Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра _____Безпеки інформаційних технологій_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Застосування технології розпізнавання мови для удосконалення систем_____
тестування на проникнення (Penetration Testing Tools) _____
(тема)

Виконав:
студент 2 курсу, групи _____БІКСм-20-1_____
_____Фатєєв О.С._____
(прізвище, ініціали)
Спеціальність _____125 Кібербезпека_____
(код і повна назва спеціальності)
Освітня програма__«Безпека інформаційних_
і комунікаційних систем»_____
(повна назва освітньої програми)
Керівник _____доц. Власов А. В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _____Халімов Г.З.__
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Безпеки інформаційних технологій
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 125 Кібербезпека

(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна, або освітньо-наукова)

Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри: _____ Халімов Г.З.
(підпис)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Фатєєву Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Застосування технології розпізнавання мови для удосконалення систем тестування на проникнення (Penetration Testing Tools)
затверджена наказом по університету від 08.11.2021 р. № 1685 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2021 р.

3. Вихідні дані до роботи
Процеси проведення тестування на проникнення, інструментальні засоби для проведення тестування, проблемні питання щодо застосування різних програмних засобів для проведення
тестування

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз процесів тестування на проникнення; програмні засоби та інструменти для проведення тестування на проникнення; способи програмної реалізації системи розпізнавання мови; створення адаптивного інтерфейсу для опрацювання вихідних даних програмних засобів для проведення тестування на проникнення

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) презентаційний матеріал у вигляді слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір здобувачем теми кваліфікаційної роботи	02.09.2021	Виконано
2	Затвердження плану і завдання кваліфікаційної роботи	09.11.2021	Виконано
3	Аналіз завдання, пошук та аналіз літературних джерел за темою роботи	10.11.2021-18.11.2021	Виконано
4	Виконання кваліфікаційної роботи	19.11.2021-30.11.2021	Виконано
5	Оформлення пояснювальної записки	01.12.2021-12.12.2021	Виконано
6	Здача на перевірку та підпис кваліфікаційної роботи керівнику	10.12.2021	
7	Проходження перевірки на плагіат та нормоконтроль кваліфікаційної роботи	10.12.2021	
8	Допуск завідувачем кафедри до захисту кваліфікаційної роботи	13.12.2021	
9	Захист кваліфікаційної роботи	14.12.2021	

Дата видачі завдання _____ _____20__ р.

Студент _____
(підпис)

Керівник роботи _____ _____доцент Власов А. В.__
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 95 с., 49 рис., 3 табл., 13 джерел.

ПРОТОКОЛ, СЕРВЕР, ШЛЮЗ, ТЕСТУВАННЯ НА ПРОНИКНЕННЯ, РОЗПІЗНАВАННЯ МОВИ, ВРАЗИВІСТЬ, REVERSE SHELL, HTTP, FTP, SMT, SSH, PENTESTING, MALWARE, FIREWALL, CTF, LINUX, WINDOWS, SMB, TCP, UDP, NMAP

Об’єкт дослідження – процеси та системи тестування на проникнення.

Предмет дослідження – методи розпізнавання мови для вдосконалення систем тестування на проникнення (для обробки вхідних і вихідних даних).

Мета роботи – визначення та обґрунтування впровадження в системи тестування на проникнення методів розпізнавання мови для підвищення швидкості процесів тестування.

Методи дослідження – аналіз напрямків удосконалення взаємодії з інструментами для тестування на проникнення, моделювання засобів для застосування розпізнавання мовної інформації для взаємодії з інструментами для тестування на проникнення.

ABSTRACT

Master’s thesis 95 pages, 49 figures, 3 tables, 13 sources.

PROTOCOL, SERVER, GATEWAY, PENETRATION TESING, SPEECH RECOGNITION, VULNERABILITY, REVERSE SHELL, HTTP, FTP, SMT, SSH, PENTESTING, MALWARE, FIREWALL, CTF, LINUX, WINDOWS, SMB, TCP, UDP, NMAP

The object of research is penetration testing processes and systems.

Subject of research - is language recognition methods for improving penetration testing systems (for processing input and output data).

The purpose of the work is to determine and substantiate the introduction of language recognition testing methods into testing systems to increase the speed of penetration testing processes.

Research methods - analysis of ways to improve the interaction with tools for penetration testing, modeling tools for the applying speech recognition for interaction with tools for penetration testing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IT – інформаційна технологія

API – Application Programming Interface, інтерфейс створення додатків

DNS – Domain Name System, система доменних імен

URL – Uniform Resource Locator, система уніфікованих адрес

VPN – Virtual Private Network, віртуальна приватна мережа

FTP – File Transfer Protocol, протокол передачі даних

SMTP – Simple Mail Transfer Protocol, протокол передачі пошти

POP3 – Протокол отримання електронної пошти з серверу

SMB – Server Message Block, мережевий протокол обміну файлами

TCP – Transmission Control Procotol, протокол контролю передачі

UDP – User Datagram Protocol, протокол передачі даних

SYN – Повідомлення, яке надсилає клієнт у момент підключення до серверу

IMAP – Продвинутий протокол отримання електронної пошти з серверу

CTF – Capture The Flag, змагання з вирішення задач зі знаходження ключів у сфері інформаційної безпеки

DDoS – Denial of Service, відмова в обслуговуванні

SSH (Secure Shell) – мережевий протокол прикладного рівня, що дозволяє забезпечувати віддалене керування операційною системою та тунелювання TCP-з'єднань.

ВСТУП

За останні декілька років важливість інформаційної безпеки як основи для забезпечення захисту життєво важливих інтересів людини і громадянина, суспільства та держави істотно збільшилася внаслідок стрімкого підвищення попиту, розробки та всебічного використання засобів електронних комунікацій в різноманітних сферах: державному управлінні, освіті, медицині, електронній комерції, соціальному житті суспільства так і окремої людини. Суттєво збільшились обсяги та інтенсивність використання різних інформаційних ресурсів і як наслідок збільшилася потреба в забезпеченні всіх категорій інформаційної безпеки цих ресурсів.

Визначення проблемних питань щодо організації інформаційної безпеки певного ресурсу та подальших шляхів їх усунення досягається шляхом реалізації різних процесів тестування спеціалістами з інформаційної безпеки з обов’язковим етапом тестування на проникнення (в разі використання інформаційно-комунікаційних технологій). Зазвичай кожен тест на проникнення передбачає дуже велику кількість етапів, починаючи від аналізу обсягу роботи і закінчуючи формуванням висновків про поточні вразливості. Під час проведення кожного етапу тестування на проникнення, спеціаліст, або група спеціалістів за інформаційної безпеки повинні бути дуже сфокусовані на процесі тестування, його організації та не звертати увагу, не відволікатися на менш важливі деталі. Таким чином, коли спеціаліст формує та визначає новий шлях для аналізу (обходу) системи захисту, наприклад міжмережевого екрану, він відволікається на те, щоб ввести команду для запуску в командній стрічці певного інструменту для проведення конкретно заданого сканування. В такі моменти дуже легко загубити фокус та через це не помітити важливу деталь чи особливість.

Розробка системи розпізнавання мови для запуску інструментів командної стрічки повинна вирішити цю проблему, адже при проведенні

тестування експерт зазвичай паралельно переглядає лог файли (файл журналу) декількох інструментів. При цьому застосовуючи голосову команду або їх набір є можливим запустити інший інструмент та не відволікаючись на набір команди здійснити поточний аналіз етапу тестування.

Впровадження такої системи розпізнавання мови допоможе не тільки поліпшити фокус на виконанні певних задач, а й підвищить швидкість у проведенні тестування на проникнення, дозволить реалізувати більший набір комбінацій етапів та тестів за визначений часовий проміжок.

1 ПОНЯТТЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ ТА ЙОГО ТИПИ

1.1 Що таке тестування на проникнення

Тест на проникнення, — це спроба метод оцінки безпеки IT-інфраструктури шляхом безпечного використання вразливостей. Ці вразливості можуть існувати в операційних системах, службах і програмному коді, неправильних конфігураціях або ризиках, які виникають внаслідок поведінки та дій кінцевого користувача. Такі оцінки також корисні для перевірки ефективності захисних механізмів, а також дотримання кінцевим користувачем визначеної політики безпеки.

Тестування на проникнення зазвичай виконується з використанням ручних або автоматизованих технологій для систематичної компрометації серверів, кінцевих точок, веб-додатків, бездротових мереж, мережевих пристроїв, мобільних пристроїв та інших потенційних точок впливу. Після того, як вразливості були успішно використані в певній системі, фахівці з тестування можуть спробувати використати зламану систему для запуску наступних експлойтів на інших внутрішніх ресурсах, зокрема, намагаючись поступово досягти більш високих рівнів безпеки та глибшого доступу до електронних активів та інформації через підвищення привілеїв

Інформація про будь-які вразливості безпеки, успішно використані за допомогою тестування на проникнення, зазвичай об’єднується та представляється менеджерам з безпеки, щоб допомогти цим фахівцям зробити стратегічні висновки та визначити пріоритетність відповідних заходів із усунення. Основна мета тестування на проникнення полягає в тому, щоб оцінити придатність систем до безпечного функціонування, визначити будь-які пов’язані з цими вразливостями наслідки, визначити найбільш вразливі компоненти та забезпечити виправлення цих

вразливостей.

Сканери вразливостей - це автоматизовані інструменти, які досліджують середовище, а після завершення створюють звіт про виявлені вразливості. Ці сканери часто перераховують ці вразливості за допомогою ідентифікаторів загальновідомих вразливостей інформаційної безпеки CVE (Common Vulnerabilities and Exposures), які надають інформацію про відомі слабкі місця. Сканери можуть виявити тисячі вразливостей, тому кількість вразливостей, що потребують подальшого аналізу може бути достатньо великою. Крім того, ці оцінки не враховують обставини кожного окремого IT-середовища. Для цього в першу чергу і застовують тестування на проникнення [1].

Хоча сканування вразливостей дає цінну картину наявних потенційних недоліків безпеки, тести на проникнення можуть додати додатковий контекст, перевіривши, чи можна використати вразливості для отримання доступу до вашого середовища. Тести на проникнення також можуть допомогти визначити пріоритети планів відновлення на основі того, що становить найбільший ризик.

1.2 Тестування веб-додатку

1.2.1 Поняття тестування веб-додатку

Тестування на проникнення веб-додатку – це процес, за допомогою якого експерти з кібербезпеки моделюють реальну кібератаку на веб-програми, веб-сайти або веб-сервіси для виявлення ймовірних загроз.

Це робиться для того, щоб визначити поточні вразливості, які можуть бути легко використані кіберзлочинцями. Всередині організації веб-сервери, доступні локально або в хмарі, піддаються високому ризику потенційної атаки з боку шкідливих джерел.

За допомогою тестування на проникнення експерти з кібербезпеки

проводять серію змодельованих атак, які відтворюють фактичні несанкціоновані кібератаки, перевіряють масштаби вразливості та виявляють лазівки та ефективність загальної системи безпеки додатків в організації.

1.2.2 Фаза аналізу обсягу роботи

Важливо визначити обсяг роботи, цілі фірми та цілі фірми щодо безпеки. На цьому етапі спеціаліст з інформаційної безпеки також визначає віртуальні та фізичні активи, які використовує фірма. Спеціалісту з інформаційної безпеки потрібно протестувати систему через такі типи тестування, як тестування Blackbox, тестування "білого ящика", тестування "сірого ящика".

1.2.3 Фаза збору інформації

Цей крок є критичним для аналізу налаштування веб-додатку. Цей збір розвідувальної інформації в першу чергу поділяється на два типи:

- пасивна фаза;
- активна фаза.

На пасивній фазі тестер збирає інформацію, яка легко доступна в Інтернеті, не виконуючи взаємодію безпосередньо з програмою. Наприклад, за допомогою синтаксису Google («site:*.domain.com») для визначення субдоменів програми або за допомогою машини Wayback для перевірки архівних версій веб-сайту.

Активна фаза полягає в тому, що тестер проникнення досліджує цільові системи, щоб отримати корисну інформацію, яка може бути використана в подальшому аналізі. Наприклад, аналіз програмних засобів, що використовуються у веб-додатку для визначення версії використовуваної технології, виконання пошуку DNS, ініціювання з'явлення сторінок помилок, вивчення вихідного коду тощо.

1.2.4 Фаза проведення аналізу та виявлення вразливостей

Зрозумівши критичні контрольні точки в системі, тестер ручки може потім детально вивчити ймовірні вектори атаки.

Це включає в себе сканування цільової програми на наявність уразливостей за допомогою таких сканерів, як Zed Attack Proxy (ZAP)/ Burp suite pro або Acunetix, щоб зрозуміти, як програма реагує на різні спроби вторгнення, і визначити лазівки в безпеці.

Мета ручного тестера тут полягає в тому, щоб визначити, чи загрожує життєво важлива інформація компанії.

Дані, зібрані за допомогою різних вичерпних процесів, тепер можна проаналізувати. Важливо перевіряти невідповідності, піклуючись про дані та визначаючи загрози.

Цей крок включає виконання різних методів експлуатації проти вразливостей, виявлених під час фази сканування, таких як ін’єкція SQL для отримання несанкціонованого доступу до бази даних, використання інструментів грубої сили для обходу авторизації, завантаження шкідливих сценаріїв на сервер додатків для отримання доступу до оболонки командного рядка.

1.3 Тестування мобільного додатку

1.3.1 Фаза аналізу обсягу роботи

Тестування на проникнення мобільного додатку – це процес, за допомогою якого експерти з кібербезпеки моделюють реальну кібератаку на мобільний додаток для виявлення ймовірних загроз.

Після запуску проекту від клієнта збирається інформація про обсяг/ціль. У разі тестування на проникнення мобільних додатків ця інформація включатиме двійкові файли програми (.ipa та/або .apk), будь-які застосовні IP-адреси та URL-адреси для серверів API в області, облікові дані

для автентифікації (2 набори облікових даних для кожна роль, що тестується), і список будь-яких конфіденційних або обмежених частин програми, які не слід сканувати чи використовувати [7].

1.3.2 Фаза збору інформації

Після офіційного початку тестування клієнту надсилається сповіщення про початок. Перший етап включає збір розвідувальної інформації з відкритим кодом, який включає огляд загальнодоступної інформації та ресурсів. Метою цього етапу є виявлення будь-якої конфіденційної інформації, яка може допомогти під час наступних етапів тестування, яка може включати адреси електронної пошти, імена користувачів, інформацію про програмне забезпечення, посібники користувача, повідомлення на форумі тощо. Крім того, цей крок включатиме пошук конфіденційної інформації які не повинні бути загальнодоступними, наприклад внутрішні комунікації, інформація про зарплату або інша потенційно шкідлива інформація. Нарешті, для кожного з наданих двійкових файлів програми код декомпілюється і шукатиме конфіденційну чи корисну інформацію, таку як коментарі або жорстко закодовані значення [7].

Для цієї фази моделювання загроз потребується для оцінки типів загроз, які можуть вплинути на цілі, що знаходяться в межах. Типи атак і ймовірність реалізації цих загроз слугуватимуть для визначення рейтингів/пріоритетів ризиків, які призначаються вразливостям під час оцінки. Також визначається перспектива тестування (зовнішнє, внутрішнє, з автентифікацією, без автентифікації тощо), щоб переконатися в достовірності виявлених вразливостей. Ця фаза також повинна включати ручне виявлення та сканування мобільних додатків, декомпіляцію бінарних файлів для розуміння функціональності, визначення бізнес-функціональності та нормального використання як з точки зору автентифікації, так і без аутентифікації, а також розуміння будь-яких відкритих кінцевих точок пов’язаних API. Також буде використано використання проксі-сервера

програми для оцінки трафіку на рівні пакетів і заголовків відповідей [7].

1.3.3 Фаза проведення аналізу та виявлення вразливостей

Етап аналізу вразливості охоплює перерахування всіх цілей/додатків, що входять до сфери дії, як на рівні мережі, так і на рівні прикладних програм. На мережевому рівні можна запустити сканування портів, аналіз банерів і сканування вразливостей, щоб оцінити поверхню атаки всіх активів, які знаходяться в області дії. На прикладному рівні, починаючи з етапу не автентифікації, а потім переходячи до кожної ролі автентифікації (в області дії користувачів) буде запущено автоматичне сканування вразливостей. Для кожного із мобільних бінарних файлів, що входять до сфери дії, буде проводитися як статичний, так і динамічний аналіз, щоб виявити проблеми компіляції, непотрібні дозволи, неправильне локальне зберігання даних, жорстко закодовану інформацію тощо. Ручна ідентифікація вразливостей, пов’язаних із поданням форми та точки введення програми, буде здійснюватися вручну, включаючи ін’єкційні атаки (SQL, команда, XPath, LDAP, XXE, XSS), аналіз помилок, завантаження файлів тощо [7].

1.4 Тестування бездротової мережі

1.4.1 Поняття тестування бездротової мережі

Термін Wi-Fi відноситься до технології бездротової мережі, яка використовує радіохвилі для встановлення бездротових мережевих з'єднань. Через природу Wi-Fi та його методів надання доступу до мережі зловмисники часто вирішують проникнути в систему, зламавши її мережу Wi-Fi та відповідні пристрої інфраструктури. Будинки також знаходяться під загрозою, особливо через зростання кількості пристроїв і приладів, підключених до IoT.

Тестування на проникнення бездротового зв’язку складається з шести

основних етапів, включаючи розвідку, виявлення бездротових мереж, дослідження вразливостей, експлуатацію, звітування та усунення. Ці тести проводяться в основному для підтримки безпечної розробки програмного коду протягом усього його життєвого циклу. Помилки в кодуванні, особливі вимоги або відсутність знань про вектори кібератак є основною метою виконання цього типу тесту на проникнення. Етапи проведення тестування на проникнення WiFi зображені на рисунку 1.1.

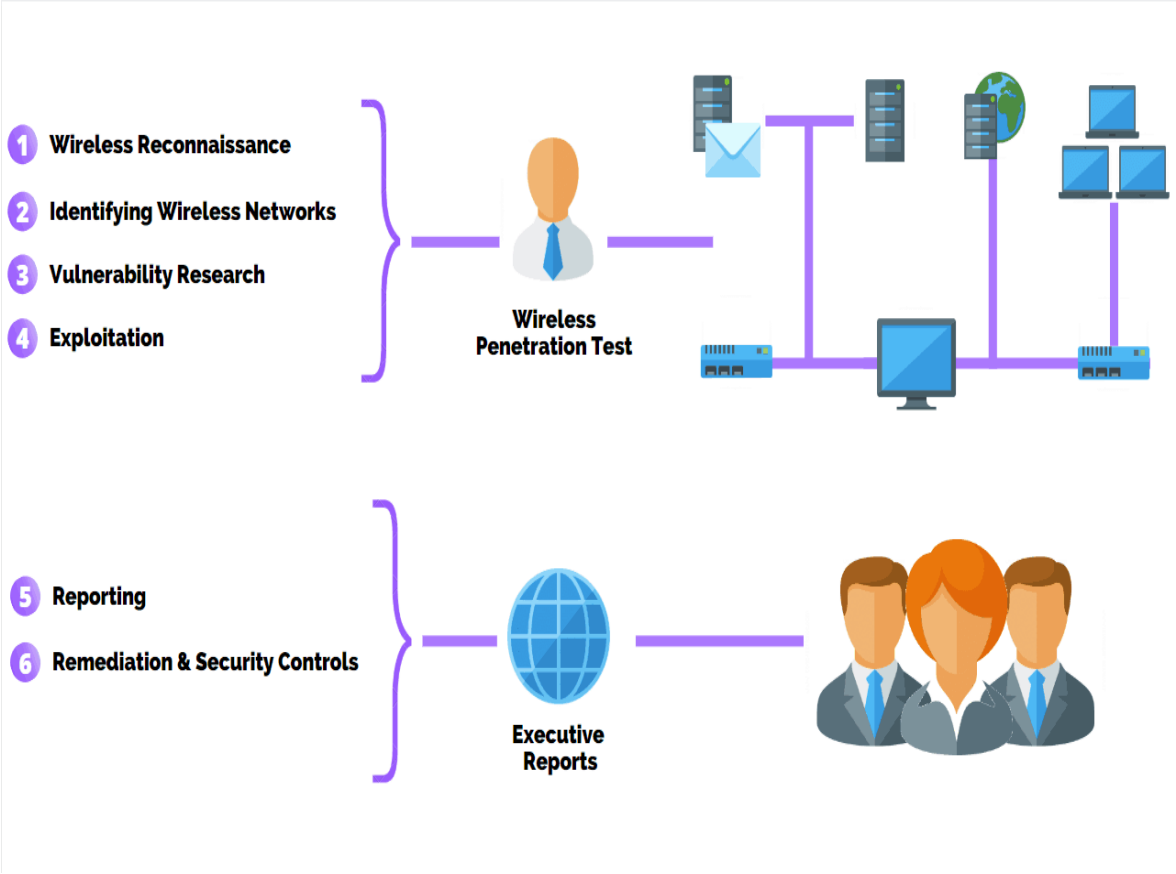


Рисунок 1.1 – Основні етапи проведення тестування на проникнення Wi-Fi

Тестування на проникнення бездротового зв’язку передбачає виявлення та перевірку з’єднань між усіма пристроями, підключеними до Wi-Fi компанії. Ці пристрої включають ноутбуки, планшети, смартфони та будь-які інші пристрої Інтернету речей (IoT).

Тести на проникнення бездротового зв’язку зазвичай проводяться на стороні клієнта, оскільки для того, щоб отримати доступ до нього, спеціаліст

з інформаційної безпеки повинен бути в зоні дії бездротового сигналу.

1.4.2 Фаза збору інформації

Перш ніж перейти безпосередньо до проведення тестування на проникнення, першим кроком у кожному процесі тестування на проникнення є фаза збору інформації. Через природу Wi-Fi інформація, збирається за допомогою WarDriving[13]. Це метод збору інформації, який включає об'їзд приміщення для виявлення сигналів Wi-Fi. Для цього необхідно мати наступне обладнання:

- автомобіль або будь-який інший транспортний засіб;
- ноутбук і Wi-Fi антена;
- адаптер бездротової мережі;
- програмне забезпечення для захоплення та аналізу пакетів.

Більшість зібраної інформації є корисною, але зашифрованою, оскільки більшість, якщо не всі компанії використовують найновіший протокол Wi-Fi: WPA2. Цей протокол Wi-Fi захищає точку доступу за допомогою шифрування та використовує автентифікацію EAPOL.

1.4.3 Фаза виявлення назв бездротових мереж

Наступним кроком у тестуванні на проникнення Wi-Fi є сканування або ідентифікація бездротових мереж. Перед цим етапом необхідно встановити бездротову карту в режимі «монітор», щоб увімкнути захоплення пакетів і вказати свій інтерфейс WLAN. Приклад переведння карти в режим монітору зображено на рисунку 1.2[13].

```
airmon-ng start wlan0
```

Рисунок 1.2 – Переведення карти в режим монітору

Після того, бездротова карта почне прослуховувати бездротовий трафік, необхідно почати процес сканування за допомогою airodump, щоб сканувати трафік на різних каналах. Приклад команди для початку сканування наведено на рисунку 1.3.

```
airodump-ng wlan0mon
```

Рисунок 1.3 – Початок сканування мережевого трафіку

Важливим кроком у зменшенні робочого навантаження під час процесу сканування є змусити airodump захоплювати трафік лише на певному каналі, приклад команди для виконання цієї дії зображено на рисунку 1.4.

```
airodump-ng -c 11 --bssid 00:01:02:03:04:05 -w dump wlan0mon
```

Рисунок 1.4 – Захоплення трафіку лише з певного каналу

1.4.4 Фаза виявлення вразливостей

Після пошуку точок доступу Wi-Fi шляхом сканування наступний етап тесту буде зосереджений на виявленні вразливостей у цій точці доступу. Найпоширенішою вразливістю є процес 4-стороннього рукостискання, коли зашифрований ключ обмінюється між точкою доступу Wi-Fi і клієнтом аутентифікації [13].

Коли користувач намагається здійснити автентифікацію на точці доступу Wi-Fi, генерується та передається попередній спільний ключ.

Під час передачі ключа зловмисний хакер може винюхувати ключ і грубим примусом перевести його в автономний режим, щоб спробувати отримати пароль[13].

Приведемо приклад застосування інструменту пакету Airplay NG для

виконання експлуатації шляхом:

- деавтентифікація законого клієнта;
- фіксація початкового 4-стороннього рукостискання, коли законний клієнт повторно підключається;
- запуск офлайн-атаки словника, щоб зламати захоплений ключ;
- деавтентифікація законного клієнту [13].

Для фіксування 4-стороннього рукостискання, яке відбувається, коли кожен клієнт аутентифікується в точці доступу, необхідно скасувати автентифікацію законного клієнта, який уже підключений, приклад виконання цієї дії зображено на рисунку 1.5.

```
aireplay-ng --deauth 5 -a 00:01:02:03:04:05 -c 00:04:05:06:07:08 wlan0mon
```

Рисунок 1.5 – Деаунтифікація користувача

1.4.5 Фаза експлуатації

Зйомка першого рукостискання. Під час процесу захоплення трафіку після надісланих пакетів «де-аутентифікації» можна побачити багато живої інформації щодо запущеної атаки «деавторизації» [13].

```
CH  2 ][ Elapsed: 1 min ]

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

DE:EF:CA:CA:65:AF    0  54     1034       11    3   2  54e  WPA2 CCMP   PSK  AmIRootYet

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

DE:EF:CA:CA:65:AF  10:A5:D0:EB:99:E6    0    0e- 1   1469     2596
```

Рисунок 1.6 – Продовження атаки після відключення справжнього користувача

Під час зйомки початкового рукостискання можна побачити номер каналу, час, що минув, BSSID (MAC-адреса), кількість маяків та багато іншої інформації.

Час, необхідний для успішного виконання цього, залежить від відстані між хакером, точкою доступу та клієнтом, якого потрібно відключити.

Після запису 4-стороннього рукостискання можна зберегти його у файлі «.cap» [13]. Зберігаючи весь цей захоплений трафік у файл «.cap», можна швидко відкрити файл у Wireshark – популярний інструмент аналізатора мережевих протоколів, щоб підтвердити, дійсно всі 4 етапи рукостискання були зафіксовані. Після цього перехоплення пакетів припиняється.

Останній крок на етапі експлуатації — зламати захоплений чотиристоронній ключ рукостискання та витягувати пароль. Для цього навіть не потрібно використовувати додаткові інструменти для злому паролів, такі як JohntheRipper або Hydra. Можна просто використовувати модуль Aircrack-ng пакета aireplay-ng[13]. На рисунку 1.7 можна бачити приклад зламування паролів за допомогою aircrack-ng.

```
aircrack-ng -b 00:01:02:03:04:05 dump-01.cap
```

Рисунок 1.7 – Приклад зламування паролів за допомогою aircrack-ng

1.4.6 Приклади інших атак

Інші практичні атаки на бездротові мережі включають розгортання шахрайської точки доступу всередині компанії. Ця атака використовує використання несанкціонованої точки доступу Wi-Fi, розгорнутої всередині будівель компанії [13].

Основна ідея полягає в тому, щоб подолати сигнали законної точки доступу в мережі компанії (або використовувати глушники сигналу Wi-Fi,

щоб зробити авторизовану точку доступу недоступною) і змусити співробітників підключитися до несанкціонованої точки доступу. Якщо це пройде успішно, зловмисник матиме контроль над усім трафіком, який проходить через цю точку доступу.

1.5 Найбільш популярні типи вразливостей

Оскільки цінність даних зростає, роль кібербезпеки в підтримці операцій підприємства значно зросла. Для того, щоб організації могли успішно розвивати нові ділові відносини, вони повинні мати можливість захистити дані клієнтів і співробітників від порушення. Досягнення такого рівня безпеки вимагає всебічного розуміння вразливостей кібербезпеки та методів, які учасники загроз використовують для отримання доступу до мережі.

Можливість ефективно керувати вразливими місцями не тільки покращує програми безпеки, але й допомагає обмежити вплив успішних атак. Саме тому наявність встановленої системи управління вразливістю стало необхідністю для організацій у різних галузях. Нижче наведений аналіз поширених типів вразливостей кібербезпеки, а також надано рекомендації щодо виявлення вразливостей і керування ними у ваших системах.

Вразливість кібербезпеки — це будь-яка слабкість інформаційних систем, внутрішнього контролю або системних процесів організації, які можуть використовуватися кіберзлочинцями. Через точки вразливості кіберзловмисники можуть отримати доступ до вашої системи та збирати дані. Що стосується загальної системи безпеки вашої організації, уразливості кібербезпеки надзвичайно важливо контролювати, оскільки прогалини в мережі можуть призвести до повномасштабного порушення системи [5].

Вразливості відрізняються від кіберзагроз тим, що вони не впроваджуються в систему, вони присутні з самого початку. Дуже рідко вразливості виникають в результаті дій кіберзлочинців, натомість вони

зазвичай викликані недоліками архітектури та побудови (операційних систем, інших пристроїв, їх програмним забезпеченням, неправильними налаштуваннями мережі). І навпаки, кіберзагрози виникають у результаті зовнішньої події, наприклад, завантаження співробітником вірусу або атаки соціальної інженерії [5].

Розглянемо їх основні типи.

1.5.1 Неправильні налаштування системи

Неправильна конфігурація системи виникає в результаті того, що мережеві активи мають вразливі налаштування або різні засоби контролю безпеки. Поширеною тактикою, яку використовують кіберзлочинці, є перевірка мереж на неправильні конфігурації та прогалини, які можна використати. Оскільки все більше організацій використовують цифрові рішення, зростає ймовірність неправильної конфігурації мережі, тому важливо співпрацювати з досвідченими фахівцями з безпеки під час впровадження нових технологій.

Основними інструментами для виявлення подібних вразливостей є наступні програмні застосунки:

- nmap;
- enum4linux;
- LinEnum;
- iptables;
- Metasploit;
- Dirbuster;
- Nessus;
- Acunetix Scanner.

1.5.2 Застаріле програмне забезпечення або програмне забезпечення без виправлень

Невиправлені вразливості можуть використовуватися кіберзлочинцями для здійснення атак і крадіжки цінних даних. Подібно до неправильної конфігурації системи, кібер-зловмисники досліджують мережі, шукаючи невиправлені системи, які вони можуть зламати. Щоб обмежити цей ризик, важливо встановити графік керування виправленнями, щоб усі нові системні виправлення впроваджувалися відразу після їх випуску [3].

Основними інструментами для виявлення подібних вразливостей є наступні програмні застосунки:

- nmap (із застосуванням параметру sV);
- Metasploit;
- BeEf;
- Accuentix Scanner;
- Nessus.

1.5.3 Відсутні або слабкі облікові дані авторизації

Поширеною тактикою зловмисників є проникнення в мережу шляхом вгадування облікових даних співробітника. Важливо ознайомити співробітників з найкращими методами кібербезпеки, щоб їхні дані для входу не можна було легко використати для отримання доступу до мережі [5].

Основними інструментами для виявлення подібних вразливостей є наступні програмні застосунки:

- BurpSuite;
- Joth The Ripper;
- THC Hydra;
- AirCrack.

1.5.4 Шкідливі інсайдерські загрози

Співробітники, які мають доступ до критично важливих систем, можуть ділитися інформацією, яка дозволяє кіберзлочинцям проникнути в мережу. Інсайдерські загрози може бути важко відстежити, оскільки всі дії співробітників виглядатимуть законними, а отже, майже не викликають червоних прапорців. Щоб допомогти боротися з цими загрозами, подумайте про інвестування в рішення для контролю доступу до мережі та сегментуйте свою мережу на основі стажу та досвіду співробітників [4].

1.5.5 Відсутність шифрування даних або погане їх використання

Мережі з відсутнім або поганим шифруванням дозволяють зловмисникам перехоплювати зв’язок між системами, що призводить до порушення. Коли погано або не зашифрована інформація переривається, кібер-зловмисники можуть витягти критичну інформацію та ввести неправдиву інформацію на сервер. Це може підірвати зусилля організації щодо дотримання вимог кібербезпеки та призвести до значних штрафів з боку контролюючих органів [3].

Основними інструментами для виявлення подібних вразливостей є наступні програмні застосунки:

- AirCrack;
- Password Cracker;
- Jorh The Ripper.

1.5.6 Вразливості нульового дня

Загрози нульового дня – це специфічні вразливості програмного забезпечення, відомі зловмиснику, але ще не виявлені організацією. Це означає, що доступного виправлення немає, оскільки постачальник системи ще не повідомив про вразливість. Вони надзвичайно небезпечні, оскільки неможливо захиститися від них, поки не буде здійснено напад. Щоб

обмежити ймовірність атаки нульового дня, важливо залишатися старанним і постійно контролювати свої системи на предмет виявлення вразливостей [4].

Основними інструментами для виявлення подібних вразливостей є наступні програмні застосунки:

- Nikto2;
- Netsparker;
- Nessus;
- Acunetix Scanner.

1.5.7 Вразливості соціальної інженерія

Соціальна інженерія – це термін, який використовується для широкого спектру шкідливих дій, що здійснюються через взаємодію людей. Вразливості спрямовані на використання та впровадження маніпуляцій (з оцінкою інтересів користувачів та їх психології), щоб обманом змусити користувачів зробити помилки безпеки або поширити конфіденційну інформацію.

Атаки соціальної інженерії відбуваються в один або кілька етапів.

На рисунку 1.8 можна бачити опис етапів проведення такої атаки [3].

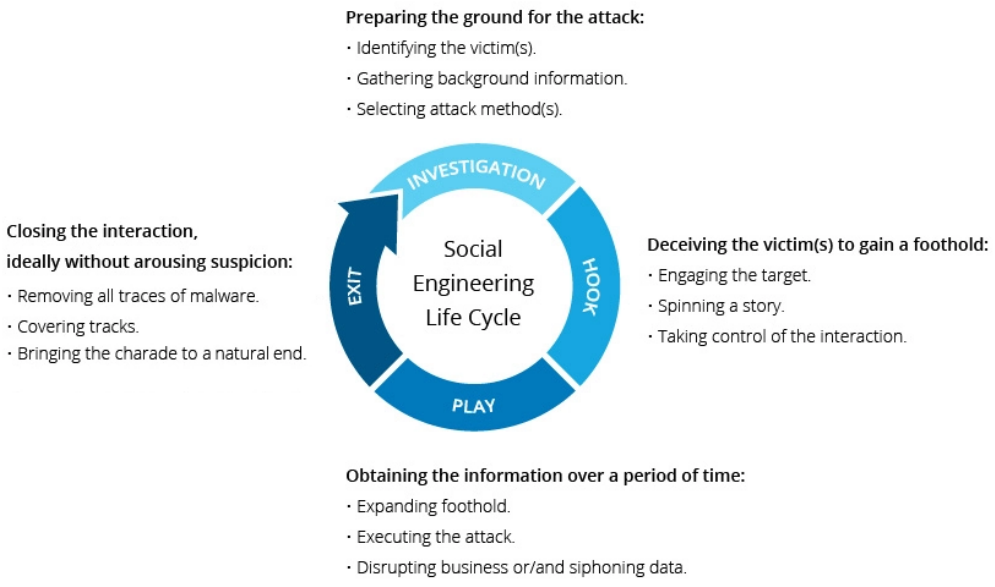


Рисунок 1.8 - Життєвий цикл атаки соціальної інженерії

Зловмисник спочатку досліджує передбачувану жертву, щоб зібрати необхідну довідкову інформацію, таку як потенційні точки входу та слабкі протоколи безпеки, необхідні для продовження атаки. Потім зловмисник намагається завоювати довіру жертви та надавати стимули для подальших дій, які порушують методи безпеки, наприклад, розкриття конфіденційної інформації або надання доступу до критичних ресурсів [3].

Особливо небезпечною соціальну інженерію вважають за те, що вона спирається на людські помилки, а не на вразливості програмного забезпечення та операційних систем. Помилки, допущені легальними користувачами, набагато менш передбачувані, тому їх важче виявити та запобігти, ніж вторгнення на основі шкідливого програмного забезпечення.

Після проведення аналізу найбільш поширених типів вразливостей можна зробити висновок - більшість типів вразливостей, які базуються в першу чергу на неправильній конфігурації системи, використанні застарілого програмного забезпечення, використанні слабких облікових даних для авторизації та вразливості нульового дня.

Для визначенні можнливих напрямків реалізації атак й використовують програмні інструменти та сканери для перевірки надійності системи (пошуку та виявлення вразливостей).

Найбільш популярними та доступними для більшості користувачів є: nmap, enum4liunx, LinEnum та Metasploit. Проведено аналіз для яких саме типів вразливостей доцільно застосовувати той або інший програмний інструмент з метою комплексного проведення тестування системи.

2 ІНСТРУМЕНТИ ДЛЯ ПРОВЕДЕННЯ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

2.1 Найбільш популярні операційні системи для проведення тестування на проникнення

2.1.1 Аналіз Kali Linux

Kali Linux заснований на дистрибутиві Debian-Linux і спеціально розроблений для проведення цифрової криміналістичної експертизи та тестування на проникнення. Він підтримується та оновлюється на регулярній основі Offensive Security Ltd під керівництвом Маті Ахароні, Девона Кернса та Рафаеля Герцога, які є основними розробниками. Kali поставляється з попередньо встановленими понад 300 програмами для тестування на проникнення і може бути встановлена як основна операційна система на жорсткому диску, live CD/USB і навіть може працювати як віртуальна машина за допомогою певного програмного забезпечення віртуалізації. Kali Linux підтримує як 32-розрядні, так і 64-розрядні зображення для використання на машинах x86, і навіть підтримує різні плати розробки, такі як Raspberry Pi, BeagleBone, Odroid, CuBox тощо. Слід зауважити, що Kali Linux не рекомендується розробниками для використання у якості повсякденної операційної системи через те, що вона не є орієнтованою на анонімність та конфіденціальність.

На рис 2.1 можна бачити категорії програмного забезпечення, що впроваджується Kali Linux.

Збір інформації: ці інструменти використовуються для збору інформації щодо DNS, IDS/IPS, мережевого сканування, операційної системи, маршрутизації, SSL, SMB, VPN, VOPI, SNMP, електронної пошти та VPN. Для оцінки вразливостей ці інструменти отримують доступ до

мережі і виявляють вразливості в кількох серверах баз даних.

Рисунок 2.1 – Категорії програмного забезпечення Kali Linux

Веб-додатки - ці інструменти пов’язані з веб-додатками, такими як системи керування вмістом, експлуатація баз даних, проксі-сервери додатків, сканери веб-вразливостей тощо [2].

Атаки паролем: ці інструменти здійснюють атаки на паролі за допомогою грубої сили, атаки за словниками тощо.

Інструменти експлуатації: вони використовують уразливості цільових систем для мереж, Інтернету та баз даних, а також для здійснення атак соціальної інженерії.

Перегляд і спуфінг - ці інструменти перевіряють мережу та веб-трафік.

Підтримка доступу: ці інструменти забезпечують доступ до цільової машини, для оцінки задніх дверей операційної системи та для тунелювання.

Інструменти звітності: вони документують процес тестування на проникнення [2].

Системні служби - ці інструменти містять різні служби тестування пера, такі як служби Apache, MySQL, SSH і Metasploit.

Основні характеристики:

- у комплекті з більш ніж 600 інструментами для тестування на проникнення порівняно з ОС Backtrack;

- повністю безкоштовний з відкритим вихідним кодом і відданий GitHub, з доступним вихідним кодом кожного пакета;

- Kali Linux відповідає FHS (стандарту ієрархії файлової системи) для пошуку двійкових файлів, а також підтримує файли та бібліотеки;

- Kali Linux пропонує спеціальне ядро для індивідуального кодування та виправлення останніх оновлень;

- кожен пакет у ньому підписаний GPG і, отже, розробниками відповідає високим стандартам якості;

- підтримка ARMEL і ARMHF дозволяє встановлювати Kali Linux на різних інших пристроях, таких як Pi, Odroid, Chromebook тощо.

2.1.2 Аналіз BackBox Linux

BackBox — це дистрибутив Linux на базі Ubuntu, спрямований на допомогу етичним хакерам і спеціалістам з інформаційної безпеки для проведення тестування на проникнення. BackBox OS розроблена з метою бути швидкою, легкою в експлуатації та мінімальним середовищем робочого столу. Ключова перевага BackBox полягає в тому, що його власні репозиторії програмного забезпечення оновлюються через регулярні проміжки часу, щоб підтримувати стабільність та популярність розповсюдження для реальних операцій.

Дистрибутив BackBox містить понад 70 інструментів для виконання завдань, починаючи від веб-аналізу та аналізу мережі до стрес-тестування, аналізу, оцінки вразливостей, криміналістичної експертизи та експлуатації.

На рис 2.2 можна бачити категорії програмного забезпечення, що впроваджується BackBox Linux [2].

Рисунок 2.2 – Категорії програмного забезпечення Backbox Linux

Основні характеристики:

- BackBox Linux є однією з перших платформ, що підтримує хмару для тестування на проникнення;

- це повністю автоматизоване та ненав’язливе, не вимагає жодних агентів або змін конфігурації мережі для регулярного автоматичного резервного копіювання конфігурації;

- економить час і усуває потребу відстеження окремих мережевих пристроїв;

- завдяки підтримці робочого столу XFCE BackBox вважається швидким у роботі і підходить навіть для старих конфігураційних систем;

повністю дружній до хакерів завдяки підтримці, яку він отримує для створення власного Launchpad PPA та надсилання пакету розробникам, які, у свою чергу, миттєво вносять свій внесок у BackBox Linux.

2.1.3 Аналіз Parrot OS

Операційна система (ОС) Parrot Security заснована на Debian

GNU/Linux у поєднанні з ОС Frozenbox і Kali Linux, щоб забезпечити етичним хакерам найкращий у своєму класі досвід проникнення та тестування безпеки в реальних середовищах. Команда Frozenbox також призначена для оцінки вразливості та зменшення її наслідків, комп'ютерної криміналістики та анонімного перегляду веб-сторінок. Parrot Security OS використовує репозиторії Kali для всіляких оновлень пакетів і для інтеграції нових інструментів. Він використовує середовище робочого столу MATE за допомогою диспетчера дисплея LightDM, щоб забезпечити простий у використанні графічний інтерфейс та полегшене середовище для аналітиків комп'ютерних систем для виконання всіх видів криміналістичної експертизи, оцінки вразливостей та криптографії.

Інтерфейс системи наведено на рисунку 2.3 [2].

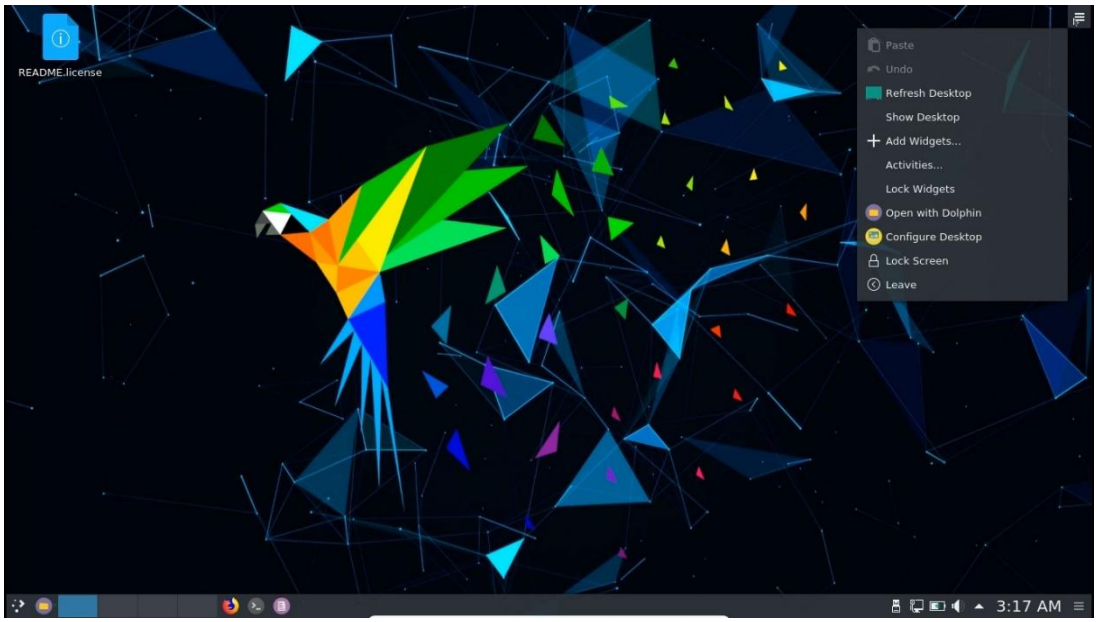


Рисунок 2.3 – Інтерфейс операційної системи Parrot OS

Основні характеристики:

- має спеціальні засоби антикриміналістичної експертизи, інтерфейси для GPG, Cryptsetup і підтримку LUKS, Truecrypt і VeraCrypt;

- підтримує мову програмування FALCON 1.0, кілька компіляторів, налагоджувачів і платформи Qt5 і .NET/mono;

- підтримує Anonsurf, включаючи анонімізацію всієї ОС, TOR, анонімних мереж I2P і не тільки;

спеціальне видання Parrot Cloud, розроблене для серверів, містить легкі дистрибутиви ОС Parrot без графічних, бездротових і криміналістичних інструментів і діє як VPS або виділений сервер із лише корисними інструментами безпеки. Щоб отримати глибокі знання про онлайн-навчання з кібербезпеки.

Таблиця 2.1 – Порівняльна характеристика проаналізованих операційних систем

	Kali Linux	Parrot OS	BackBox
Кількість встановлених інструментів	600	250	70
Вимоги до навичок користувача	Великі	Середні	Середні
Орієнтованість на конфіденційність	Відсутня	Присутня	Присутня

Аналіз операційних систем з метою використання їх на проникнення показав, що операційна система Kali Linux є найбільш вимогливою до навиків користувача, та є достатньо складною для початківців у сфері інформаційної безпеки. Також слід зауважити, що кількість інструментів для проведення тестування, що встановлені за замовчуванням на Kali Linux дуже сильно перевищує їх кількість у операційних системах Parrot OS та BackBox Linux. Через те, що Kali Linux потребує знання та навичок використання більшої кількості команд для конфігурації системи та проведення тестування на проникнення, було обрано саме цю операційну систему для тестування системи розпізнавання мови.

2.2 Найбільш популярні інструменти для проведення тестування на проникнення, що використовують командну стрічку

2.2.1 Інструмент командного рядка Nmap

Програмний інструмент командного рядка Nmap – це скорочення від Network Mapper. Це інструмент командного рядка Linux з відкритим кодом, який використовується для сканування IP-адрес і портів у мережі та виявлення встановлених програм [9].

Nmap дозволяє адміністраторам мережі знаходити, які пристрої запущені в їхній мережі, виявляти відкриті порти та служби та виявляти вразливості.

Основні можливості:

- можливість швидкого розпізнавання всіх пристроїв, включаючи сервери, маршрутизатори, комутатори, мобільні пристрої тощо в одній або кількох мережа;

- допомагає визначити служби, запущені в системі, включаючи веб-сервери, DNS-сервери та інші поширені програми. Nmap також може виявляти версії програми з достатньою точністю, щоб допомогти виявити наявні вразливості;

- Nmap може знайти інформацію про операційну систему, запущену на пристроях. Він може надавати детальну інформацію, як-от версії ОС, що полегшує планування додаткових підходів під час тестування на проникнення;

- під час аудиту безпеки та сканування цільової машини є можливість використовувати Nmap для атаки на системи за допомогою наявних скриптів із механізму сценаріїв Nmap;

- Nmap має графічний інтерфейс користувача під назвою Zenmap. Це допомагає вам розробляти візуальні відображення мережі для кращого

використання та звітності [9].

2.2.2 Приклади застосування Nmap

Сканування списку активних пристроїв у мережі є першим кроком у відображенні мережі. Для цього можна використовувати два типи сканування:

- сканування Ping – сканує список пристроїв, які працюють у певній підмережі;
- сканування одного хоста – сканує один хост на 1000 добре відомих портів. Ці порти використовуються такими популярними службами, як SQL, SNTP, apache та іншими, приклад сканування зображено на рисунку 2.4;
- приховане сканування – виконується шляхом надсилання пакету SYN та аналізу відповіді. Приховане сканування ніколи не завершує тристороннє рукостискання, що ускладнює визначення цілі системи сканування;
- сканування версій – пошук версій програми є важливою частиною тестування на проникнення. Це полегшує життя, оскільки є можливість знайти наявну вразливість у базі даних Common Vulnerabilities and Exploits (CVE) для певної версії служби, та потім використовувати його для атаки на машину за допомогою інструменту експлуатації, наприклад Metasploit[9]. Приклад такого сканування зображено на рисунку 2.5

Рисунок 2.4 – Приклад роботи сканування одного хоста

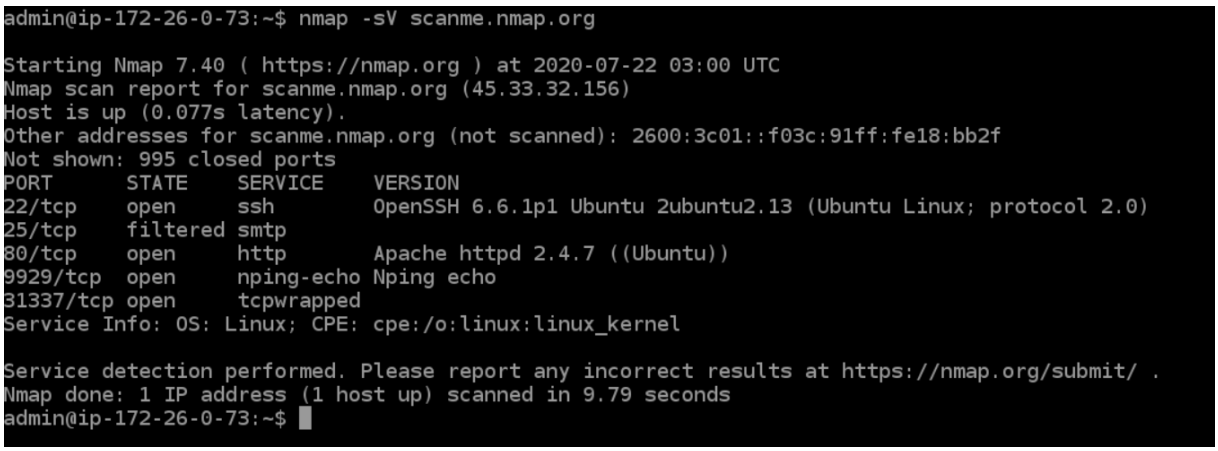


Рисунок 2.5 – Приклад роботи сканування версій

Також Nmap можна використовувати для знаходження інформації, що стосується версії ядра операційної системи, також можна використовувати агресивне сканування. Приклади додаткового функціоналу Nmap:

- сканування ОС – Nmap може надавати інформацію про базову операційну систему за допомогою відбитків пальців TCP/IP. Nmap також спробує знайти час роботи системи під час сканування ОС.

Приклад такого сканування зображено на рисунку 2.6.

- агресивне сканування – Nmap має агресивний режим, який дозволяє виявляти ОС, версії, сканувати сценарії та трасувати.

Приклад такого сканування зображено на рисунку 2.7.

- сканування кількох хостів - Nmap має можливість сканувати декілька хостів одночасно.

- сканування портів – сканування портів є однією з найважливіших функцій Nmap. Сканувати порти можна багатьма способами;

- детальність і експорт результатів сканування - тестування на проникнення може тривати кілька днів або навіть тижнів. Експорт результатів Nmap може бути корисним, щоб уникнути зайвої роботи та допомогти у створенні остаточних звітів. Давайте розглянемо кілька способів експорту результатів сканування Nmap;

- вихід у форматі XML - сканування Nmap також можна експортувати в XML. Це також бажаний формат файлу для більшості інструментів для

тестування пера, що дозволяє легко аналізувати його під час імпортування результатів сканування;

- довідка Nmap містить усі параметри та їх опис. Це часто зручно, враховуючи кількість аргументів командного рядка, з якими постачається Nmap [9].

Приклад виконання такої команди зображено на рисунку 2.8.

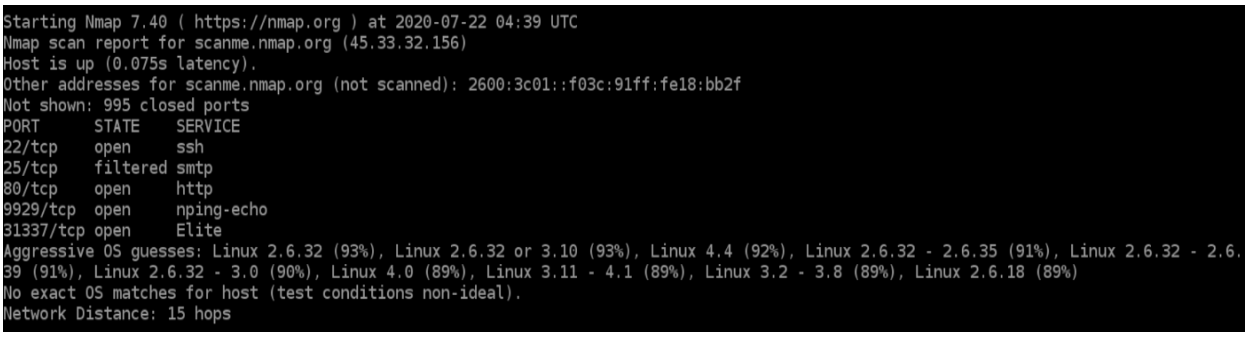


Рисунок 2.6 – Приклад роботи сканування операційної системи

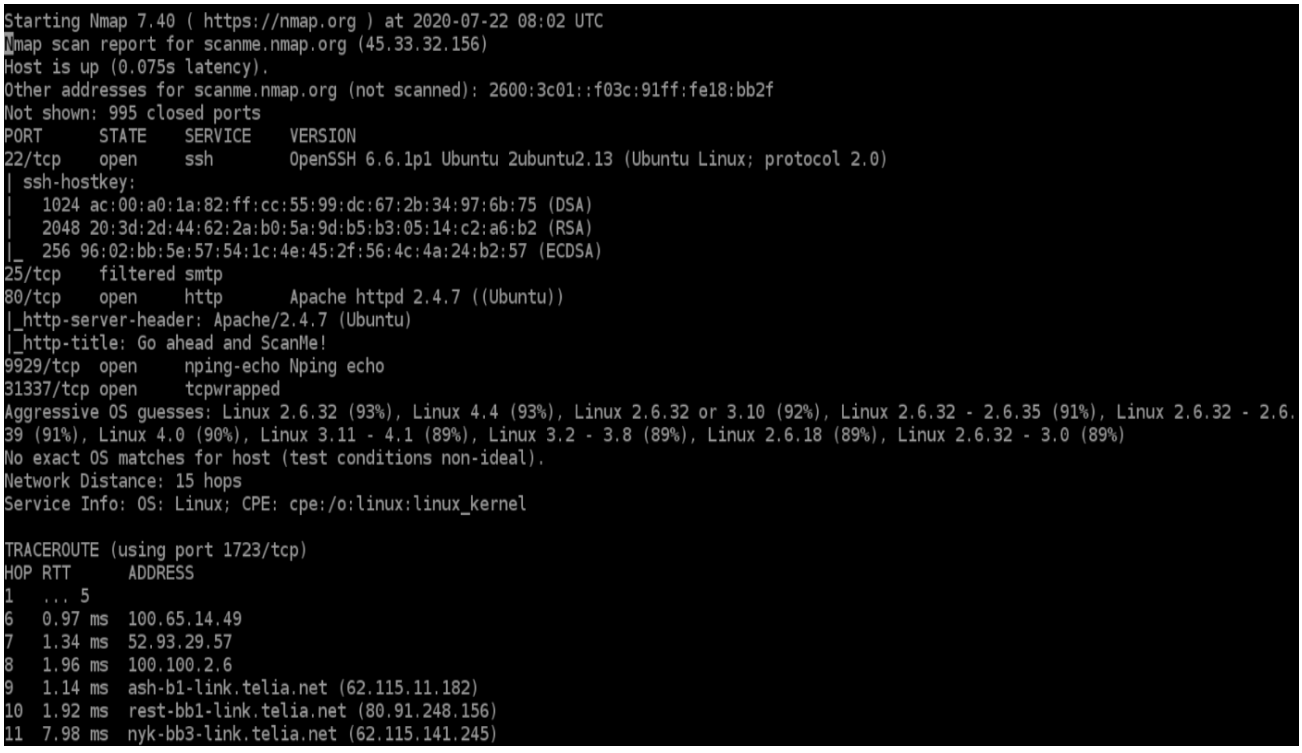


Рисунок 2.7 – Приклад роботи агресивного сканування

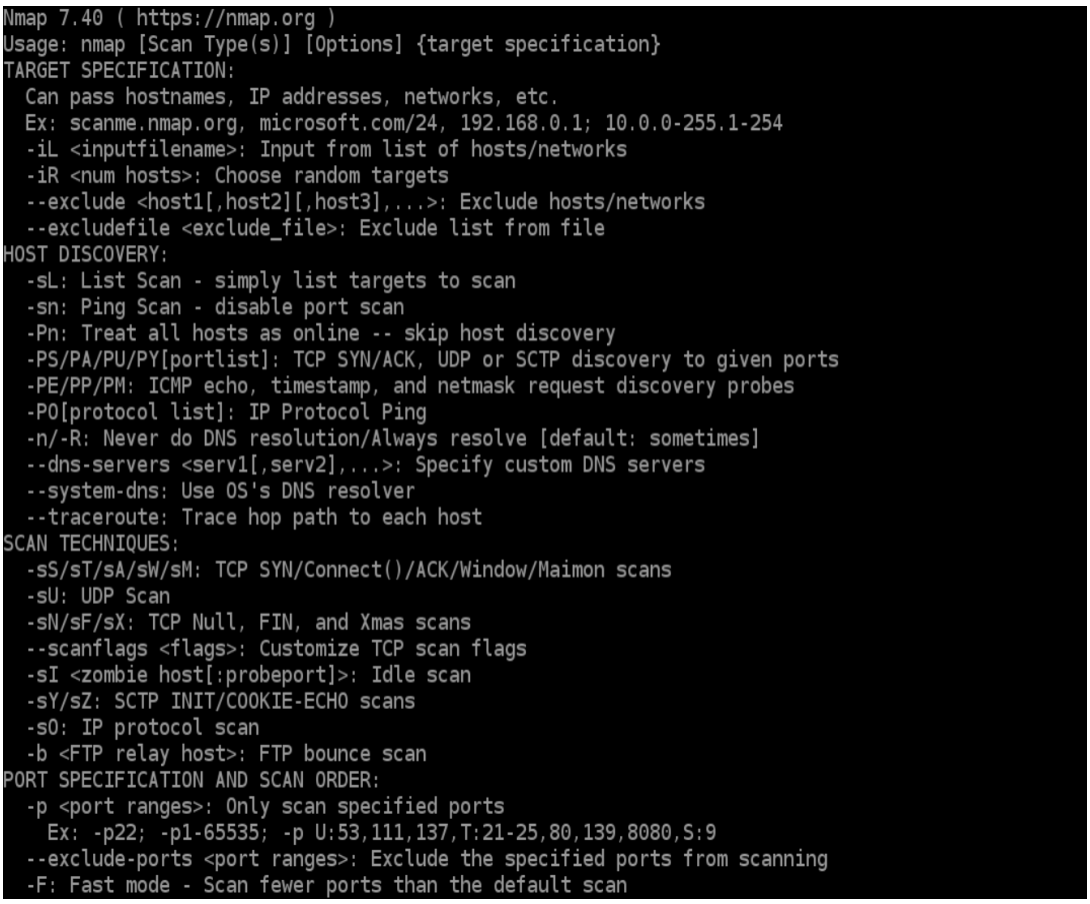


Рисунок 2.8 – Приклад роботи виводу документації

Приклади використання інструменту nmap, що зображено на прикладі 2.1 розподілемо на наступні категорії за складністю, для того щоб створити набір команд для проведення тестування системи розпізнавання мови:

- прості, мають від один або два параметри;
- середні, мають від двох до чотирьох;
- складні, мають від чотирьох параметрів.

```
nmap server2.tecmint.com
nmap -v server2.tecmint.com
nmap 192.168.0.101 192.168.0.102 192.168.0.103
nmap 192.168.0.* --exclude 192.168.0.100
nmap -v -O --osscan-guess 192.168.1.1
nmap -v -sS -A -T4 192.168.2.5
```

Приклад 2.1 – Приклади команд для застосування інструменту nmap

На прикладі 2.1 зображено шість команд, по дві команди на кожну

категорію. Таким чином легко бачити, що перші дві команди можна віднести до категорії простих команд, другі дві команди – до категоріх команд середньої складності, останні дві команди – до команд високої складності.

2.2.3 Інструмент командного рядка LinEnum

LinEnum — це простий сценарій bash, який виконує звичайні команди, пов’язані з підвищенням привілеїв, заощаджуючи час і дозволяючи докладати більше зусиль для отримання root. Це не ідеальне рішення, оскільки можуть бути помилкові спрацьовування або речі, які він пропускає, тому завжди добре перевіряти елементи вручну після запуску сценарію [10].

На рисунку 2.9 можна бачити список параметрів, які можна застосовувати для запуску цього інструменту. Опис цих параметрів:

- k, використання ключового слова;
- e, шлях до файлу для експорту результату виконання;
- t, включення додаткових тестів;
- r, вказання назви файлу висновку аналізу;
- h, відображення допоміжної інформації.

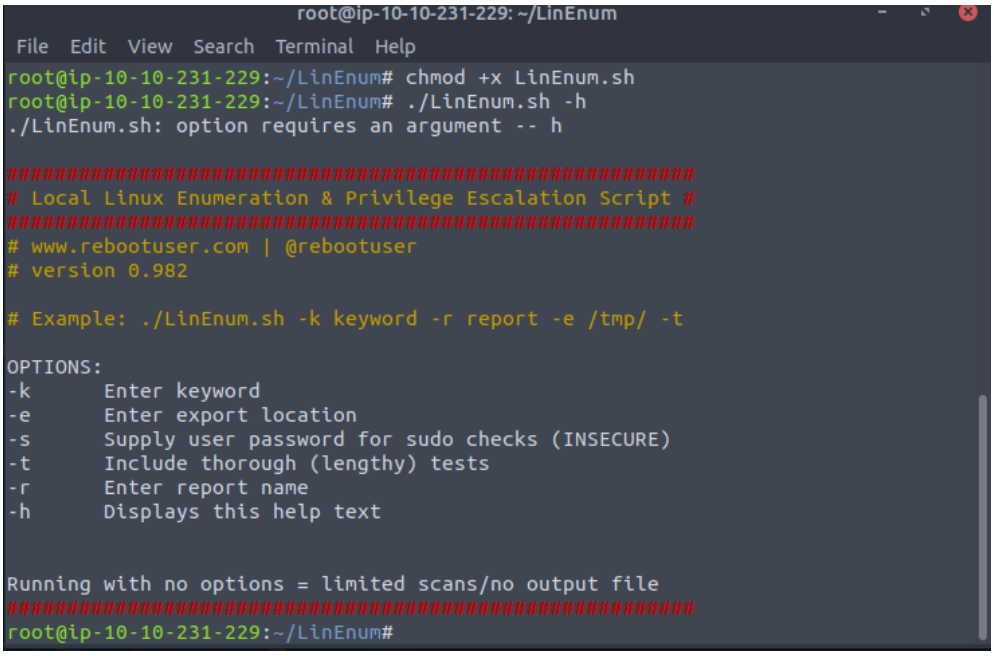


Рисунок 2.9 – Допоміжна інформація інструменту LinEnum

Приклад застосування LinEnum:

```
www-data@metasploitable:/var/tmp$ chmod +x LinEnum.sh
#############################################################
# Local Linux Enumeration & Privilege Escalation Script #
#############################################################
# www.rebootuser.com
# version 0.96

[-] Debug Info
[+] Thorough tests = Disabled

Scan started at:
Wed Aug  8 10:23:33 EDT 2018

### SYSTEM ##################################################
[-] Kernel information:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686 GNU/Linux

[-] Kernel information (continued):
Linux version 2.6.24-16-server (buildd@palmer) (gcc version 4.2.3
(Ubuntu 4.2.3-2ubuntu7)) #1 SMP Thu Apr 10 13:58:00 UTC 2008

[-] Specific release information:
DISTRIB_ID=Ubuntu
DISTRIB_DESCRIPTION="Ubuntu 8.04"
### USER/GROUP ##############################################
[-] Current user/group info:
uid=33(www-data) gid=33(www-data) groups=33(www-data)

[-] Users that have previously logged onto the system:
Username         Port     From             Latest
root              pts/0    :0.0             Wed Aug  8 09:46:22 -0400
2018
msfadmin          tty1                      Wed Aug  8 09:47:11 -0400
2018

[-] Who else is logged on:
 10:23:33 up 38 min,  2 users,  load average: 0.01, 0.02, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
msfadmin tty1     -                09:47   35:59m  0.19s  0.09s -bash
root     pts/0    :0.0             09:46   37:14m  0.04s  0.04s -bash
```

Приклад 2.2 – Застосування скрипту LinEnum

Наведемо приклади команд для запуску інструменту LinEnum, щоб створити вибірку команд для тесування системи розпізнавання мови. На прикладі 2.3 зображено набір команд різної складності. Перші дві команди віднесемо до складних команд, другу і третю віднесемо до команд середьої складності, та останні дві – до простих.

```
./LinEnum.sh -k keyword -r report -e /tmp/ -t
```

```
./LinEnum.sh -k keyword -r report -e /tmp/
./LinEnum.sh -k keyword -r report
./LinEnum.sh -k keyword -t
./LinEnum.sh -k keyword
./LinEnum.sh -t
```

Приклад 2.3 – Перелік команд для запуску інструменту LinEnum з різними параметрами

2.2.4 Інструмент командного рядка enum4linux

Enum4linux — це інструмент для перерахування інформації з систем Windows і Samba. Enum4linux написаний на PERL і в основному є обгорткою інструментів Samba smbclient, rpclient, net і nmblookup. Тому пакет samba є залежністю.

```
root@kali:~# enum4linux -h
enum4linux v0.8.9 (http://labs.portcullis.co.uk/application/enum4linux/)
Copyright (C) 2011 Mark Lowe (mrl@portcullis-security.com)

Simple wrapper around the tools in the samba package to provide similar
functionality to enum.exe (formerly from www.bindview.com). Some
additional
features such as RID cycling have also been added for convenience.

Usage: ./enum4linux.pl [options] ip

Options are (like "enum"):
    -U        get userlist
    -M        get machine list*
    -S        get sharelist
    -P        get password policy information
    -G        get group and member list
    -d        be detailed, applies to -U and -S
    -u user   specify username to use (default "")
    -p pass   specify password to use (default "")

The following options from enum.exe aren't implemented: -L, -N, -D, -f

Additional options:
    -a        Do all simple enumeration (-U -S -G -P -r -o -n -i).
              This opion is enabled if you don't provide any other
options.
    -h        Display this help message and exit
    -r        enumerate users via RID cycling
    -R range   RID ranges to enumerate (default: 500-550,1000-1050,
implies -r)
    -K n          Keep searching RIDs until n consective RIDs don't
correspond to
              a username.  Impies RID range ends at 999999. Useful
      against DCs.
    -l        Get some (limited) info via LDAP 389/TCP (for DCs only)
    -s file   brute force guessing for share names
    -k user      User(s) that exists on remote system (default:
```

```
administrator,guest,krbtgt,domain admins,root,bin,none)
                    Used to get sid with "lookupsid known_username"
                  Use commas to try several users: "-k admin,user1,user2"
          -o        Get OS information
          -i        Get printer information
          -w wrkg   Specify workgroup manually (usually found automatically)
          -n        Do an nmblookup (similar to nbtstat)
          -v         Verbose.  Shows full commands being run (net, rpcclient,
etc.)
      RID cycling should extract a list of users from Windows (or Samba) hosts
      which have RestrictAnonymous set to 1 (Windows NT and 2000), or "Network
      access: Allow anonymous SID/Name translation" enabled (XP, 2003).

      NB: Samba servers often seem to have RIDs in the range 3000-3050.

      Dependancy info: You will need to have the samba package installed as
this
      script is basically just a wrapper around rpcclient, net, nmblookup and
      smbclient.                                  Polenum                from
http://labs.portcullis.co.uk/application/polenum/
      is required to get Password Policy info
```

Приклад 2.4 – Застосування скрипту enum4linux

Наведемо приклади команд для запуску інструменту enum4linux, щоб створити вибірку команд для тесування системи розпізнавання мови. На прикладі 2.5 наведено перелік команд для запуску інструменту enum4linux з різними параметрами.

```
enum4linux -U
enum4linux -U -M
enum4linux -U -M -P
enum4linux -U -M -P -G
enum4linux -U -M -P -G -u username
enum4linux -U -M -P -G -u username –p password
```

Приклад 2.5 – Приклади команд для запуску enum4linux з різними параметрами

2.2.6 Інструмент командного рядка netstat

Команда Linux netstat надає скарбницю інформації про мережеві підключення, порти, які використовуються, і процеси, які їх використовують.

Слухаючий сокет називається сервером, а сокет, який запитує з'єднання з прослуховуючим сокетом, називається клієнтом. Ці назви не мають нічого спільного з апаратними чи комп’ютерними ролями. Вони просто визначають

роль кожного гнізда на кожному кінці з’єднання [11].

Команда netstat дозволяє дізнатися, які сокети підключені, а які прослуховуються. Це означає, що він повідомляє, які порти використовуються і які процеси їх використовують. Він може показати вам таблиці маршрутизації та статистику про ваші мережеві інтерфейси та багатоадресні з’єднання.

Приклад роботи цього інструменту зображено на рисунку 2.10.

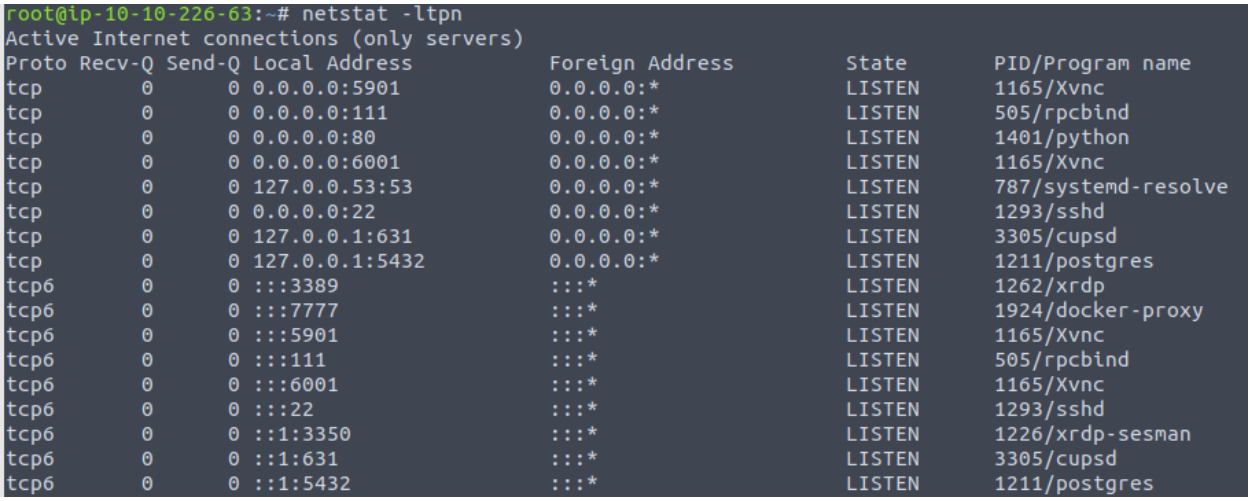


Рисунок 2.10 – Приклад роботи netstat

Після аналізу доміжної інформації інструменту netstat, що зображено на рисунку 2.11, було зроблено вибірку команд для тестування системи розпізнавання мови.

Список цих команд відображено на прикладі 2.6

```
netstat -r
netstat -r -g
netstat -l -t -p
netstat -l -t -p -n
netstat -a -l -t -p -n
netstat -a -l -t -p –fib
```

Приклад 2.6 – Приклади команд для запуску netstat з різними параметрами

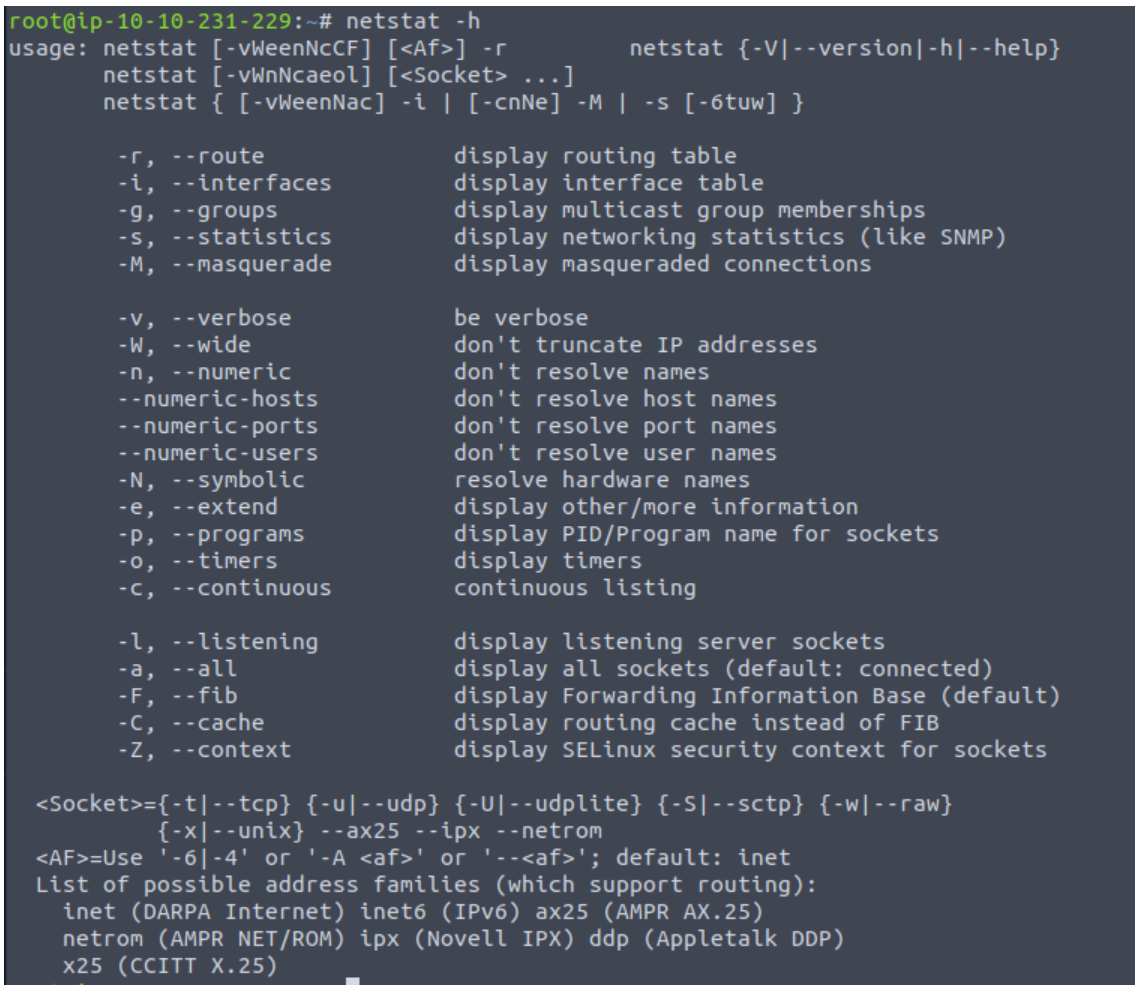


Рисунок 2.11 – Допоміжна інформація інструменту netstat

2.2.7 Інструмент командного рядка John The Ripper

John The Ripper підтримує кілька стандартних технологій шифрування для систем UNIX і Windows. (ред. Mac базується на UNIX). JtR автоматично визначає шифрування хешованих даних і порівнює його з великим текстовим файлом, який містить популярні паролі, хешуючи кожен пароль, а потім зупиняючи його, коли знаходить відповідність. Просто.

John The Ripper також включає власні списки слів із поширеними паролями для понад 20 мов. Ці списки слів надають John The Ripper тисячі можливих паролів, з яких він може генерувати відповідні значення хешування, щоб зробити припущення високого значення цільового пароля. Оскільки більшість людей вибирають паролі, які легко запам’ятовуються, John The Ripper часто дуже ефективний навіть зі своїми готовими списками

слів паролів. John The Ripper входить до Kali Linux [12].

Список технологій шифрування, доступних у John The Ripper:

- крипт UNIX(3);
- «bigcrypt»;
- BSDI розширений на основі DES;
- FreeBSD MD5 (linux та Cisco IOS);
- OpenBSD Blowfish;
- Kerberos/AFS;
- Windows LM (на базі DES);
- хеші SHA-crypt (нові версії Fedora та Ubuntu);
- хеші SHA-crypt і SUNMD5 (Solaris).

Основні режими John The Ripper для зламу паролів — це одиничний режим злому, режим списку слів і інкрементальний. Режим одиночного зламування є найшвидшим і найкращим режимом, якщо є повний файл паролів для зламу. Режим списку слів порівнює хеш із відомим списком потенційних збігів паролів. Інкрементальний режим є найпотужнішим і, можливо, не завершиться.

Найпростіший спосіб спробувати зламати пароль – це дозволити John The Ripper пройти через ряд поширених режимів злому [12].

Приклад роботи цього інструменту зображено на рисунку 2.12

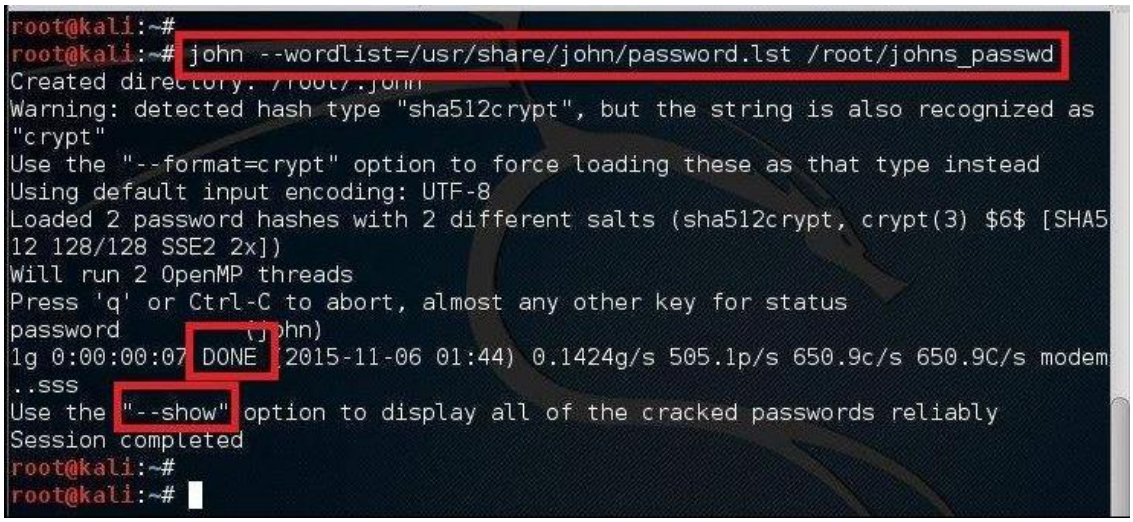


Рисунок 2.12 – Приклад роботи John the Ripper

Наведемо приклади команд для запуску інструменту John the Ripper, щоб створити вибірку команд для тесування системи розпізнавання мови.

В прикладі 2.7 наведено перелік команд для запуску інструменту John the Ripper з різними параметрами.

```
john mypasswd.txt
john --show mypasswd.txt
john --show --shells=-/etc/expired mypasswd.txt
john --show --users=0 mypasswd.txt
john --wordlist=password.txt --rules mypasswd.txt
john -w=password.txt -r -u mypasswd.txt
```

Приклад 2.7 – Приклади команд для запуску John the Ripper з різними параметрами

2.2.8 Інструмент командного рядка Metasploit

Фреймворк Metasploit є дуже потужним інструментом, який можуть використовувати кіберзлочинці, а також етичні хакери для дослідження систематичних вразливостей у мережах і серверах. Оскільки це фреймворк з відкритим вихідним кодом, його можна легко налаштувати та використовувати з більшістю операційних систем.

За допомогою Metasploit команда спеціалістів з інформаційної безпеки може використовувати готовий або налаштований код і впроваджувати його в мережу для виявлення слабких місць. Ще один вид пошуку загроз, коли недоліки виявлені та задокументовані, інформація може бути використана для усунення системних недоліків і визначення пріоритетів рішень.

Metasploit тепер включає понад 1677 експлойтів, організованих на 25 платформах, включаючи Android, PHP, Python, Java, Cisco тощо. Структура також несе майже 500 корисних навантажень, деякі з яких включають:

- корисне навантаження командної оболонки, яка дозволяє користувачам запускати сценарії або випадкові команди на хості;

- динамічні корисні навантаження, які дозволяють тестувальникам створювати унікальні корисні навантаження, щоб уникнути антивірусного програмного забезпечення;

- корисне навантаження Meterpreter, що дозволяє користувачам захоплювати монітори пристроїв за допомогою VMC і брати на себе сеанси або завантажувати та завантажувати файли;

- статичні корисні навантаження, які забезпечують переадресацію портів і зв'язок між мережами.

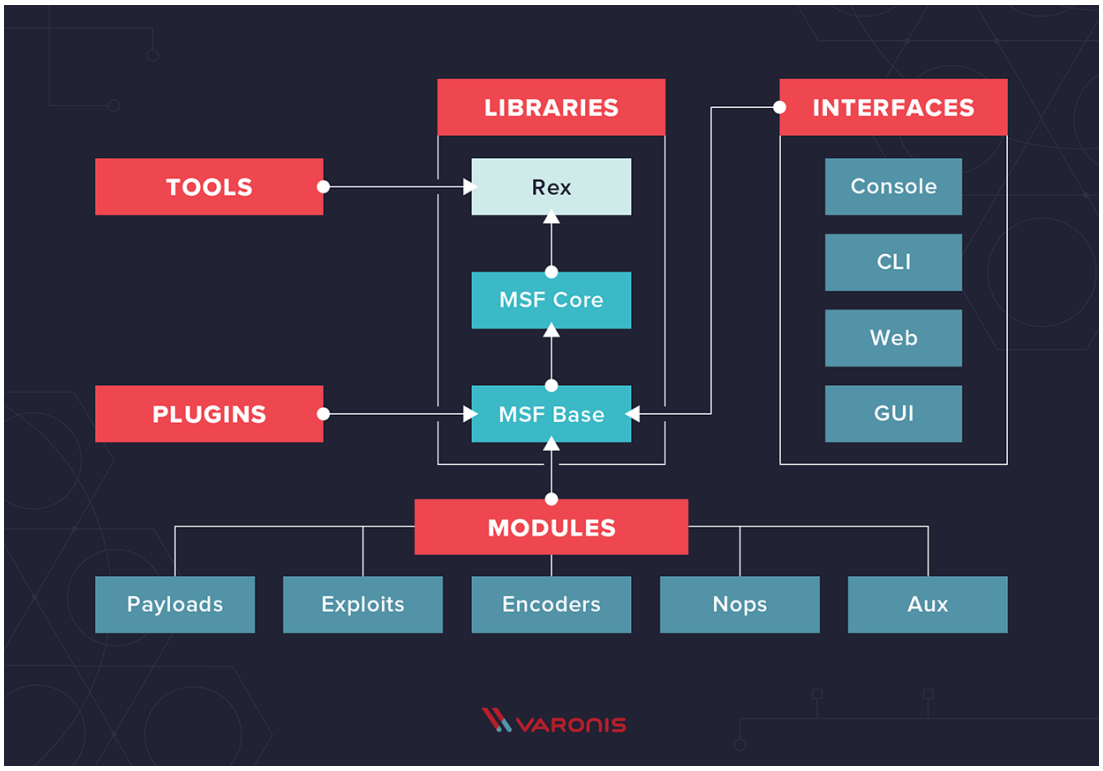


Рисунок 2.13 – Архітектура компонентів додатку Metasploit

Модулі є основними компонентами Metasploit Framework. Модуль – це частина програмного забезпечення, яка може виконувати певну дію, наприклад, сканувати або використовувати. Модулі класифікуються за типом, а потім за протоколом.

Існує кілька типів модулів, як зображено на рисунку 2.13. Тип модуля залежить від призначення модуля та типу дії, яку виконує модуль. Нижче наведено типи модулів, які доступні в Metasploit Framework:

- exploit – модуль експлойта виконує послідовність команд, спрямованих на конкретну вразливість, знайдену в системі або програмі;

- auxiliary – допоміжний модуль. До допоміжних модулів відносяться

сканери, фазери та атаки відмови в обслуговуванні;

- post-exploitation - модуль після експлуатації дозволяє вам збирати більше інформації або отримати подальший доступ до експлуатованої цільової системи;

- payload - це код оболонки, який запускається після того, як експлойт успішно скомпрометує систему.

- NOP generator– генератор NOP створює серію випадкових байтів, які можна використовувати для обходу стандартних сигнатур IDS та IPS NOP. Використовуйте генератори NOP для заповнення буферів.

сховище даних є основним компонентом Metasploit Framework. Сховище даних дозволяє інтерфейсам налаштовувати параметри, корисні навантаження для виправлення кодів операцій і експлойти для визначення параметрів. Сховище даних можна поділити на 2 типи:

- глобальне сховище даних. Щоб визначити параметр глобального сховища даних, потрібно використовувати команду setg. Усі модулі можуть використовувати опцію сховища даних;

- сховище даних модуля – щоб визначити параметр сховища даних на рівні модуля, потрібно використовувати команду set. Тільки модуль, для якого буде визначено опцію сховища даних, може використовувати його.

2.2.9 Проведення сканування за допомогою Metasploit

Discovery scan – це внутрішній сканер Metasploit. Він використовує Nmap для виконання базового сканування портів TCP і запускає додаткові модулі сканера для збору додаткової інформації про цільові хости. За замовчуванням сканування виявлення включає сканування UDP, яке надсилає проби UDP на найбільш відомі порти UDP, такі як NETBIOS, DHCP, DNS і SNMP. Discovery scan перевіряє приблизно 250 портів, які зазвичай відкриті для зовнішніх служб і частіше перевіряються під час тесту на проникнення.

Під час використання Discovery scan, Metasploit Pro автоматично додає

дані хоста до проекту. Є можиливість переглянути дані хоста, щоб краще зрозуміти топологію мережі та визначити найкращий спосіб використання кожної цілі. Часто топологія мережі дає уявлення про типи додатків і пристроїв, які має ціль.

Сканування виявлення вразливостей розділяють на чотири окремі фази:

- ping-сканування;
- сканування портів;
- визначення ОС і версії;
- імпорт даних.

Перший етап сканування виявлення (ping-сканування) визначає, чи хости знаходяться в мережі. Discovery scan встановлює параметр -PI, який повідомляє Nmap про виконання стандартної перевірки ping ICMP. Цільові надсилається один ICMP Echo-запит. Якщо є ехо-відповідь ICMP, хост вважається «підключеним» або онлайн. Якщо хост підключений до мережі, Discovery scan включає хост у сканування портів.

Під час другого етапу (сканування портів) Metasploit Pro запускає Nmap, щоб визначити порти, які відкриті, і служби доступні на цих портах. Nmap надсилає запити до різних портів і класифікує відповіді, щоб визначити поточний стан порту. Сканування охоплює широкий спектр часто відкритих портів, таких як HTTP, telnet, SSH і FTP.

Discovery scan використовує параметри Nmap за замовчуванням для визначення ОС та її версії.

На останньому етапі (імпорту даних) Metasploit Pro імпортує дані в проект після того, як Nmap збере всі дані та створить власний звіт. Metasploit Pro використовує службову інформацію для надсилання додаткових модулів, які націлені на виявлені служби, і для пошуку додаткових даних у цілі. Наприклад, якщо сканування виявлення обстежує ціль за допомогою зондів telnet, цільова система може повернути запит на вхід.

Наведемо приклади команд для запуску інструменту Metasploit, щоб створити вибірку команд для тесування системи розпізнавання мови.

На прикладі 2.8 наведено перелік команд для запуску та використання інструменту Metasploit на базовому рівні.

```
msfconsole
search module_name
use module_name
use number_of_the_module
info
set variable_name value
back
```

Приклад 2.8 – Приклади команд для запуску John the Ripper з різними параметрами

Для практичного дослідження та тестування системи розпізнавання мови створено вибірку команд для, яку зображено на прикладі 2.9.

```
nmap server2.tecmint.com
nmap -v server2.tecmint.com
nmap 192.168.0.101 192.168.0.102 192.168.0.103
nmap 192.168.0.* --exclude 192.168.0.100
nmap -v -O --osscan-guess 192.168.1.1
nmap -v -sS -A -T4 192.168.2.5
./LinEnum.sh -k keyword -r report -e /tmp/ -t
./LinEnum.sh -k keyword -r report -e /tmp/
./LinEnum.sh -k keyword -r report
./LinEnum.sh -k keyword -t
./LinEnum.sh -k keyword
./LinEnum.sh -t
netstat -r
netstat -r -g
netstat -l -t -p
netstat -l -t -p -n
netstat -a -l -t -p -n
netstat -a -l -t -p –fib
john mypasswd.txt
john --show mypasswd.txt
john --show --shells=-/etc/expired mypasswd.txt
john --show --users=0 mypasswd.txt
john --wordlist=password.txt --rules mypasswd.txt
john -w=password.txt -r -u mypasswd.txt
msfconsole
search module_name
use module_name
use number_of_the_module
info
set variable_name value
back
```

Приклад 2.9 – Вибірка команд для тестування системи розпізнавання мови

Для запуску інструментів для проведення тестування на проникнення, та взаємодії з деякими з них було проаналізовано наступні інструменти:

- Nmap;
- LinEnum;
- enum4linux;
- netstat;
- John The Ripper;
- Metasploit.

Кожен з інструментів може мати безліч комбінацій параметрів для запуску або взаємодії з ним.

Команди було поділено на умовні три рівні:

- прості;
- середьої складності;
- складні.

Розподіл команд на 3 різні рівні дає змогу протестувати вплив системи розпізнавання мови на швидкість взаємодії з інстурментами тестування, та як наслідок на оцінку впливу на швидкість проведення тестування на проникнення.

3. ОПИС ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Функціональне призначення програмного застосунку

Даний програмний застосунок створений покращити проведення тестування на проникнення шляхом скорочення часу на написання важких команд з великою кількістю параметрів, та вирішує наступні задачі:

- запуск інструментів для проведення тестування на проникнення за допомогою засобів операційної системи;
- розпізнавання мови та перетворення їх у строки;
- опрацьовування строкових даних, їх валідація;
- перетворення допоміжної інформації інструментів до файлів формату XML для подальшого використання компонентами цього програмного застосунку;
- взаємодія з користувачем за допомогою засобів графічного інтерфейсу;
- конфігурація вводу та виводу інструментів для проведення тестування голосовими командами.

Проаналізувавши наявність подібних програмних застосунків можна дійти висновку, що на момент написання цього програмного застосунку – аналогів у відкритому доступі не існує.

3.2 Програмні засоби для реалізації системи розпізнавання мови

Для реалізації цієї системи розпізнавання мови було обрано 2 мови програмування C++ та Python. C++ використовується для створення ключових компонентів програми, а саме:

- компонент для прослухування звуків, та роспізнавання мови;

- компонент для управління запуском та виключенням інструментів, якими керує система;

- компонент для керування контекстом виконання, наприклад розпізнавання команди «show» у контексті операційної системи, або у контексті користуванням інструментом Metasploit;

- компонент для взаємодії з користувачем за допомогою графічного інтерфейсу;

- компонент для керування вхідними та вихідними данними інструментів, якими керує система;

- компонент для валідації команд, які були розпізнані системою;

- компонент для тестування інших компнентів за допомогою використання Gtest.

Мова Python використовується для розробки додаткових програмних додатків, а саме:

- додаток для перетворення документації інструментів для проведення тестування на проникнення у файли формату XML, для подальшого використання системою;

- додаток для створення засобів тестування системи за домогою реалізації інтеграційних тестів.

3.3 Бібліотеки і фреймворки

3.3.1 Бібліотека для розпізнавання мови

Для розпізнавання мови була вибрана бібліотека CMU Sphinx, через наступні причини:

- можливість легко інтегрувати в любий проект;
- є кросплатформеною;
- має власний словник символів для розпізнавання;
- підтримує англійську та російську мови;

- має велику підтримку спільноти.

Приклад застосування CMU Sphinx наведено на рисунку 3.1.

```
config = cmd_ln_init(NULL, ps_args(), TRUE,
                     "-hmm", MODELDIR "/en-us/en-us",
                     "-lm", MODELDIR "/en-us/en-us.lm.bin",
                     "-dict", MODELDIR "/en-us/cmudict-en-us.dict",
                     NULL);
if (config == NULL) {
    fprintf(stderr, "Failed to create config object, see log for details\n");
    return -1;
}

ps = ps_init(config);
if (ps == NULL) {
    fprintf(stderr, "Failed to create recognizer, see log for details\n");
    return -1;
}

fh = fopen("goforward.raw", "rb");
if (fh == NULL) {
    fprintf(stderr, "Unable to open input file goforward.raw\n");
    return -1;
}

rv = ps_start_utt(ps);

while (!feof(fh)) {
    size_t nsamp;
    nsamp = fread(buf, 2, 512, fh);
    rv = ps_process_raw(ps, buf, nsamp, FALSE, FALSE);
}

rv = ps_end_utt(ps);
hyp = ps_get_hyp(ps, &score);
printf("Recognized: %s\n", hyp);

fclose(fh);
ps_free(ps);
cmd_ln_free_r(config);
```

Приклад 3.1 – Приклад інтеграції CMU Sphinx у проект

3.3.2 Програмний застосунок для опрацьовування документації інструментів командної стрічки

Застосунок для опрацьовування документації інструментів командної стрічки необхідний у випадку автоматизації опрацьовування документації нових інструментів командної стрічки, котрі є необхідність отримати. Такий застосунок має виявити чи підтримує інструмент командної стрічки параметр -h на вході, щоб відобразити усі доступні команди та їх значення.

Наведено приклад застосування такого застосунка за допомогою аналізу інструменту командної стрічки netstat.

Допоміжна інформація цього програмного застосунка наведено на рисунку 3.2.

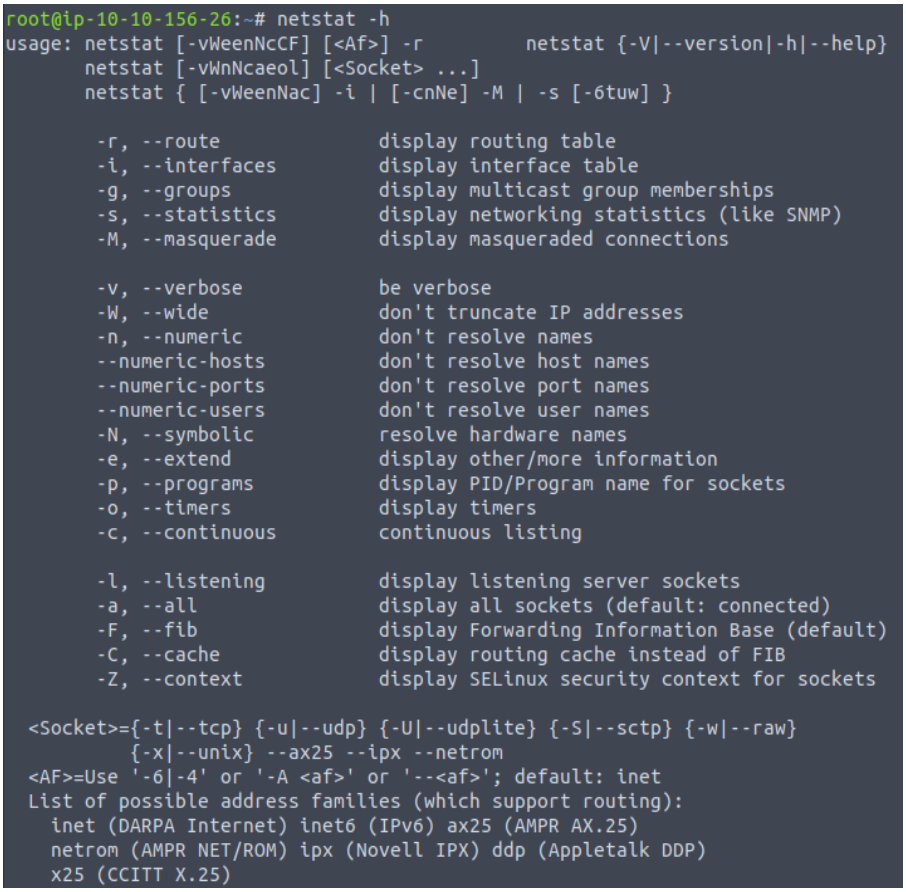


Рисунок 3.2 – Вікно допоміжної інформації інструменту командної стрічки netstat

Таким чином, найважливіша інформація (параметри та їх значення) розташовуються у таблиці. Відповідно є можливим створити (написати) скрипт для розбору цієї інформації, наприклад використовуючи мову Python та згенерувати потрібний нам XML файл за необхідними даними.

Приклад генерованого XML файлу зображено на рисунку 3.3.

Дані, що знаходяться у таких файлах використовуються для конфігурації компонентів системи, а саме:

- компонентів, що є абстракціями певних інструментів. Такі дані

використовуються для ініціалізації необхідних параметрів для запуску певного інструменту;

- компонентами, що відслідковують список інструментів, що підтримує поточна версія системи.

```
<parameter>
    <name>-r</name>
    <meaning>display routing table</meaning>
</parameter>
<parameter>
    <name>-i</name>
    <meaning>display interface table</meaning>
</parameter>
<parameter>
    <name>-g</name>
    <meaning>display multicast group memberships</meaning>
</parameter>
<parameter>
    <name>-s</name>
    <meaning>display networking statistics (like SNMP)</meaning>
</parameter>
<parameter>
    <name>-M</name>
    <meaning>display masqueraded connections</meaning>
</parameter>
<parameter>
    <name>-v</name>
    <meaning>be verbose</meaning>
</parameter>
<parameter>
    <name>-W</name>
    <meaning>don't truncate IP addresses</meaning>
</parameter>
<parameter>
    <name>-n</name>
```

Рисунок 3.3 – Результат роботи додатку з опрацьовування допоміжної інформації інструменту netstat у вигляді XML файлу

3.3.3 Бібліотека TinyXML 2 для парсингу xml файлів

TinyXML 2 аналізує XML-документ і створює з нього об’єктну модель документа (DOM), яку можна прочитати, змінити та зберегти. XML означає «розширювана мова розмітки». Це мова розмітки загального призначення для опису довільних даних. Усі ці випадкові формати файлів, створені для зберігання даних програми, можна замінити на XML. Один парсер для всього.

TinyXML 2 випускається під ліцензією ZLib, тому можна використовувати його у відкритому коді або комерційному коді. Деталі ліцензії знаходяться у верхній частині кожного вихідного файлу.

XMLDocument — це об’єкт C++, як і будь-який інший, який може бути в стеку або бути створений і видалений у купі.

TinyXML 2 повідомляє номер рядка будь-яких помилок у XML-документі, які неможливо правильно проаналізувати. Крім того, всі вузли (елементи, оголошення, текст, коментарі тощо) та атрибути мають номер рядка, записаний під час аналізу. Це дозволяє програмі, яка виконує додаткову перевірку аналізованого XML-документа (наприклад, перевірку DTD, реалізованої в програмі), повідомляти інформацію про номер рядка для повідомлень про помилку.

Приклад застосування бібліотеки TinyXML 2 наведено в прикладі 3.1

```
void write_app_settings_doc( )
{
TiXmlDocument doc;
TiXmlElement* msg;
      TiXmlDeclaration* decl = new TiXmlDeclaration( "1.0", "", "" );
doc.LinkEndChild( decl );

TiXmlElement * root = new TiXmlElement( "MyApp" );
doc.LinkEndChild( root );

TiXmlComment * comment = new TiXmlComment();
comment->SetValue(" Settings for MyApp " );
root->LinkEndChild( comment );

TiXmlElement * msgs = new TiXmlElement( "Messages" );
root->LinkEndChild( msgs );

msg = new TiXmlElement( "Welcome" );
msg->LinkEndChild( new TiXmlText( "Welcome to MyApp" ));
msgs->LinkEndChild( msg );
```

```
msg = new TiXmlElement( "Farewell" );
msg->LinkEndChild( new TiXmlText( "Thank you for using MyApp" ));
msgs->LinkEndChild( msg );

TiXmlElement * windows = new TiXmlElement( "Windows" );
root->LinkEndChild( windows );

TiXmlElement * window;
window = new TiXmlElement( "Window" );
windows->LinkEndChild( window );
window->SetAttribute("name", "MainFrame");
window->SetAttribute("x", 5);
window->SetAttribute("y", 15);
window->SetAttribute("w", 400);
window->SetAttribute("h", 250);

TiXmlElement * cxn = new TiXmlElement( "Connection" );
root->LinkEndChild( cxn );
cxn->SetAttribute("ip", "192.168.0.1");
cxn->SetDoubleAttribute("timeout", 123.456); // floating point attrib

dump_to_stdout( &doc );
doc.SaveFile( "appsettings.xml" );
}
```

Приклад 3.1 – Приклад застосування бібліотеки TinyXML

3.4 Опис найбільш важливих компонент системи

Таким чином пропонується для впровадження комплексна система для проведення тестування, що включає в себе: реалізацію розпізнавання мови, опрацьовування вхідної та вихідної інформації інструментів командної стрічки для проведення тестування на проникнення. Дана система повинна містити купу взаємопов’язаних компонентів для повної та коректної їх роботи.

3.4.1 Перетворювач мови у текст

Для реалізації розпізнавання мови у цій системі було вирішено використовувати CMU Sphinx, для розпізнавання невеликої кількості слів лише для одного користувача. Компонент що є відповідальним за розпізнавання мови, повинен встановити слухача, та кожен раз, коли слухач розпізнає якесь слово, зберігати його, та повідомляти усю систему про розпізнане слово.

Інтерфейс такого перетворювача, що використовує CMU Sphinx для розпізнавання мови зображено на рисунку 3.4.

```
/**
 * @brief Class for launching voice listener
 */
class VoiceListener
{
public:
    VoiceListener(std::mutex &mutex_,
                  std::condition_variable &conditionalVariable,
                  std::vector<std::string> &recognizedWords_);

    virtual  ~VoiceListener() = default;

public:

    virtual Result start() = 0;

protected:
    std::mutex & mutex_;
    std::condition_variable & conditionVariable_;
    std::vector<std::string> & recognizedWords_;
};

/**
 * @brief Class for launching CMUSphinx in terminal and setup listener.
 */
class CMUSphinxVoiceListener : public VoiceListener
{
public:
    CMUSphinxVoiceListener(std::mutex &mutex_,
                           std::condition_variable &conditionalVariable,
                           std::vector<std::string> &recognizedWords_);

    Result start() override;
};
```

Рисунок 3.4 – Інтерфейс слухача.

3.4.2 Перевірка підтримки розпізнаної команди

Після того, як слухачем було розпізнано будь-яку команду від користувача, необхідно перевірити, чи підтримується вона у використовуваній версії системи, чи ні. Це можна зробити за допомогою відповідного компонента, що має в собі тримати певний список команд користувача, зо зараз підтримуються. Слід зауважити, що команди можуть мати контекст операційної системи, або контекст певного інструменту

командної стрічки для проведення тестування на проникнення. Тобто користувач може бажати як запустити інструмент командної стрічки для проведення тестування на проникнення, так і будь яку команду операційної системи.

На рисунку 3.5 зображений інтерфейс такого компонента.

```
/**
 * @brief Class for validation commands
 */
class CommandValidator
{
public:
    CommandValidator(std::mutex &mutex_,
                     std::condition_variable &conditionalVariable,
                     std::vector<std::string> &recognizedWords_);

    virtual ~CommandValidator() = default;

public:

    virtual Result initializeSupportedCommandNames(const std::string & pathToToolNamesPreset) = 0;
    virtual Result validateRecognizedCommand() = 0;

private:
    std::mutex & mutex_;
    std::condition_variable & conditionVariable_;
    std::vector<std::string> & recognizedWords_;

    std::vector<std::string> supportedToolNames_;
};

/**
 * @brief Class for validation tool commands
 */
class ToolCommandValidator : public CommandValidator
{
public:
    ToolCommandValidator(std::mutex &mutex_,
                     std::condition_variable &conditionalVariable,
                     std::vector<std::string> &recognizedWords_);

public:
    Result initializeSupportedCommandNames(const std::string & pathToToolNamesPreset) override;
    Result validateRecognizedCommand() override;
};
```

Рисунок 3.5 – Інтерфейс валідатора команд запуску інструментів командної стрічки.

Метод initializeSupportedCommandNames використовується для

ініціалізації усіх команд, що підтримуються поточною версією системи.

Метод validateRecognizedCommand намагається знайти розпізнану команду у списку команд що підтримуються поточною версією системи.

3.4.3 Вікна для взаємодії з користувачем

Для взаємодії з користувачем потрібно реалізувати певний адаптер між класами вікон, що пропонує QT та цією системою. Для вирішення цієї задачі необхідно створити 2 типи вікон:

- стейт-вікна, вікна, що повідомляють користувача про завершення якоїсь дії, або просто відображають невелику кількість інформації;

- дата-вікна, вікна, що запрошують від користувача певну інформацію, параметри, необхідні для виконання певної команди.

Приклад інтерфейсу таких класів зображено на рисунку 3.6.

```
class Window
{
public:
    explicit Window(Ui::MainWindow * ui);
    ~Window() = default;

    [[nodiscard]] Ui::MainWindow * getUi() const;

protected:
    Ui::MainWindow * ui_;
};

class StatusWindow : public Window
{
public:
    enum class WindowsStatus
    {
        SUCCESS = 0,  ///< Represents the action, that was successfully done, and we want to inform user about that
        WARNING,      ///< Represents the action, that was user has to be warned about
        ERROR,        ///< Represents the action, that was failed, and we want to inform user about that
        UNDEFINED
    };
public:
    Result setStatusExplanation(const std::string & statusExplanation);
    Result setStatus(WindowsStatus currentWindowStatus);
};

class DataRequestWindow : public Window
{
public:
    Result addRadioButtonsRow(const std::string & rowDescription,
                              const std::vector<std::string> & radioButtonsRow);

    Result addInputField(const std::string & inputDescription);
};
```

Рисунок 3.6 – Приклад інтерфейсу класів вікон.

3.4.4 Вилучення параметрів від користувача

Для того щоб отримати будь-яку інформацію від користувача, система повинна мати функціонал для взаємодії з користувачем. В даному випадку засобом для взаємодії є розпізнавання мови або графічні діалогові вікна, через які користувач може вводити інформацію яка від нього потребується. Для отримання інформації від користувача в даному випадку використовуються поля для введення інформації, або радіо кнопки.

Для генерації вікон для інтерактивної взаємодії з користувачем використовуються класи WindowGenerator, StatusWindowGenerator, DataRequestWindowGenerator.

Інтерфейси цих класів зображені на рисунках 3.7, 3.8 та 3.9.

```
/**
 * Class for generating interactive windows for user
 */
class WindowGenerator
{
protected:
    WindowGenerator() = default;

    static std::unique_ptr<WindowGenerator> windowsGenerator_;

public:
    static WindowGenerator* getInstance();

public:
    virtual Result generateWindow(const Window & window) = 0;

private:
    std::vector<std::vector<std::string>> radioButtons_;
};
```

Рисунок 3.7 – Інтерфейс узагальнюючого класу генераторів вікон

```cpp
/**
 * Class for generation Error/Warning/Success popups for user
 */
class StatusWindowGenerator : public WindowGenerator
{
public:
    enum class WindowsStatus
    {
        SUCCESS = 0,  ///< Represents the action, that was successfully done, and we want to inform user about that
        WARNING,      ///< Represents the action, that was user has to be warned about
        ERROR,        ///< Represents the action, that was failed, and we want to inform user about that
        UNDEFINED
    };

public:
    Result setStatusExplanation(StatusWindow & window, const std::string & statusExplanation);
    Result setStatus(StatusWindow & window, WindowsStatus currentWindowStatus);

};
```

Рисунок 3.8 – Інтерфейс класу генерації вікон, що відображують статус

```cpp
/**
 * Class for generation popup with question and input boxes(buttons, input fields). Or sequences of such popups
 */
class DataRequestWindowGenerator : public WindowGenerator
{
public:

    Result addRadioButtonsRow(DataRequestWindow & window,
                              const std::string & rowDescription,
                              const std::vector<std::string> & radioButtonsRow);

    Result inputField(DataRequestWindow & window, const std::string & inputDescription);
};
```

Рисунок 3.9 – Інтерфейс класу генерації вікон, що відображують статус

Метод setStatusExplanation класу StatusWindowGenerator використовується для додавання інформації, що повинна бути відображена користувачу

Метод setStatus класу StatusWindowGenerator використовується для вибору статусу, від якого залежить, яка картинка буде відображена користувачеві. Приклад таких вікон відображено на рисунку 3.8.

Метод addRadioButtonsRow класу DataRequestWindowGenerator додає до вікна список радіокнопок та їх опис, що складається з небільше ніж 15 символів.

Метод inputField класу DataRequestWindowGenerator додає до вікна

поле вводу інформації та його опис. Приклад такого вікна зображений на рисунку 3.10.

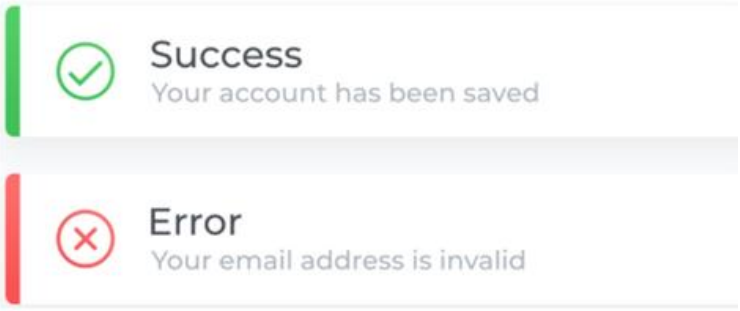


Рисунок 3.10 – Приклад вікна зі статусом проведення певної операції

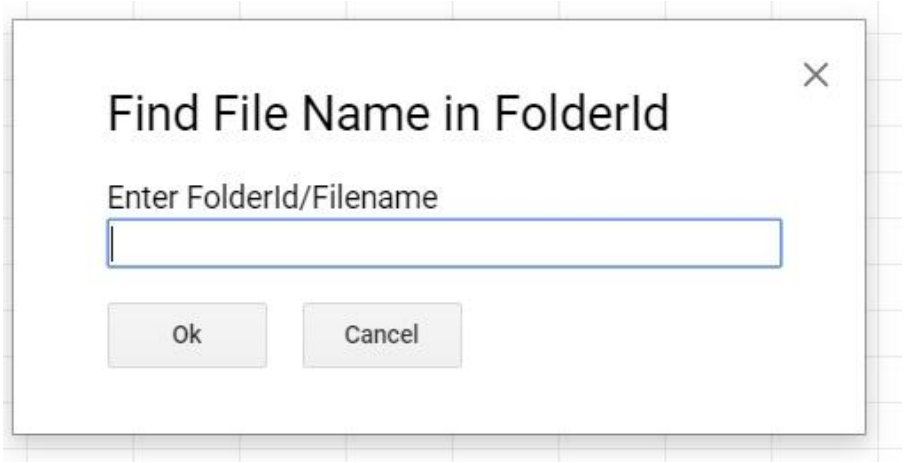


Рисунок 3.11 – Приклад вікна із полем для вводу інформації

3.4.5 Логування необхідної інформації

Для відстеження поточного стану системи, дій які виконуються, виявлення помилок тощо, необхідний компонент котрий буде записувати кожен крок що виконується в системі.

Інтерфейс класу Logging представлено на рисунку 3.12.

Перерахування LoggingLevel представляє собою розподілення логування на різні группи.

Метод getLevel призначений повертати поточний рівень логування компонента, у кого цей метод був викликаний.

```
class Logging
{
public:
    enum LoggingLevel
    {
        LOG_DISABLED,   ///< Disable all traces
        LOG_ERROR,      ///< Errors, enabled by default.
        LOG_WARNING,    ///< Warnings, enabled by default.
        LOG_INFO,       ///< Important information, enabled by default.
        LOG_DEBUG,      ///< Verbose information (high amount of information, debugging only).
        NUM_LOGS        ///< It tells us how many log levels have been defined
    };
public:

    Logging () = default;
    virtual ~Logging () = default;

    /**
     * @brief Returns the currently enabled logging level
     * @return The current logging level threshold
     */
    [[nodiscard]] LoggingLevel getLevel() const;

    void setLevel (LoggingLevel level);

    /**
     * See LoggingBD.h
     */
    void logDebug   (const char * message, ...) const;
    void logInfo    (const char * message, ...) const;
    void logWarning (const char * message, ...) const;
    void logError   (const char * message, ...) const;

};
```

Рисунок 3.12 – Інтерфейс класу для логування

Метод setLevel призначений встановити поточний рівень логування компонента, у кого цей метод був викликаний.

Метод logDebug друкує інформацію, що повинна відображатися на рівні LOG_DEBUG.

Метод logInfo друкує інформацію, що повинна відображатися на рівні LOG_INFO.

Метод logWarning друкує інформацію, що повинна відображатися на рівні LOG_WARNING.

Метод logError друкує інформацію, що повинна відображатися на рівні LOG_ERROR.

3.4.6 Запуск доступних додатків командної стрічки

Для того щоб запустити будь-який додаток командної стрічки, необхідно знати:

- назву цього додатку;
- файл, у який буде спрямовано вивід цього додатку;
- список необхідних параметрів та їх опис, для запуску.

Для збереження такого типу інформації необхідно створити абстракцію, клас що буде зберігати всі ці дані.

Приклад інтерфейсу такого класу зображено на рисунку 3.13.

```
using ApplicationParameterName = std::string;
using ApplicationParameterDescription = std::string;

class Application
{
public:
    explicit Application(const std::string & applicationName);
    ~Application() = default;

public:
    Result  start() const;
    void    stop()  const;

public:
    Result initializeRequiredParameters(std::map<ApplicationParameterName, ApplicationParameterDescription> & requiredParameters);

    void setOutputFilename(const std::string & filename);
    void setParameters(const std::string & parameters);

    std::string getApplicationName() const;
    std::string getOutputFilename() const;
    std::string getProvidedParameters() const;
    uint32_t getApplicationPID() const;
private:

    std::string applicationName_;
    std::string outputFileName_;

    std::map<ApplicationParameterName, ApplicationParameterDescription> requiredParameters_;
    std::string providedParameters_;
    std::ifstream outputFile_;

    uint32_t currentPID_;
};
```

Рисунок 3.13 – Інтерфейс класу що представляє додаток

Метод initializeRequiredParameters необхідний для зчитування параметрів та їх опису з XML файлу, за допомогою парсера TinyXML2, та подальшого збереження цієї інформації у класі.

Метод setOutputFilename необхідний для збереження назви файлу для виводу додатку.

Метод getApplicationName() повертає назву додатку.

Метод getOutputFilename() повертає назву файлу, у який буде записано вивід запущеного додатку.

Метод getApplicationPID() повертає PID процесу додатку.

Метод start() необхідний для запуску додатку, за допомогою команд операційної системи, та збереження PID процесу у відповідній змінній.

Метод stop() необхідний для завершення програми.

Для запуску, зупинення додатків командної стрічки для проведення тестування на проникнення необхідно створити клас, що матиме змогу контролювати усі запущені системою додатки.

Приклад інтерфейсу такого класу зображено на рисунку 3.14.

```
using ApplicationName = std::string;

/**
 * @brief Class for managing tools lifecycle(launch, kill)
 */
class ToolManager
{
public:
    ToolManager() = default;
    ~ToolManager() = default;
public:
    Result startApplication(std::string & applicationName);
    Result stopApplication (Application & application);
    Result stopApplication (uint32_t applicationPID);

    Result initializeSupportedApplicationNames(const std::string & pathToApplicationNamesPreset);

    Result requestApplicationParametersFromUser(Application & application);

    void listAllRunningApplications() const;
private:
    ToolOutputManager toolOutputManager_;
    ToolInputManager  toolInputManager_;

    std::map<ApplicationName, Application> runningApplications_;
    std::vector<std::string> supportedApplicationNames_;
};
```

Рисунок 3.14 – Інтерфейс класу, що керує додатками командної стрічки

Метод startApplication() необхідний для запуску додатку командної стрічки за допомогою команд операційної системи, та подальшого її додавання у змінну runningApplications_.

Метод stopApplication() необхідний для зупинення додатку командної стрічки за допомогою використання методу stop() класу Application, або за домогою зупинення процессу операційною системою за певним PID процесу.

Метод initializeSupportedApplicationNames() необхідний для ініціалізації списку інструментів командної стрічки для проведення тестування на проникнення, що підтримуються поточною версією системи.

Метод requestApplicationParametersFromUser() для генерації певної кількості вікон для запрошування необхідних даних для запуску додатку командної стрічки від користувача за допомогою класу ToolInputManager.

Метод listAllRunningApplications() для виводу назв додатків командної стрічки, що зараз запущені.

Інтерфейс такого класу зображено на рисунку 3.14.

```
/**
 * @brief Class for managing tool input parameters.
 */
class ToolInputManager
{
public:
    ToolInputManager() = default;
    ~ToolInputManager() = default;

public:
    Result requestApplicationDataFromUser(Application & application);

private:
    DataRequestWindowGenerator dataRequestWindowGenerator_;

};
```

Рисунок 3.14 – Інтерфейс класу, що керує запрошуванням інформації від користувача для додатків командної стрічки

Для управління необхідними даними для взаємодії користувача с

додатком командної стрічки для проведення тестування на проникнення необхідно створити клас, що буде здатним запрошувати необхідні дані для коректної роботи додатку за допомогою використання класів, що генерують певні вікна, наприклад DataRequestWindowGenerator.

Було проаналізовано архітектуру необхідних для коректної та стабільної роботи компонентів системи, а саме:

- валідація розпізнаних команд;
- конроль запуску та вимкнення інструментів для тесутвання;
- парсинг файлів з розширенням XML;
- логування;
- взаємодія з користувачем за допомогою графічного інтерфейсу;
- інструмент для створення конфігураційних файлів.

Визначені та проаналізовані необхідні методи кожного класу, що є необхідним для створення системи.

4. ЗАСТОСУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ МОВИ НА ПРИКЛАДІ ВИРІШЕННЯ CTF

4.1 Вибір декількох завдань для використання інструментів командної стрічки та їх аналіз

Для проведення порівняльного аналізу використання різних інструментів командної стрічки та подальшого впровадження розпізнавання мови обрано декілька типових CTF задач з їх стислим описом.

4.1.1 Tryhackme Network Servies

Однією з CTF задач для наведення прикладу застосування системи розпізнавання мови для проведення тестування на проникнення було обрано кімнату Network Services на ресурсі для вивчення технік та розвитку необхідних навичок для проведення тестування на проникнення Tryhackme.

TryHackMe – це онлайн-платформа, яка навчає кібербезпеці за допомогою коротких ігрових лабораторій у реальному світі.

Кімната Network Services була обрана з наступних причин:

- необхідно застосування інструменту командної стрічки nmap;
- необхідно застосування інструменту командної стрічки enum4linux;
- необхідно застосування інструменту командної стрічки nmap scripts.

На рисунку 4.1 можна бачити кількість етапів, які необхідно виконати для успішного завершення цієї кімнати.

Розглянемо етап Enumeration SMB детальніше. На цьому етапі необхідно запустити інструмент командної стрічки enum4linux для виявлення SMB Shares на цільовій машині. Enum4linux — це інструмент, який використовується для перерахування спільних ресурсів SMB в системах Windows і Linux. По суті, це обгортка навколо інструментів у пакеті Samba і

дозволяє легко швидко витягти інформацію з цільового призначення, що стосується SMB. Він встановлений за замовчуванням на Parrot і Kali.

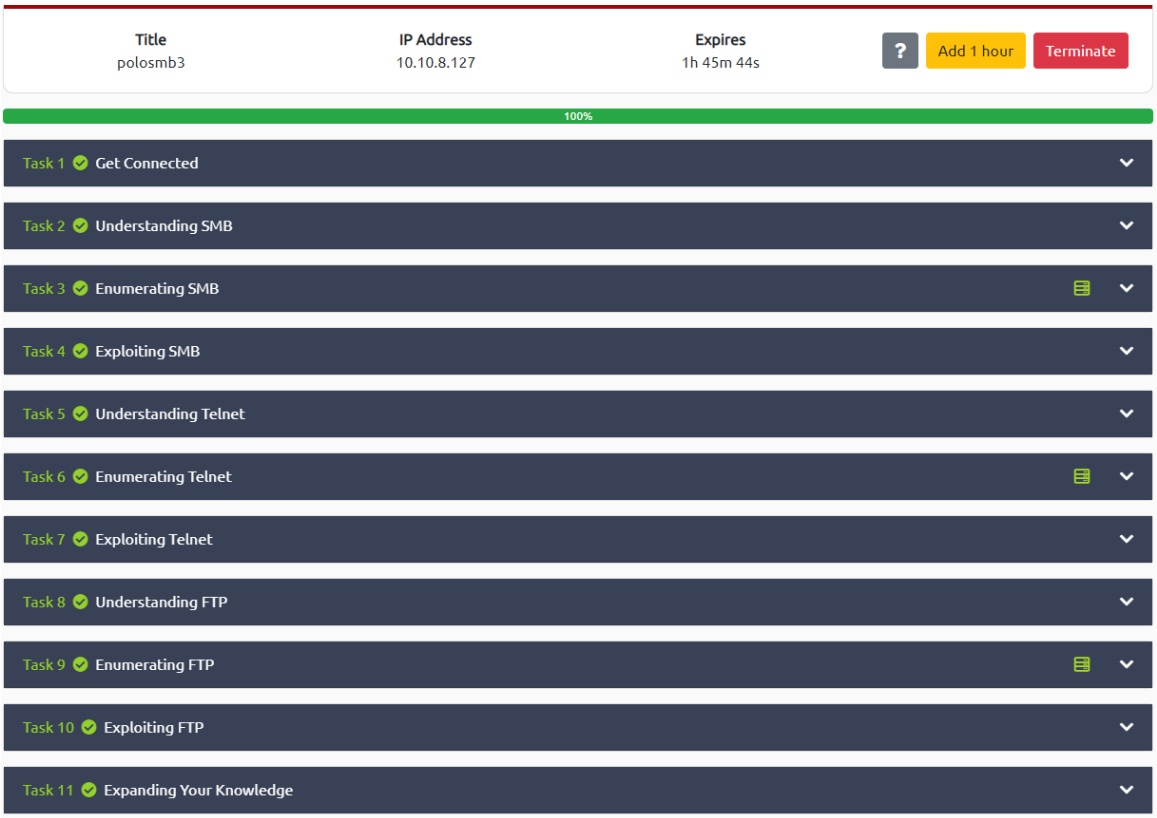


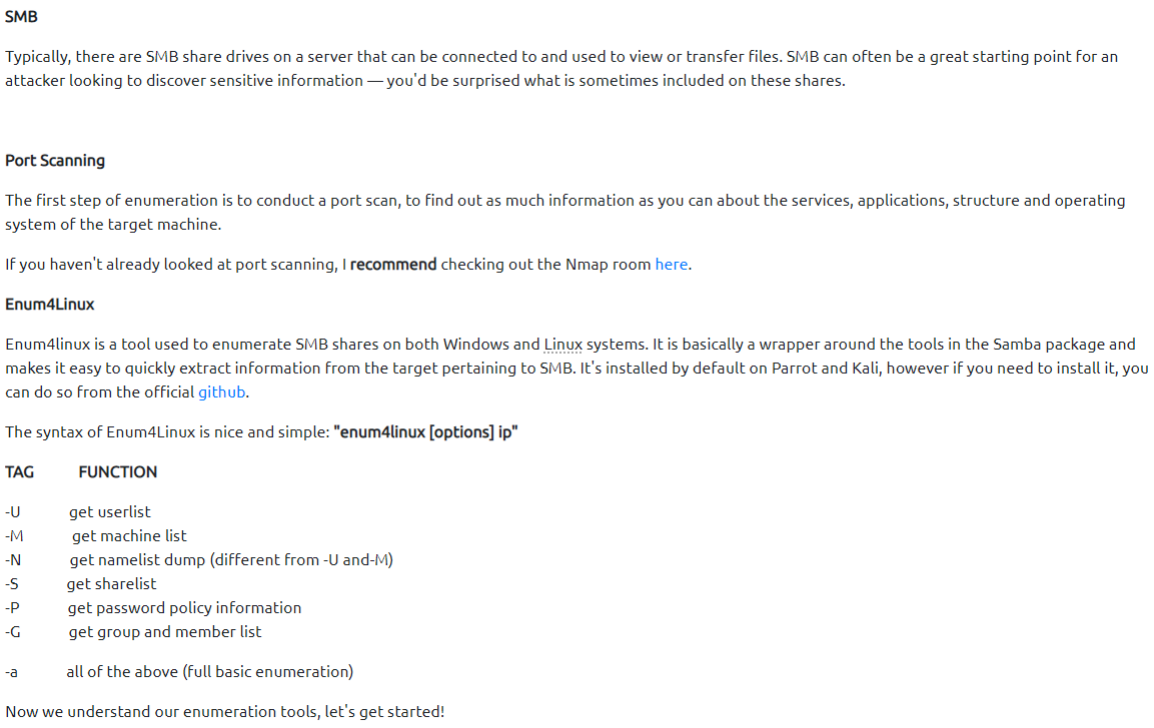
Рисунок 4.1 – Етапи кімнати Network Services

На рисунку 4.2 наведено кількість параметрів, їх назву та значення для запуску enum4linux.

Для проведення сканування цільової машини використовується nmap. Перейшовши за цим посилання можна бачити кімнату, що складається з великої кількості етапів вивчення можливостей інструменту nmap.

На рисунку 4.3 зображені все етапи кімнати з розвитку навичок використання nmap.

Розглянемо окремо етап 4, у якому перелічені найбільш популярні типи сканування та параметри, за допомогою яких можливо запустити ці процеси сканування.



SMB

Typically, there are SMB share drives on a server that can be connected to and used to view or transfer files. SMB can often be a great starting point for an attacker looking to discover sensitive information — you'd be surprised what is sometimes included on these shares.

Port Scanning

The first step of enumeration is to conduct a port scan, to find out as much information as you can about the services, applications, structure and operating system of the target machine.

If you haven't already looked at port scanning, I **recommend** checking out the Nmap room here.

Enum4Linux

Enum4linux is a tool used to enumerate SMB shares on both Windows and Linux systems. It is basically a wrapper around the tools in the Samba package and makes it easy to quickly extract information from the target pertaining to SMB. It's installed by default on Parrot and Kali, however if you need to install it, you can do so from the official github.

The syntax of Enum4Linux is nice and simple: **"enum4linux [options] ip"**

TAG	FUNCTION
-U	get userlist
-M	get machine list
-N	get namelist dump (different from -U and-M)
-S	get sharelist
-P	get password policy information
-G	get group and member list
-a	all of the above (full basic enumeration)

Now we understand our enumeration tools, let's get started!

Рисунок 4.2 – Опис інструменту командної стрічки enum4linux

Перелік типів сканування та параметри для їх запуску:

- sT, сканування TCP;
- sS, сканування SYN, або полу відкрите сканування;
- sU, сканування UDP;
- sN, сканування TCP Null;
- sF, сканування TCP Fin;
- sX, сканування TCP Xmas.

Детальний опис етапу 4 кімнати Nmap зображено на рисунку 4.4.

Наявність завдання в цій кімнаті, що включає в себе запуск інструменту командної стрічки для сканування портів цільової машини nmap та інструменту командної стрічки для сканування цільової машини на наявність спільних ресурсів SMB є дуже гарною можливістю продемонструвати користь застосування системи розпізнавання мови для запуску інструментів командної стрічки.

Рисунок 4.3 – Етапи вивчення інструменту nmap в Nmap

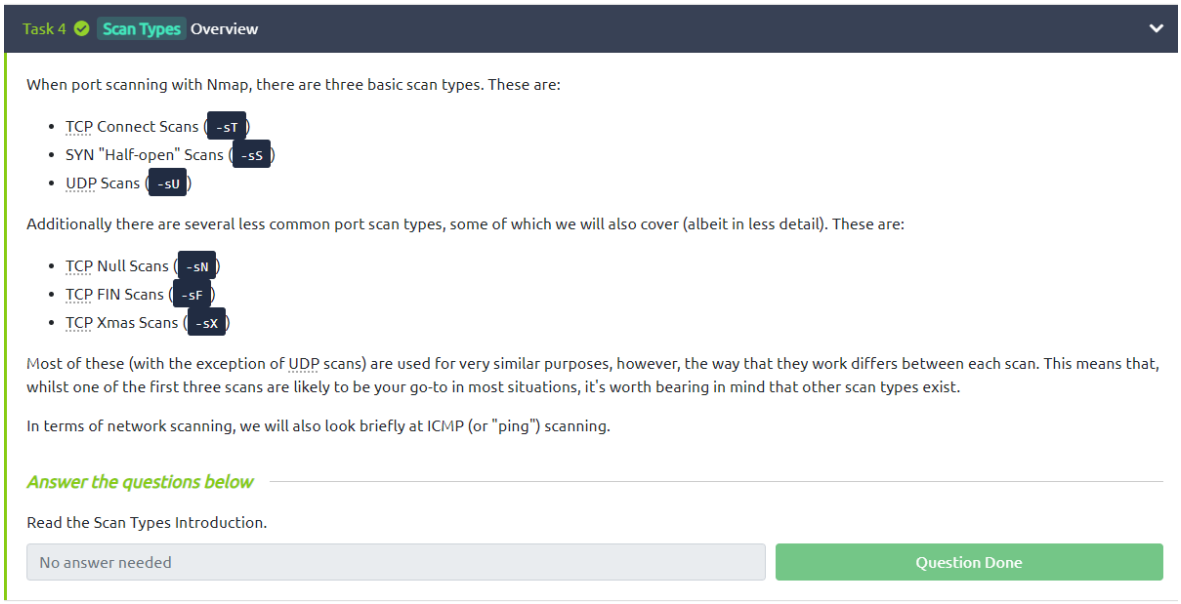


Рисунок 4.4 – Детальний опис етапу 4 кімнати Nmap

4.1.2 Перевірка портів, що використовуються

Для отримання необхідної інформації (після логування за допомогою SSH на цільову машину та для перевірки портів, що використовуються) зазвичай застосовують сканування всього діапазон портів від 1 до 65535. Ця ситуація досить розповсюджена на етапах пост-експлуатації або у щоденній роботі системного адміністратора.

Для аналізу стану портів, та виявлення процесів, що їх використовують, дуже зручно застосувати можливості інструменту командної стрічки netstat для сканування. Цей інструмент має дуже широкий спектр можливостей, для кожної з яких виділено певний параметр.

Повний список параметрів зображено на рисунку 4.5.

```
usage: netstat [-vWeenNcCF] [<Af>] -r         netstat {-V|--version|-h|--help}
       netstat [-vWnNcaeol] [<Socket> ...]
       netstat { [-vWeenNac] -i | [-cnNe] -M | -s [-6tuw] }

        -r, --route              display routing table
        -i, --interfaces         display interface table
        -g, --groups             display multicast group memberships
        -s, --statistics         display networking statistics (like SNMP)
        -M, --masquerade         display masqueraded connections

        -v, --verbose            be verbose
        -W, --wide               don't truncate IP addresses
        -n, --numeric            don't resolve names
        --numeric-hosts          don't resolve host names
        --numeric-ports          don't resolve port names
        --numeric-users          don't resolve user names
        -N, --symbolic           resolve hardware names
        -e, --extend             display other/more information
        -p, --programs           display PID/Program name for sockets
        -o, --timers             display timers
        -c, --continuous         continuous listing

        -l, --listening          display listening server sockets
        -a, --all                display all sockets (default: connected)
        -F, --fib                display Forwarding Information Base (default)
        -C, --cache              display routing cache instead of FIB
        -Z, --context            display SELinux security context for sockets

  <Socket>={-t|--tcp} {-u|--udp} {-U|--udplite} {-S|--sctp} {-w|--raw}
           {-x|--unix} --ax25 --ipx --netrom
  <AF>=Use '-6|-4' or '-A <af>' or '--<af>'; default: inet
  List of possible address families (which support routing):
    inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
    x25 (CCITT X.25)
```

Рисунок 4.5 – Список параметрів та їх опис інструменту командної стрічки netstat

В даному випадку для вирішення даної задачі необхідно використати наступні параметри:

- l, відображення сокетів, що знаходяться в режимі прослуховування;
- t, відображення тільки TCP сокетів;
- p, відображення назв додатків, що використовують ці сокети;
- n, не виконувати резолвінг ip адрес.

4.2 Приклад застосування системи розпізнавання мови для запуску nmap

Наведемо приклад застосування системи розпізнавання мови для того, щоб запустити інструмент командної стрічки для проведення сканування цільової машини nmap.

Для цього спочатку проаналізуємо допоміжну інформацію цього інструменту, яка зображена на рисунку 4.6.

За допомогою додатку для парсингу допоміжної інформації інструментів командної стрічки сформуємо XML файл, який буде містити параметри та їх опис.

Приклад такого файлу зображено на рисунку 4.7.

Після того, як файл було перевірено, наступним кроком потрібно викликати команду «nmap». Для цього потрібно вголос сказати «nmap» та дочекатися появлення вікна, що повідомить про успішно розпізнану команду «nmap».

Приклад такого вікна зображено на рисунку 4.8

Після вікна, що інформує про успішне розпізнавання команди слід очікувати ще одне вікно, у якому будуть відображені параметри, необхідні для запуску інструменту nmap. У такому вікні повинні бути відображені всі параметри, що вказані в допоміжній інформації додатку, с можливістю вибору тих, що користувач хоче застосувати.

Приклад такого вікна зображено на рисунку 4.9.

```
root@ip-10-10-49-248:~/Desktop# nmap -h
Nmap 7.60 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
```

Рисунок 4.6 – Список параметрів та їх опис інструменту командної стрічки nmap

```
<parameter>
    <name>-sL:</name>
    <meaning>List Scan - simply list targets to scan</meaning>
</parameter>
<parameter>
    <name>-sn:</name>
    <meaning>Ping Scan - disable port scan</meaning>
</parameter>
<parameter>
    <name>-Pn:</name>
    <meaning>Treat all hosts as online -- skip host discovery</meaning>
</parameter>
<parameter>
    <name>-PS/PA/PU/PY[portlist]:</name>
    <meaning>TCP SYN/ACK, UDP or SCTP discovery to given ports</meaning>
</parameter>
<parameter>
    <name>-PE/PP/PM:</name>
    <meaning>ICMP echo, timestamp, and netmask request discovery probes</meaning>
</parameter>
```

Рисунок 4.7 – Список параметрів та їх опис для nmap у форматі XML

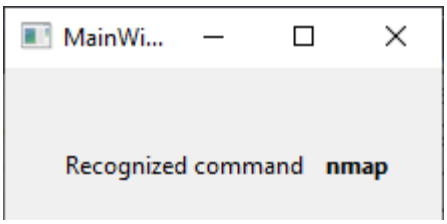


Рисунок 4.8 – Приклад вікна з інформуванням про успішне розпізнавання команди

Після появлення такого вікна, користувач повинен вголос промовити слово «use», або «don’t use». Після того, як система розпізнавання мови розпізнає ці слова, вона знайде найпершу пару радіокнопок, та відмітить кнопку «use», або «don’t use», після цього, користувач повинен промовити свій вибір ще раз. Після розпізнавання нової команди, система зробить вибір наступного пункту, що не був використаний до цього часу. Тобто у разі промовляння вдруге «don’t use» система відмітить кнопку за надписом «don’t use» у другому пункти, що має опис «-sn, Ping Scan - disable port scan».

Приклад такого вікна зображено на рисунку 4.10

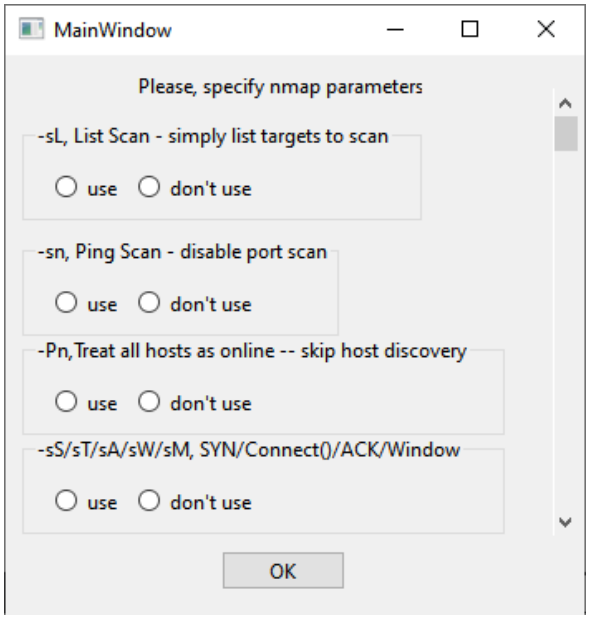


Рисунок 4.9 – Приклад вікна з вибором необхідних для інструменту параметрів

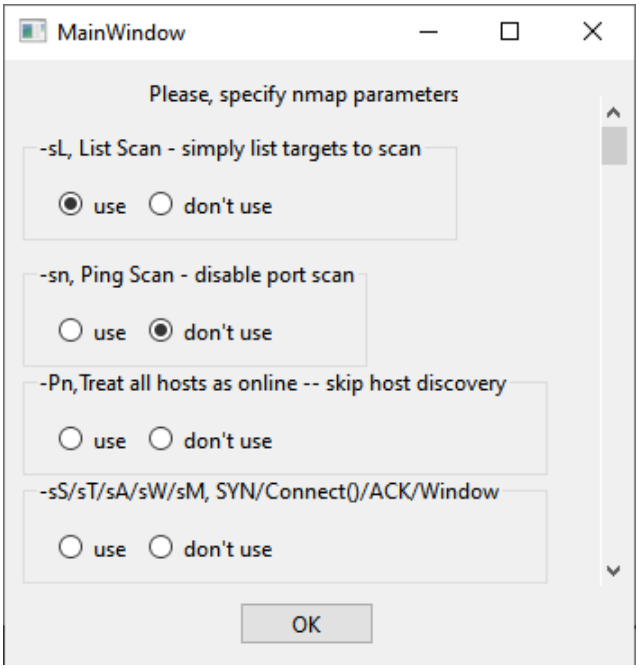


Рисунок 4.10 – Приклад вікна у процесі вибору необхідних для інструменту параметрів

Після того, як користувач вибере усі необхідні йому параметри, він повинен промовити «ОК» вголос, та після цього побачити новий термінал з запущеним інструментом командної стрічки nmap.

Приклад роботи цього інструменту зображено на рисунку 4.11.

```
root@ip-10-10-249-116:~# nmap -sS -sV 10.10.226.131

Starting Nmap 7.60 ( https://nmap.org ) at 2021-12-13 08:42 GMT
Nmap scan report for ip-10-10-226-131.eu-west-1.compute.internal (10.10.226.131)
Host is up (0.016s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protoco
l 2.0)
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 02:9D:2A:8F:60:63 (Unknown)
Service Info: Host: POLOSMB; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.09 seconds
root@ip-10-10-249-116:~#
```

Рисунок 4.11 – Приклад роботи інструменту для сканування nmap

4.3 Приклад застосування системи розпізнавання мови для запуску enum4linux

За прикладом використання системи розпізнавання мови для запуску інструменту для проведення сканування цільової машини nmap, користувач може запустити інструмент командної стрічки enum4linux вголос промовивши «enum4linux».

Приклад вікна інформування про успішне розпізнавання команди зображено на рисунку 4.12.

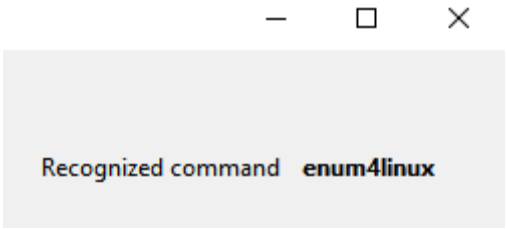


Рисунок 4.12 – Приклад вікна з інформуванням про успішне розпізнавання команди

Далі, після появлення вікна з вибором параметрів для запуску enum4linux, яке зображено на рисунку 4.13, за допомогою вимовляння слів «use», або «don’t use» і в кінці промовивши «OK», користувач матиме змогу бачити процес роботи інструменту для виявлення SMB файлів у відкритому терміналі.

Процес роботи інструменту командної стрічки enum4linux зображено на рисунку 4.14.

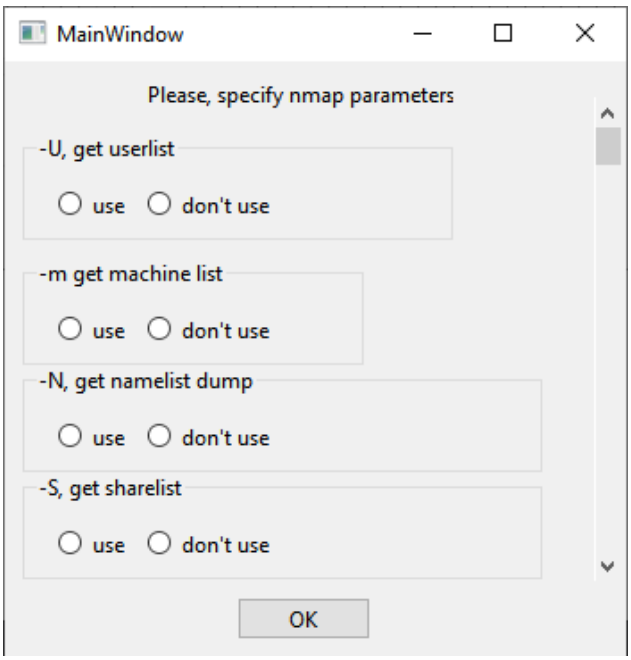


Рисунок 4.13 – Приклад вікна у процесі вибору необхідних для інструменту параметрів

```
root@ip-10-10-249-116:~# enum4linux -S -N 10.10.226.131
WARNING: polenum.py is not in your path.  Check that package is installed and your PATH is sane.
WARNING: ldapsearch is not in your path.  Check that package is installed and your PATH is sane.
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Mon Dec 13 08:52:07
 2021

 ==========================
|    Target Information    |
 ==========================
Target ........... 10.10.226.131
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 =====================================================
|    Enumerating Workgroup/Domain on 10.10.226.131    |
 =====================================================

[+] Got domain/workgroup name: WORKGROUP

 ====================================
|    Session Check on 10.10.226.131    |
 ====================================
[+] Server 10.10.226.131 allows sessions using username '', password ''

 ============================================
|    Getting domain SID for 10.10.226.131    |
 ============================================
Domain Name: WORKGROUP
Domain Sid: (NULL SID)
[+] Can't determine if host is part of domain or part of a workgroup

 ========================================
|    Name Enumeration on 10.10.226.131    |
 ========================================
[E] Internal error.  Not implemented in this version of enum4linux.

 ========================================
|    Share Enumeration on 10.10.226.131    |
 ========================================
WARNING: The "syslog" option is deprecated

        Sharename       Type      Comment
        ---------       ----      -------
        netlogon        Disk      Network Logon Service
        profiles        Disk      Users profiles
```

Рисунок 4.14 – Робота інструменту командної стрічки enum4linux

4.4 Часові показники щодо проведення тестування

4.4.1 Часові показники на прикладі тестування задачі CTF Chronos Vulnhub Chronos

CTF Chronos Vulnhub Chronos — це машина, що відноситься до категорії простих задач від Vulnhub від AL1ENUM.

CTF Chronos Vulnhub була обрана для аналізу часових показників (часових витрат) на тестування через те, що потребує для повного проходження виконнання команд, які можливо класифікувати як прості команди (мають в складі від одного до трьох параметрів), але всі ці команди виконуються в різних контекстах. Відповідно користувач повинен постійно відкривати нові директоріі або термінали.

Під час проходження цієї CTF задачі було додано підтримку наступних команд до системи розпізнавання мови:

- nmap, для сканування цільової машини;
- netcat(nc), для встановлення слухача під’єднань по сокету за певним портом;
- python -m httpserver 80, для швидкого створення локального серверу;
- etc /cat/hosts/, для швидкого відображення файлу hosts;
- etc /cat/shadow/, для швидкого відображення файлу shadow;
- etc /cat/passwd/, для швидкого відображення файлу passwd.

Під час проходження цієї CTF задачі було проведено запуску перелічених команд за допомогою проголошення вголос певних команд, та набору їх їз застосуваннням командної стрічки. Нижче приведено порівняльну таблицю результатів замірів.

Результати порівняльного аналізу часових показників (із застосуванням командної стрічки та з застосуванням розробленого програмного застосунку - системи розпізнавання мови) на тестування наведено в таблиці 4.1.

Таблиця 4.1 – Порівняльна характеристика замірів виконання команд

	Час запуску команди із застосуванням командної стрічки, в секундах	Час запуску команди із застосуванням системи розпізнавання мови, в секундах
Nmap -A 192.168.1.110	8 с	6 с
nc -lvp 8888	4.5 с	1.9 с
python -m httpserver 80	7 с	2.5 с
etc /cat/hosts/	4.2 с	2.5 с
etc /cat/shadow/	5.7 с	1.9 с
etc /cat/passwd/	4.6 с	2.6 с

4.4.2 Часові показники на прикладі тестування задачі CTF Pickle Rick

Завдання є легким за складністю, при умові, що користувач має базові навички з користування наступними інструментами тестування:

- dirbuster;
- nmap;
- python http server;
- netcat;
- clipboard.

CTF Pickle Rick була обрана через те, що для повного проходження потребує використання команд з переліку вище, але з більшою кількістю параметрів, ніж були використані команди у CTF Chronos Vulnhub (середній рівень складності команд).

Під час проходження цієї CTF задачі було додано підтримку наступних команд до системи розпізнавання мови:

- nmap, для сканування цільової машини;
- netcat(nc), для встановлення слухача під’єднань по сокету за певним портом;
- python -m httpserver 80, для швидкого створення локального серверу;

- dirbuster, сканування директорій серверу;
- whoami, для перевірки поточного ім’я користувача.
- clipboard, для застосування буферу для збереження IP-адрес.

Під час проходження цієї CTF задачі було проведено запуску перелічених команд за допомогою проголошення вголос певних команд, та набору їх з застосуванням командної стрічки.

Результати порівняльного аналізу часових показників (із застосуванням командної стрічки та з застосуванням розробленого програмного застосунку - системи розпізнавання мови) на тестування наведено в таблиці 4.2.

Таблиця 4.2 – Порівняльна характеристика замірів виконання команд

	Час запуску команди із застосуванням командної стрічки, в секундах	Час запуску команди із застосуванням системи розпізнавання мови, в секундах
nmap -sC -sV 10.10.43.98	7.5 с	9 с
dirb http://10.10.43.98	6.3 с	5.7 с
dirb http://10.10.43.98 -X .php	8.4 с	8.5 с
nc -lvp 8080	4 с	2.5 с
whoami	1.5 с	2 с

За результатами отриманих часових показників (таблиці 4.1 та 4.2) є можливим зробити наступний висновок. Впровадження системи розпізнавання мови має позитивний вплив на швидкість проведення тестування на проникнення, але не за всіма тестами.

Найбільше прискорення в роботі з інструментами для проведення тестування на проникнення досягається в наступних випадках:

- під час необхідності відобразити зміст якогось стандартного файлу, наприклад файлів hosts, passwd та shadow;

- під час використання найбільш поширених наборів команд за параметрами, наприклад запуск серверу за допомогою Python, або прослуховувача для певного порту, наприклад за допомогою NetCat;

- під час використання сканерів, коли IP-адреса вже скопійована в буфер, це дуже пришвидшує роботу команди, так як користувач може голосовою командою «source ip clipboard» дістати потрібну IP-адресу не вводячи її руками, що значно може пошвидшити виконання роботи.

Також така система має і певні недоліки, що були виявлені в наступних сценаріях:

- за наявності не дуже потужного апаратного забезпечення система не буде ефективною, бо витрачається деякий час на розпізнавання команд;

- за умови використання інструментів для проведення сканування з параметрами, що не є дуже поширеними є необхідність вводити дані власноруч (це потребує додаткового часу). За рахунок цього система розпізнавання мови може опрацьовувати запит користувача довше, або з не значним прискоренням.

Застосування такої системи розпізнавання мови може покращити швидкість проведення тестування на окремих етапах сканування, аналізу цільової системи. Особливо добре ця система показала себе під час виконання команд, що є дуже поширеними, наприклад встановлення прослуховувача на визначений порт, або запуск серверу за допомогою Python. При виконанні подібних дій з командної стрічки (фахівець з інформаційної безпеки або тестувальник) має власноруч відкривати нові вікна терміналу, шукати місце для розміщення нового вікна, тощо. В таких випадках система розпізнавання мови економить час на подібні дії, та зберігає фокус експерта на тестування або визначення певної вразливості під час виконання своєї роботи (окремих їх етапів).

В цілому мої пропозиції потребують подальшого удосконалення та уніфікації системи розпізнавання мови до різних тестів або їх послідовності.

ВИСНОВОК

У першому розділі розглянуто процес тестування на проникнення, проаналізовано фази тестування на проникнення веб-додатків, мобільних додатків, Wi-Fi точок доступу, наведено опис найбільш поширених типів вразливостей та інструментів для пошуку та виявлення вразливостей.

У другому розділі розглянуто та проаналізовано програмні інструменти для проведення тестування на проникнення, та визначено які компоненти доцільно використовувати для проаналізованих типів вразливостей.

У третьому розділі розглянуті найбільш важливі компоненти системи, що пропонується, їх призначення для забезпечення коректного та якісного розпізнавання команд користувача у різних ситуаціях, розглянуто програмні засоби, що використовуються для реалізації системи розпізнавання мови та взаємодії з користувачем.

У четвертому розділі розглянуто приклади застосування системи розпізнавання мови у вигляді вирішення CTF завдань, зроблено моделювання застосування системи розпізнавання мови при тестуванні на проникнення, визначені часові показники виконання певних команд та окремих етапів (запуск з використанням терміналу та командної стрічки, так і виконаних з впровадженням програмної компоненти розпізнавання мови).

Проаналізувавши необхідні компоненти та розглянувши приклади застосування розробленої системи розпізнавання мови для запуску інструментів з метою тестування на проникнення можливо зробити висновок, що така система може стати дуже важливим та корисним помічником для спеціаліста з інформаційної безпеки або адміністратора безпеки, особливо за умов необхідності дуже часто (наприклад щоденно або декілька разів на добу) проводити тестування на проникнення в різні системи.

При перевірці першої версії компонентів системи стало зрозуміло, що дизайн всієї системи тестування повинен бути модульним, та не повинен

вимагати від розробника створювати декілька класів кожен раз, коли користувач бажає додати підтримку нового інструменту командної стрічки. Саме задля цього слід запровадити генерацію необхідних файлів у файли формату XML, що й було виконано в роботі. Це дозволить системі тестування не бути залежною від розробника, та кожен користувач матиме змогу підготувати конфігураційні файли у форматі XML власними способами з метою підготовки та налаштування власних сценаріїв тестування для різних ОС.

Порівняльний аналіз часових показників тестування (за етапами CTF задач) довів, що запропонований підхід з застосовуванням розпізнавання мови (для введення команд) дійсно підвищує швидкість тестування на проникнення через витрачення меншої кількості часу на окремі етапи проведення тестування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Tryahckme [Електронний ресурс] – Режим доступу: In. https://www.tryhackme.com

2. Medium [Електронний ресурс]. – Режим доступу: https://medium.com/lotus-fruit/top-10-operating-systems-for-ethical-hackers-and-penetration-testers-2020-list-b523b611cdbb

3. Reid F. An Analysis of Anonymity in the Bitcoin System [Електронний ресурс] / F. Reid and M. Harrigan // Security and Privacy in Social Networks. – Springer New York, 2013. – P. 197–223. – Режим доступу: ttps://users.encs.concordia.ca/~clark/biblio/bitcoin/Reid%202011.pdf.

4. What are the Types of Cyber Security Vulnerabilities? [Електронний ресурс] / Sarah Meiklejohn, Marjori Pomarole, Grant Jordan. – Режим доступу:

5. https://www.logsign.com/blog/what-are-the-types-of-cyber-security-vulnerabilities/

6. Common Types Of Network Security Vulnerabilities In 2021 [Електронний ресурс] / Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. – Режим доступу: https://purplesec.us/common-network-vulnerabilities/

7. Triaxiom security blog [Електронний ресурс] – Режим доступу: https://www.triaxiomsecurity.com/our-mobile-application-penetration-testing-methodology/

8. PentestPeople security blog [Електронний ресурс] – Режим доступу: https://www.pentestpeople.com/web-application-penetration-testing/

9. FreeCodeCamp online school and coureses [Електронний ресурс] – Режим доступу: https://www.freecodecamp.org/news/what-is-nmap-and-how-to-use-it-a-tutorial-for-the-greatest-scanning-tool-of-all-time/

10. CloudSavvyIt security blog [Електронний ресурс] – Режим доступу: https://www.cloudsavvyit.com/6424/how-to-use-the-snort-intrusion-detection-system-on-linux/

11. HowToGeek online security courses [Електронний ресурс] – Режим доступу: https://www.howtogeek.com/513003/how-to-use-netstat-on-linux/

12. Varonis forum [Електронний ресурс] – Режим доступу: https://www.varonis.com/blog/john-the-ripper/

13. Purplesec security blog [Електронний ресурс] – Режим доступу: https://purplesec.us/perform-wireless-penetration-test/