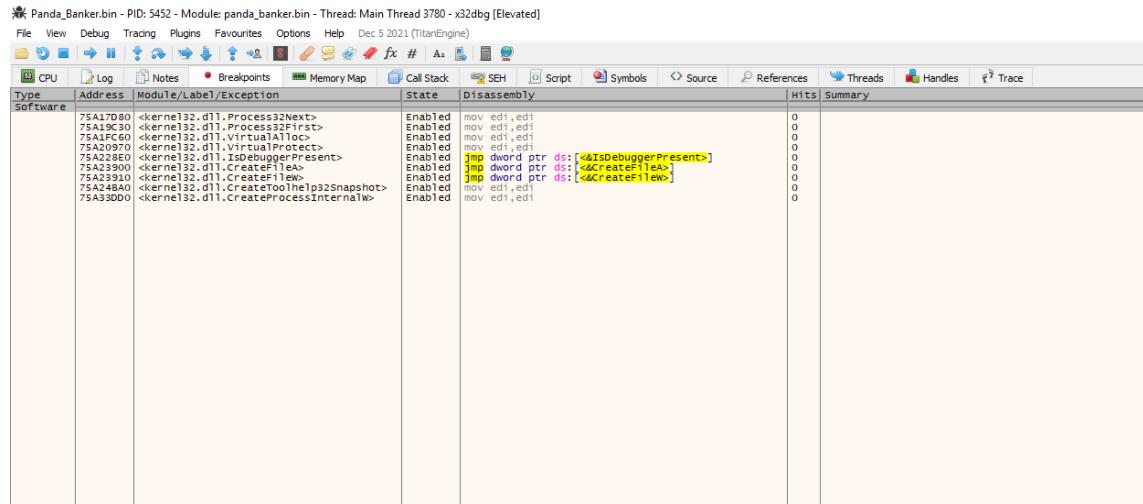
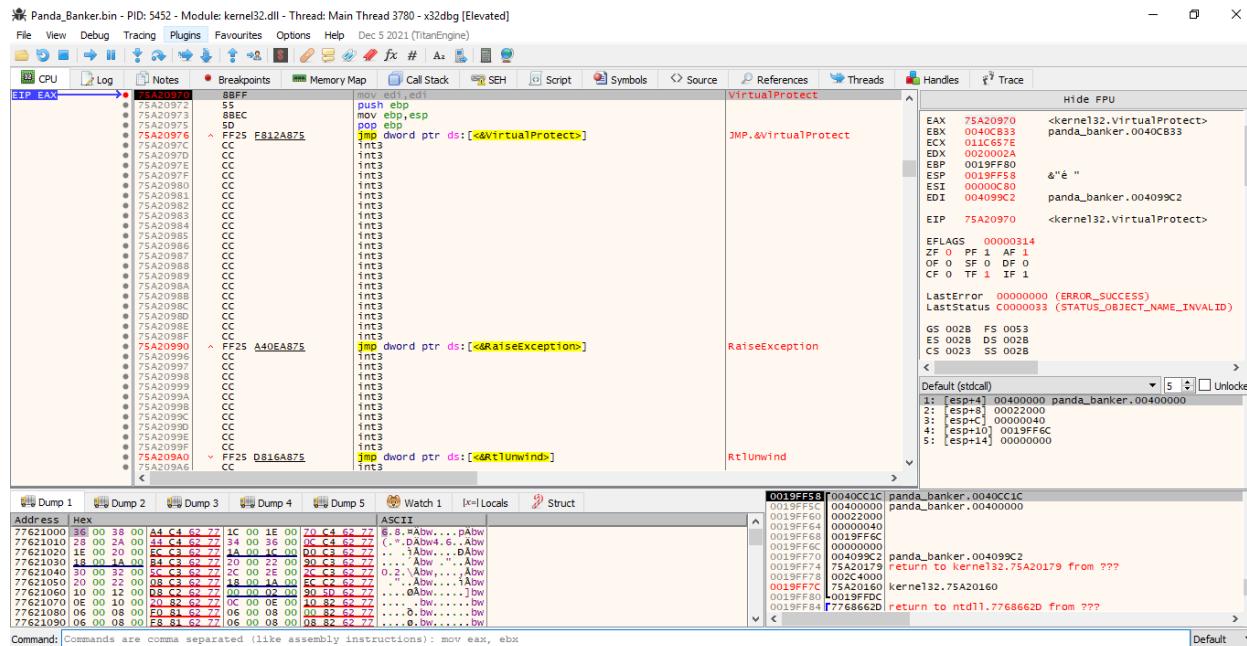


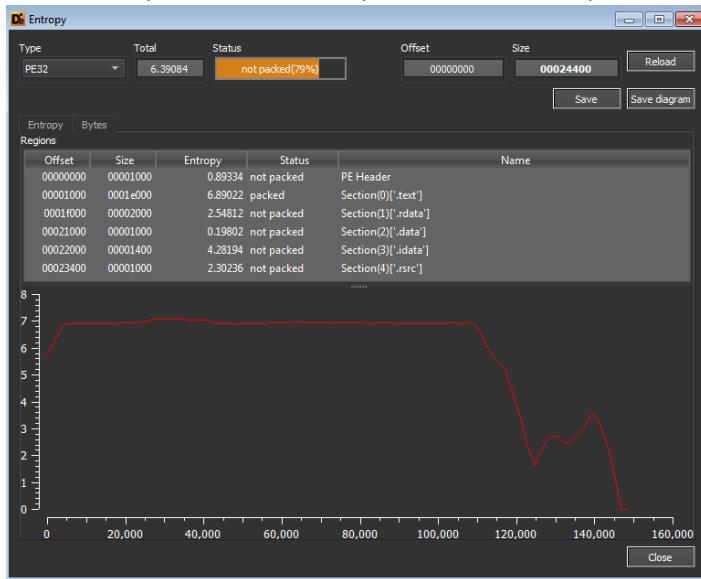
Here we load the sample in x32dbg and then put breakpoints on VirtualAlloc, VirtualProtect, IsDebuggerPresent, CreateToolhelp32Snapshot(enumarate list of running process in memory), Process32First, Process32Next, CreateFileA, CreateFileW, CreateProcessInternalW.



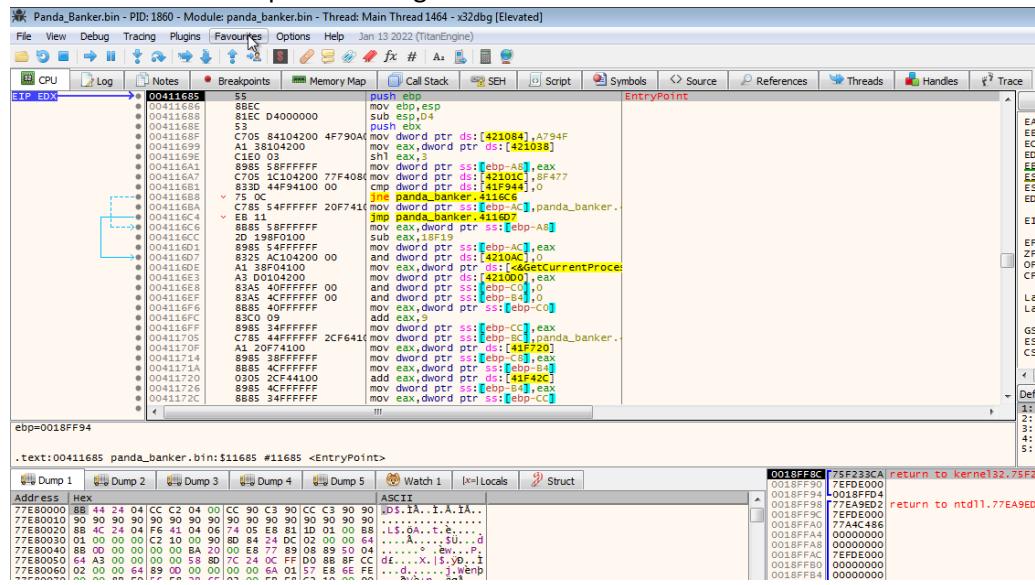
Now we hit our first breakpoint at VirtualProtect(it will change the permission bits for the memory region which is passed to it in second parameter), here just press f9.



Here initially we load the sample at Detect It Easy.

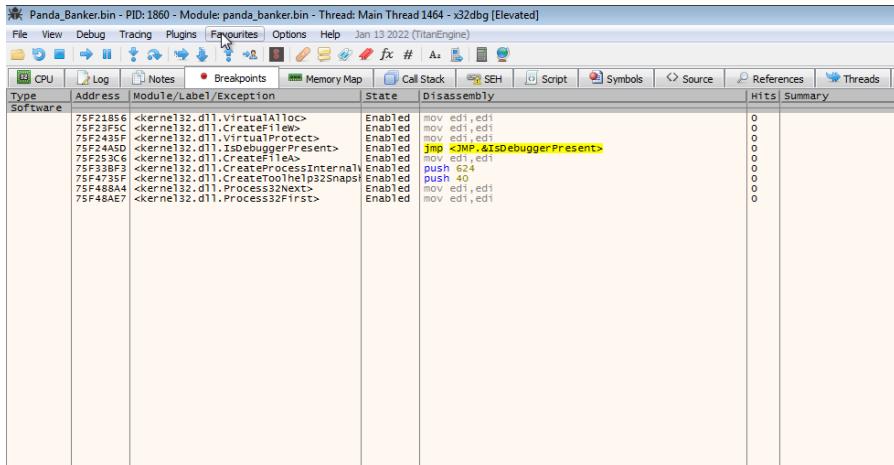


Now we load the sample in x32dbg.

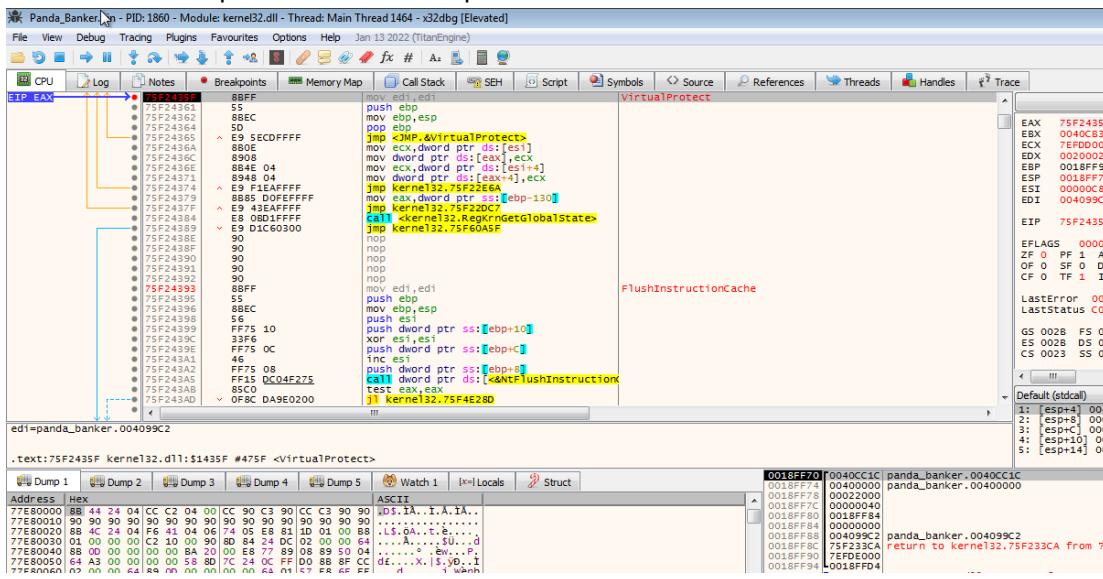


Here we load the sample in x32dbg and then put breakpoints on VirtualAlloc, VirtualProtect, IsDebuggerPresent, CreateToolhelp32Snapshot(enumarate list of running process in memory),

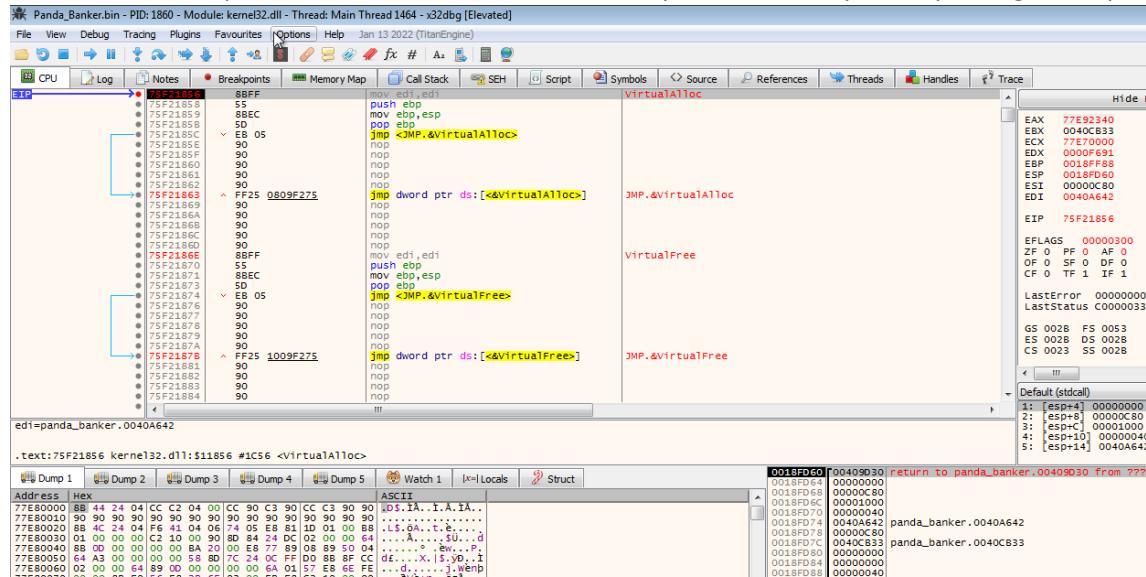
`Process32First`, `Process32Next`, `CreateFileA`, `CreateFileW`, `CreateProcessInternalW`.



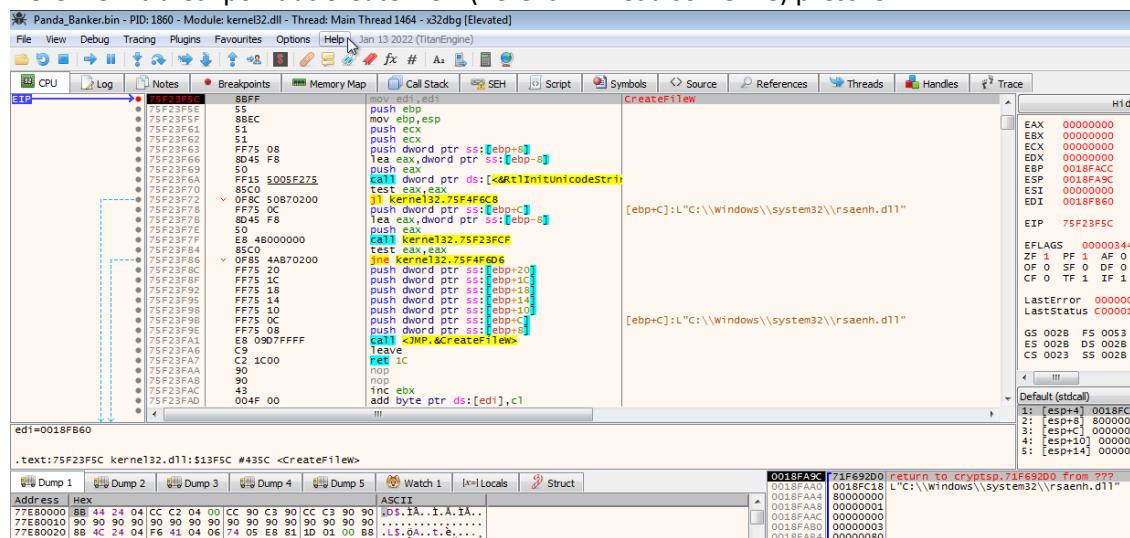
Here we hit breakpoint at VirtualAlloc press f9.



Here we hit breakpoint at VirtualAlloc (it will allocate space in memory for unpacking code) press f9.



Here we hit breakpoint at CreateFileW (here it will read some file) press f9.



Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system.

```

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
CreateFileW
EIP 75F23F5C <kernel32.CreateFileW
EAX 0054FD00 L:"C:\populkiller.exe"
EBX 00000000
ECX 0054FD00 L:"C:\populkiller.exe"
EDX 7EF0001C
EBP 00000001
ESP 0018F230
ESI 00043F "1475"
EDI 0000043F L:n
EIP 75F23F5C <kernel32.CreateFileW
EFLAGS 00000034
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 1 IF 1
Lasterror 00000002 (ERROR_FILE_NOT_FOUND)
Laststatus C0000034 (STATUS_OBJECT_NAME_NOT_FOUND)
GS 002B FS 0053
ES 002B DS 002B
CS 0023 SS 002B
Default (stcall) 5 Unl
1: [esp+4] 0054FD00 L:"C:\populkiller.exe"
2: [esp+8] 00000000
3: [esp+C] 00000003
4: [esp+10] 00000000
5: [esp+14] 00000003

```

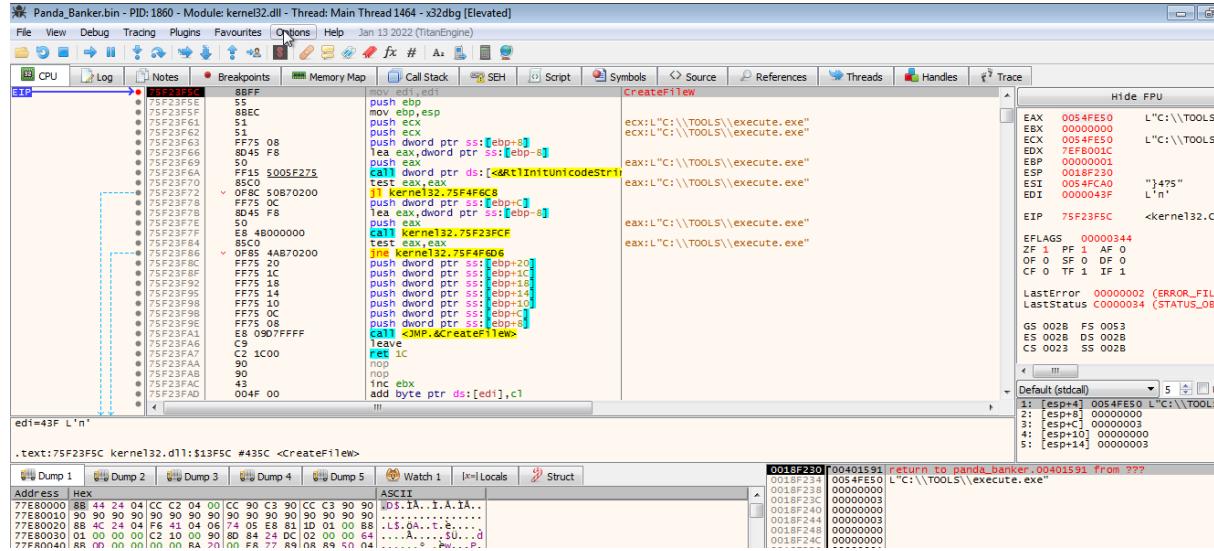
Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system.

```

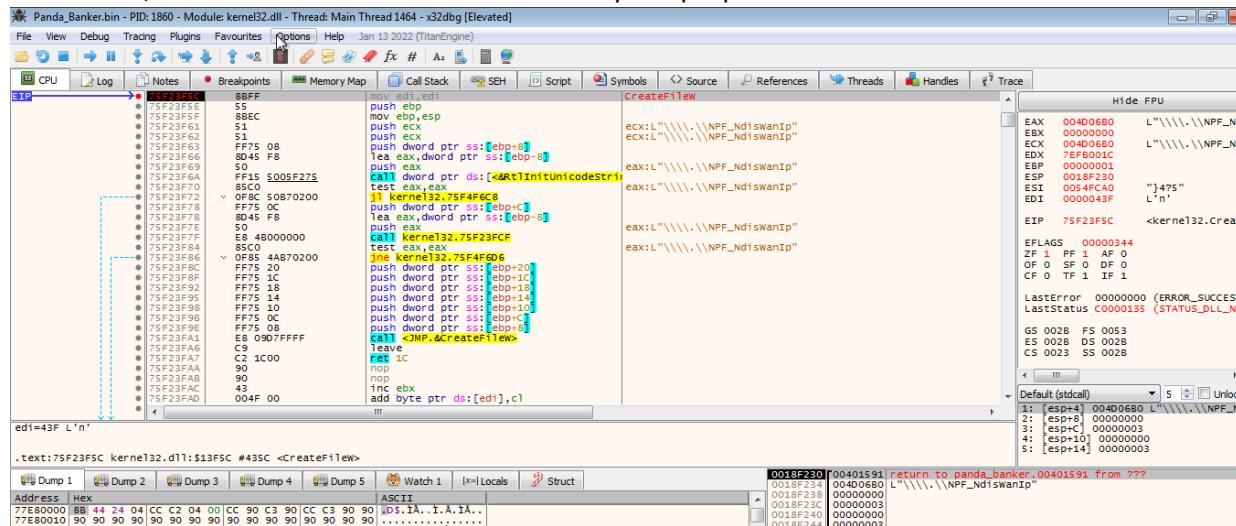
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
CreateFileW
EIP 75F23F5C <kernel32.CreateFileW
EAX 0054FDAB L:"C:\stimulator.exe"
EBX 00000000
ECX 0054FDAB L:"C:\stimulator.exe"
EDX 00000001
EBP 0018F230
ESP 0018F230
ESI 0018F230
EDI 0000043F "1475"
EIP 75F23F5C <kernel32.CreateFileW
EFLAGS 00000034
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 1 IF 1
Lasterror 00000002 (ERROR_FILE_NOT_FOUND)
Laststatus C0000034 (STATUS_OBJECT_NAME_NOT_FOUND)
GS 002B FS 0053
ES 002B DS 002B
CS 0023 SS 002B
Default (stcall) 5 Unl
1: [esp+4] 0054FDAB L:"C:\stimulator.exe"
2: [esp+8] 00000000
3: [esp+C] 00000003
4: [esp+10] 00000000
5: [esp+14] 00000003

```

Here we again hit the breakpoint at `CreateFileW`, it is checking here whether the following files are present in our system.

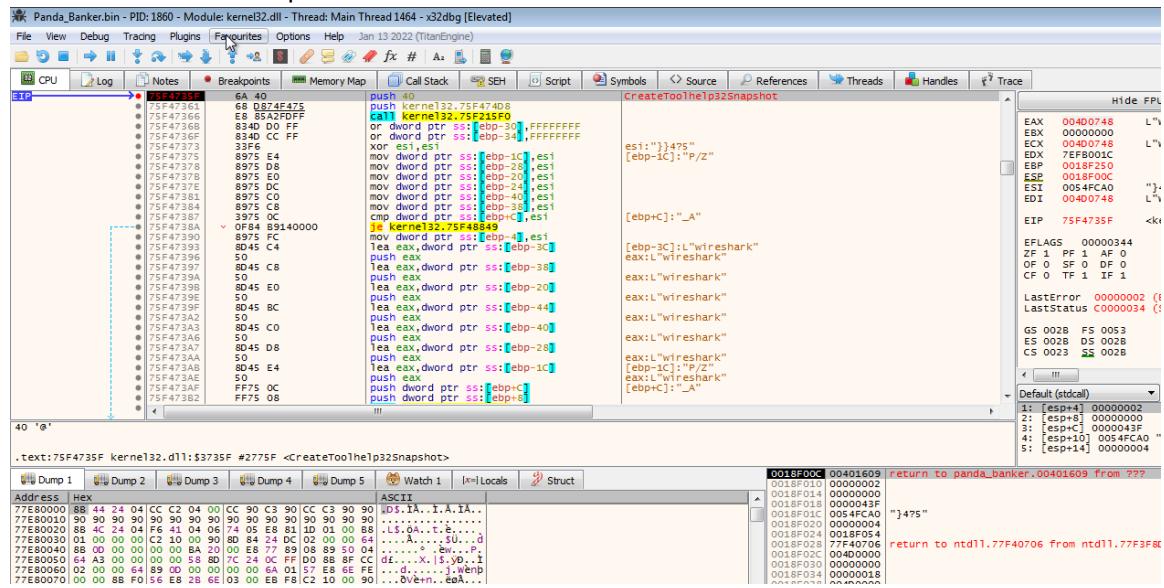


Here we again hit the breakpoint at `CreateFileW`, it is checking here whether the following files are present in our system, here it is the library of wireshark if we try to run the debugging will be terminated, so here we need to uninstall the library Winpcap and wireshark.

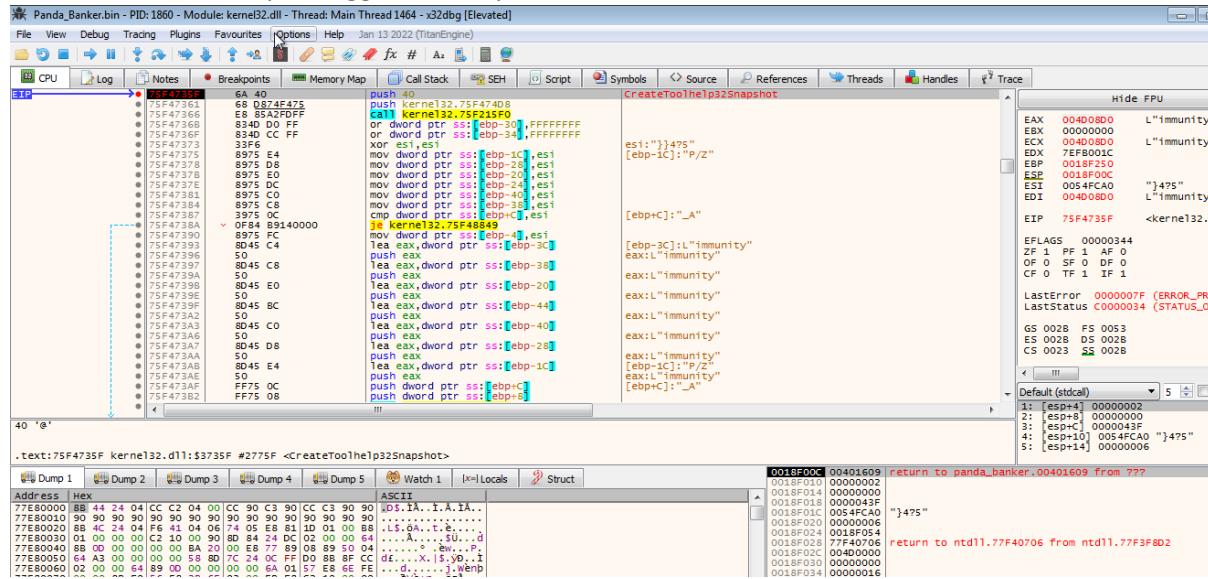


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory). here it is checking whether the wireshark is running in memory since we did not

start wireshark we can press f9 here.

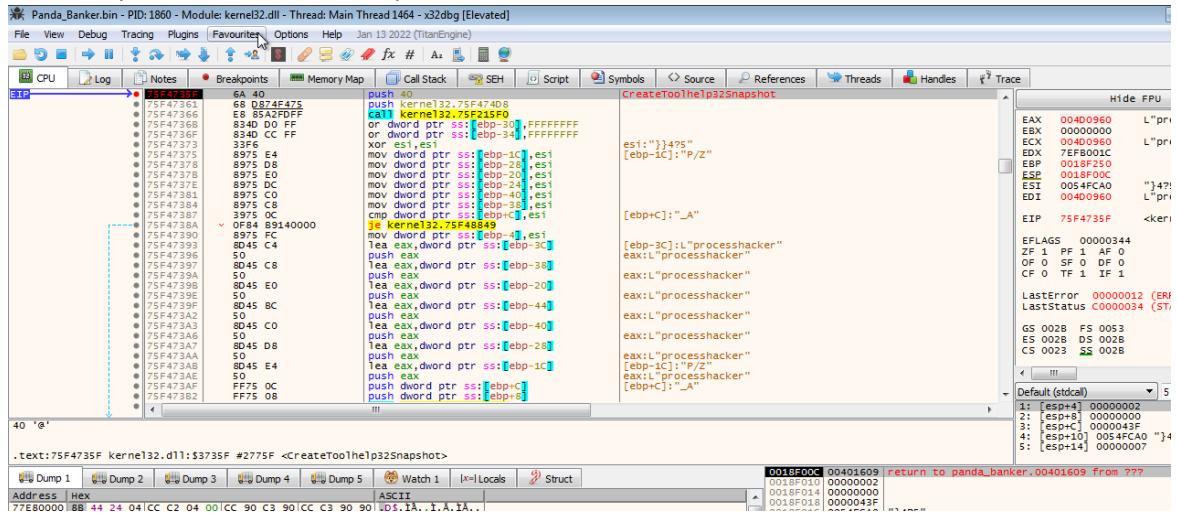


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the immunity debugger is running in memory since we did not start immunity debugger we can press f9 here.

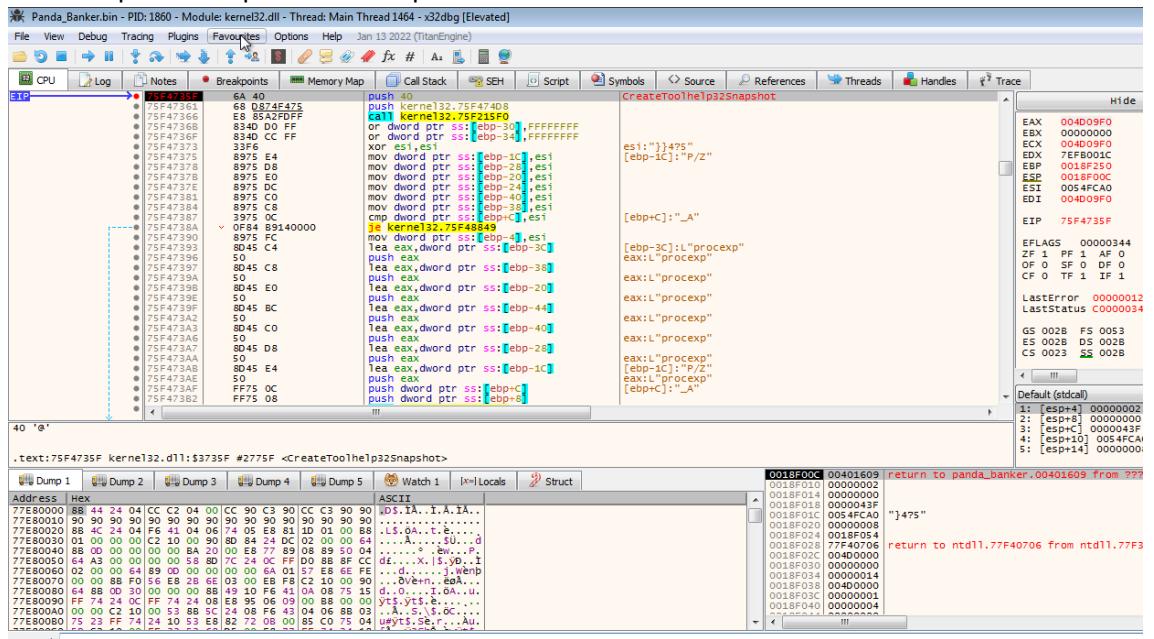


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the process hacker is running in memory since we did

not start process hacker we can press f9 here.

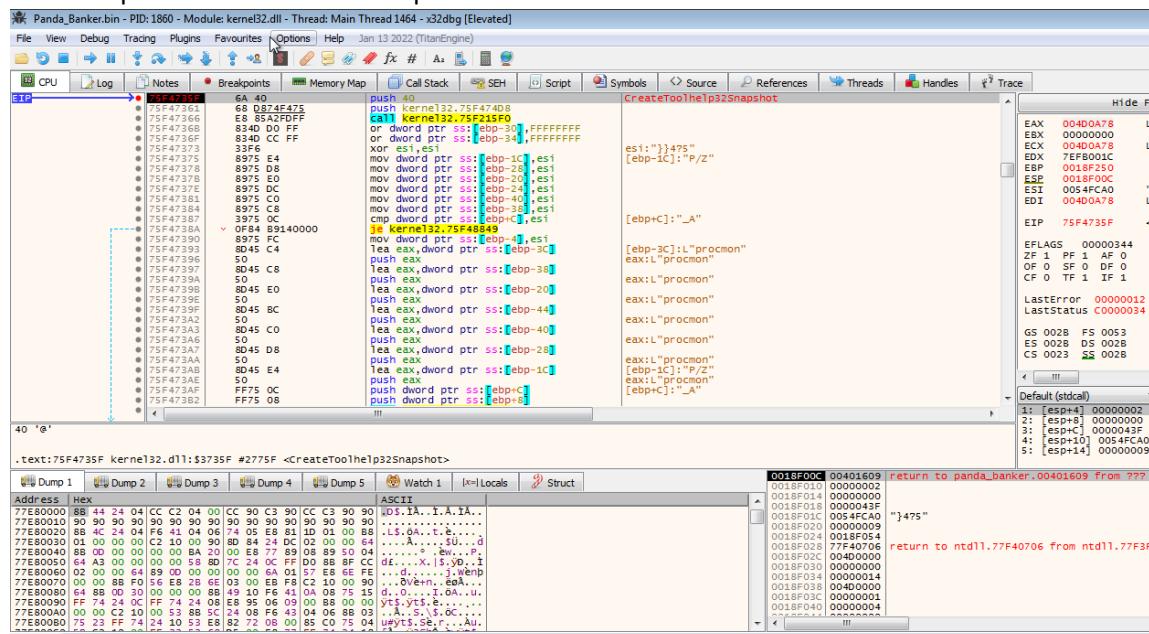


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the process explorer is running in memory since we did not start process explorer we can press f9 here

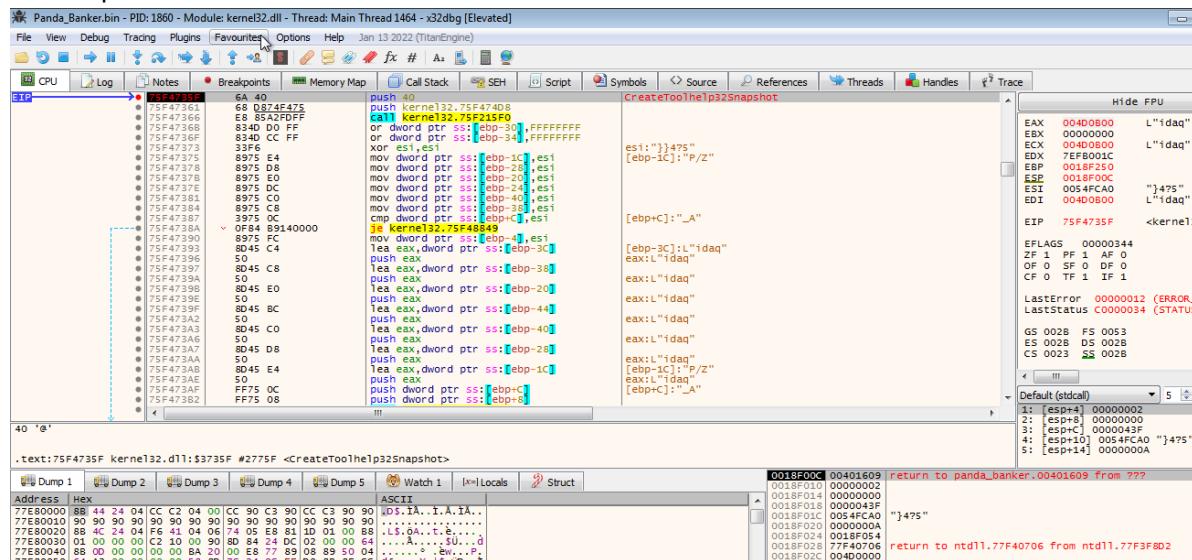


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the process monitor is running in memory since we did

not start process monitor we can press f9 here.

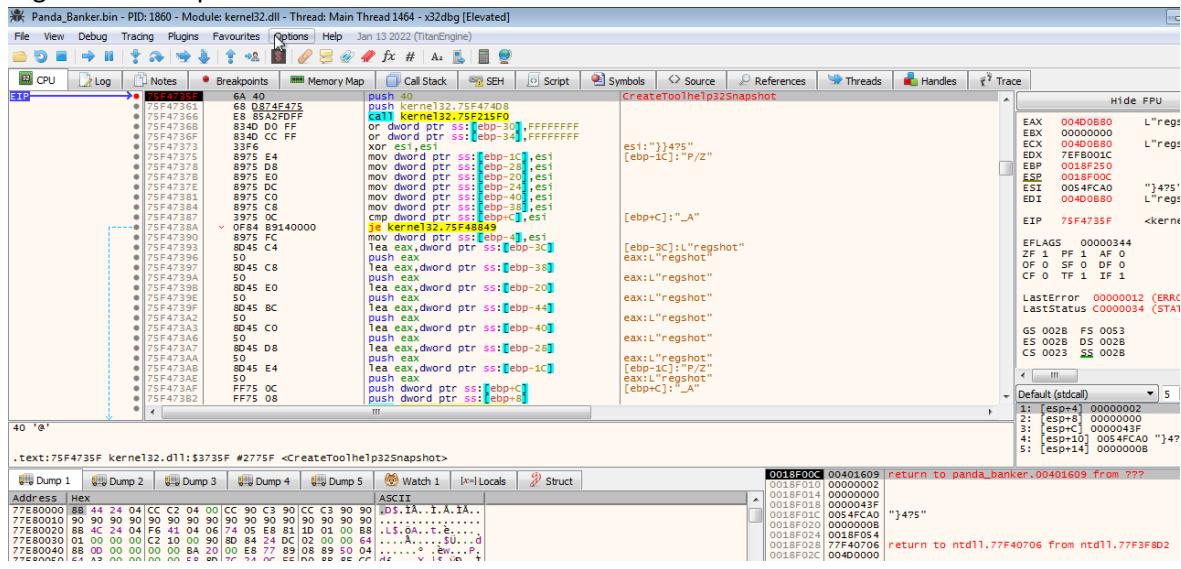


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the IDA is running in memory since we did not start IDA we can press f9 here.

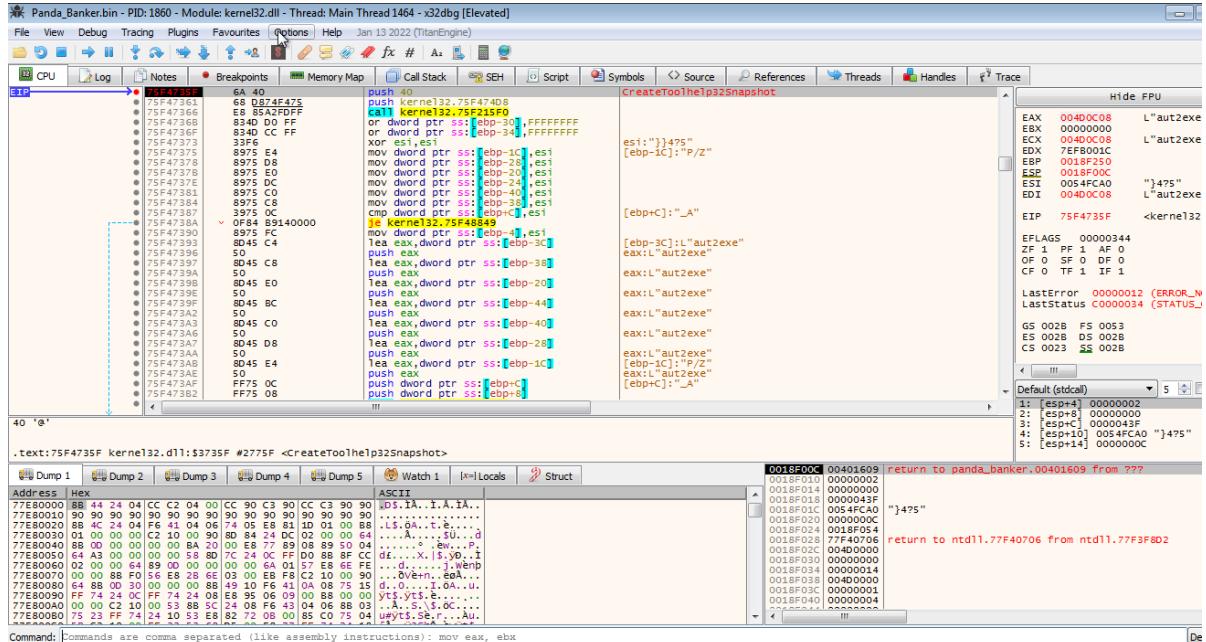


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the regshot is running in memory since we did not start

regshot we can press f9 here.

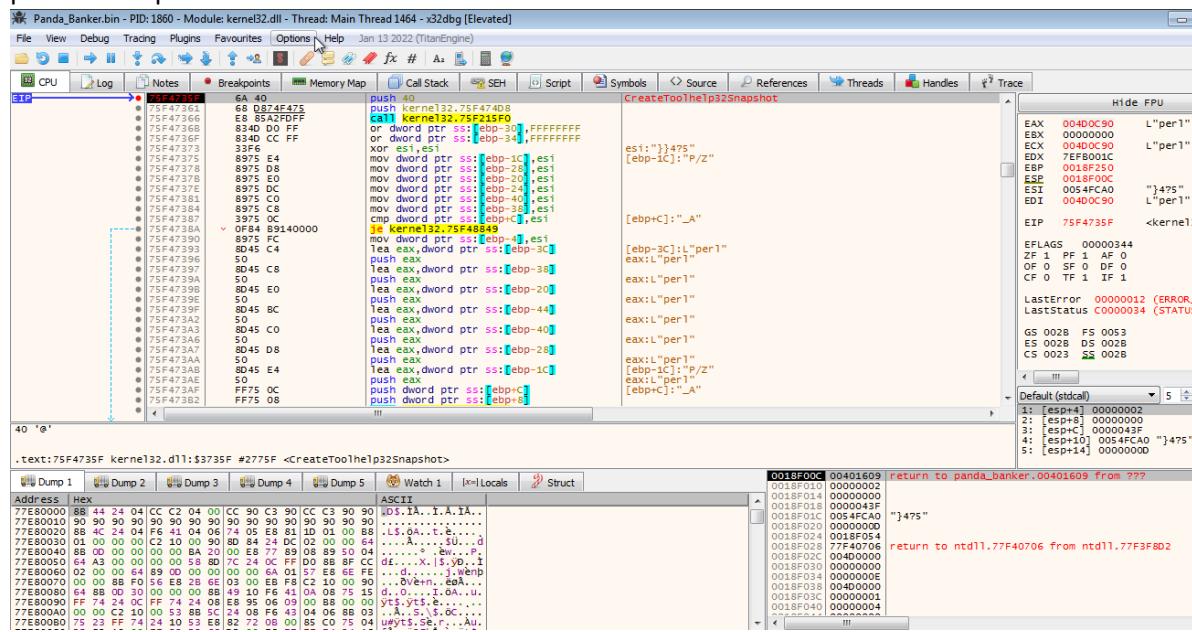


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the autoit is running in memory since we did not start autoit we can press f9 here.

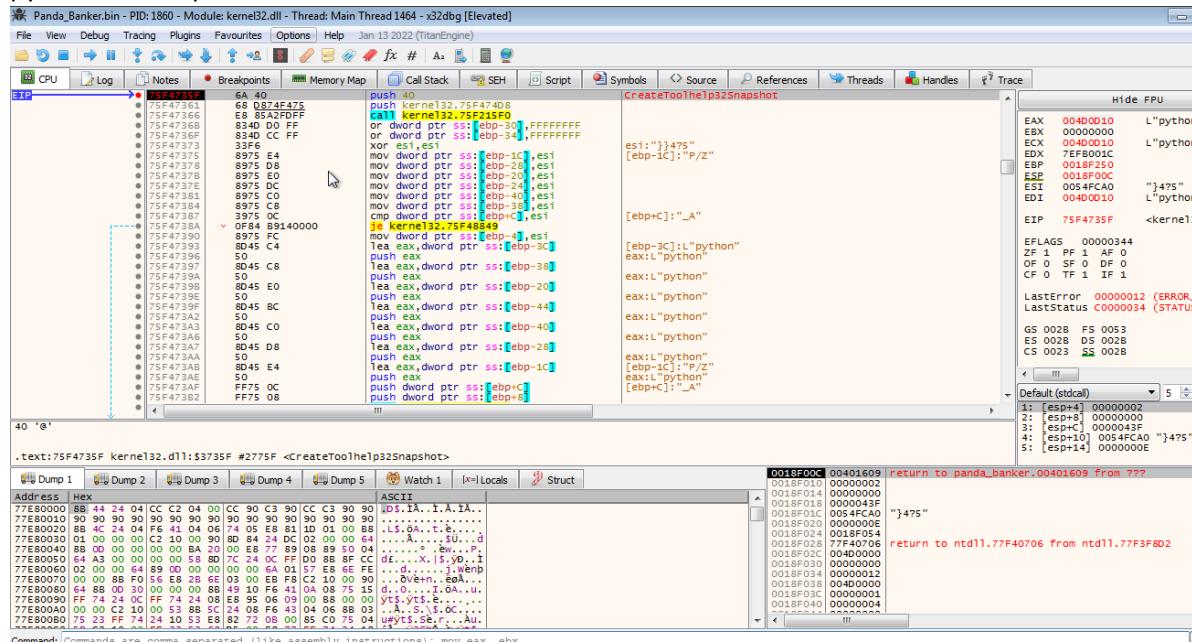


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the perl is running in memory since we did not start

perl we can press f9 here.

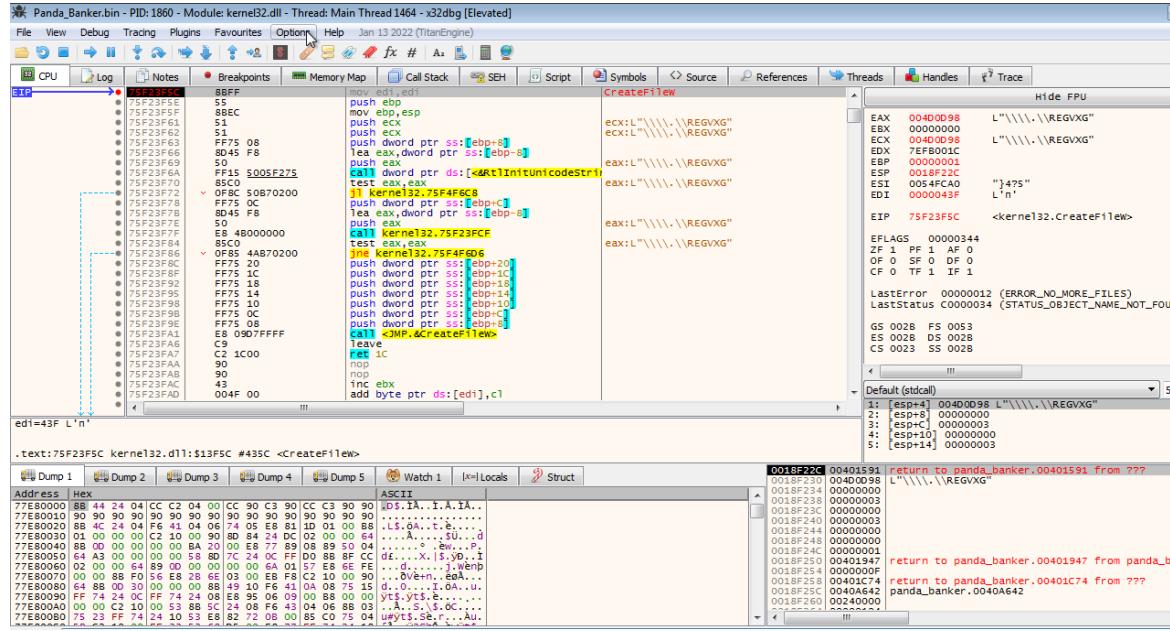


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the python is running in memory since we did not start python we can press f9 here.

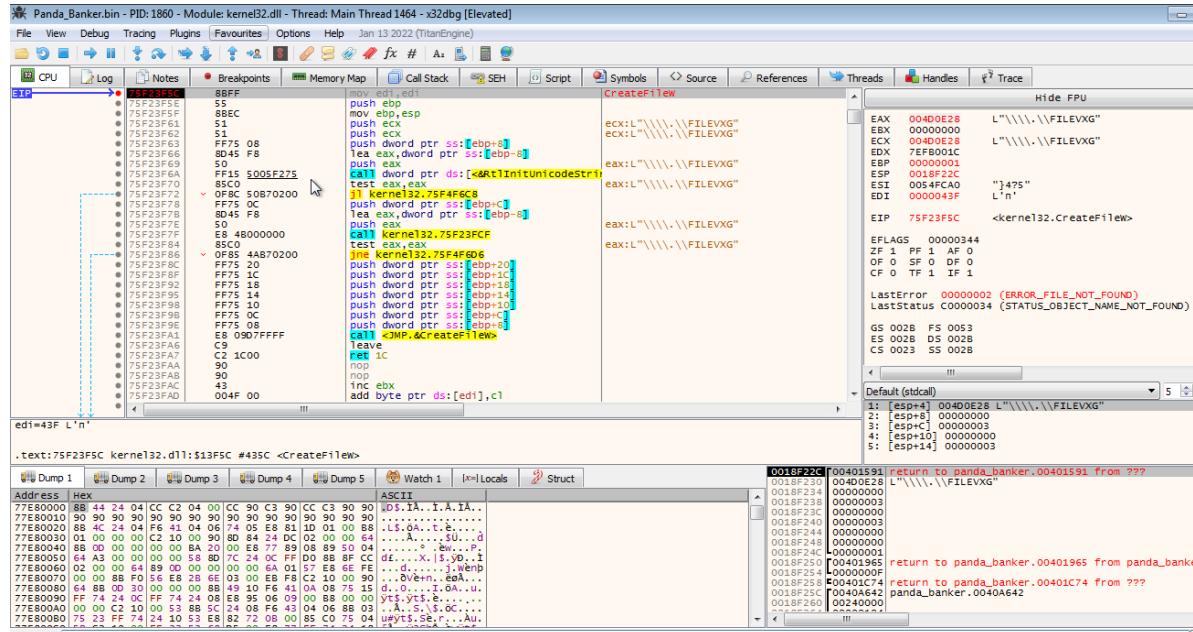


Command: Commands are comma separated (like assembly instructions): mov eax, ebx

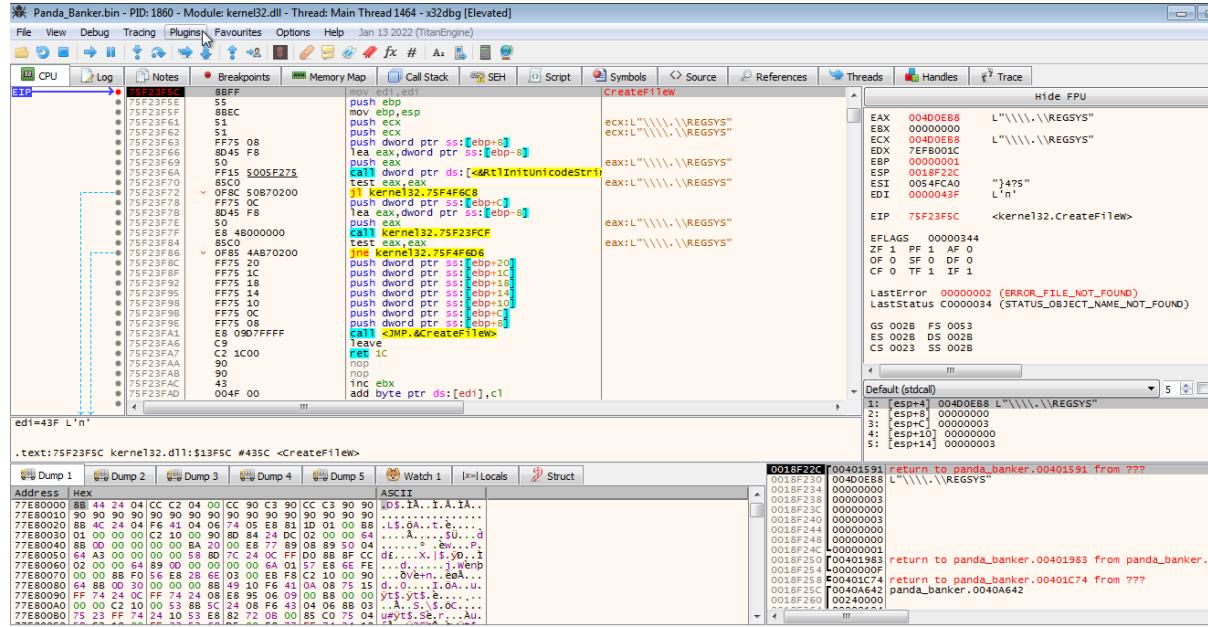
Here we again hit the breakpoint at `CreateFileW`, it is checking here whether the following files are present in our system.



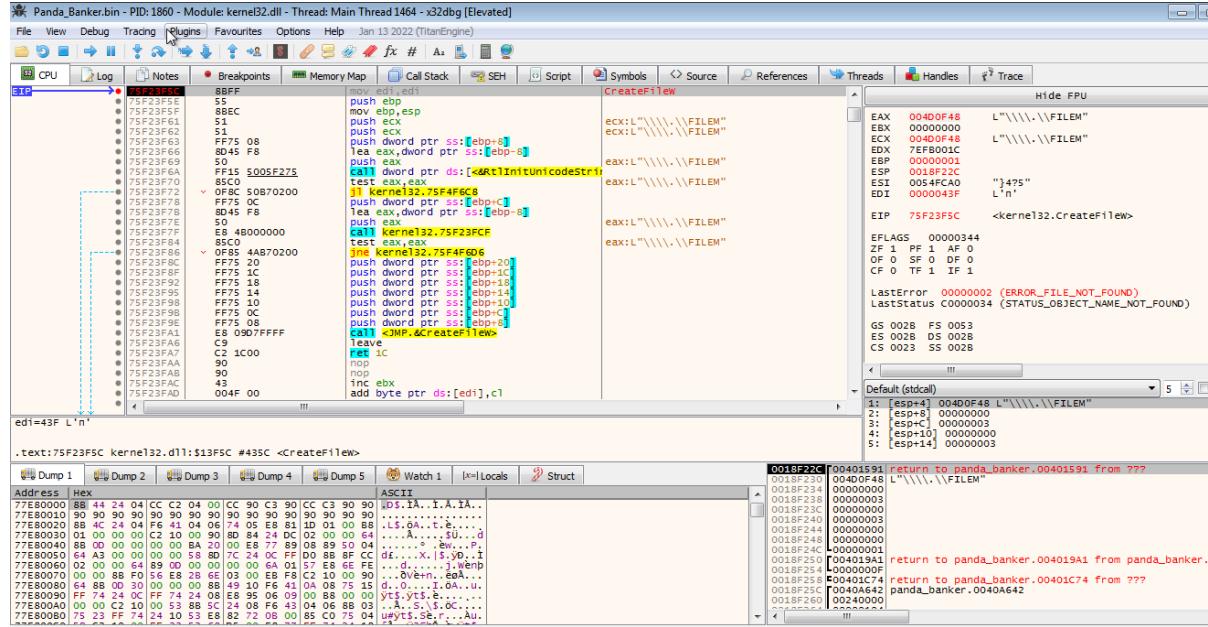
Here we again hit the breakpoint at `CreateFileW`, it is checking here whether the following files are present in our system.



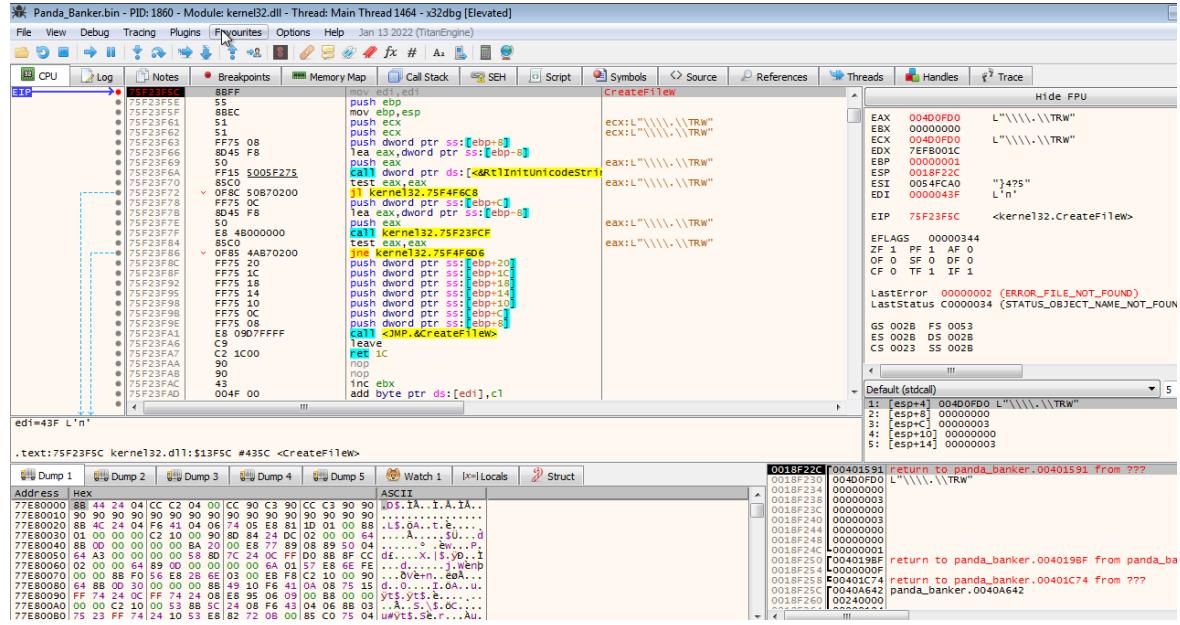
Here we again hit the breakpoint at `CreateFileW`, it is checking here whether the following files are present in our system.



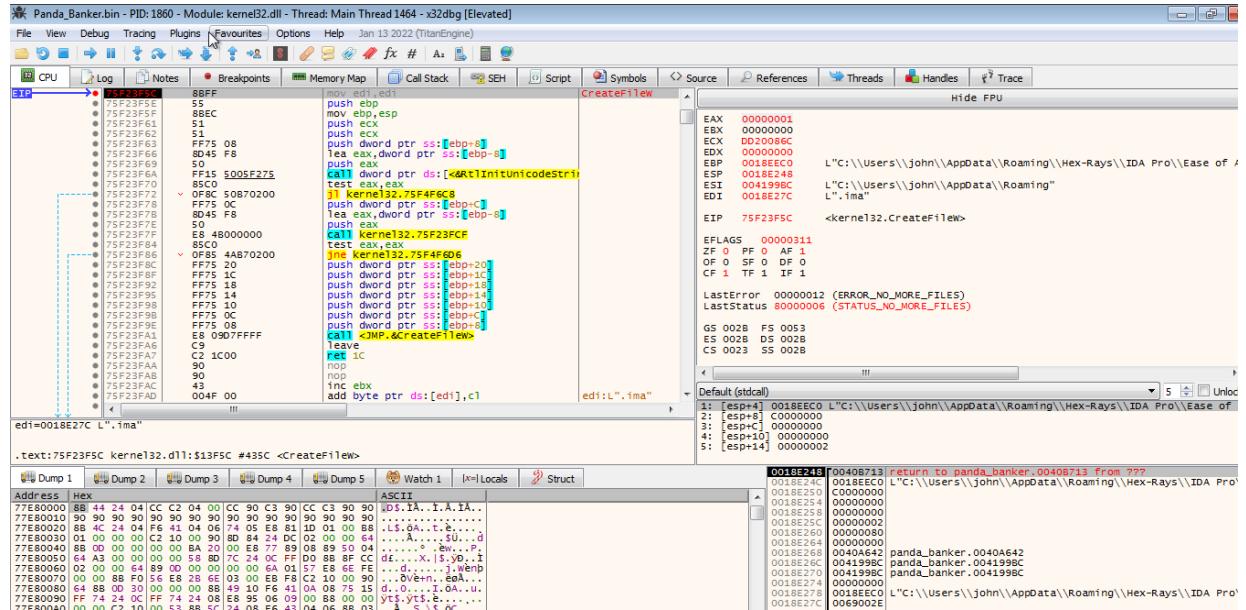
Here we again hit the breakpoint at `CreateFileW`, it is checking here whether the following files are present in our system.



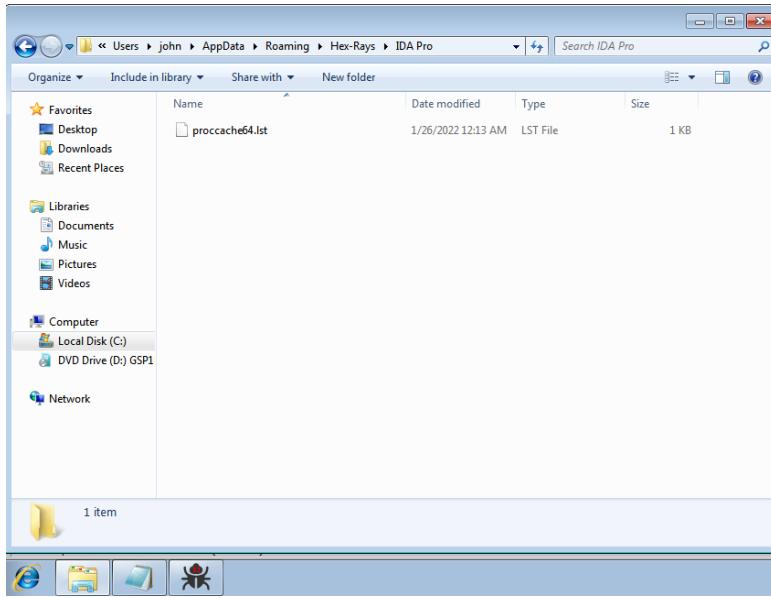
Here we again hit the breakpoint at `CreateFileW` , it is checking here whether the following files are present in our system .



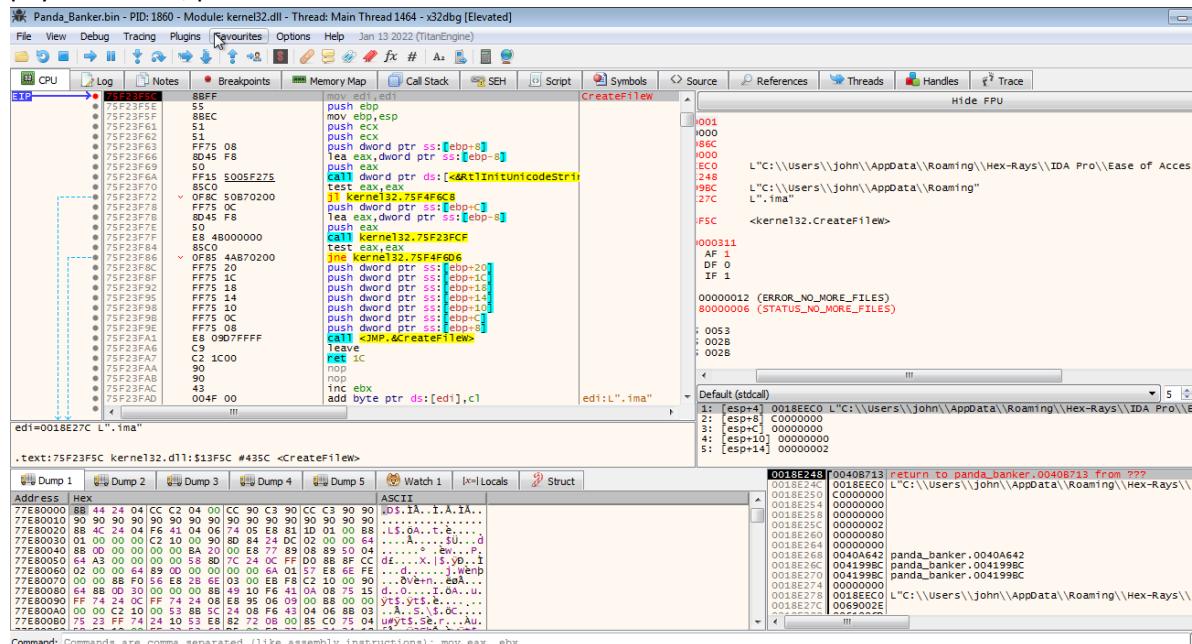
Here we again hit breakpoint at CreateFileW, but here it is accessing a location , might be trying to drop payload there , press f9 .



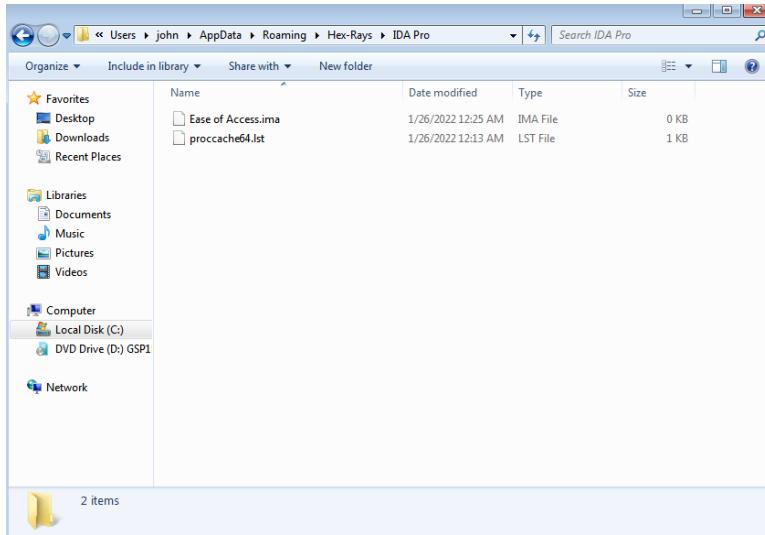
Here we view the location and till now it has drop the following file.



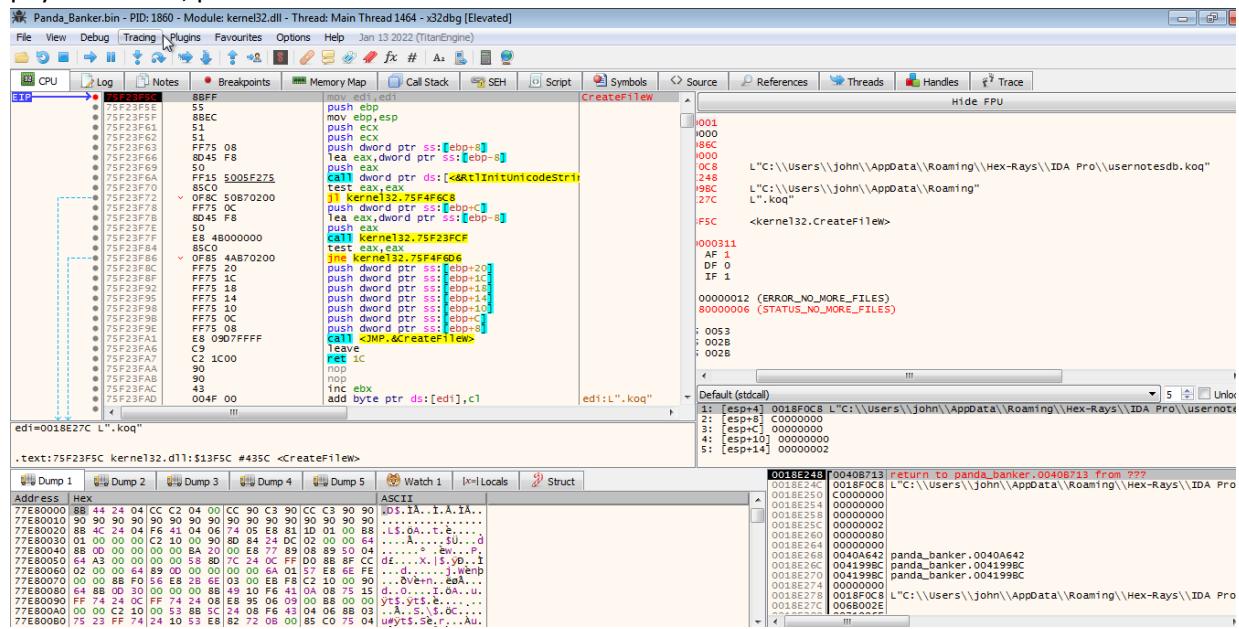
Here we again hit breakpoint at CreateFileW, but here it is accessing a location , might be trying to drop payload there , press f9 .



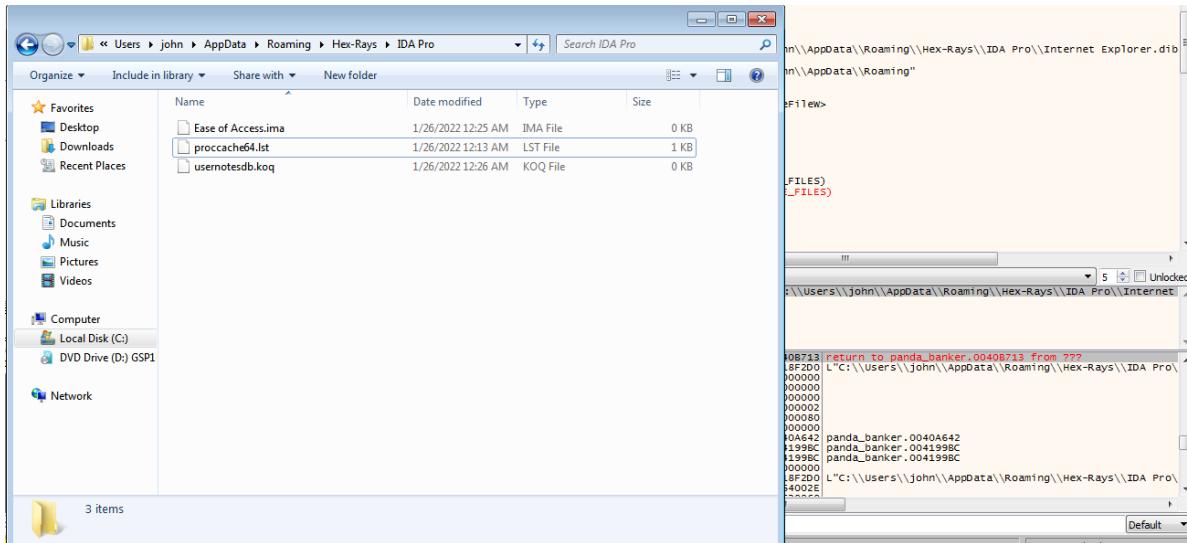
Here we view the location and till now it has drop the following files.



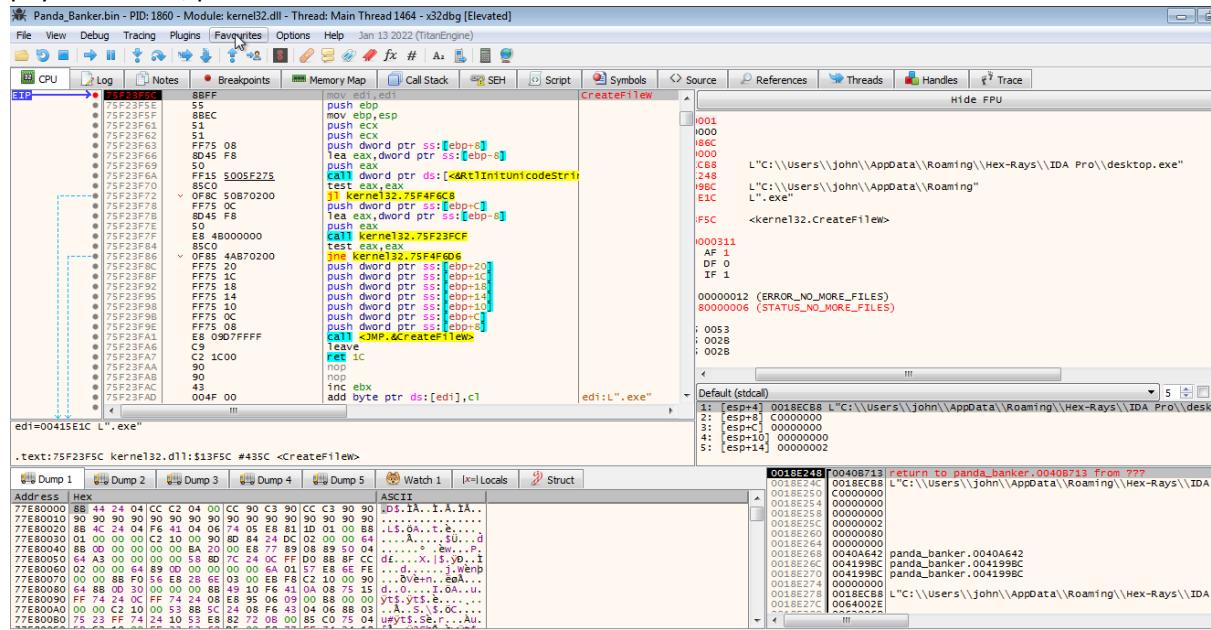
Here we again hit breakpoint at CreateFileW, but here it is accessing a location , might be trying to drop payload there , press f9.



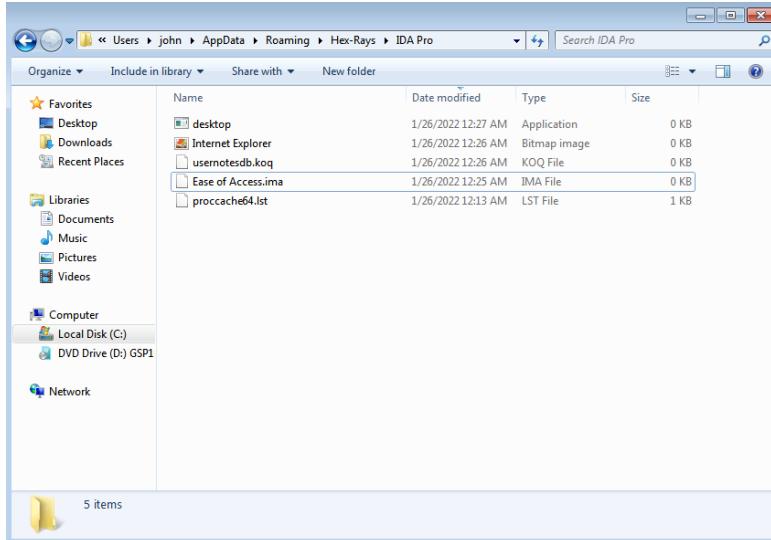
Here we view the location and till now it has drop the following files.



Here we again hit breakpoint at CreateFileW, but here it is accessing a location , might be trying to drop payload there , press f9.



Here we view the location and till now it has drop the following files .

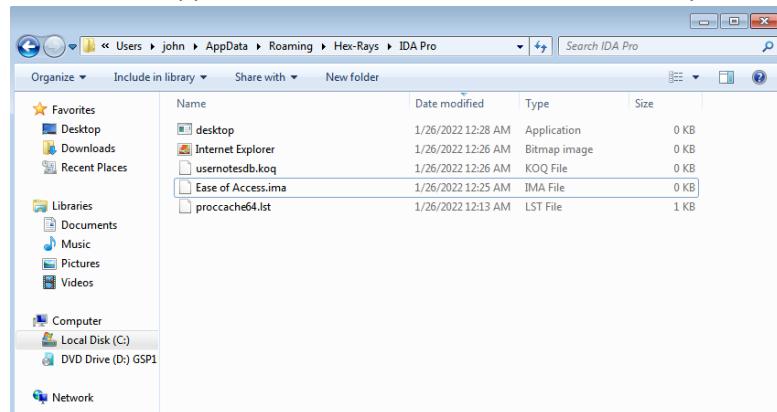


Here we again hit breakpoint at CreateFileW, but here it is trying to write in it , press f9.

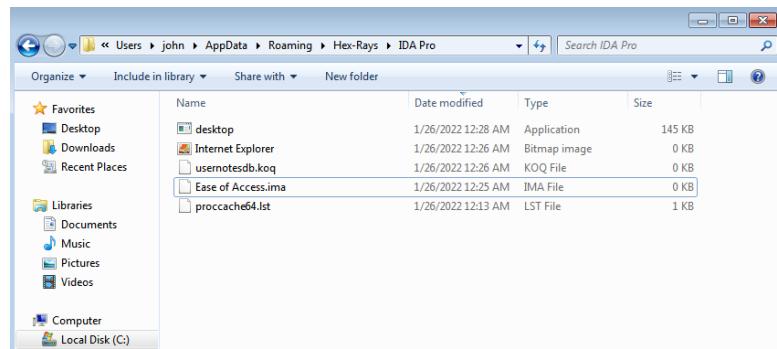
Register	Value
eax	0018ECB8
ebx	004F00
ecx	L"C:\\\\Users\\\\John\\\\AppData\\\\Roaming\\\\Hex-Rays\\\\IDA Pro\\\\desktop.exe"
edx	00000000
esi	00000000
edi	0018ECB8
ebp	0018ECC8
esp	0018ECC8
ebp	0018ECC8
cs	00400000
ss	00400000
ds	00400000
es	00400000
fs	00400000
gs	00400000

Here we can see the full path of the previously dropped exe.

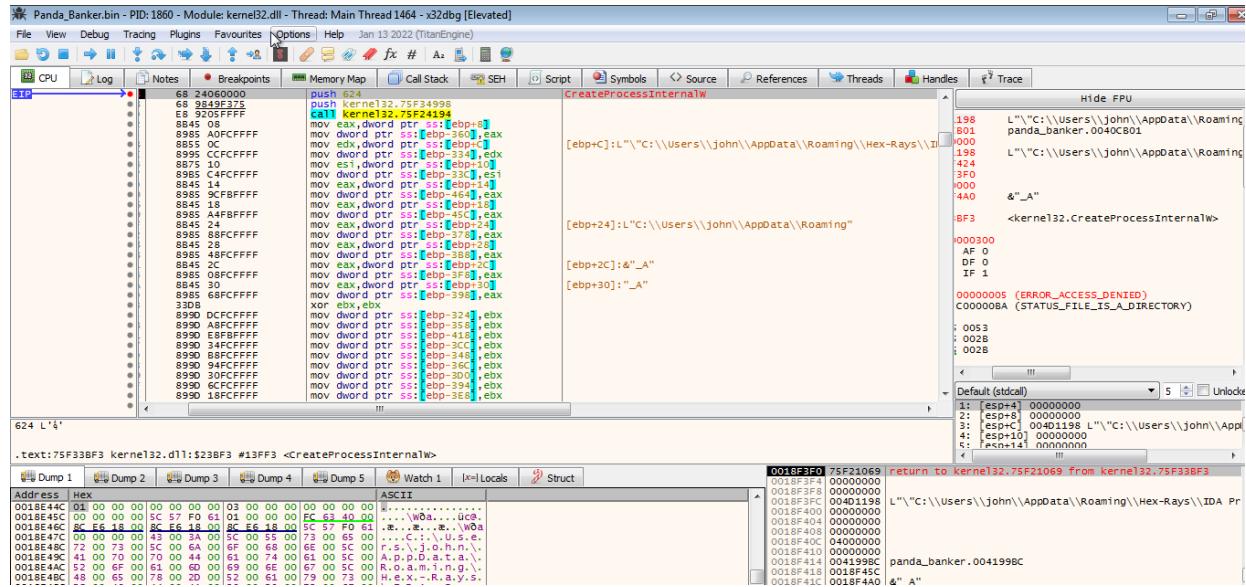
This is the dropped exe ,Note that here till now it has only created the empty exe, now press f9.



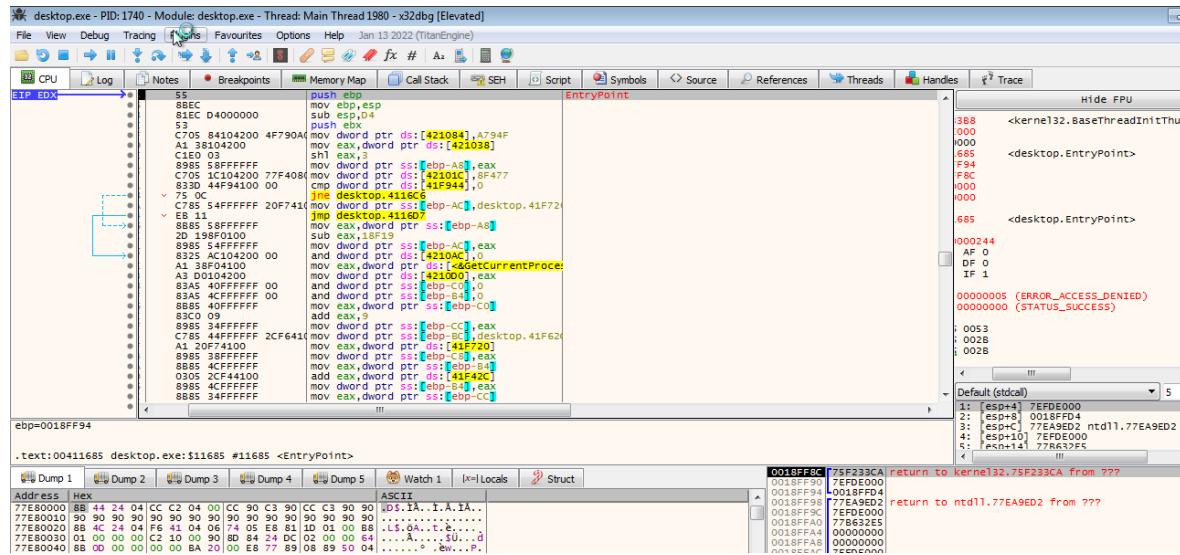
Here it has written to the exe file created and press f9 , now we can compare the hash of this exe and panda banker and they both are same.



Here we hit breakpoint at CreateProcessInternalW, now it is probably going to run the dropped file , now we can load the dropped exe in x32 dbg.



Here we load that exe .



Now we put breakpoint then put breakpoints on VirtualAlloc, CreateFileW, CreateProcessInternalW, WriteProcessMemory, CreateToolhelp32Snapshot(enumarate list of running

process in memory), CreateRemoteThread, ,ZwWriteVirtualMemory,NtResumeThread , press f9.

```

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
Type Address Module/Label/Exception State Disassembly Hits Summary
Software
75F21B86 <kernel32.dll.VirtualAlloc> Enabled mov edi,edi 0
75F21B8E <kernel32.dll.VirtualAlloc> Enabled mov edi,edi 0
75F33BF3 <kernel32.dll.CreateProcessInternal> Enabled push 64 0
75F3D9E0 <kernel32.dll.WriteProcessMemory> Enabled mov edi,edi 0
75F4435E <kernel32.dll.CreateToolhelp32Snapshot> Enabled push 40 0
75F44404 <kernel32.dll.CreateThread> Enabled mov edi,edi 0
77E8FE04 <ntdll.dll.ZwWriteVirtualMemory> Enabled mov eax,37 0
77E90058 <ntdll.dll.NtResumeThread> Enabled mov eax,4F 0

```

Here we hit breakpoint at VirtualAlloc just press f9.

```

CPU Log Notes Breakpoints VirtualAlloc Call Stack SEH Script Symbols Source References Threads Handles Trace
8BF mov edi,edi
55 push ebp
8BEC mov ebp,esp
5D pop ebp
EB 05 jmp <JMP.&VirtualAlloc>
90 nop
90 nop
90 nop
90 nop
90 nop
90 mov edi,edi
8BF push ebp
8BEC mov ebp,esp
5D pop ebp
EB 05 jmp <JMP.&VirtualFree>
90 nop
90 nop
90 nop
90 nop
90 nop
90 mov edi,edi
FF25 0809E275 jmp dword ptr ds:[<&VirtualAlloc>] JMP.&VirtualAlloc
90 nop
90 nop
90 nop
90 nop
90 nop
90 mov edi,edi
FF25 1009F275 jmp dword ptr ds:[<&VirtualFree>] JMP.&VirtualFree
90 nop
90 nop
90 nop
90 nop
90 nop
90

```

Registers:

```

edi=desktop.0040A642

```

Stack Dump:

```

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals Struct
Address Hex ASCII
77E80000 88 44 24 04 CC C2 04 00 CC 90 C3 90 CC C3 90 90 .0$..IA..I.A.IA..
77E80010 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .L$..A..t.e...A...
77E80020 88 4C 24 04 F6 41 04 06 74 05 E8 81 10 01 00 B8 .L$..A..t.e...A...
77E80030 01 00 00 00 C2 10 00 90 BD 84 24 DC 02 00 64 ....A...$U..d
77E80040 88 00 00 00 00 BA 20 00 E8 77 09 08 B9 50 00 00 ....B...P...
77E80050 64 A3 00 00 00 98 BD FC 24 OC FF D9 8B CC DE...X..S..O..I

```

Registers:

```

0018FD60 00409030 return_to_desktop.00409030 from ?
0018FD68 00000000
0018FD6C 00000000
0018FD70 00001000
0018FD74 00000000
0018FD78 0040A642 desktop.0040A642
0018FD79 00000000

```

Here we hit breakpoint at CreateFileW ,here it is opening a dll just press f9.

```

CPU Log Notes Breakpoints CreateFileW Call Stack SEH Script Symbols Source References Threads Handles Trace
8BF mov edi,edi
55 push ebp
8BEC mov ebp,esp
51 push ecx
FF75 08 push dword ptr ss:[ebp+8]
8045 F8 test eax,eax
F1F1 5005F275 call dword ptr ds:[<_KtInitUnicodeString>]
85C0 test eax,esx
000870200 KtInitUnicodeString
FF75 0C push dword ptr ss:[ebp+C]
8D45 F8 lea eax,[ebp+C]
push byte ptr ss:[ebp+8]
FF75 14 push byte ptr ss:[ebp+14]
FF75 18 push dword ptr ss:[ebp+18]
FF75 1A push dword ptr ss:[ebp+1A]
FF75 1C push dword ptr ss:[ebp+1C]
FF75 1E push dword ptr ss:[ebp+1E]
FF75 20 push dword ptr ss:[ebp+20]
FF75 24 push dword ptr ss:[ebp+24]
FF75 28 push dword ptr ss:[ebp+28]
C9 leave
FF75 2C ret ic
90
90
43 inc ebx
004F 00 add byte ptr ds:[edi],c1

```

Registers:

```

edi=0018FB60

```

Stack Dump:

```

Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals Struct
Address Hex ASCII
77E80000 88 44 24 04 CC C2 04 00 CC 90 C3 90 CC C3 90 90 .0$..IA..I.A.IA..
77E80010 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .L$..A..t.e...A...
77E80020 88 4C 24 04 F6 41 04 06 74 05 E8 81 10 01 00 B8 .L$..A..t.e...A...
77E80030 01 00 00 00 C2 10 00 90 BD 84 24 DC 02 00 64 ....A...$U..d
77E80040 88 00 00 00 00 BA 20 00 E8 77 09 08 B9 50 00 00 ....B...P...
77E80050 64 A3 00 00 00 98 BD FC 24 OC FF D9 8B CC DE...X..S..O..I

```

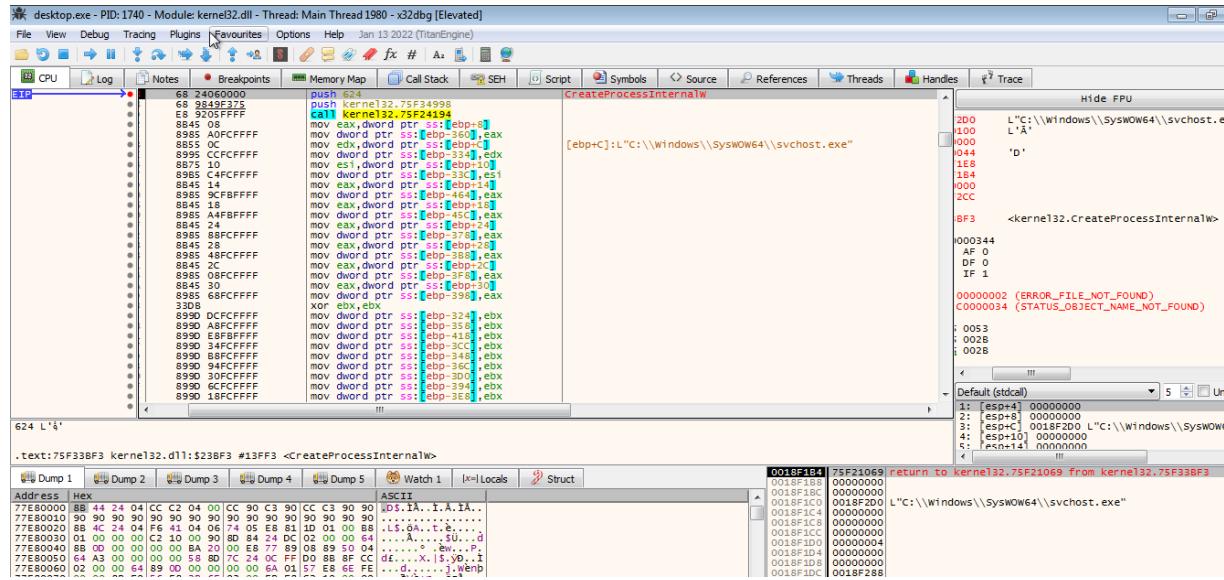
Registers:

```

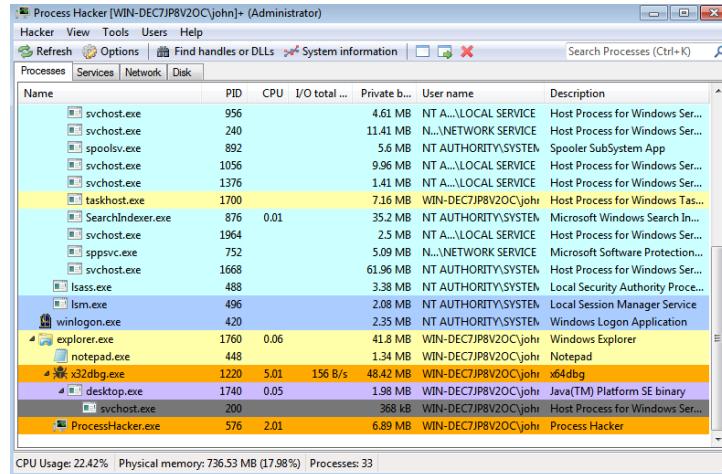
0018FA9C 71F692D0 return_to_Cryptsp.71F692D0 from ???
0018FC18 L'C:\Windows\system32\rsaenh.dll
0018FAA8 00000001
0018FAAC 00000000
0018FAB0 00000003
0018FAB2 00000000
0018FAB8 00000000
0018FABC 0018FB48

```

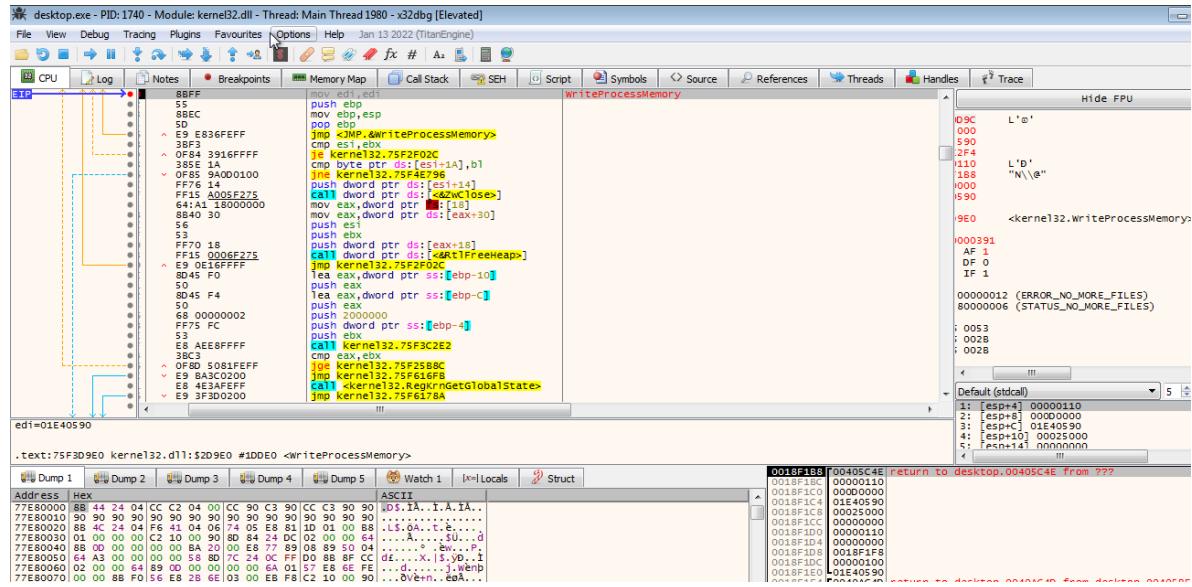
Now we hit breakpoint at CreateProcessInternalW, here it is probably writing in the memory of svchost.exe so we can use process hacker to verify it, press f9.



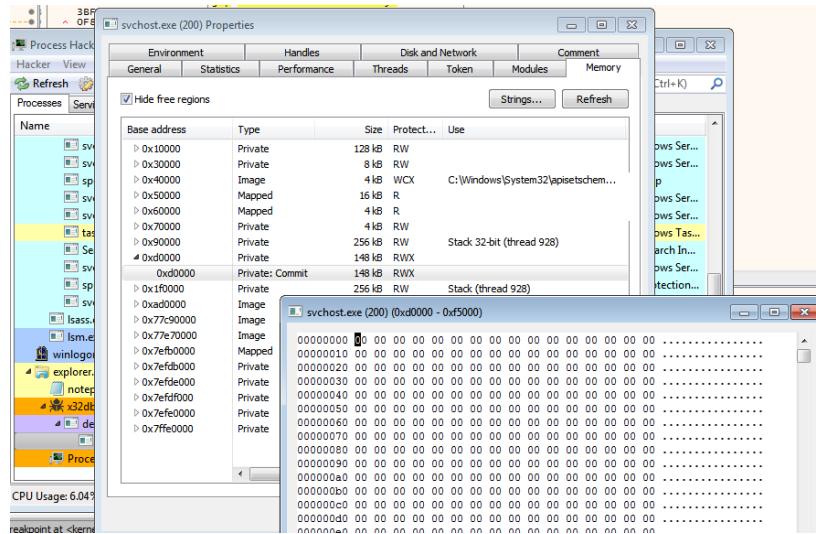
Here we can see now it has created a process svchost.exe in suspended state.



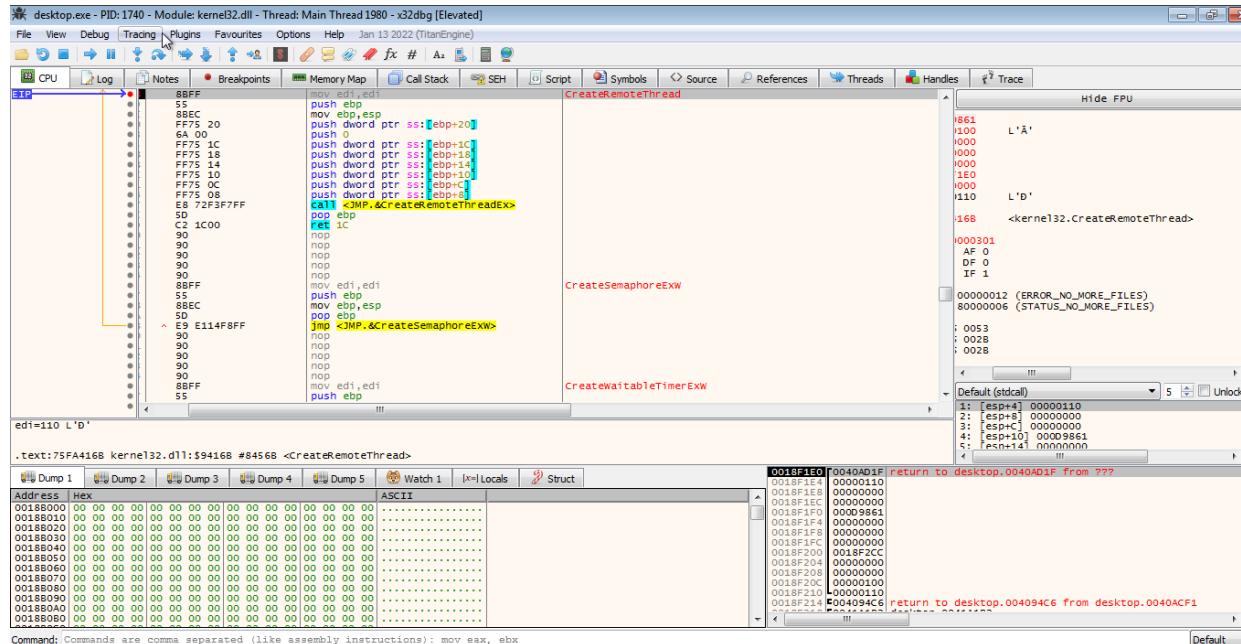
Now here we hit breakpoint at WriteProcessMemory here it is in memory of svchost.exe , here we can note the second parameter for this call i.e D0000 this is the region where it is writing to.



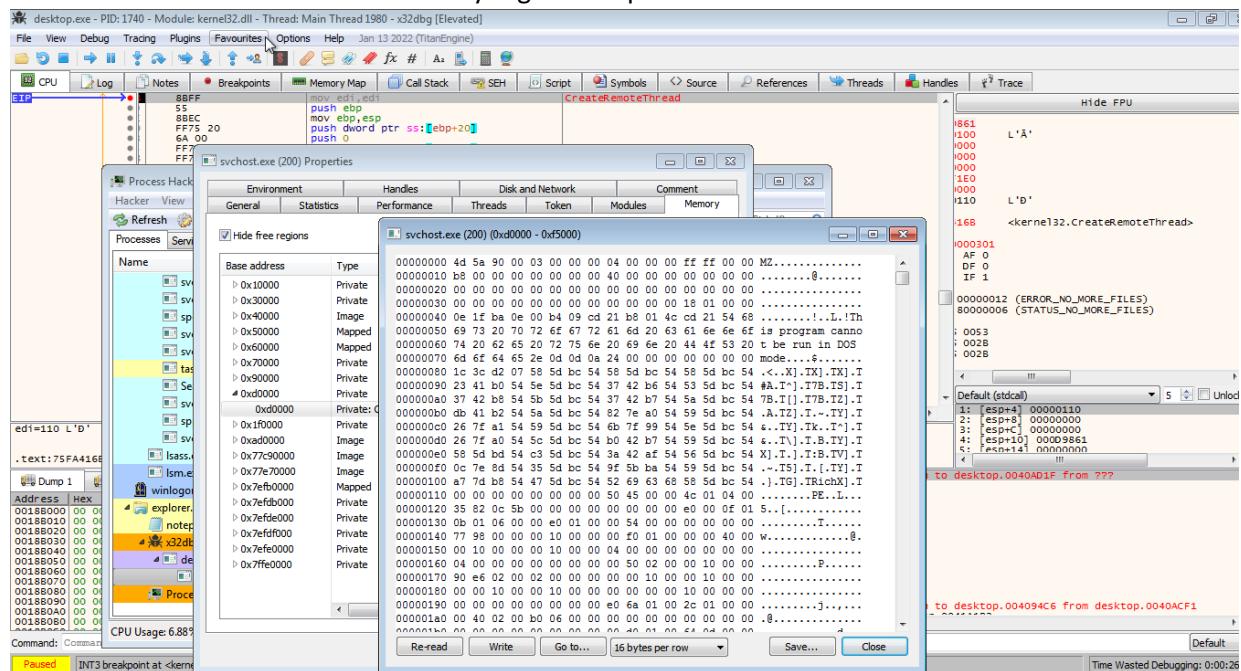
Here in process hacker we analysed memory for that address i.e d0000 we can also note here that permission of this region is read,write and execute, here this malware is trying to write to this location and then will execute it, here now just press f9.



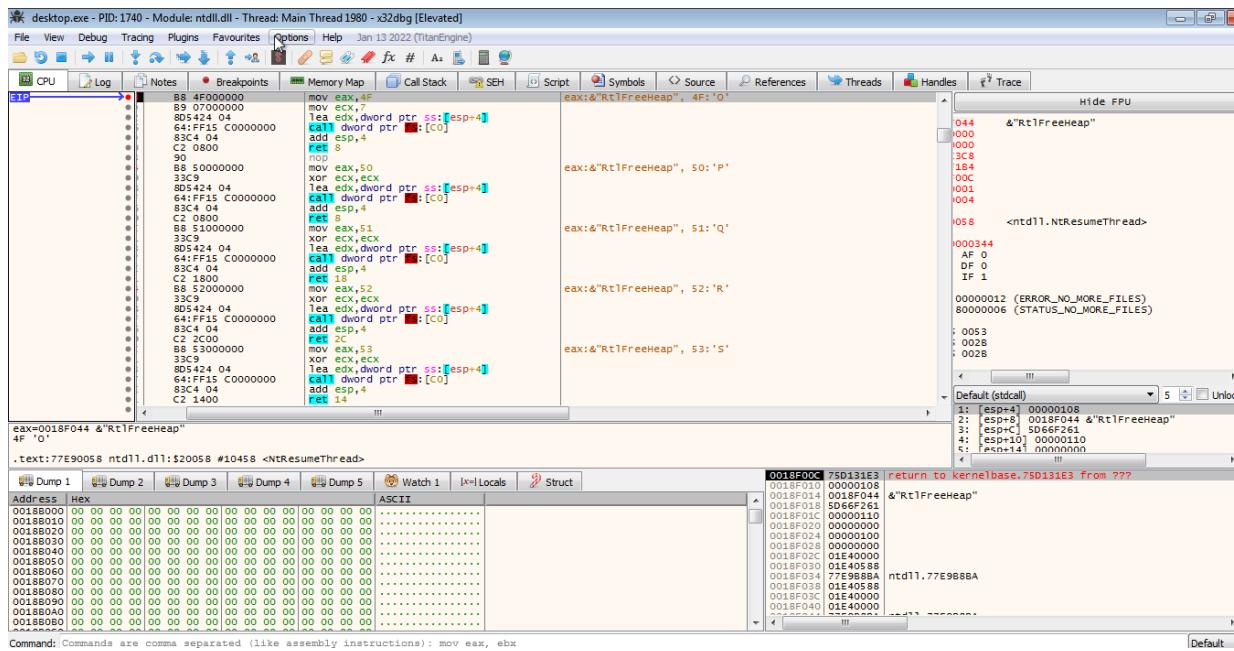
Now we hit breakpoint at CreateRemoteThread , and now it had written to svchost.exe we can verify it by viewing it in process hacker.



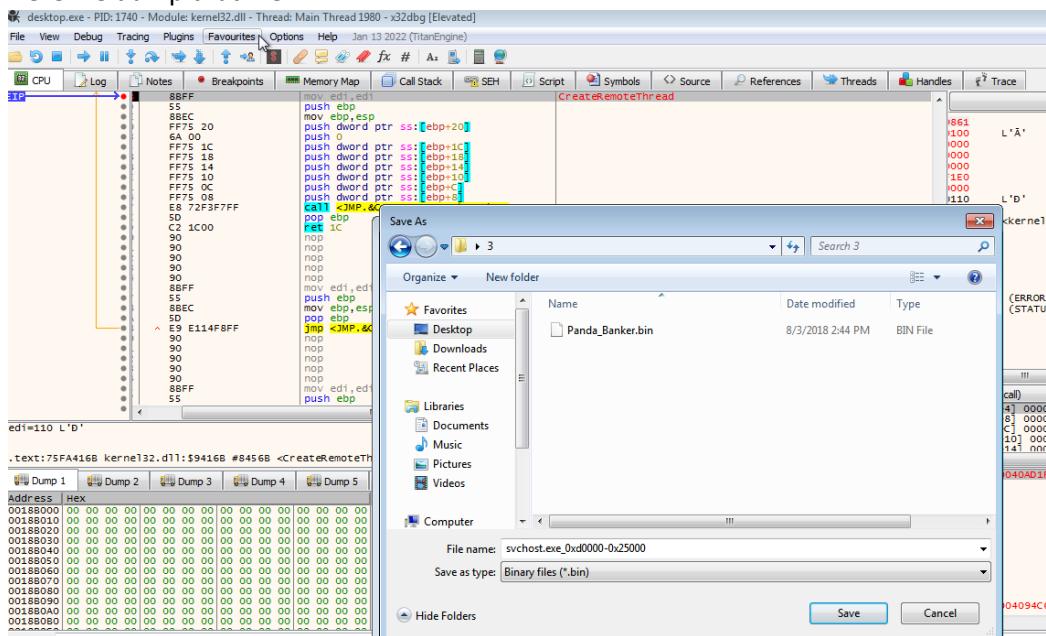
Here it has now written to that memory region and press f9.



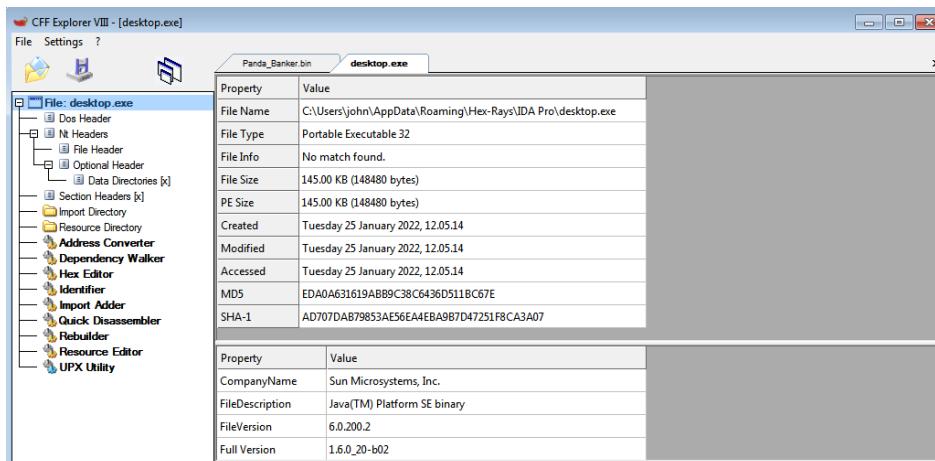
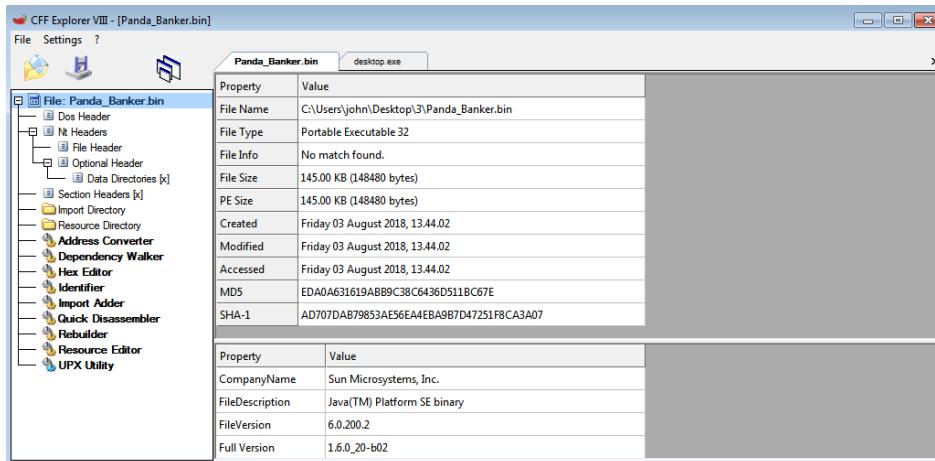
Now here we can see that now it is trying to resume thread created so now we can dump that memory region



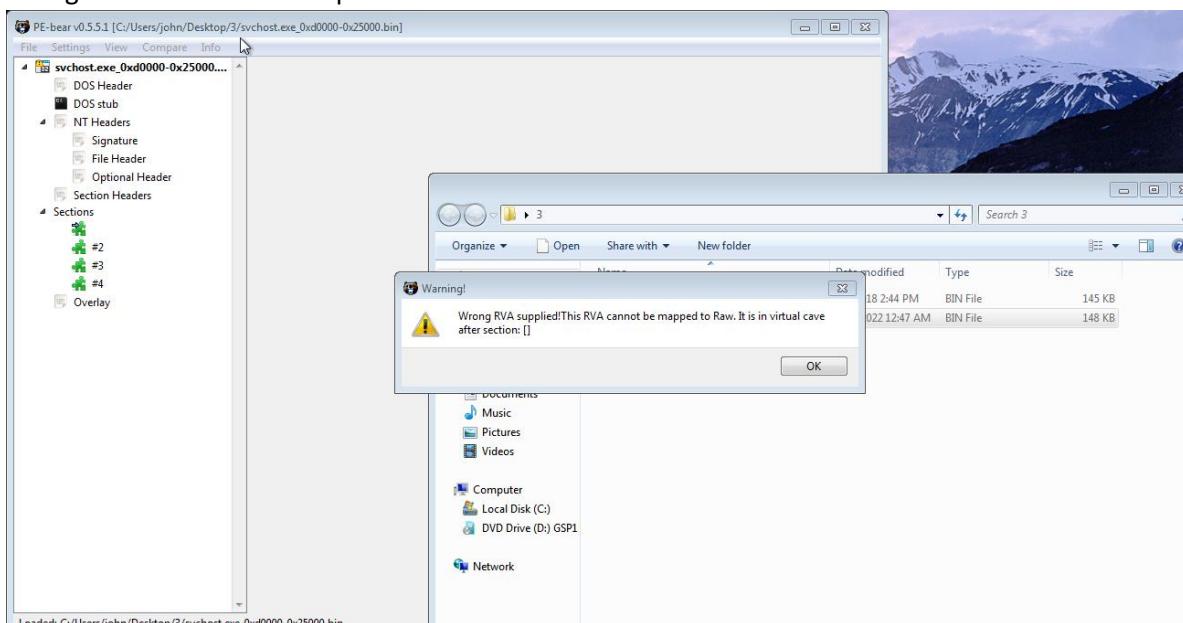
Here we dump that file.



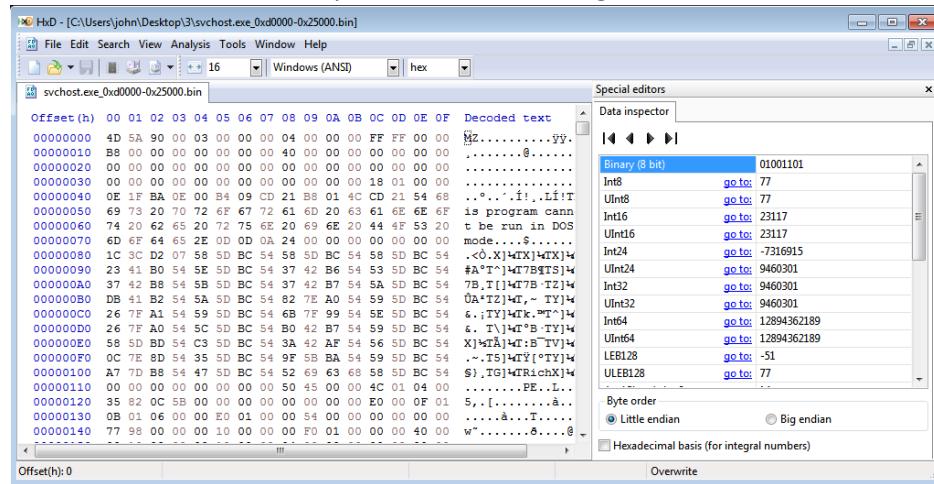
Now we load that file in CFF Explorer.



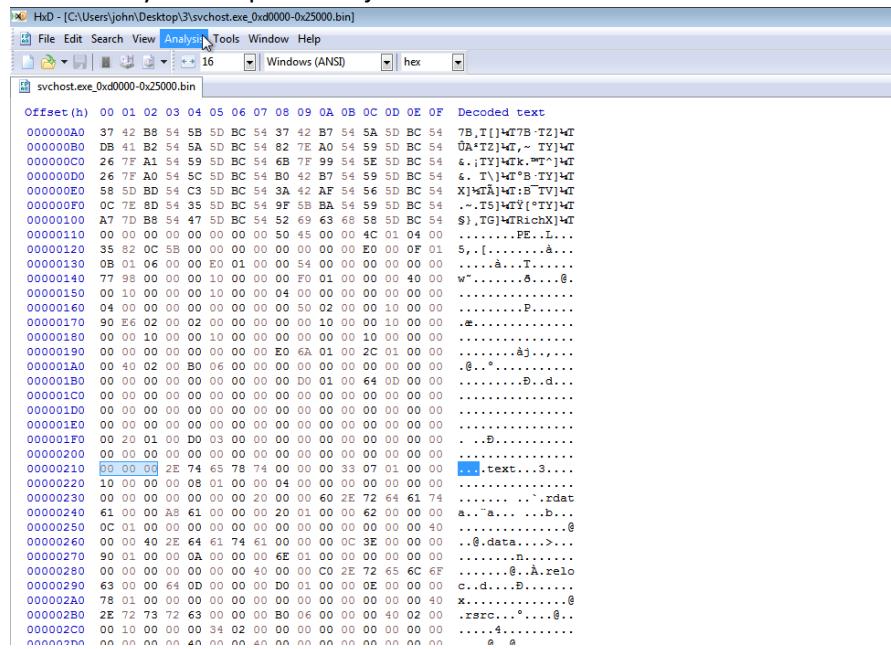
Here now we will fix the section alignments , section headers so we load the sample in PE-bear, here it is showing warning because file had been mapped to the memory therefore all the section header are out of alignment now close the pe-bear .



Now we have load the sample in hex editor to align the section header.



Here we can note that section header got three null bytes that means it is not aligned , so cut these three nul bytes and put them just before the end of section headers.



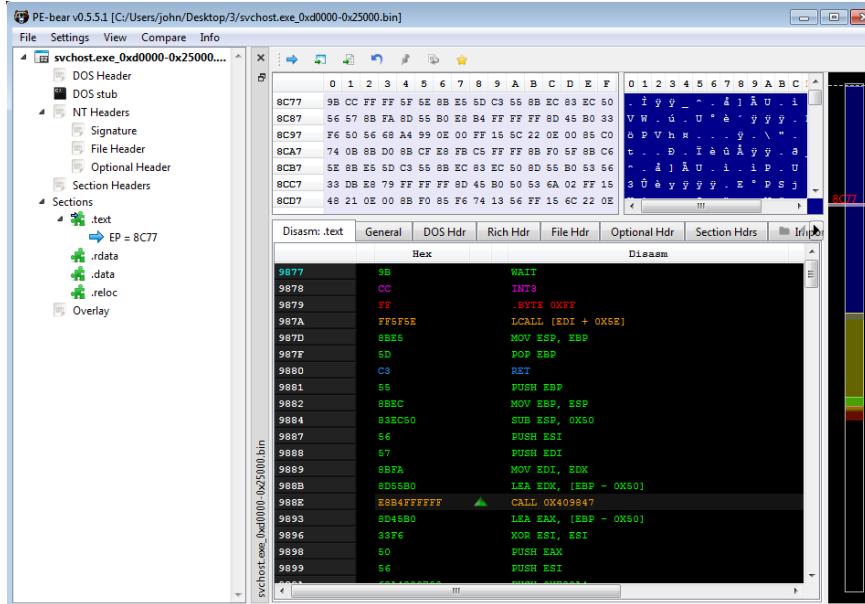
Here we have inserted those null bytes.

The screenshot shows the HxD hex editor interface with the file 'svchost.exe_0xd0000-0x25000.bin' open. The 'Analysis' tab is selected. The code window displays assembly-like instructions and their corresponding decoded text. A series of null bytes (00) are inserted at various memory addresses, such as 0x000000A0 through 0x000000F0, which are highlighted in blue. The text column shows characters like '7B,T[1]WT'B-TZ]4T', '0A:T2]4T,~ TY]4T', and 'X]WT'B-TV]4T'. The 'Decoded text' column shows the actual characters or symbols from the assembly.

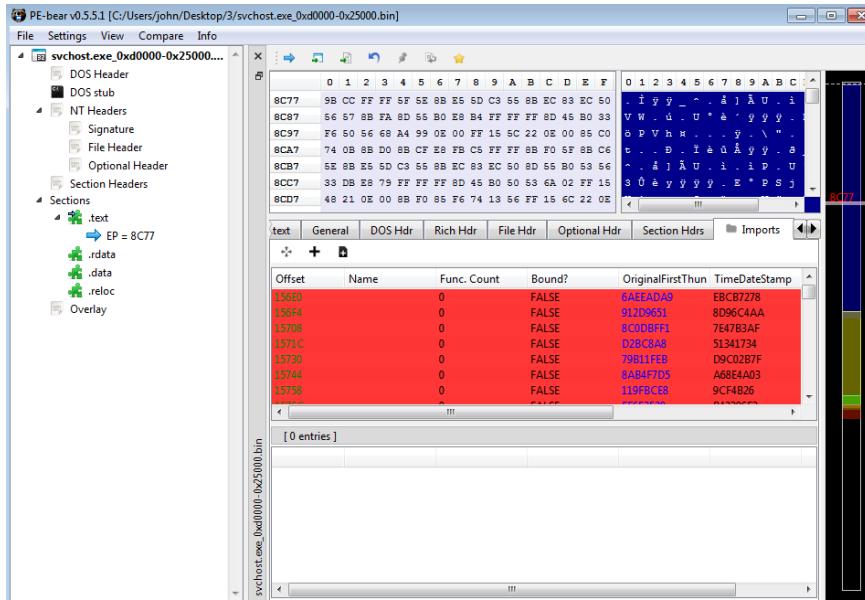
Here now we can see that it is properly aligned , now save it.

The screenshot shows the HxD hex editor interface with the same file and settings as the previous one. The 'Analysis' tab is still selected. The code window now shows the same assembly and text as before, but the null bytes (00) have been removed, indicating proper alignment. The text and decoded text columns remain the same as in the first screenshot.

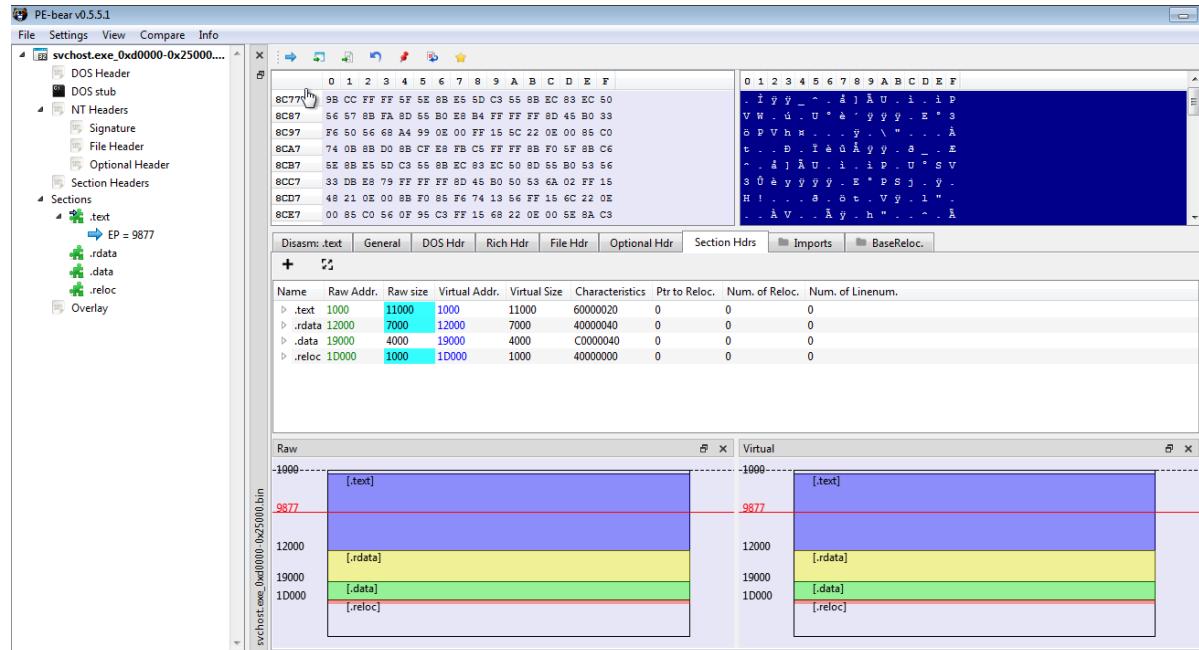
Now we load the sample in PE-bear , here we can note that now name of section headers are visible to us now we will unmap the section headers.



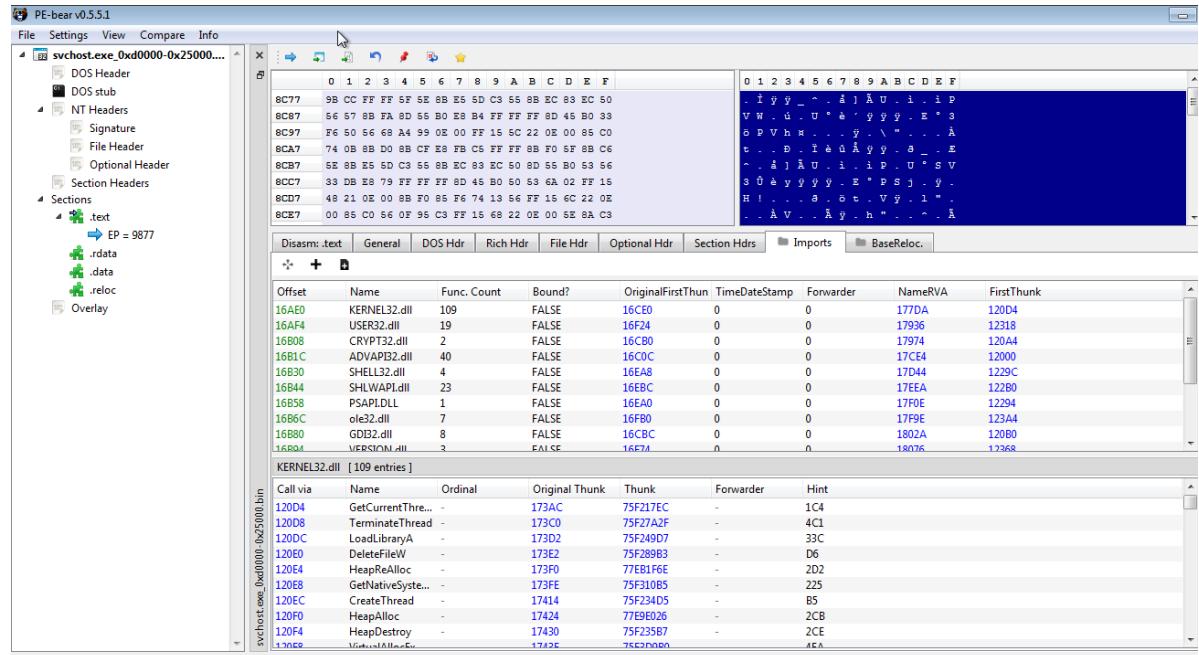
Here we can see that all the imports are distorted so now we need to fix them too.



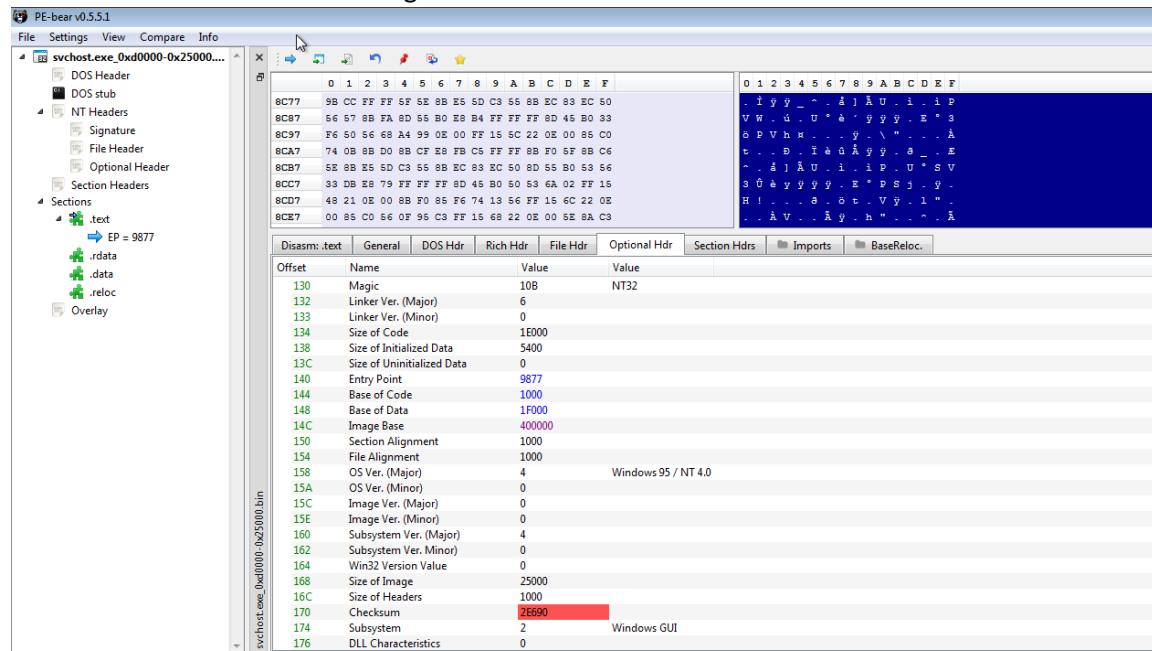
Unmap is necessary because all the raw address is mapped from memory so we need to make the raw address same as virtual address and then calculate the raw size and make virtual size same as raw size.



Here now we can see that all imports are fixed now.



Now here we need to fix the Image base to d0000.



Here we have fixed it and save it.

