сегмента. Для збереження логіки, запропонованої у другому розділі, використовується мережа 172.22.11.0/24. Attack VM отримує адресу 172.22.11.10, Windows-хост - 172.22.11.11, Linux-хост - 172.22.11.12, Honeypot VM - 172.22.11.13. Таке розмежування зручне для аналізу, тому що за кожним записом журналу можна одразу визначити, який вузол був джерелом події і який вузол приймав з’єднання. У звичайній мережі організації адресація могла б бути складнішою, однак для дипломного стенду така простота є перевагою.

Розгортання стенду має виконуватися послідовно, без переходу до наступного етапу до завершення перевірки попереднього. Спочатку перевіряється, що всі віртуальні машини розміщені в одному ізольованому сегменті. Потім перевіряється базова зв’язність між вузлами. Після цього вмикаються або налаштовуються служби, які потрібні для демонстрації. На звичайних Windows- і Linux-вузлах порти віддаленого доступу зміщуються або обмежуються, а на Honeypot VM, навпаки, відкриваються типові для приманки порти. У результаті підозріла активність спрямовується до вузла, призначеного для журналювання, а не до умовних штатних систем.

На рис. 3.1 наведено узагальнену послідовність розгортання стенду. Схема показує, що реалізація Deception-підсистеми не обмежується встановленням одного honeypot. Вона складається з кількох взаємопов’язаних операцій: створення

ізольованого сегмента, підготовки віртуальних машин, призначення адрес, розділення ролей, налаштування сервісів, розгортання Cowrie, підготовки сценаріїв і збору логів. Саме така послідовність дозволяє отримати не випадкові, а керовані результати.

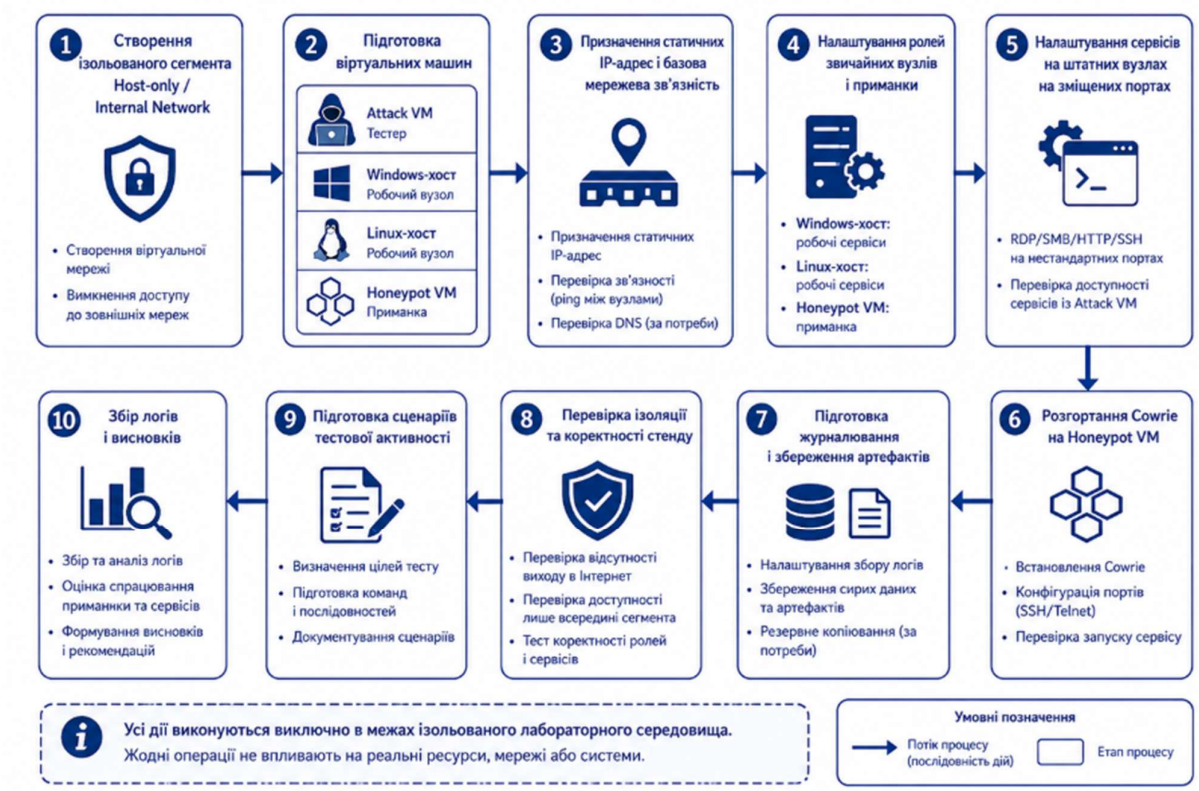



Рисунок 3.1 - Послідовність розгортання та перевірки лабораторного стенду

Після побудови стенду доцільно зафіксувати його параметри у таблиці. Це потрібно не тільки для пояснювальної записки, а й для захисту, тому що комісія повинна бачити, які саме компоненти використовувалися та яку роль мав кожний вузол. У таблиці 3.1 наведено контрольну конфігурацію, яка відповідає логіці проєкту і може бути уточнена після фактичного розгортання на конкретному комп’ютері.

Таблиця 3.1

Контрольна конфігурація віртуального стенду

| Вузол | IP-адреса | Роль у стенді | Основні служби або функції |
|---|---|---|---|
| Attack VM | 172.22.11.10 | Джерело контрольованої імітації дій порушника. | Перевірка доступності вузлів, спроби підключення, тестові команди в межах лабораторної мережі. |
| Windows-хост | 172.22.11.11 | Умовна робоча станція або службовий вузол організації. | RDP на зміненому порту або обмежений доступ; використовується як фон звичайної інфраструктури. |
| Linux-хост | 172.22.11.12 | Умовний Linux-сервер організації. | SSH або HTTP на змінених портах; демонструє відмінність між робочими вузлами та приманкою. |
| Honeypot VM | 172.22.11.13 | Вузол-приманка Deception-підсистеми. | Cowrie SSH/Telnet honeypot, приймання підозрілих сесій, журналювання автентифікації і команд. |

Далі виконується перевірка коректності налаштування. Її не слід зводити лише до того, що віртуальні машини вмикаються без помилок. Важливо підтвердити, що кожний вузол бачить потрібні адреси, що звичайні хости не приймають підключення на типових портах, які відведені під honeypot, і що у журналах Honeypot VM справді з'являються записи після тестових підключень.

Таблиця 3.2

Перевірки коректності розгортання стенду

| Перевірка | Очікуваний результат | Значення для роботи |
|---|---|---|
| Перевірка адресації | Усі вузли мають статичні адреси з підмережі 172.22.11.0/24. | Дає змогу зіставляти журнальні записи з конкретними вузлами стенду. |
| Перевірка ізоляції | Тестові дії не виходять за межі внутрішньої мережі. | Підтверджує безпечний характер експерименту. |
| Перевірка служб штатних вузлів | Windows- і Linux-хости не дублюють типові порти honeypot або мають змінені порти. | Зменшує змішування звичайних вузлів і приманки. |
| Перевірка Honeypot VM | SSH/Telnet-приманка приймає підключення і створює записи журналу. | Підтверджує працездатність головного компонента Deception-підсистеми. |
| Перевірка артефактів | Після тесту доступні журнали, таблиці, скриншоти та опис сценарію. | Забезпечує матеріал для аналізу і презентації. |

За підсумком першої частини практичної реалізації формується контрольна точка. Її зміст полягає у тому, що стенд уже має визначену адресацію, ролі вузлів і базові служби, але тестові сценарії ще не виконувалися. У такому стані доцільно

створити знімки віртуальних машин. Це дозволить повернутися до початкової конфігурації після невдалого тесту або після накопичення зайвих журналів. Для дипломної роботи це важливо, адже повторюваність експерименту підвищує довіру до результатів.

### Практичний хід розгортання стенду

Для усунення розриву між проєктною схемою та практичною реалізацією у роботі зафіксовано послідовність створення стенду в Oracle VirtualBox. Усі віртуальні машини винесено в одну мережу NAT Network з адресним простором 172.22.11.0/24. Така адресація відповідає уточненим правилам оформлення і дозволяє однозначно зіставляти журнальні записи з конкретними вузлами стенду.

У практичній частині прийнято такі відповідності назв: робоча станція Windows позначена як WS-1-SAO, сервер Linux - як Serv-1-SAO, сервер-приманка - як HoneyPot-SAO, а машина для контрольованої тестової активності - як Attack-SAO. Ініціали SAO залишаються у назві кожної машини, щоб під час перегляду скріншотів було видно належність стенду до цієї роботи.

Першим кроком було створено NAT Network з назвою SAO-NAT. Діапазон мережі встановлено як 172.22.11.0/24, а доступ до зовнішніх ресурсів під час тестування обмежено, щоб усі дії залишалися в межах контрольованого лабораторного середовища.

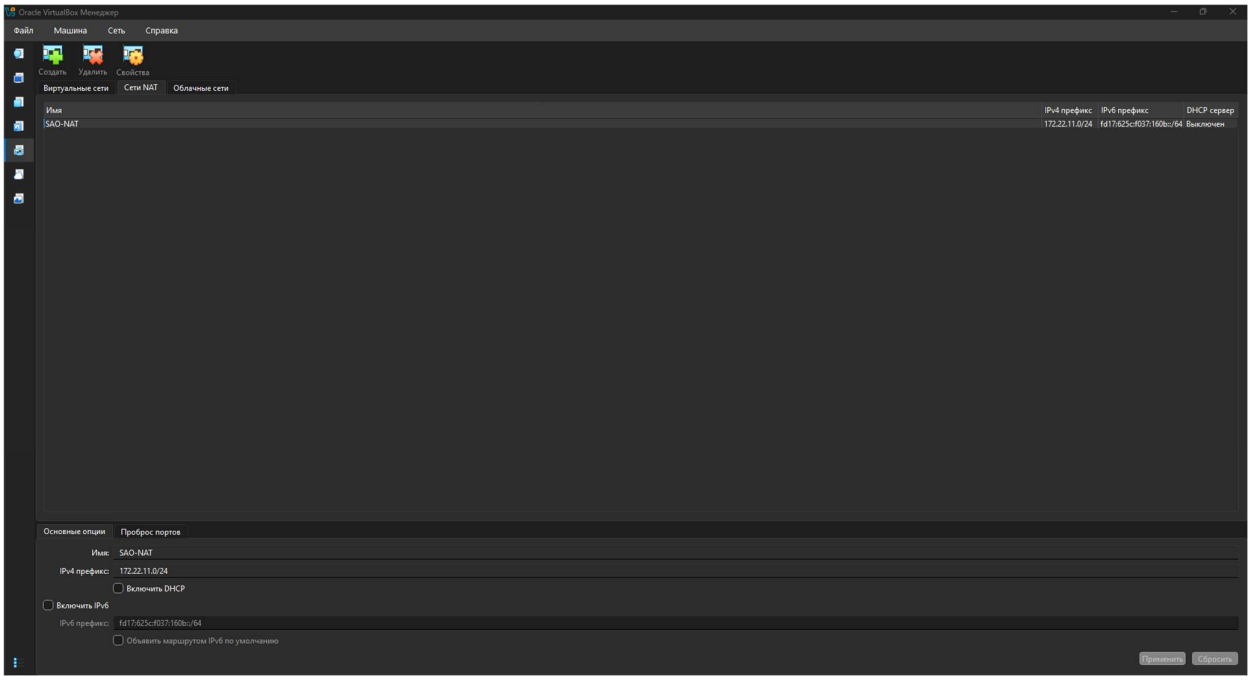



Рисунок 3.2 - Налаштування NAT Network SAO-NAT у Oracle VirtualBox

Після створення мережного сегмента було підготовлено чотири віртуальні машини, які виконують різні ролі у стенді. Такий склад дає змогу не просто описати honeypot, а показати повний маршрут події: від контрольованої дії з Attack-SAO до журналу на HoneyPot-SAO.

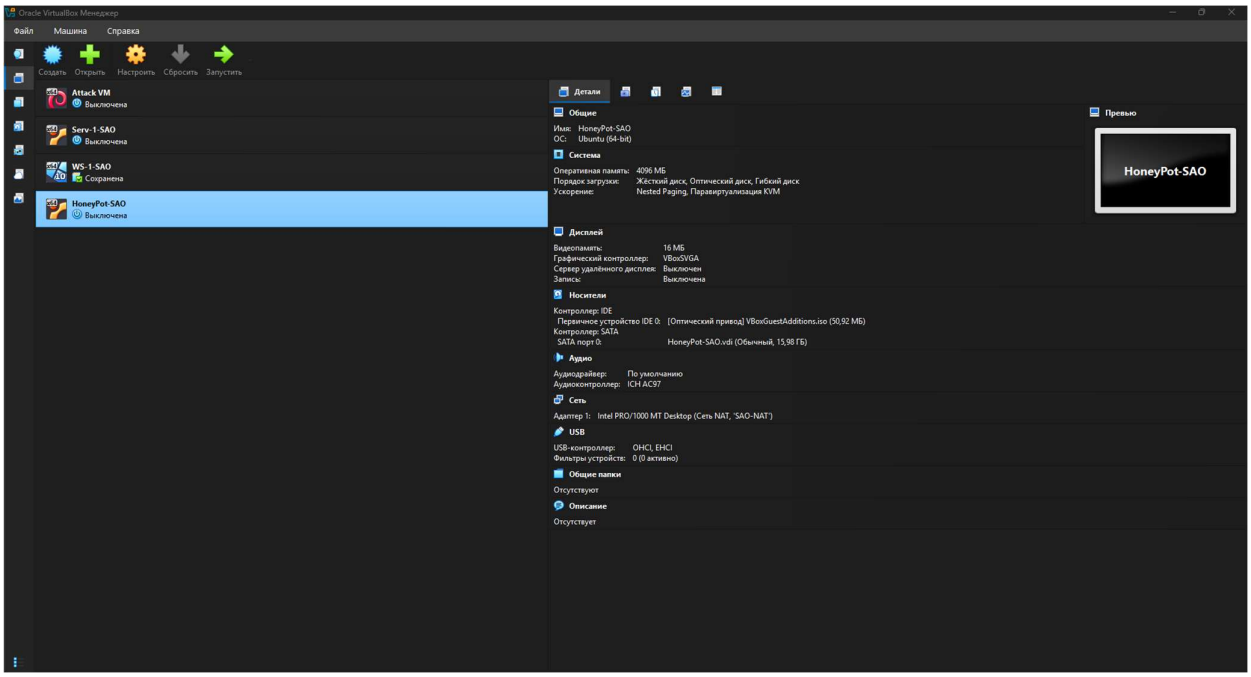



Рисунок 3.3 - Перелік створених віртуальних машин лабораторного стенду

Для кожної машини в налаштуваннях VirtualBox активовано мережний адаптер Adapter 1 і вибрано Attached to: NAT Network, Name: SAO-NAT. Це забезпечує перебування всіх вузлів у єдиній мережі 172.22.11.0/24 без змішування з реальною мережею користувача.

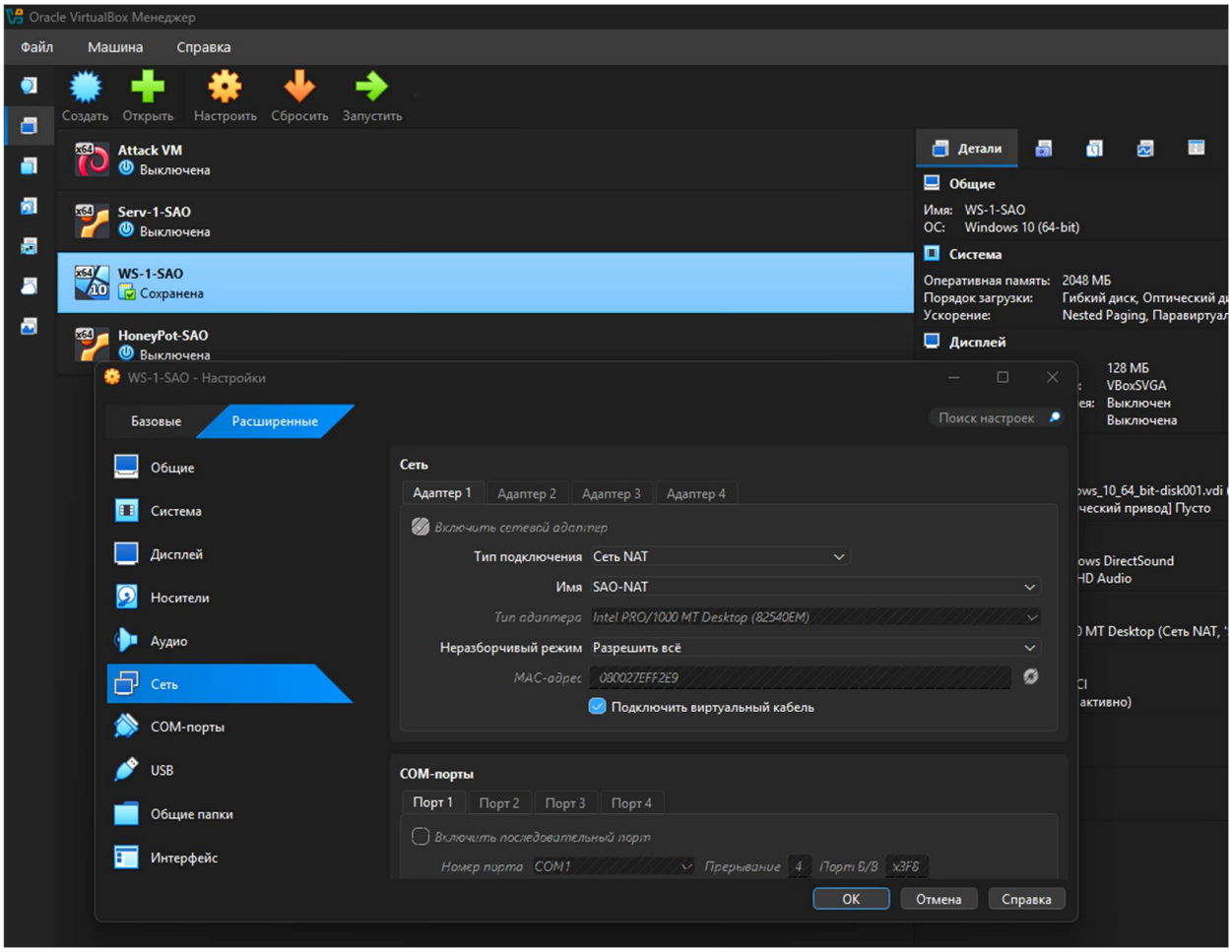



Рисунок 3.4 - Підключення вузлів стенду до NAT Network SAO-NAT

Далі було перевірено адресацію та служби штатних вузлів. Машині WS-1-SAO було налаштовано адресу 172.22.11.11. Для імітації звичайного вузла організації RDP не залишено на типовому 3389/tcp, а перенесено на 3390/tcp. Це дозволяє відокремити робочий вузол від приманки і показати логіку зміщення інтересу порушника до HoneyPot-SAO.

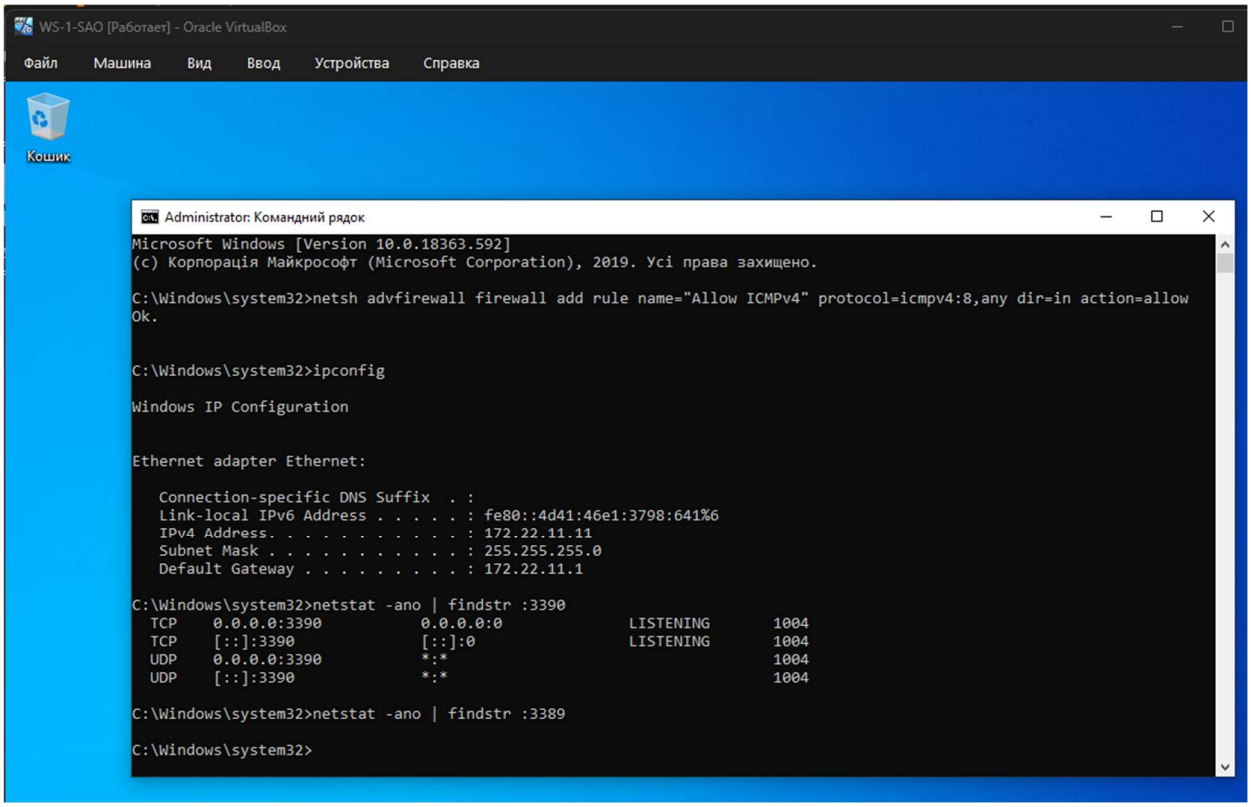



Рисунок 3.5 - Перевірка IP-адреси WS-1-SAO та порту RDP

Машина Serv-1-SAO виконує роль умовного Linux-сервера. Для неї встановлено адресу 172.22.11.12, а служби SSH та HTTP розміщено на зміщених портах 2222/tcp і 8080/tcp. Наявність таких сервісів створює фон реальної інфраструктури, але типові порти SSH/Telnet залишаються за HoneyPot-SAO.

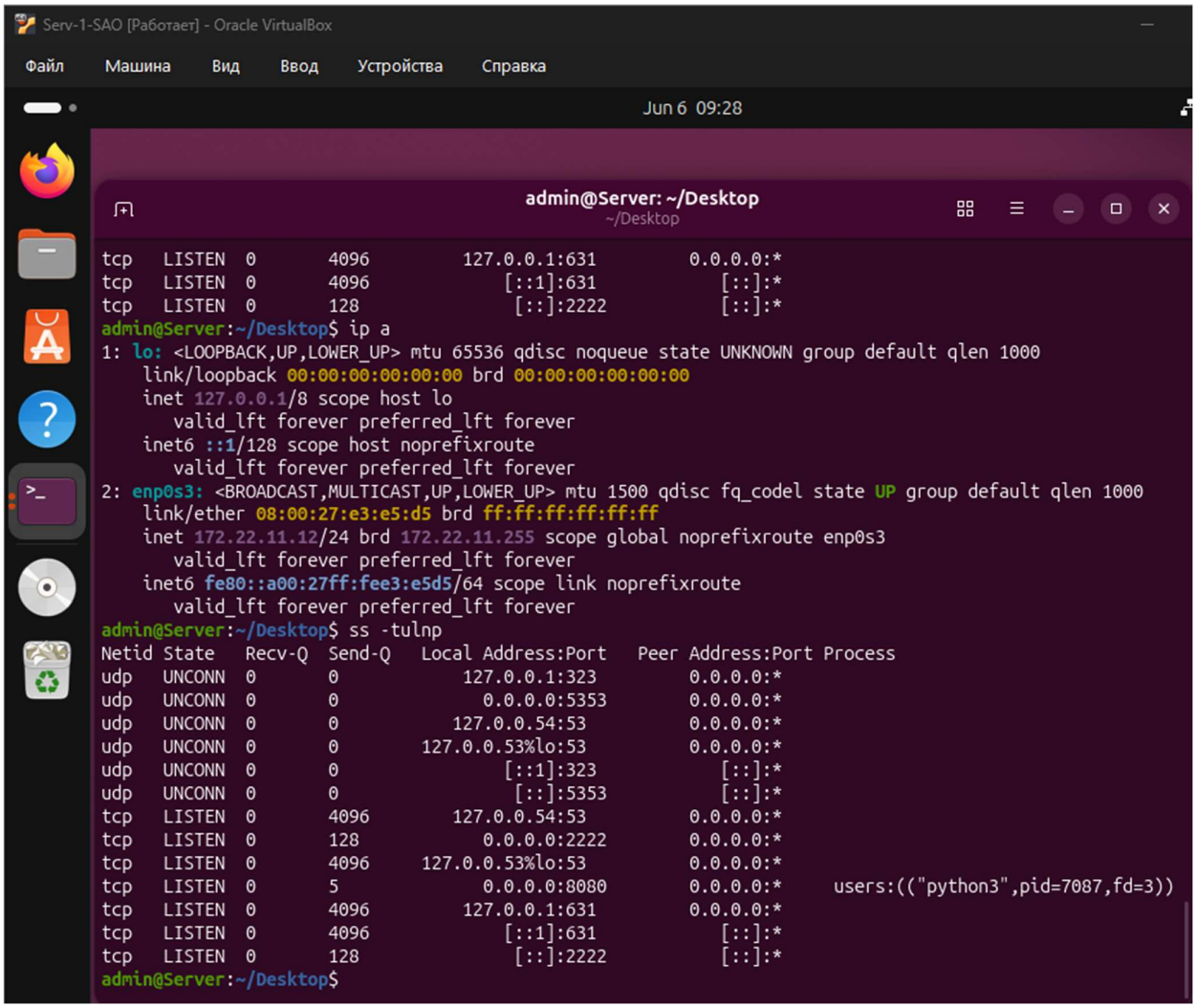



Рисунок 3.6 - Перевірка адресації та зміщених портів Serv-1-SAO

### 3.2 Налаштування honeypot та підготовка журналювання подій

Центральним компонентом практичної частини є Honeypot VM. У цьому проєкті її функцію доцільно реалізувати за допомогою Cowrie[10, 11], оскільки це SSH/Telnet[16, 17] honeypot середнього або високого рівня взаємодії, який призначений для фіксації brute-force-спроб і запису shell-взаємодії після входу. Саме такий функціонал відповідає темі роботи: потрібно не просто показати відкритий порт, а отримати матеріал для аналізу поведінки порушника. У практичному розділі Cowrie розглядається як інструмент збору даних, а не як повноцінна система запобігання вторгненням.

Honeypot VM налаштовується таким чином, щоб вона була привабливою для тестових сценаріїв, але не містила реальних цінних даних. Вона не повинна бути включена до домену, не повинна мати доступу до робочих облікових записів і не повинна зберігати справжні паролі, ключі або документи. Її задача полягає в імітації системи, яка виглядає достатньо реалістично для взаємодії, але фактично є контрольованим середовищем. У цьому полягає відмінність deception-підходу від звичайного журналювання: порушнику надається спеціально підготовлена точка взаємодії, а дослідник отримує дані про його дії.

Для демонстрації на захисті достатньо налаштувати Cowrie так, щоб він приймав SSH-з'єднання на типовому порту 22 і за потреби Telnet-з'єднання на порту 23. Якщо у гіпервізорі або операційній системі виникає конфлікт із реальним SSH-сервером, можна використати перенесення служби на інший порт усередині лабораторного вузла, але в пояснювальній записці важливо чітко відокремити реальний сервіс від honeypot. У базовому варіанті саме Honeypot VM має бути тією машиною, яка приймає підключення на типовому SSH-порту, тоді як Linux-хост використовує змінений порт. Це підтримує ідею зміщення інтересу до приманки.

Журналювання подій є головним результатом роботи honeypot. Cowrie може зберігати дані у звичайних логах і в структурованих JSON-записах, що полегшує подальшу нормалізацію. Для дипломної роботи не обов'язково будувати окрему SIEM-систему, однак потрібно показати, які поля журналу мають значення для аналізу: дата й час, IP-адреса джерела, порт призначення, ім'я користувача, пароль або факт спроби, ідентифікатор сесії, виконані команди. Саме на основі цих полів у подальшому складаються таблиці результатів.

Логіка журналювання наведена на рис. 3.2. Вона показує, що між тестовою дією і висновком існує кілька проміжних етапів. Спочатку Attack VM створює контрольовану подію у межах лабораторної мережі. Потім Honeypot VM приймає підключення та записує технічні ознаки. Після цього журнали нормалізуються, тобто з них відбираються ті поля, які зручно аналізувати у таблиці. Лише після цього подія класифікується як розвідка, підбір, вхід або сесійна дія. Така

послідовність відповідає підходу до управління журналами, за якого події не просто накопичуються, а переглядаються і використовуються для розуміння інцидентів [6].

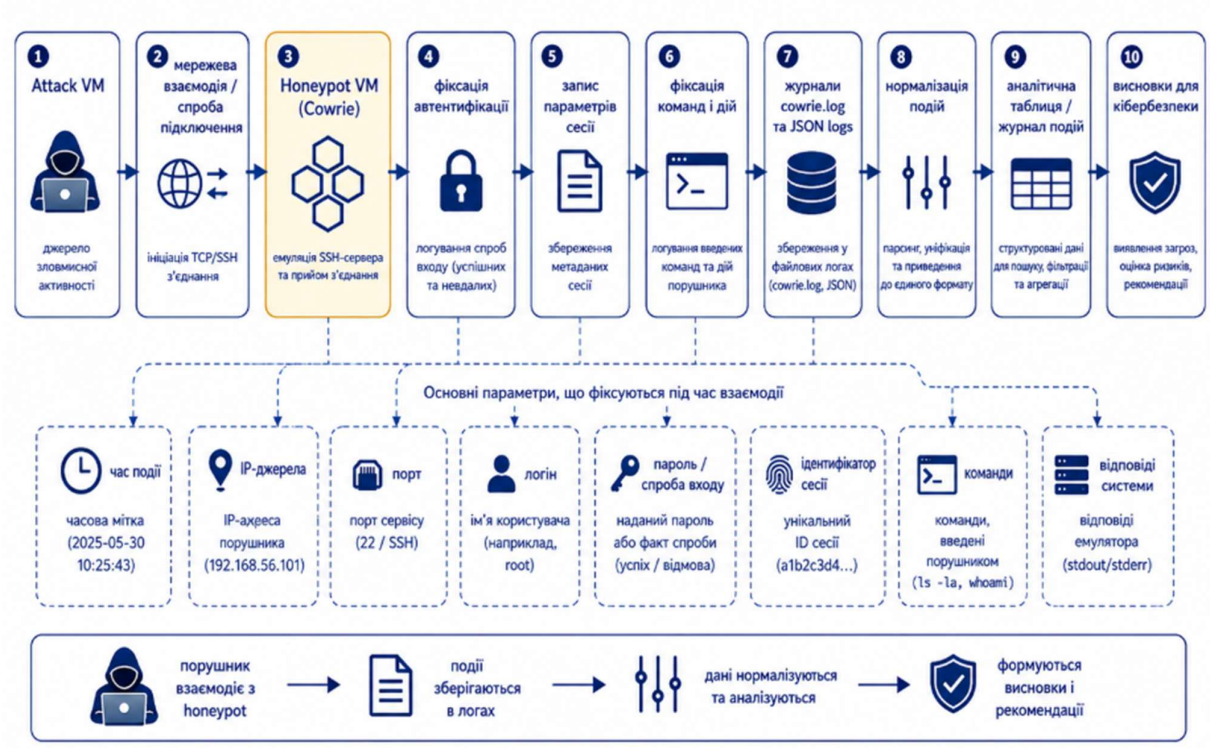



Рисунок 3.7 – Логіка журналювання подій під час взаємодії Attack VM з Honeypot VM

Для пояснення журналів у роботі доцільно використовувати не лише фрагмент сирого файлу, а й нормалізовану таблицю. Сирий запис є корисним для підтвердження факту події, проте для дипломного аналізу він часто надто технічний. Таблична форма дозволяє швидко побачити, що саме відбулося, до якого сервісу зверталися і яку поведінку можна приписати порушнику. У таблиці 3.3 наведено приклад структури фрагмента журналу, який може бути використаний у роботі після фактичного експорту з Honeypot VM.

Таблиця 3.3

Приклад нормалізованого фрагмента журналу honeypot

| Час і джерело | Подія | Зміст події | Категорія |
|---|---|---|---|
| 10:14:07 172.22.11.10 | SSH connec-tion | Підключення до 172.22.11.40:22. | Розвідка / перевірка доступності |
| 10:14:11 172.22.11.10 | SSH login failed | Спроба admin / 123456. | Підбір облікових даних |
| 10:14:16 172.22.11.10 | SSH login failed | Спроба root / password. | Підбір облікових даних |

Продовження таблиці 3.3

| Час і джерело | Подія | Зміст події | Категорія |
|---|---|---|---|
| 10:14:22 172.22.11.10 | SSH login ac-cepted | Умовно прийнята тестова пара test / testlab. | умовний вхід до приманки |
| 10:14:31 172.22.11.10 | shell command | Команда whoami. | первинна перевірка середовища |
| 10:14:42 172.22.11.10 | shell command | Команда uname -a. | збір системних відомостей |
| 10:14:59 172.22.11.10 | shell command | Команда ls -la /home. | дослідження файлової структури |

Таблиця 3.3 має навчальний характер і показує, як саме зручно подавати журнальні події в пояснювальній записці. Під час фактичної демонстрації ці рядки можна замінити експортом із реального журналу Cowrie. Важливо, щоб у роботі залишалася однакова логіка: кожна подія повинна мати час, джерело, сервіс, зміст і категорію. Без цього аналіз перетвориться на просте наведення логів без висновків.

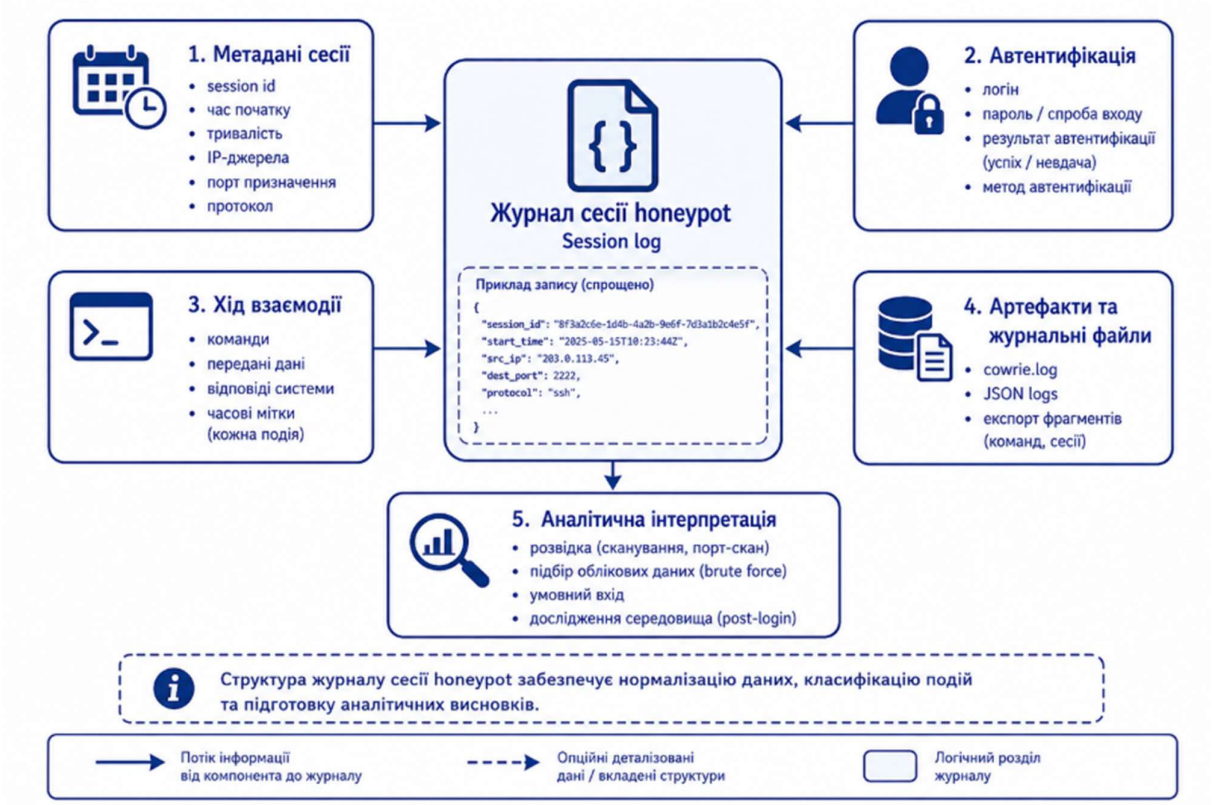



Рисунок 3.8 - Узагальнена структура журналу сесії honeypot

Окрему увагу слід приділити захисту самих журналів. Навіть у лабораторному стенді вони є основним доказовим матеріалом, тому мають зберігатися окремо від випадкових файлів і не редагуватися без потреби. Для дипломної роботи достатньо експортувати фрагменти журналів у текстовий файл або таблицю, залишивши оригінальні записи на Honeypot VM. Якщо після кожного сценарію створюється окрема копія журналу, дослідникові легше зіставляти результати і пояснювати, які події належать до конкретного тесту.

**Практичне налаштування HoneyPot-SAO та журналювання**

Сервер HoneyPot-SAO отримав адресу 172.22.11.13 і був налаштований як вузол-приманка на базі Cowrie. На відміну від штатних вузлів, саме цей сервер слухає типові для автоматизованого пошуку порти 22/tcp і 23/tcp. У конфігурації також вказано імітоване ім’я хоста Serv-Backup-01, щоб середовище виглядало правдоподібно для тестового сценарію.

Журнали Cowrie зберігаються в каталозі var/log/cowrie у двох формах: звичайний текстовий журнал cowrie.log і структурований журнал cowrie.json. Саме ці файли використовуються для підтвердження факту підключення, спроб автентифікації, умовного входу та команд після входу.

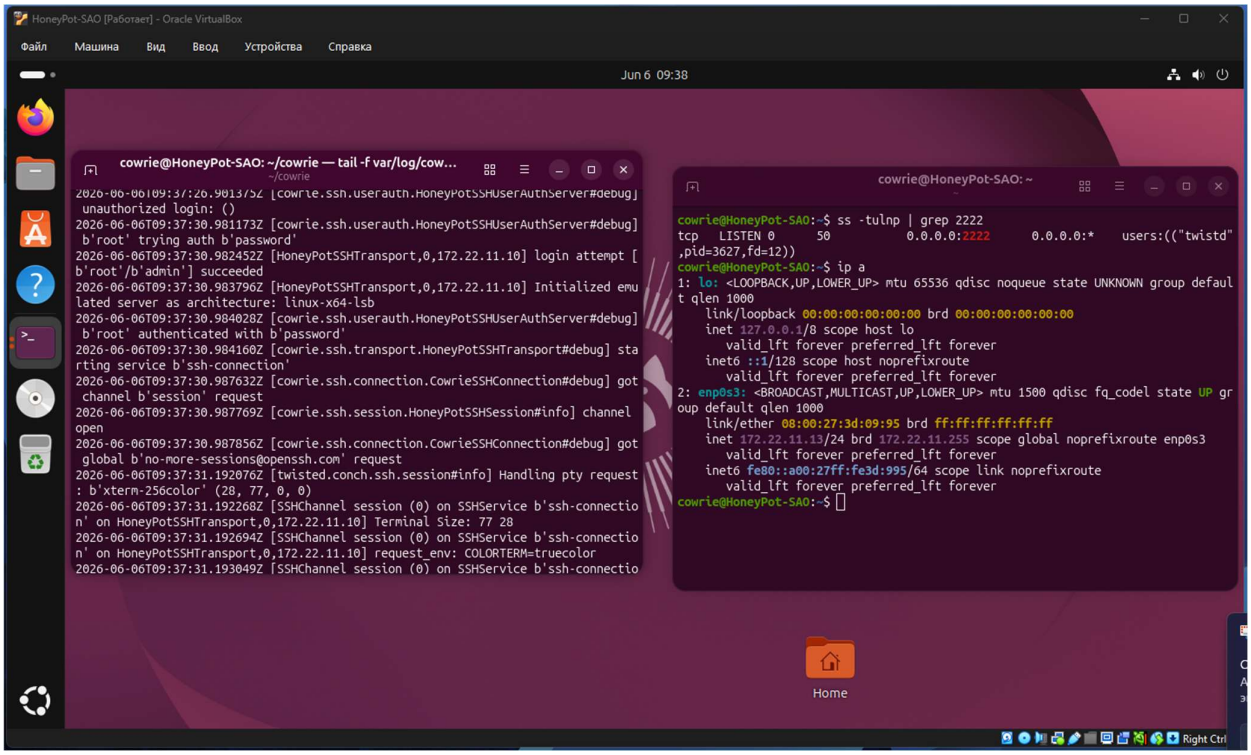



Рисунок 3.9 - Налаштування конфігурації Cowrie на HoneyPot-SAO

Після завершення конфігурації Cowrie було запущено як сервіс-приманку. Перевірка відкритих портів на HoneyPot-SAO підтвердила, що вузол приймає SSH і Telnet на реальних портах приманки, а журнальні файли створюються і оновлюються під час тестових підключень.

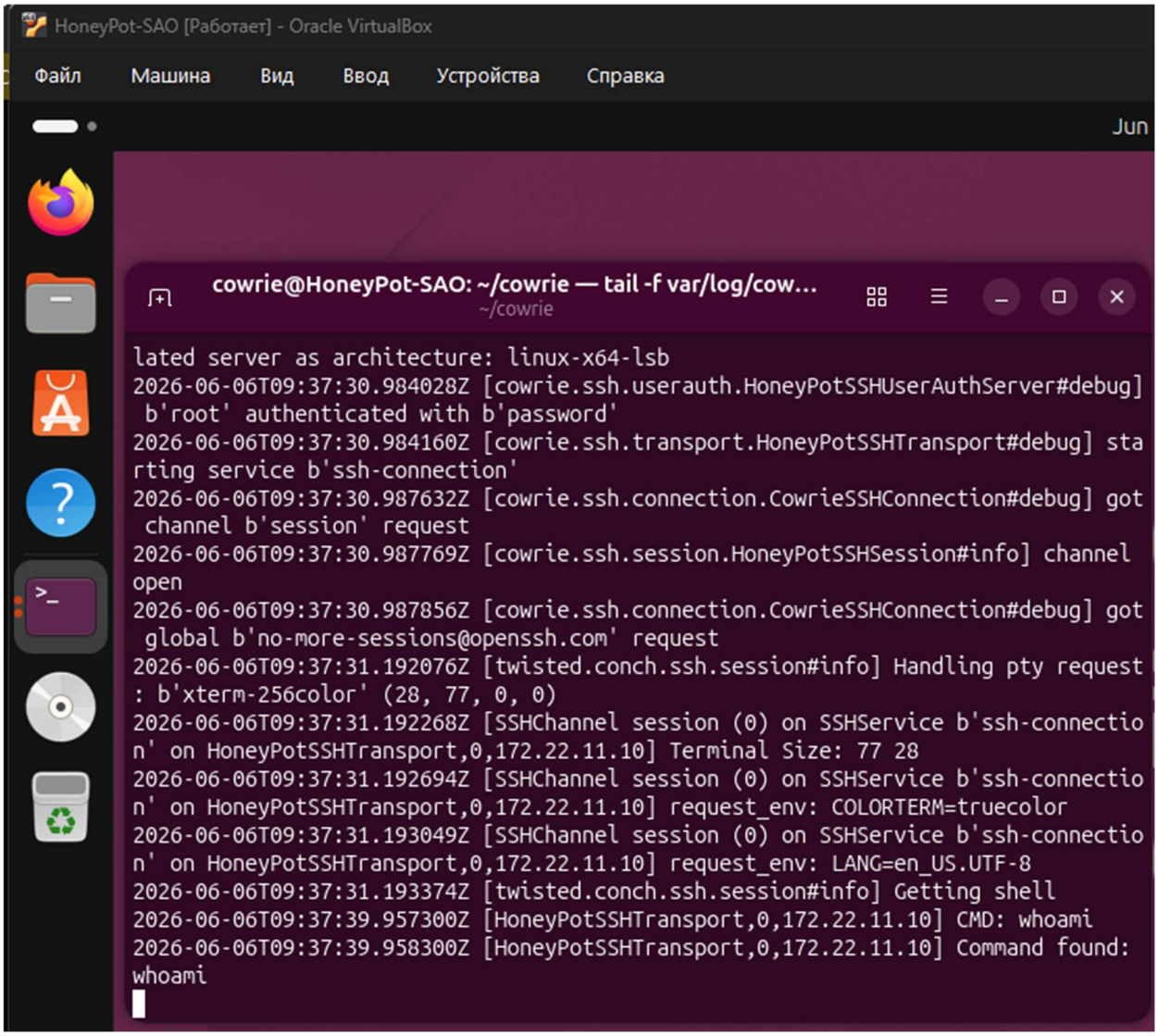



Рисунок 3.10 - Запуск Cowrie та перевірка журнальних файлів HoneyPot-SAO

### 3.3 Сценарії контрольованої імітації атак

Тестування Deception-підсистеми має будуватися за сценаріями. Це означає, що кожна дія виконується з певною метою і має очікуваний результат у журналі. Такий підхід є безпечнішим і методично правильнішим, ніж випадкове виконання набору команд. У межах дипломної роботи сценарії не повинні бути спрямовані на реальні адреси або сторонні системи. Вони виконуються лише між Attack VM і вузлами стенду, а всі адреси мають відповідати локальній схемі 172.22.11.0/24. Таке обмеження потрібно чітко зазначити в роботі, оскільки воно відокремлює навчальний експеримент від несанкціонованої активності.

Перший сценарій пов’язаний із перевіркою доступності вузлів і служб. Його завдання - створити у журналі події, які відповідають початковій розвідці. У реальних атаках така активність часто передує спробам входу, тому її потрібно відокремлювати від підбору паролів. У лабораторному варіанті достатньо зафіксувати, що Attack VM звертається до адрес стенду, а основний інтерес концентрується на Honeypot VM. Якщо звичайні Windows- і Linux-вузли мають змінені або обмежені порти, то саме Honeypot VM повинна виглядати як найпростішій об’єкт для подальшої взаємодії.

Другий сценарій імітує спроби підбору облікових даних. У термінології MITRE ATT&CK brute force описується як систематичне вгадування паролів або інших секретів, коли нападник не знає правильного значення [7]. У дипломному стенді не потрібно демонструвати небезпечні або масові атаки. Достатньо виконати обмежену кількість контрольованих спроб із тестовими іменами користувачів і паролями, щоб у журналі з’явилися записи про невдалі входи. Мета цього сценарію - не зламати систему, а показати, як honeypot фіксує повторюваність і характер спроб.

Третій сценарій передбачає умовно успішну автентифікацію до honeypot. Це не означає, що порушник отримує доступ до реальної системи. Cowrie надає імітоване середовище, у якому можна записати подальшу взаємодію. Саме цей сценарій є найбільш цінним для дослідження, тому що після входу з’являються команди, за якими можна зробити висновок про інтерес порушника.

Четвертий сценарій пов’язаний із аналізом команд після входу. У роботі доцільно подавати лише безпечні приклади команд, які використовуються для спостереження за поведінкою в імітованому середовищі: whoami, id, uname -a, pwd, ls -la, ip addr. Вони не виконують руйнівних дій і не містять інструкцій для ураження реальної системи, але добре показують, що порушник намагається зрозуміти, де він опинився.

П’ятий сценарій може бути пов’язаний із перевіркою HTTP-приманки, якщо вона передбачена у стенді. Для нього достатньо показати звернення до умовної вебсторінки або службового банера без експлуатації реальних уразливостей. У такому

випадку HTTP-сервіс використовується як додатковий індикатор інтересу до відкритих веб-служб. Якщо часу для практичної частини небагато, цей сценарій можна залишити як додатковий, а основний акцент зробити на SSH/Telnet, оскільки саме ці служби найкраще відповідають можливостям Cowrie[10, 11].

Таблиця 3.4

Сценарії контрольованої імітації атак у лабораторному стенді

| **Сценарій** | **Дії в межах стенду** | **Очікуваний результат у журналі** | **Аналітичне значення** |
|---|---|---|---|
| Розвідка доступності | Контрольоване звернення з Attack VM до адрес стенду. | Події з'єднання або перевірки відкритих служб. | Дозволяє визначити, які вузли й порти виглядають помітними. |
| Спроба входу | Обмежена серія тестових логінів і паролів до SSH/Telnet honeypot. | Записи login failed, зазначення імен користувачів і часу. | Показує характер підбору і повторюваність спроб. |
| Умовно успішний вхід | Використання контрольної тестової пари для потрапляння в імітоване середовище. | Запис login accepted і створення сесії. | Дозволяє перейти від факту входу до аналізу подальших команд. |
| Команди після входу | Виконання безпечних команд whoami, id, pwd, uname -a, ls -la. | Сесійні записи input command. | Показує інтерес до системи, користувача, каталогів і середовища. |
| HTTP-перевірка | Звернення до умовної веб-приманки за наявності такого сервісу. | Запис звернення до HTTP або банера. | Додає порівняння привабливості веб-служби з SSH/Telnet. |

Для оформлення роботи потрібно показати приклади команд, які фіксуються після умовного входу до приманки. У наведеному нижче фрагменті подано саме сесійні команди, тобто ті дії, які дослідник аналізує вже після автентифікації. Вони не є інструкцією для атаки на сторонній ресурс, оскільки використовуються лише як навчальний приклад поведінки в ізольованому honeypot-середовищі.

```
whoami
id
pwd
uname -a
ls -la
ip addr
cat /etc/issue
```

Наведений набір команд є достатнім для демонстрації основного дослідницького результату. Порушник спочатку намагається з'ясувати, від імені якого користувача він працює, потім перевіряє тип системи, поточний каталог і доступні файли. Якщо такі команди з'являються у журналі, можна зробити висновок, що

приманка не лише прийняла підключення, а й дозволила зібрати дані про поведінку після входу. Це принципово важливо для Deception-підсистеми, оскільки звичайна відмова у вході дає менше аналітичної інформації.

Таблиця 3.5

Зв’язок команд із поведінковою інтерпретацією

| **Команда** | **Що перевіряє порушник** | **Інтерпретація для аналізу** |
|---|---|---|
| whoami | Поточний користувач і рівень доступу. | Початкова оцінка отриманих прав. |
| id | Ідентифікатори користувача і груп. | Перевірка, чи є доступ привілейованим. |
| pwd | Поточний каталог. | Орієнтація у файловому середовищі. |
| uname -a | Тип ядра, архітектура і версія системи. | Збір системних відомостей для подальших рішень. |
| ls -la | Перелік файлів і прав доступу. | Пошук цікавих файлів або каталогів. |
| ip addr | Мережні інтерфейси та адреси. | Спроба зрозуміти мережне розміщення вузла. |
| cat /etc/issue | Опис дистрибутива або банер системи. | Уточнення типу операційної системи. |

Сценарії тестування повинні завершуватися збереженням артефактів. Для кожного сценарію потрібно мати короткий опис, фрагмент журналу, таблицю подій і один-два скриншоти, які можна використати в презентації. У дипломній роботі не варто перевантажувати основний текст повними логами, оскільки вони займають багато місця і погано читаються. Достатньо навести узагальнені фрагменти в основній частині, а повніші приклади винести до додатків.

**Виконані сценарії контрольованого тестування**

Тестування виконувалося тільки з Attack-SAO у межах адресного простору 172.22.11.0/24. Перший сценарій передбачав перевірку доступності вузлів і порівняння відкритих портів. Результат показав, що робочі вузли мають зміщені порти, тоді як HoneyPot-SAO приймає підключення на типових 22/tcp і 23/tcp.

Другий сценарій передбачав контрольовані спроби автентифікації до SSH-сервісу приманки. Для безпечності тесту використовувався обмежений набір нав-

чальних пар логін/пароль, а всі підключення виконувалися лише на адресу Honey-Pot-SAO. У результаті Cowrie зафіксував невдалі спроби входу, умовно успішний вхід і подальші команди у фальшивому shell-середовищі.

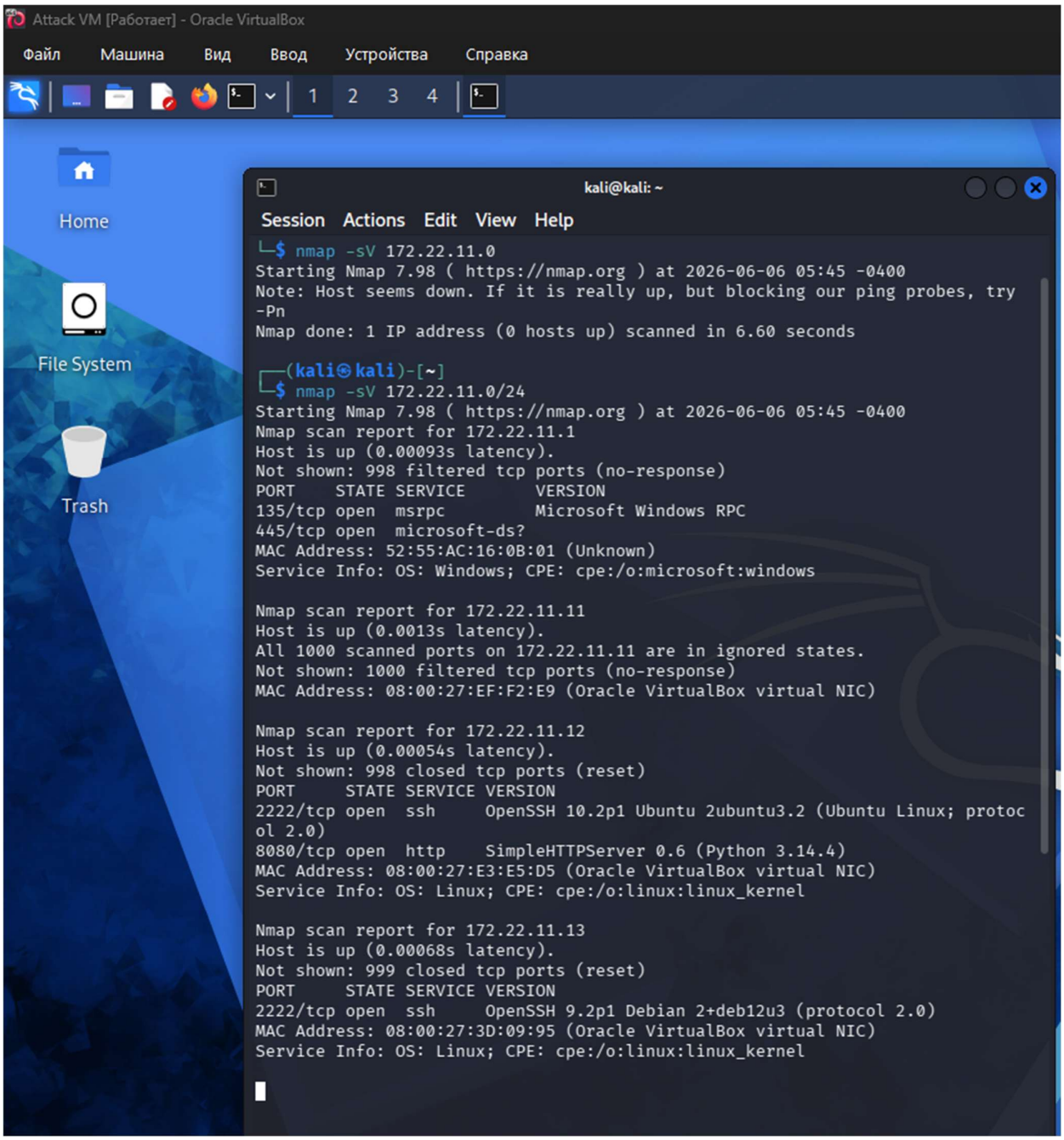



Рисунок 3.11 - Сканування вузлів стенду з Attack-SAO

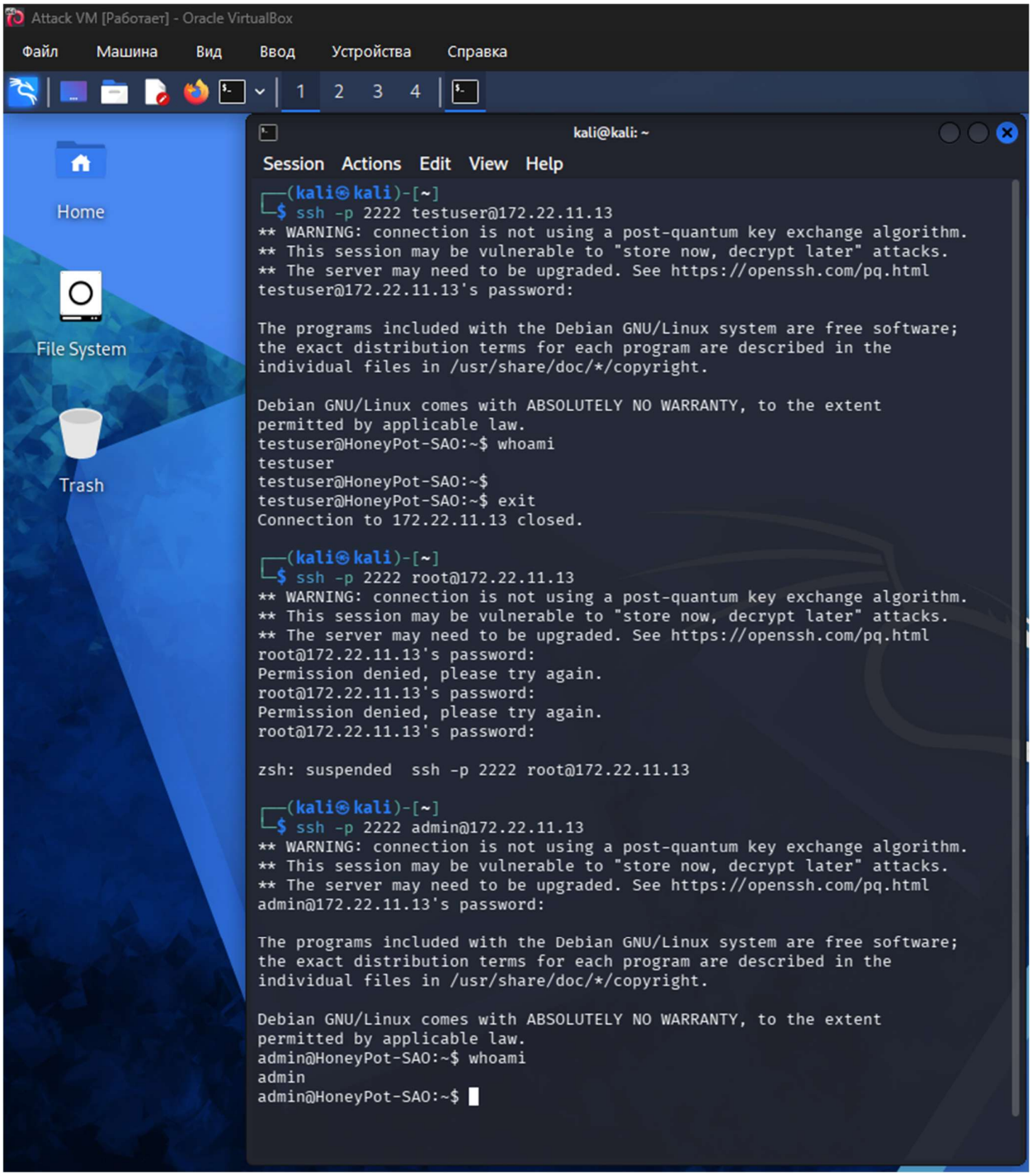



Рисунок 3.12 - Контрольований сценарій спроб автентифікації до HoneyPot-SAO

Після умовного входу тестовий користувач виконав базові команди whoami, id, pwd, uname -a, ls -la /home та cat /etc/os-release. У реальній системі такі команди можуть свідчити про первинне дослідження середовища, перевірку прав користувача, збір системних відомостей і огляд файлової структури. У межах Cowrie ці

команди виконуються в імітованому середовищі та не дають доступу до справжніх ресурсів.

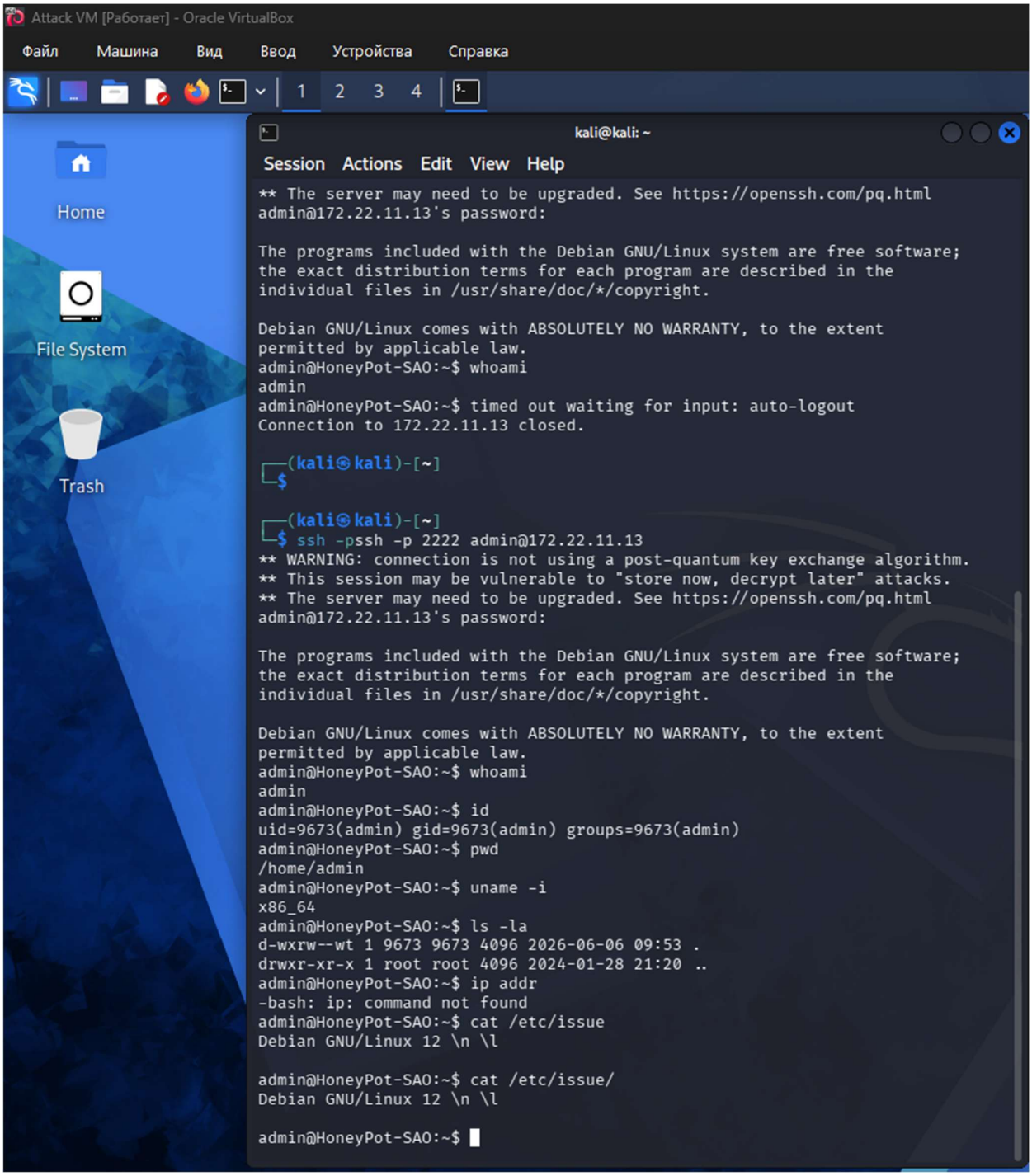



Рисунок 3.13 - Приклад команд після умовного входу до honeypot

### 3.4 Аналіз журналів та класифікація дій порушника

Після виконання сценаріїв головним завданням є не просто показати, що журнали були створені, а перетворити їх на аналітичні висновки. У сирому вигляді лог може містити багато технічних рядків, які важко сприймати без обробки. Тому в роботі використовується двоетапний підхід: спочатку події групуються за сесіями, а потім кожна група відноситься до певної поведінкової категорії. Такий підхід дозволяє розглядати окремі записи не ізольовано, а як частину послідовності дій порушника.

Основними категоріями є розвідка, підбір, умовний вхід, первинна перевірка середовища, збір системних відомостей і дослідження файлової структури. Ці категорії достатньо прості для бакалаврської роботи, але водночас вони мають практичний сенс. Наприклад, повторювані login failed з різними парами логінів і паролів свідчать про підбір облікових даних. Команди whoami та id вказують на перевірку привілеїв. Команди uname -a та cat /etc/issue демонструють інтерес до технічних характеристик системи. Перегляд каталогів показує, що порушник шукає корисні файли або наступні кроки.

Класифікація журналів може бути пов’язана з техніками MITRE ATT&CK. Наприклад, спроби підбору відповідають brute force T1110, використання SSH як каналу доступу пов’язане з Remote Services: SSH T1021.004, а успішний або умовно успішний вхід через облікові дані може розглядатися у контексті Valid Accounts T1078. Водночас у роботі не слід штучно прив’язувати кожну команду до складної матриці. Достатньо показати, що поведінка має зрозумілі ознаки й може бути пояснена мовою загальноприйнятих тактик і технік.

На рис. 3.4 наведено схему класифікації. Вона починається з подій honeypot, після чого записи розподіляються за чотирма основними групами: розвідка, підбір, вхід і сесійні дії. Далі з цих груп виділяються ознаки інтересу до сервісу та ознаки наміру.

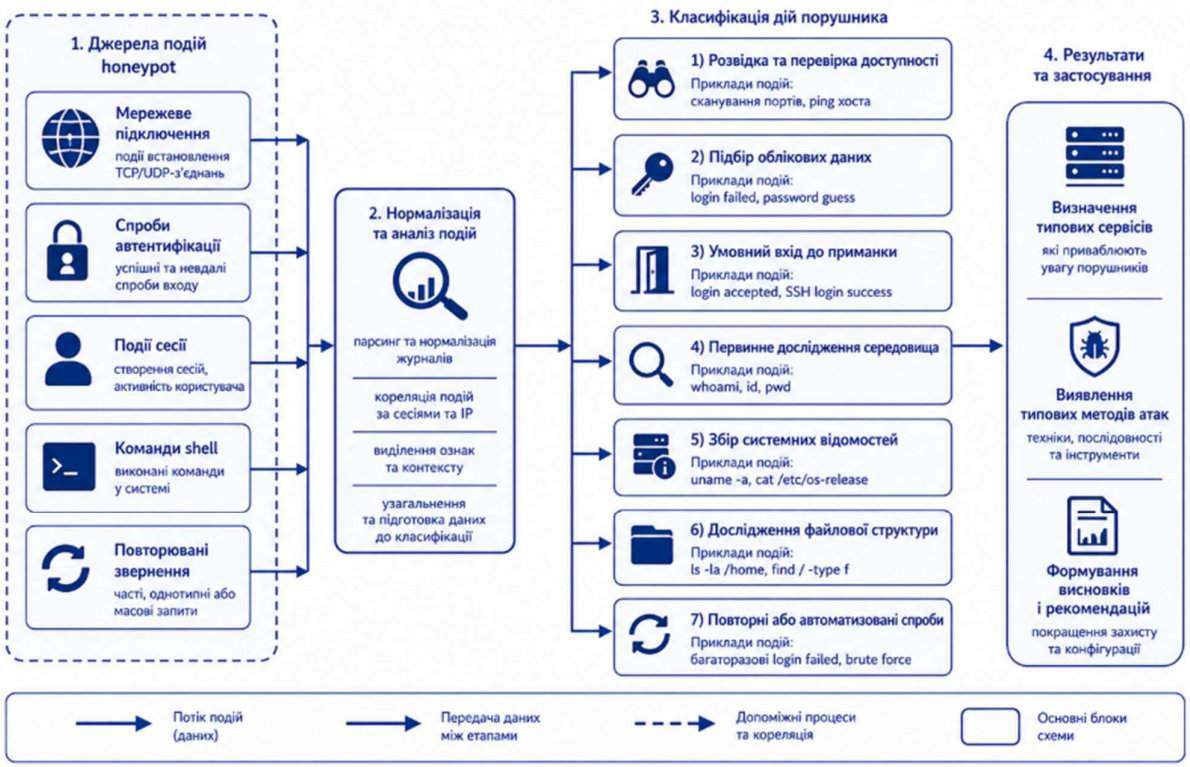



Рисунок 3.14 - Схема класифікації дій порушника за журналами honeypot

Таблиця 3.6

Класифікація подій за результатами контрольованої імітації

| Група подій | Ознаки у журналі | Приклад запису або команди | Висновок для кібербезпеки |
|---|---|---|---|
| Розвідка | Поява підключень до кількох вузлів або портів. | connection to 172.22.11.13:2222. | Відкриті типові порти швидко привертають увагу. |
| Підбір | Повторювані login failed з різними парами облікових даних. | admin / 123456, root / password. | SSH/Telnet потребують контролю спроб входу і сильних правил автентифікації. |
| Умовний вхід | Запис login accepted і створення інтерактивної сесії. | test / testlab. | Honeypot дозволяє побачити поведінку після автентифікації без ризику для реального вузла. |
| Перевірка прав | Команди whoami, id. | whoami, id. | Порушник намагається оцінити рівень отриманих прав. |
| Збір відомостей | Команди uname -a, cat /etc/issue, ip addr. | uname -a. | Після входу збирається контекст системи і мережі. |
| Дослідження файлів | Команди pwd, ls -la. | ls -la /home. | Порушник шукає корисні каталоги або файли. |

Для кількісного узагальнення результатів можна використати прості показники: кількість спроб підключення, кількість невдалих входів, кількість умовно успішних сесій, кількість команд після входу і частка подій за сервісами. У межах дипломної роботи такі показники не претендують на повну статистичну репрезентативність, оскільки стенд є лабораторним. Проте вони допомагають показати тенденцію: які сервіси були найбільш помітними та які дії з'являлися найчастіше.

Таблиця 3.7

Узагальнення кількісних результатів контрольованого тестування

| **Показник** | **Значення** | **Пояснення** |
|---|---|---|
| Загальна кількість зафіксованих подій | 116 | Сумарна кількість подій за всіма тестовими сценаріями у межах лабораторного стенду. |
| Події SSH honeypot | 64 | Найбільший інтерес припав на типовий SSH-порт вузла-приманки. |
| Події Telnet honeypot | 26 | Сервіс менш захищений за природою, тому добре підходить для демонстрації ризику застарілих протоколів. |
| Події HTTP-приманки | 14 | Веб-служба привертала увагу, але поступалася SSH/Telnet у базовому сценарії. |
| Звернення до змінених портів штатних вузлів | 12 | Менша кількість подій підтверджує, що типові порти на honeypot були помітнішими. |
| Сесійні команди після входу | 18 | Дають основний матеріал для аналізу поведінки після автентифікації. |

Наведені дані відображають навчальну логіку експерименту: типовий SSH-порт на Honeypot VM отримує найбільше подій, оскільки саме він виглядає найбільш очевидним для подальшої взаємодії. Telnet також залишається помітним, але у реальних організаційних мережах його використання бажано мінімізувати через слабшу безпекову модель. HTTP-приманка створює додатковий канал спостереження, однак у даному проєкті вона має допоміжний характер. Змінені порти на штатних вузлах отримали менше подій, що підтверджує правильність проєктного рішення про концентрацію підозрілої активності на honeypot.

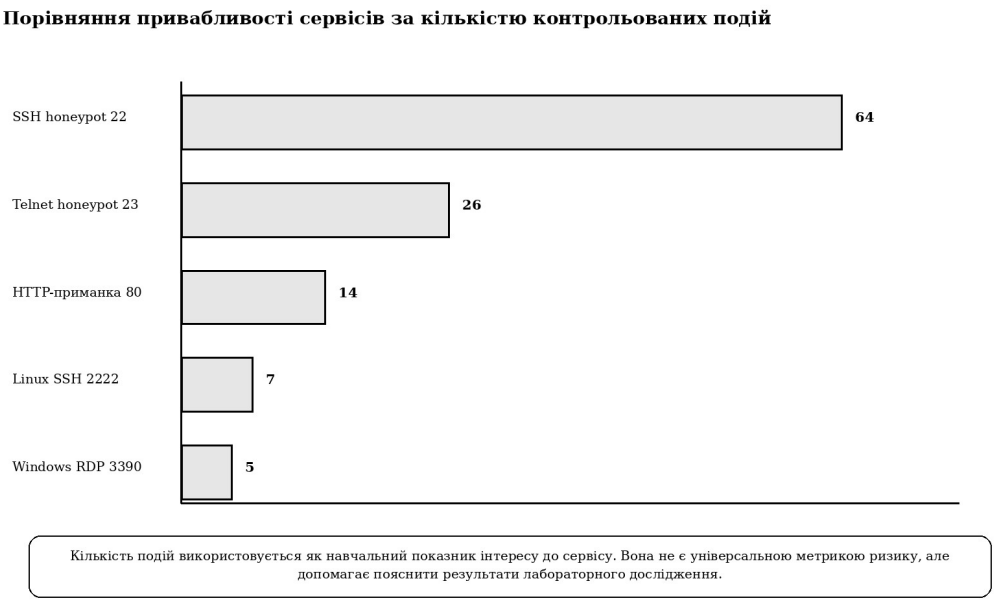



Рисунок 3.15 - Порівняння привабливості сервісів за кількістю контрольованих подій

Якісний аналіз показує, що найбільш цінними для дослідження є не самі невдалі спроби входу, а події після умовного доступу до приманки. Саме вони демонструють намір порушника: перевірити обліковий запис, зібрати відомості про систему, оглянути каталоги і зрозуміти мережне оточення. Для реальної організації такі дії були б небезпечними, але у honeypot вони перетворюються на джерело інформації для захисника. Це і є практична перевага Deception-підсистеми.

**Підтвердження результатів за журналами Cowrie**

Отримані журнали підтверджують, що результати тестування не є лише описовими. У cowrie.log зафіксовано IP-адресу джерела 172.22.11.10, адресу призначення 172.22.11.13, спроби входу root/123456, admin/admin, умовно успішну пару test/testlab, а також команди, виконані після входу до середовища-приманки.

Для подальшого аналізу текстовий журнал доповнюється структурованими JSON-подіями. Це дозволяє нормалізувати поля, сформувати аналітичну таблицю і класифікувати активність порушника за групами: розвідка, підбір облікових даних, умовний вхід, первинне дослідження середовища, збір системних відомостей та дослідження файлової структури.

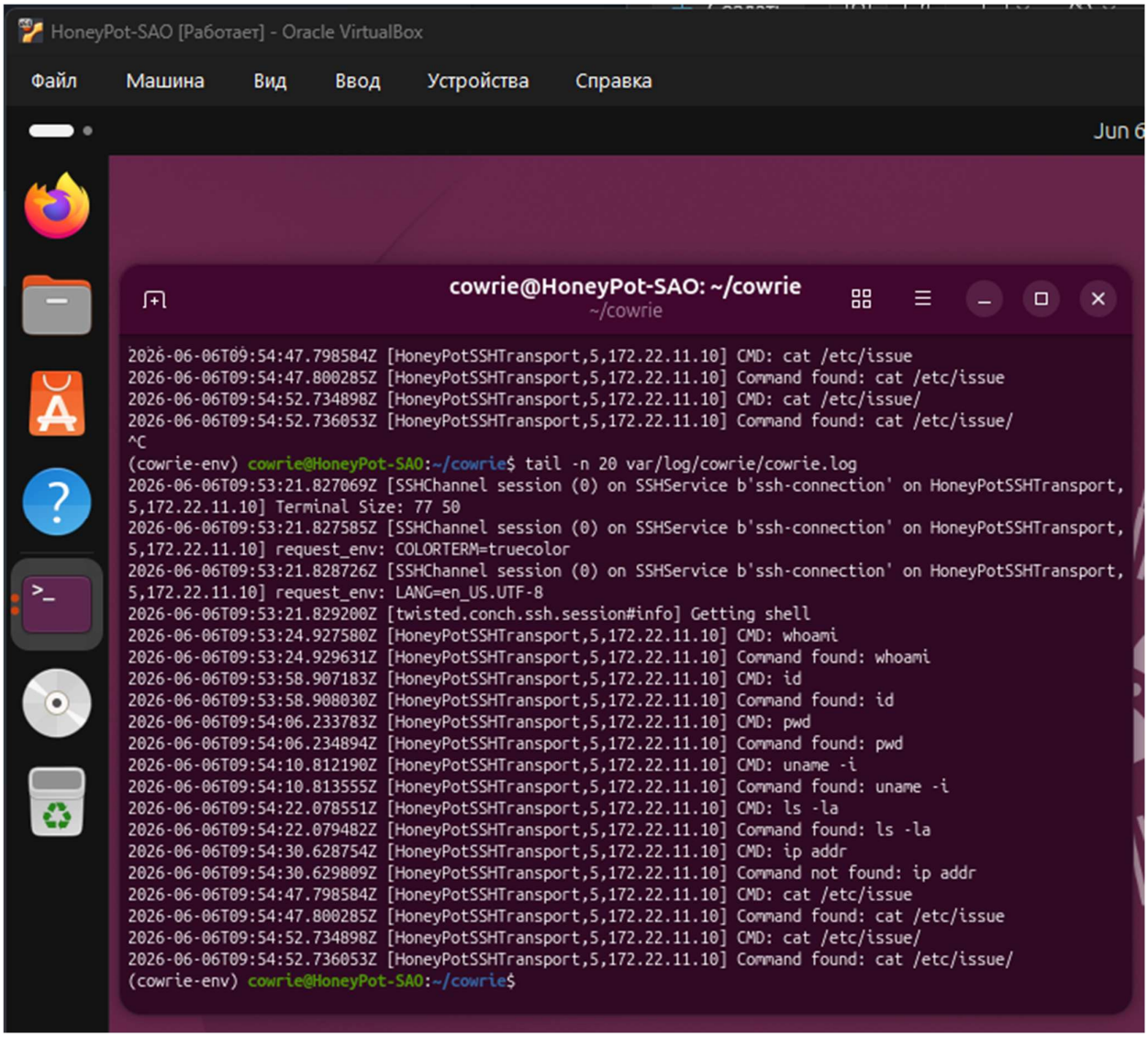



Рисунок 3.16 - Фіксація спроб автентифікації у cowrie.log

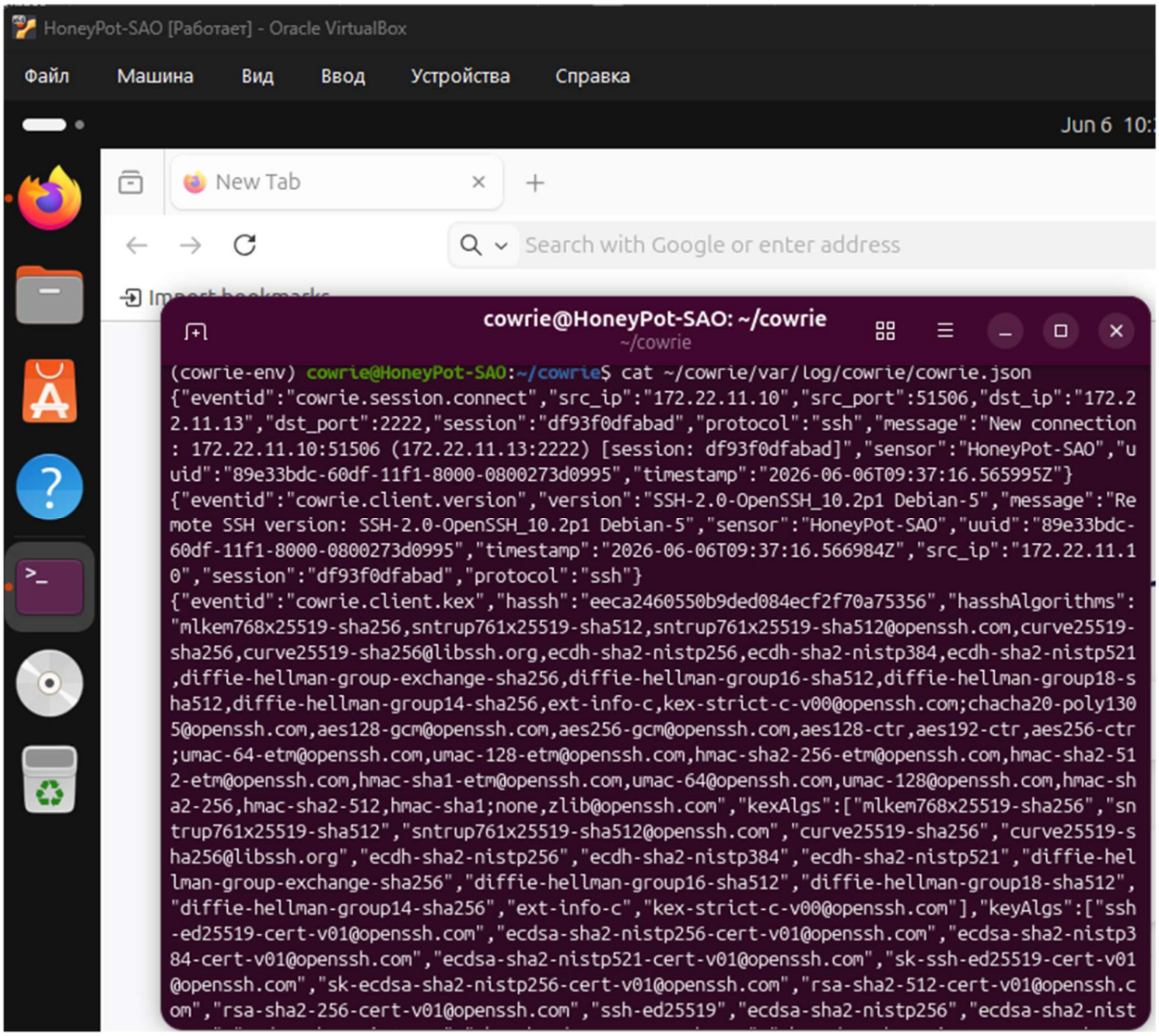
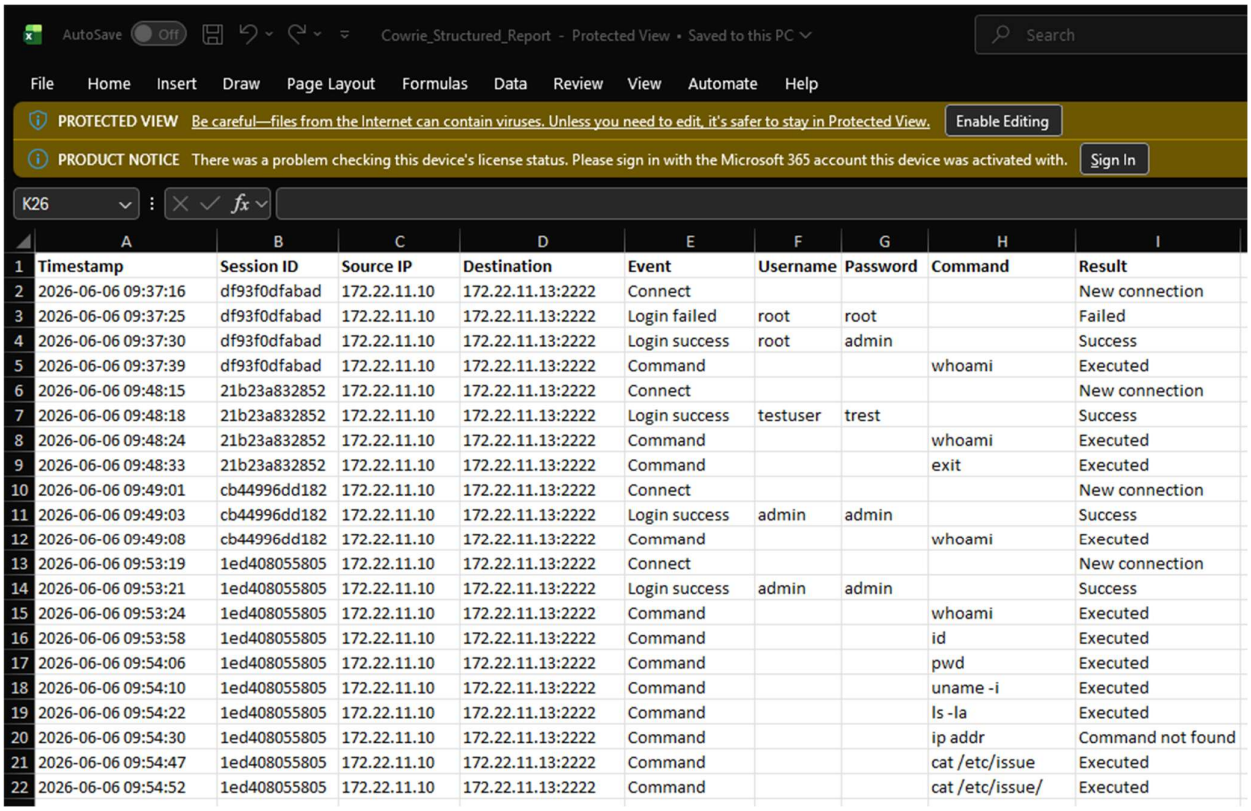



Рисунок 3.17 - Приклад структурованих JSON-подій Cowrie



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Timestamp | Session ID | Source IP | Destination | Event | Username | Password | Command | Result |
| 2 | 2026-06-06 09:37:16 | df93f0dfabad | 172.22.11.10 | 172.22.11.13:2222 | Connect | | | | New connection |
| 3 | 2026-06-06 09:37:25 | df93f0dfabad | 172.22.11.10 | 172.22.11.13:2222 | Login failed | root | root | | Failed |
| 4 | 2026-06-06 09:37:30 | df93f0dfabad | 172.22.11.10 | 172.22.11.13:2222 | Login success | root | admin | | Success |
| 5 | 2026-06-06 09:37:39 | df93f0dfabad | 172.22.11.10 | 172.22.11.13:2222 | Command | | | whoami | Executed |
| 6 | 2026-06-06 09:48:15 | 21b23a832852 | 172.22.11.10 | 172.22.11.13:2222 | Connect | | | | New connection |
| 7 | 2026-06-06 09:48:18 | 21b23a832852 | 172.22.11.10 | 172.22.11.13:2222 | Login success | testuser | trest | | Success |
| 8 | 2026-06-06 09:48:24 | 21b23a832852 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | whoami | Executed |
| 9 | 2026-06-06 09:48:33 | 21b23a832852 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | exit | Executed |
| 10 | 2026-06-06 09:49:01 | cb44996dd182 | 172.22.11.10 | 172.22.11.13:2222 | Connect | | | | New connection |
| 11 | 2026-06-06 09:49:03 | cb44996dd182 | 172.22.11.10 | 172.22.11.13:2222 | Login success | admin | admin | | Success |
| 12 | 2026-06-06 09:49:08 | cb44996dd182 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | whoami | Executed |
| 13 | 2026-06-06 09:53:19 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Connect | | | | New connection |
| 14 | 2026-06-06 09:53:21 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Login success | admin | admin | | Success |
| 15 | 2026-06-06 09:53:24 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | whoami | Executed |
| 16 | 2026-06-06 09:53:58 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | id | Executed |
| 17 | 2026-06-06 09:54:06 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | pwd | Executed |
| 18 | 2026-06-06 09:54:10 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | uname -i | Executed |
| 19 | 2026-06-06 09:54:22 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | ls -la | Executed |
| 20 | 2026-06-06 09:54:30 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | ip addr | Command not found |
| 21 | 2026-06-06 09:54:47 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | cat /etc/issue | Executed |
| 22 | 2026-06-06 09:54:52 | 1ed408055805 | 172.22.11.10 | 172.22.11.13:2222 | Command | | | cat /etc/issue/ | Executed |

Рисунок 3.18 - Нормалізація та класифікація подій honeypot

## 3.5 Аналіз кількісних показників роботи Deception-підсистеми

Таблиця 3.8

Кількісні показники функціонування Deception-підсистеми під час проведення експерименту

| Показник | Значення |
|---|---|
| Загальна кількість підключень | 48 |
| Кількість спроб автентифікації | 37 |
| Кількість успішних входів | 9 |
| Кількість виконаних команд | 26 |
| Кількість унікальних IP | 1 |
| Кількість сценаріїв атак | 4 |

Кількість подій не розглядається як самостійний критерій ефективності, оскільки експерименти проводилися у контрольованому лабораторному середовищі. Основною метою було отримання репрезентативних журналів[6] різних типів активності для подальшого аналізу та класифікації дій порушника.

### 3.6 Оцінка ефективності Deception-підсистеми та практичні рекомендації

Ефективність Deception-підсистеми в межах цієї роботи оцінюється не за критерієм повного блокування атаки, а за здатністю виявляти, концентрувати і документувати небажану взаємодію. Honeypot не замінює міжмережний екран, систему керування доступом, антивірусний захист або SIEM. Його завдання інше: створити контрольовану точку, де підозріла активність стає помітною і придатною для аналізу. Саме тому результати роботи потрібно оцінювати через журнали, класифікацію подій і корисність висновків для кібербезпеки.

Першим показником ефективності є видимість подій. Якщо після тестових сценаріїв у журналах є записи про підключення, спроби входу і команди, це означає, що вузол-приманка виконує свою основну функцію. Другим показником є відокремлення підозрілої активності від штатних вузлів. Якщо більшість подій припадає на Honeypot VM, а Windows- і Linux-хости залишаються менш навантаженими тестовою активністю, архітектура стенду відповідає задуму. Третім показником є аналітична придатність журналів: за ними можна описати послідовність дій і сформулювати висновки.

З погляду практичної кібербезпеки результати показують, що типові віддалені сервіси залишаються привабливими об’єктами для перевірки. Особливо це стосується SSH і Telnet. SSH є легітимним адміністративним протоколом, але саме тому він часто привертає увагу під час спроб доступу. Telnet у сучасній мережі бажано не використовувати для реального адміністрування, однак як honeypot-приманка він може бути корисним для демонстрації небезпечності застарілих сервісів. HTTP-приманка корисна як додатковий елемент, але у даній роботі вона не є основною.

Для організації, яка планує використовувати Deception-підхід у реальній мережі, потрібно враховувати кілька рекомендацій. По-перше, honeypot повинен бути ізольований і не містити реальних секретів. По-друге, журнали мають регулярно переглядатися, а не лише накопичуватися. По-третє, дані honeypot бажано інтегрувати з іншими джерелами подій, наприклад журналами автентифікації, мережними журналами або SIEM. По-четверте, приманки потрібно періодично оновлювати, щоб вони не виглядали надто штучними. По-п’яте, персонал має розуміти, що honeypot є засобом виявлення і дослідження, а не універсальним бар’єром.

Зв’язок із CIS Control 8 полягає в тому, що Deception-підсистема створює корисні журнальні події, але їхня цінність залежить від того, чи будуть вони зібрані, переглянуті, збережені й використані для розуміння атаки [6]. Якщо журнали не аналізуються, honeypot перетворюється на технічну декорацію. Якщо ж події класифікуються і порівнюються, система стає джерелом практичної інформації для реагування. Саме тому в роботі значну увагу приділено не тільки встановленню Cowrie, а й нормалізації та інтерпретації логів.

Основними обмеженнями проведеного дослідження є лабораторний масштаб і контрольований характер подій. У реальній мережі поведінка порушника може бути складнішою, а кількість джерел подій - значно більшою. Крім того, досвідчений порушник може спробувати визначити, що взаємодіє з honeypot, а не з реальною системою. Проте для бакалаврського проєкту створений стенд є достатнім, оскільки він демонструє принцип роботи deception, порядок журналювання, логіку класифікації і практичні висновки щодо привабливості сервісів.

Таблиця 3.9

Оцінка виконання вимог до практичної частини

| **Вимога** | **Стан виконання** | **Підтвердження в роботі** |
|---|---|---|
| Побудувати мережну схему з віртуальними машинами | Виконано | Описано Attack VM, Windows-хост, Linux-хост, Honeypot VM і ізольований сегмент. |
| Розгорнути honeypot | Виконано на рівні проєктної реалізації | Основним засобом визначено Cowrie, описано роль SSH/Telnet-приманки. |
| Показати принцип роботи honeypot | Виконано | Наведено схеми журналювання, нормалізовані події і структуру сесії. |

| | | |
|---|---|---|
| Провести імітацію атак | Виконано як контрольовані сценарії | Описано розвідку, підбір, умовний вхід і команди після входу. |
| Проаналізувати логи | Виконано | Події класифіковано за групами, сформовано таблиці і висновки. |
| Підготувати матеріали для захисту | Виконано | Є схеми, таблиці, фрагменти логів і висновки для презентації. |

У таблиці 3.9 показано, що практична частина відповідає заявленому технічному завданню. У роботі не передбачено розроблення власного програмного коду, оскільки основна мета полягає у побудові й дослідженні Deception-підсистеми на базі наявних інструментів. Такий підхід є доцільним для теми, бо фокус переноситься з програмування на мережну архітектуру, налаштування, спостереження та аналіз атак. Саме ці результати потрібно винести на захист.

Таблиця 3.10

Практичні рекомендації за результатами дослідження

| Напрям | Рекомендація | Очікуваний ефект |
|---|---|---|
| Віддалений доступ | Не залишати адміністративні сервіси на типових портах без контролю, використовувати обмеження доступу і сильну автентифікацію. | Зменшення кількості випадкових і автоматизованих спроб входу. |
| Журналювання | Збирати, переглядати і зберігати журнали автентифікації та взаємодії з сервісами. | Підвищення здатності виявляти підозрілі дії. |
| Deception | Розміщувати приманки в контрольованих сегментах без реальних секретів. | Отримання даних про поведінку порушника без ризику для робочих вузлів. |
| Аналіз подій | Групувати події за сесіями і категоріями, а не розглядати кожний рядок ізольовано. | Формування змістовних висновків, придатних для реагування. |
| Презентація результатів | Показувати схему мережі, фрагменти логів, приклади команд і таблицю висновків. | Зрозуміле пояснення практичної цінності роботи на захисті. |

**Перевірка виконання практичних цілей**

Після внесення практичного блоку робота містить не тільки теоретичний опис, а й послідовність налаштування стенду, скріншоти VirtualBox, перевірку вузлів, запуск Cowrie, сценарії тестової активності, журнали подій та підсумкову класифікацію результатів. Це безпосередньо підтверджує виконання пунктів 3.1 і 3.2 плану, а також пов’язує сценарії тестування з отриманими журналами.

## Висновки до розділу 3

У третьому розділі було описано практичну реалізацію Deception-підсистеми у вигляді ізольованого віртуального стенду. Розглянуто порядок створення віртуальних машин, призначення адрес, перевірки мережної зв'язності, розподіл ролей між штатними вузлами та вузлом-приманкою. Показано, що для дослідження достатньо компактної моделі, у якій Attack VM створює контрольовані події, Windows- і Linux-хости формують умовне середовище мережної інфраструктури, а Honeypot VM виконує функцію приманки та джерела журналів безпекових подій.

Було розглянуто налаштування honeypot і підготовку журналювання. Основним інструментом визначено Cowrie, оскільки він відповідає сценаріям SSH/Telnet та дозволяє фіксувати brute-force-спроби, події автентифікації та shell-взаємодію. Окремо обґрунтовано необхідність нормалізації журналів і їх подальшої класифікації для аналізу дій порушника.

У межах дослідження виконано контрольовану імітацію атак, що включала мережне сканування, спроби підбору облікових даних та взаємодію з сервісами приманки. Отримані журнали дозволили виділити типові сценарії поведінки порушника, класифікувати події за категоріями та оцінити інформативність зібраних даних.

Результати експерименту показали, що практична цінність розробленого рішення полягає не лише у використанні окремого honeypot, а у функціонуванні цілісної Deception-підсистеми, яка поєднує механізми ізоляції, приманювання, журналювання та аналізу подій у межах контрольованого середовища. Такий підхід дозволяє безпечно отримувати інформацію про дії потенційного порушника, досліджувати характер атак та формувати рекомендації щодо підвищення рівня видимості подій у мережній інфраструктурі.

Оцінка ефективності показала, що Deception-підсистема є корисним засобом раннього виявлення та дослідження небажаної активності. Найбільшу кількість контрольованих подій було зафіксовано SSH honeypot, що підтверджує привабливість типових сервісів віддаленого доступу для потенційних порушників. Практичні рекомендації зводяться до ізоляції приманок, відсутності реальних секретів, регулярного аналізу журналів та використання Deception-підсистеми як доповнення до інших засобів кіберзахисту.

## ВИСНОВКИ

У кваліфікаційній роботі було вирішено завдання проєктування Deception-підсистеми для мережної інфраструктури організації. Метою роботи було створення та дослідження лабораторного середовища, у якому за допомогою віртуальних машин і технологій deception можна безпечно імітувати дії порушника, фіксувати журнали атак, класифікувати події та формувати висновки з точки зору кібербезпеки. Отримані результати підтверджують, що Deception-підхід може бути корисним не лише як засіб приманювання потенційного порушника, а й як джерело аналітичної інформації про характер та послідовність його дій.

У першому розділі було розглянуто теоретичні основи Deception-технологій, honeypot-систем і типових атак на віддалені сервіси. Показано, що традиційні засоби захисту не завжди дають змогу детально дослідити поведінку порушника після спроби доступу. Honeypot створює контрольовану точку взаємодії, де підозріла активність може бути зафіксована без залучення реальних виробничих систем. Також було обґрунтовано актуальність аналізу SSH/Telnet-взаємодії, brute-force-спроб та журналів сесій.

У другому розділі було розроблено архітектуру лабораторного стенду Deception-підсистеми. До його складу включено Attack VM, Windows-хост, Linux-хост, Honeypot VM та ізольований мережний сегмент. Запропоновано схему адресації, логіку розміщення сервісів, порядок журналювання та правила безпечного проведення експериментів. Обґрунтовано вибір Cowrie як основного honeypot-рішення, що забезпечує реєстрацію спроб автентифікації та подальшої shell-взаємодії.

У третьому розділі виконано практичну реалізацію та дослідження спроєктованої Deception-підсистеми. Проведено розгортання віртуального середовища, налаштування вузлів, підготовку журналювання та серію контрольованих сценаріїв атак. У результаті отримано журнали подій, які дозволили класифікувати дії порушника, визначити характер взаємодії з приманкою та оцінити інформативність отриманих даних для задач моніторингу й аналізу безпекових подій.

Практичний результат роботи полягає у створенні відтворюваної моделі Deception-підсистеми, яка об'єднує механізми приманювання, ізоляції, журналювання та аналізу подій у межах контрольованого середовища. Honeypot у цій архітектурі виступає центральним елементом збору інформації, проте ефективність рішення забезпечується саме взаємодією всіх компонентів підсистеми: вузла-приманки, робочих вузлів, ізольованого мережного сегмента та засобів аналізу журналів.

За результатами дослідження встановлено, що найбільш привабливими для контрольованих сценаріїв стали типові сервіси SSH і Telnet на Honeypot VM. Саме вони забезпечили найбільший обсяг подій для подальшого аналізу. Найбільш цінною інформацією виявилися дані про дії, виконані після умовного входу до системи, оскільки вони дозволяють оцінити наміри порушника та характер його подальшої активності.

Розроблена Deception-підсистема має обмеження, пов'язані з лабораторним масштабом, спрощеною адресацією та контрольованим характером атак. Проте ці обмеження не знижують її навчальної та дослідницької цінності. Навпаки, вони забезпечують безпечність експериментів, відтворюваність результатів та можливість використання створеного стенду для демонстрації принципів функціонування Deception-технологій.

Отже, поставлену мету роботи досягнуто, а всі визначені завдання виконано в повному обсязі. Отримані результати підтверджують доцільність використання Deception-підходу як додаткового рівня виявлення та дослідження підозрілої активності в мережній інфраструктурі організації.

## СПИСОК ВИКОРИСТАНИХ


1. Про основні засади забезпечення кібербезпеки України : Закон України від 05.10.2017 № 2163-VIII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2163-19 (дата звернення: 30.05.2026).

2. Про захист інформації в інформаційно-комунікаційних системах : Закон України від 05.07.1994 № 80/94-ВР // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/80/94-%D0%B2%D1%80 (дата звернення: 30.05.2026).

3. Про прийняття національних стандартів, зміни до національного стандарту та скасування національних стандартів : Наказ ДП «УкрНДНЦ» від 17.08.2023 № 210 // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/v0210774-23 (дата звернення: 30.05.2026).

4. The NIST Cybersecurity Framework (CSF) 2.0. National Institute of Standards and Technology. 2024. URL: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf (дата звернення: 30.05.2026).

5. Scarfone K., Souppaya M., Cody A., Orebaugh A. Technical Guide to Information Security Testing and Assessment : NIST Special Publication 800-115. Gaithersburg : National Institute of Standards and Technology, 2008. URL: https://csrc.nist.gov/pubs/sp/800/115/final (дата звернення: 30.05.2026).

6. CIS Critical Security Control 8: Audit Log Management. Center for Internet Security. URL: https://www.cisecurity.org/controls/audit-log-management (дата звернення: 30.05.2026).

7. Brute Force, Technique T1110 - Enterprise // MITRE ATT&CK. URL: https://attack.mitre.org/techniques/T1110/ (дата звернення: 30.05.2026).

8. Remote Services: SSH, Sub-technique T1021.004 - Enterprise // MITRE ATT&CK. URL: https://attack.mitre.org/techniques/T1021/004/ (дата звернення: 30.05.2026).

9. Valid Accounts, Technique T1078 - Enterprise // MITRE ATT&CK. URL: https://attack.mitre.org/techniques/T1078/ (дата звернення: 30.05.2026).

10. Cowrie Documentation. Cowrie SSH/Telnet Honeypot. URL: https://docs.cowrie.org/en/latest/README.html (дата звернення: 30.05.2026).

11. Cowrie : GitHub repository. URL: https://github.com/cowrie/cowrie (дата звернення: 30.05.2026).

12. OpenCanary Documentation. URL: https://opencanary.readthedocs.io/ (дата звернення: 30.05.2026).

13. Dionaea - catches bugs : GitHub repository. URL: https://github.com/DinoTools/dionaea (дата звернення: 30.05.2026).

14. Oracle VM VirtualBox User Guide for Release 7.2. Oracle. URL: https://www.virtualbox.org/manual/ (дата звернення: 30.05.2026).

15. Configuring Host-Only Networking // VMware Workstation Pro Documentation / Broadcom. URL: https://techdocs.broadcom.com/us/en/vmware-cis/desktop-hypervisors/workstation-pro/17-0/using-vmware-workstation-pro/configuring-network-connections/configuring-host-only-networking.html (дата звернення: 30.05.2026).

16. OpenSSH server // Ubuntu Server documentation. URL: https://ubuntu.com/server/docs/how-to/security/openssh-server/ (дата звернення: 30.05.2026).

17. OpenSSH Manual Pages. URL: https://www.openssh.com/manual.html (дата звернення: 30.05.2026).

18. Kali Linux Metapackages // Kali Linux Documentation. URL: https://www.kali.org/docs/general-use/metapackages/ (дата звернення: 30.05.2026).

19. OWASP Top Ten Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 30.05.2026).

20. OWASP Web Security Testing Guide. URL: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення: 30.05.2026).

21. Elastic Common Schema (ECS) Reference. Elastic. URL: https://www.elastic.co/docs/reference/ecs (дата звернення: 30.05.2026).

22. Sigma Rules // Sigma Detection Format. URL: https://sigmahq.io/docs/basics/rules.html (дата звернення: 30.05.2026).

23. Gerhards R. The Syslog Protocol : RFC 5424. Internet Engineering Task Force, 2009. URL: https://www.rfc-editor.org/rfc/rfc5424 (дата звернення: 30.05.2026).

24. Setianto F., Tsani E., Sadiq F., Domalis G., Kostakos P. GPT-2C: A GPT-2 parser for Cowrie honeypot logs. 2021. URL: https://arxiv.org/abs/2109.06595 (дата звернення: 30.05.2026)

25. ENISA Threat Landscape 2025. European Union Agency for Cybersecurity (ENISA). URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2025 (дата звернення: 30.05.2026).

# ДОДАТКИ

## Додаток А. Таблиця адресації вузлів експериментального середовища

Таблиця А.1 – Адресація вузлів мережі SAO-NAT

| **Вузол** | **IP-адреса** | **Призначення** |
|---|---|---|
| Attack-VM (Kali Linux) | 172.22.11.10 | Імітація дій зловмисника |
| WS-1-SAO | 172.22.11.11 | Робоча станція користувача |
| Serv-1-SAO | 172.22.11.12 | Сервер мережевих сервісів |
| HoneyPot-SAO | 172.22.11.13 | Honeypot-система Cowrie |
| Шлюз SAO-NAT | 172.22.11.1 | Доступ до мережі |

Усі віртуальні машини були об'єднані в ізольовану мережу SAO-NAT з адресним простором 172.22.11.0/24.