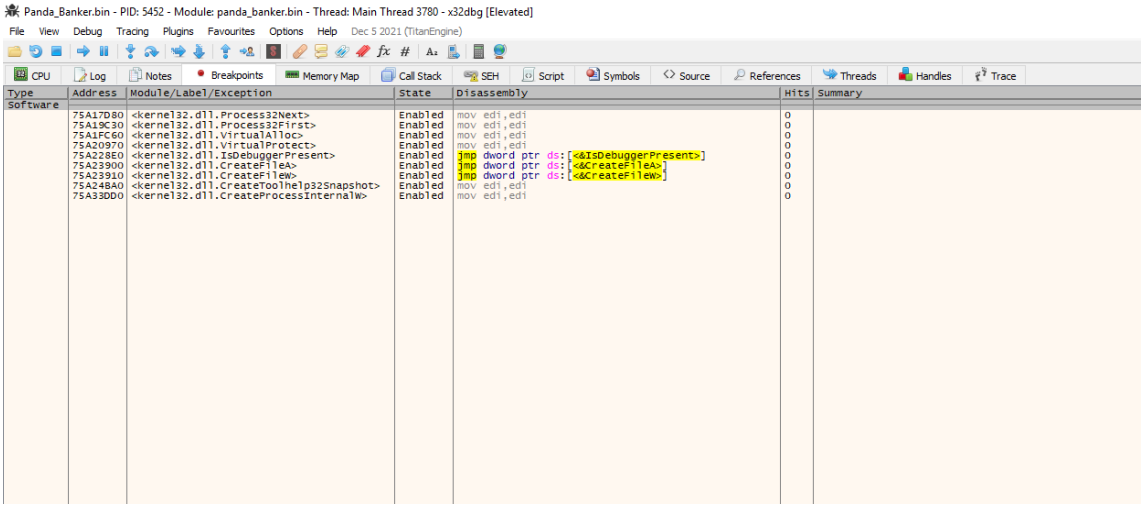
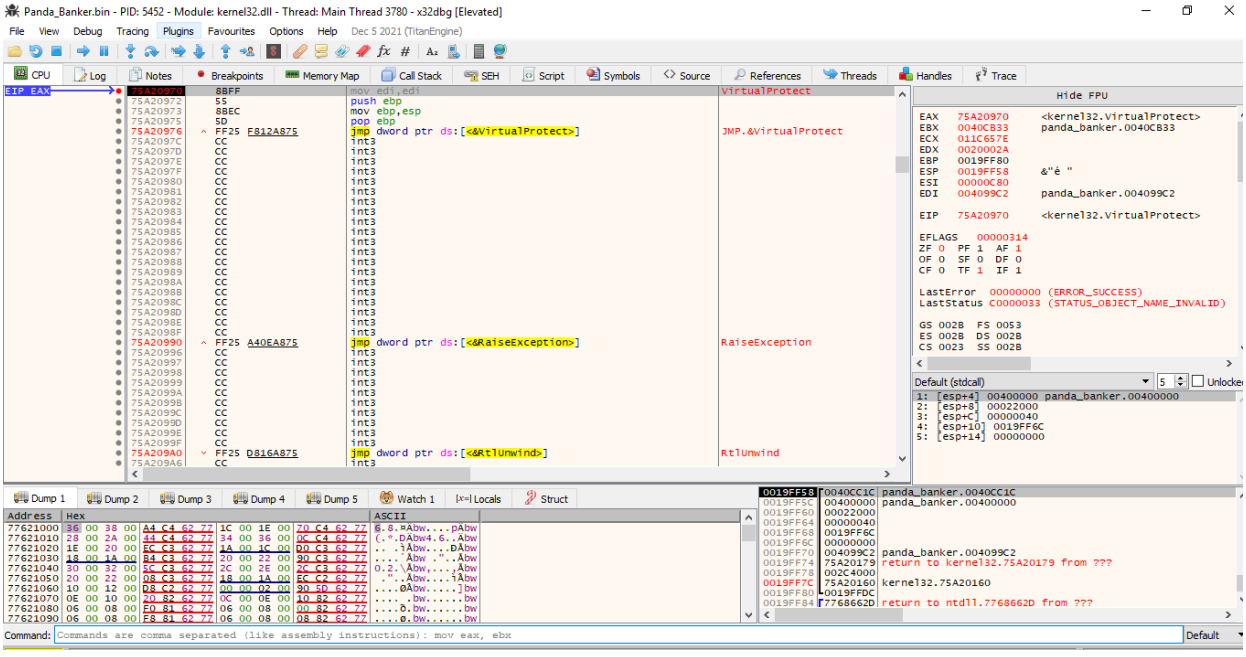
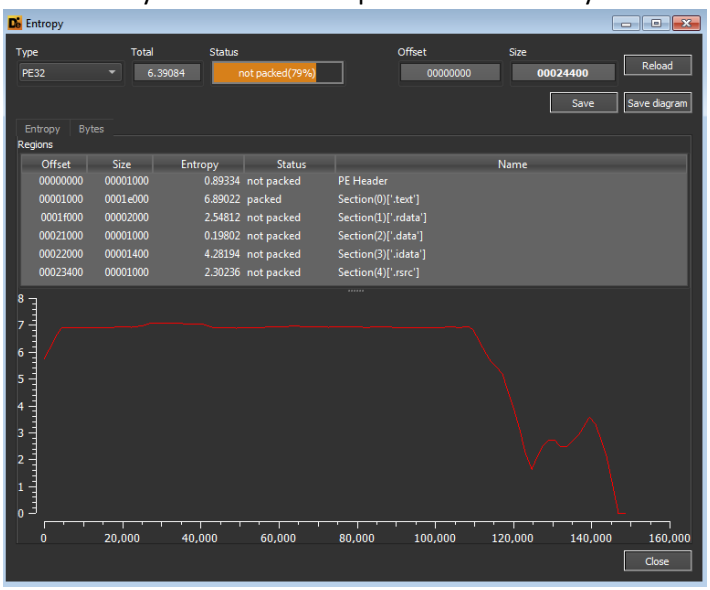
Here we load the sample in x32dbg and then put breakpoints on VirtuaalAlloc, VirtualProtect, IsDebuggerPresent, CreateToolhelp32Snapshot(enumerate list of running process in memory), Process32First, Process32Next, CreateFileA, CreateFileW, CreateProcessInternalW.



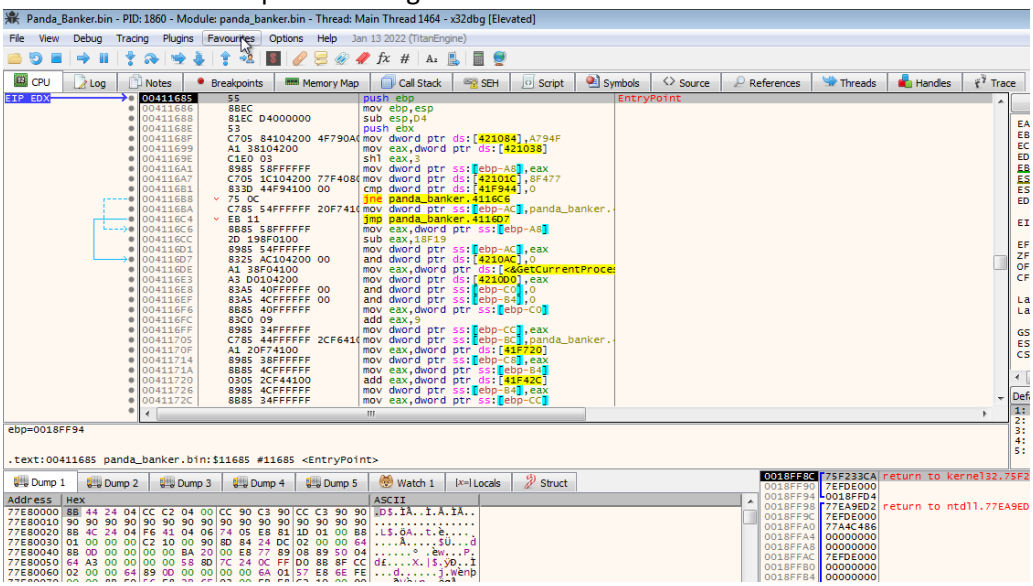
Now we hit our first breakpoint at VirtualProtect(it will change the permission bits for the memory region which is passed to it in second parameter), here just press f9.



Here initially we load the sample at Detect It Easy.

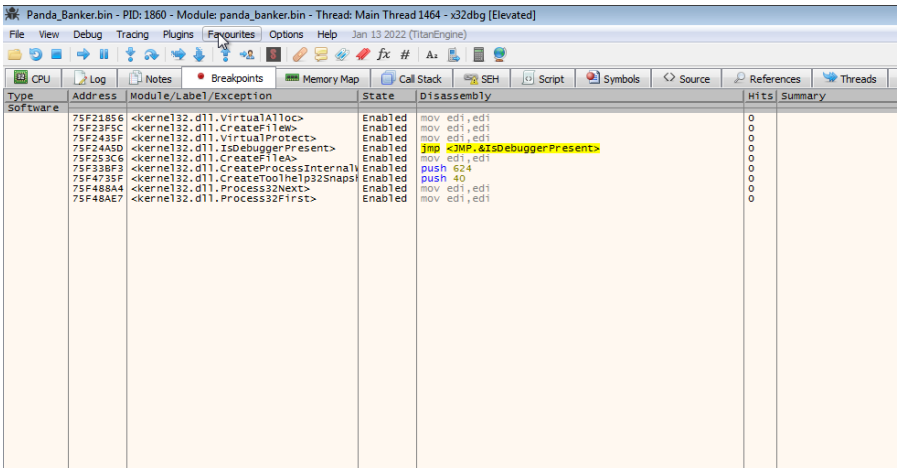


Now we load the sample in x32dbg.

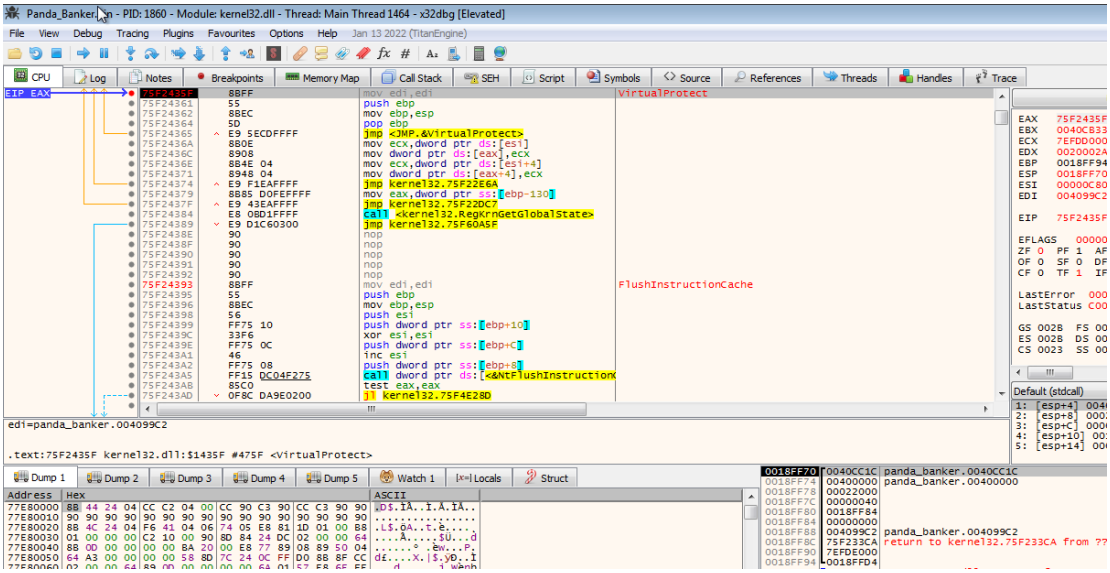


Here we load the sample in x32dbg and then put breakpoints on VirtuaalAlloc, VirtualProtect, IsDebuggerPresent, CreateToolhelp32Snapshot(enumerate list of running process in memory),

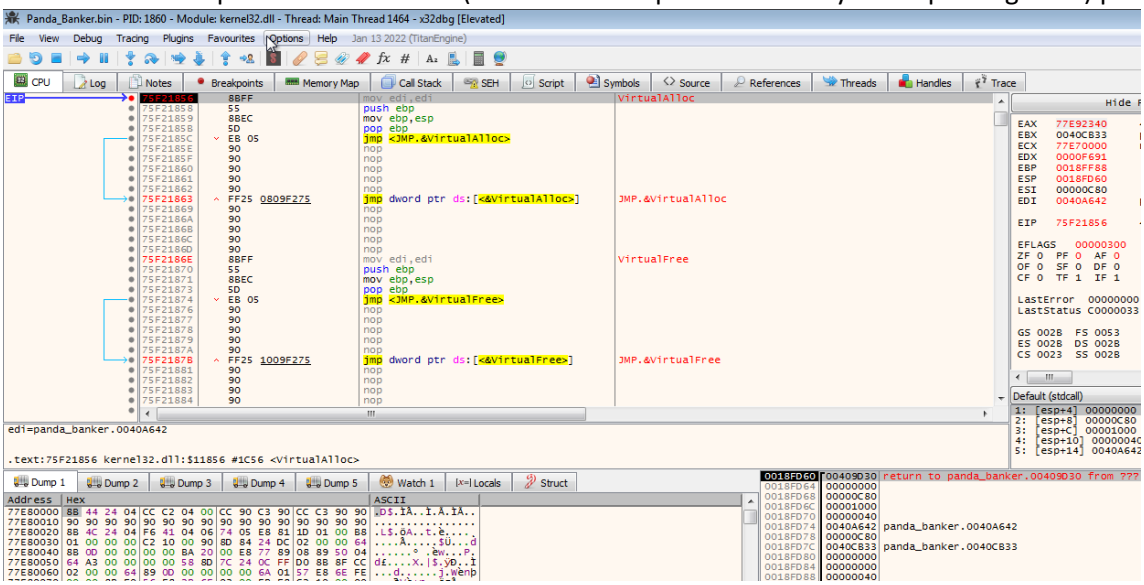
Process32First, Process32Next, CreateFileA, CreateFileW, CreateProcessInternalW.



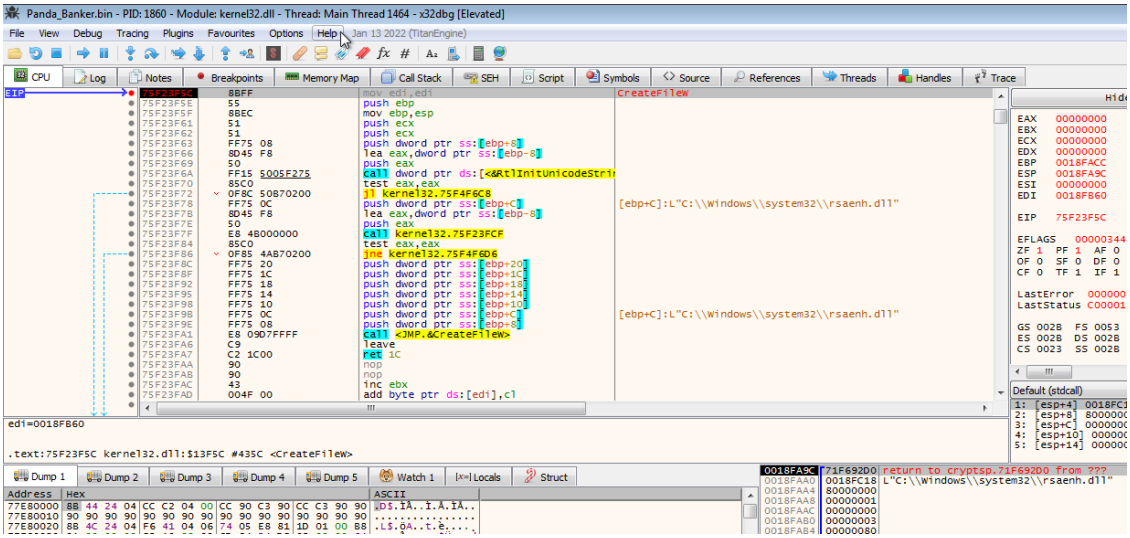
Here we hit breakpoint at VirtualAlloc press f9.



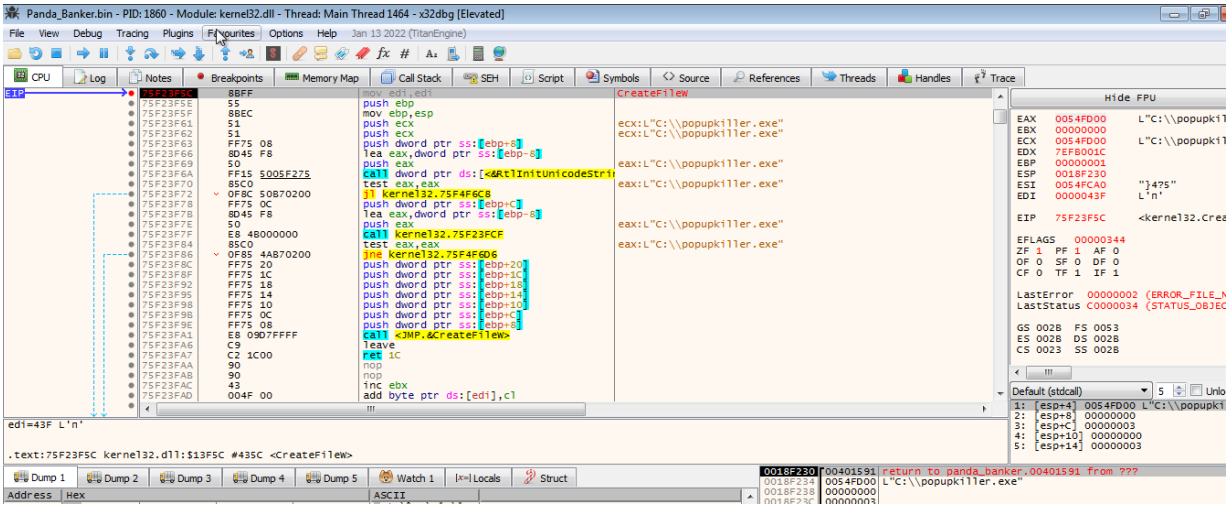
Here we hit breakpoint at VirtualAlloc (it will allocate space in memory for unpacking code) press f9.



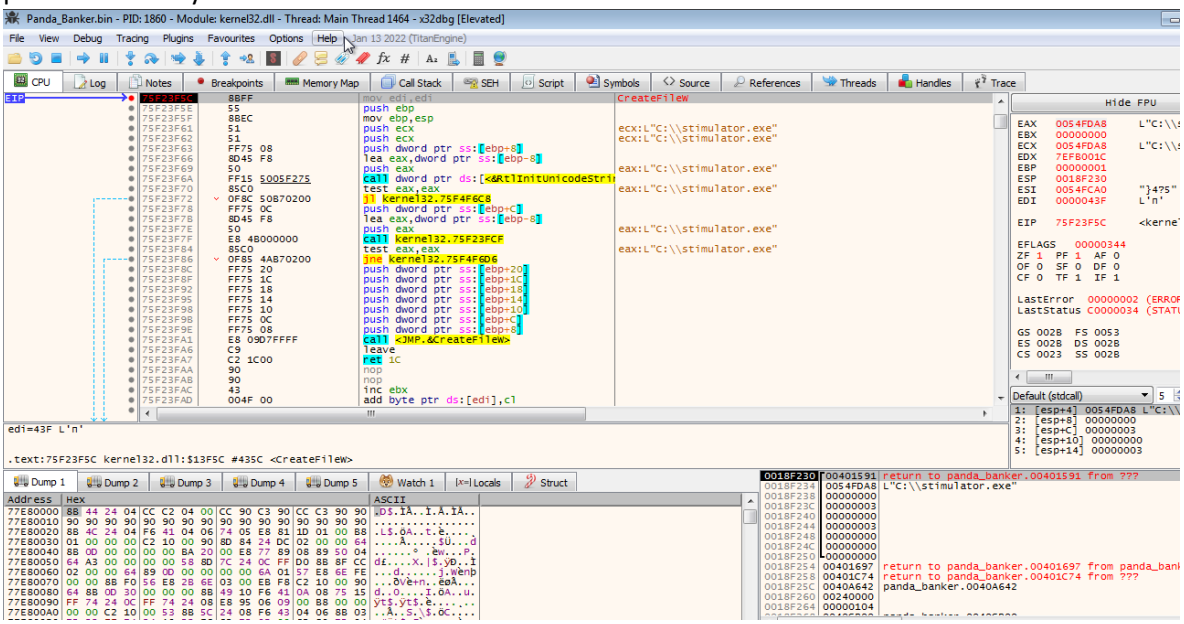
Here we hit breakpoint at CreateFileW(here it will read some file) press f9.



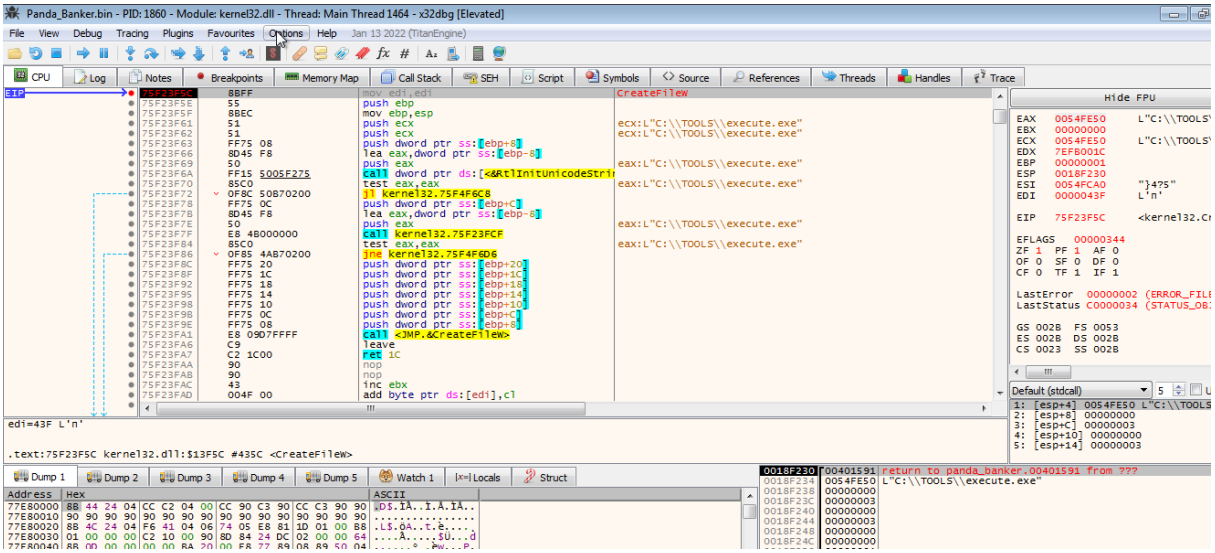
Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system.



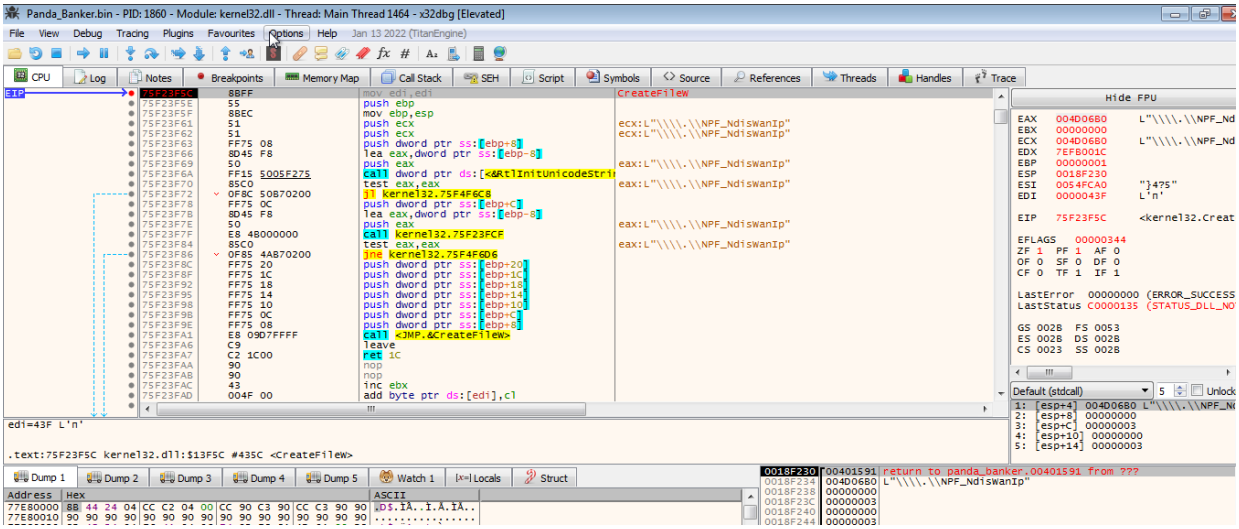
Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system.



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

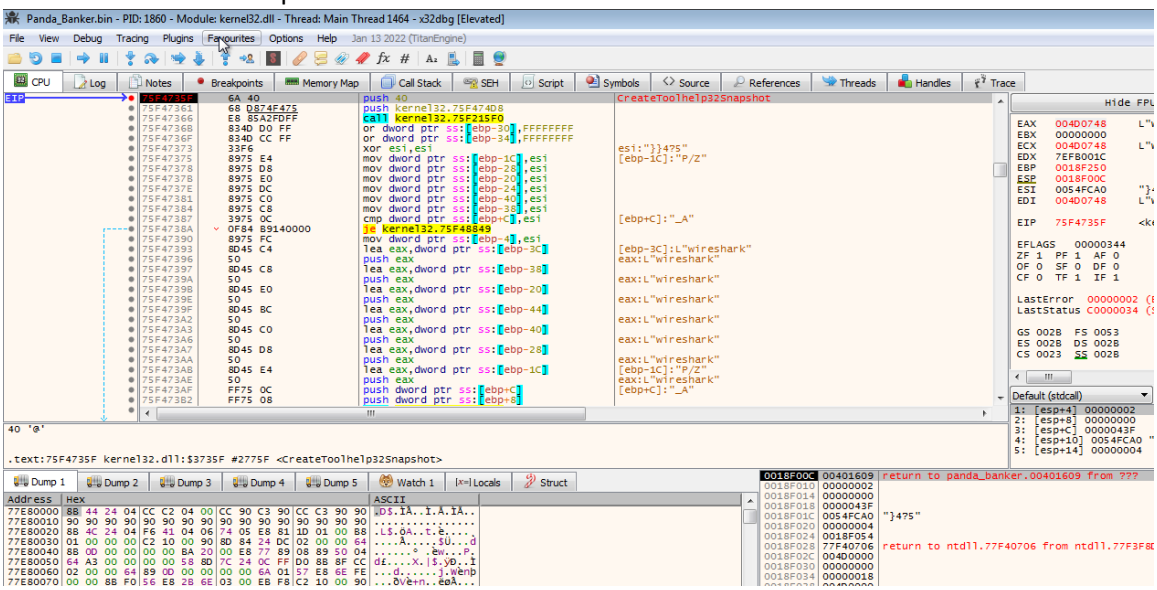


Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system, here it is the library of wireshark if we try to run the debugging will be terminated, so here we need to uninstall the library Winpcap and wireshark.

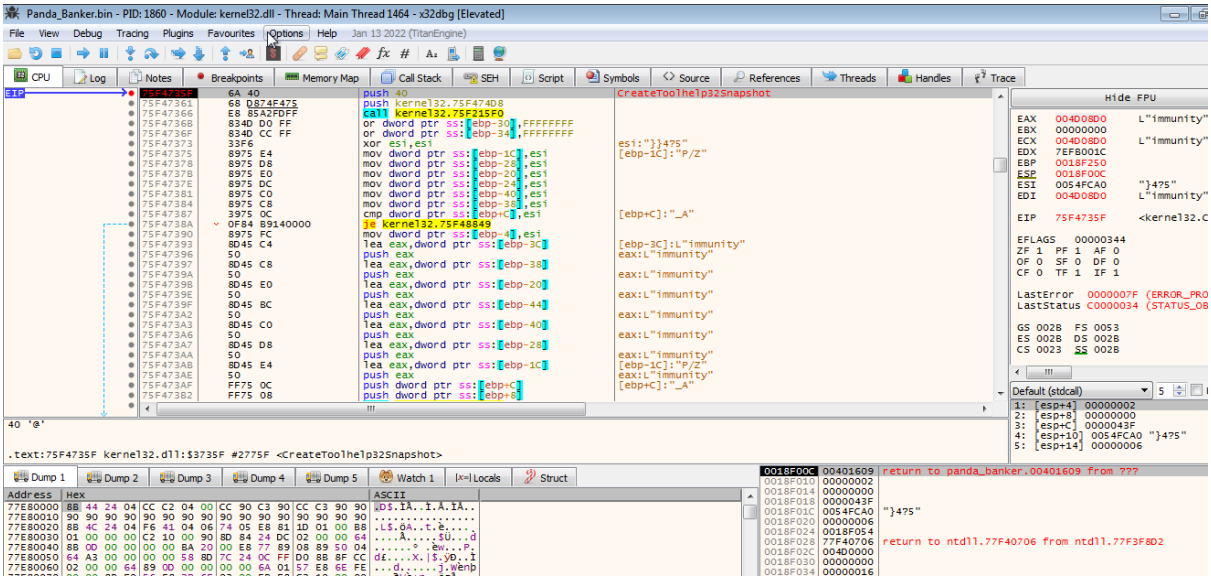


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the wireshark is running in memory since we did not

start wireshark we can press f9 here.

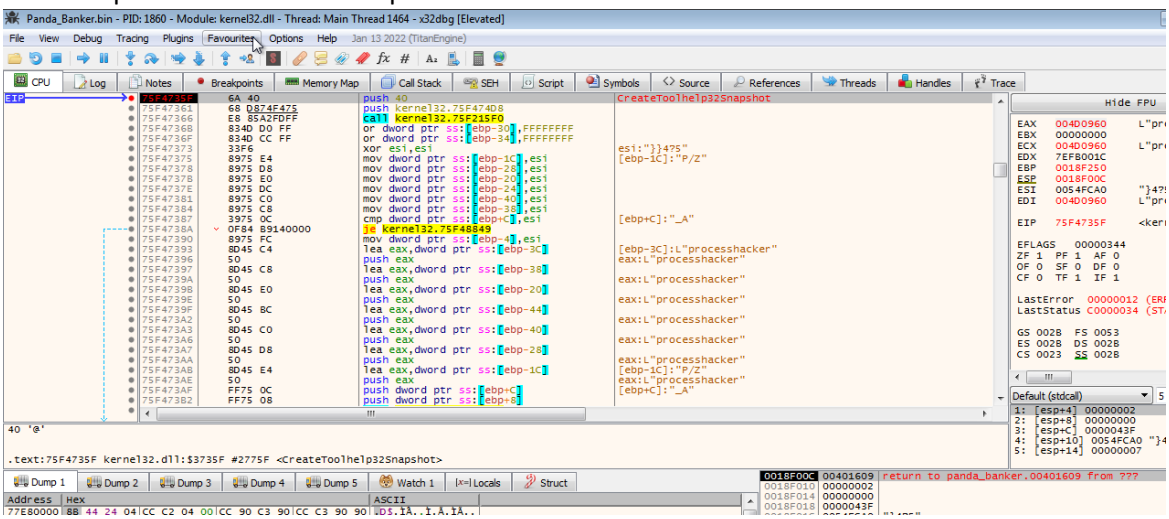


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the immunity debugger is running in memory since we did not start immunity debugger we can press f9 here.

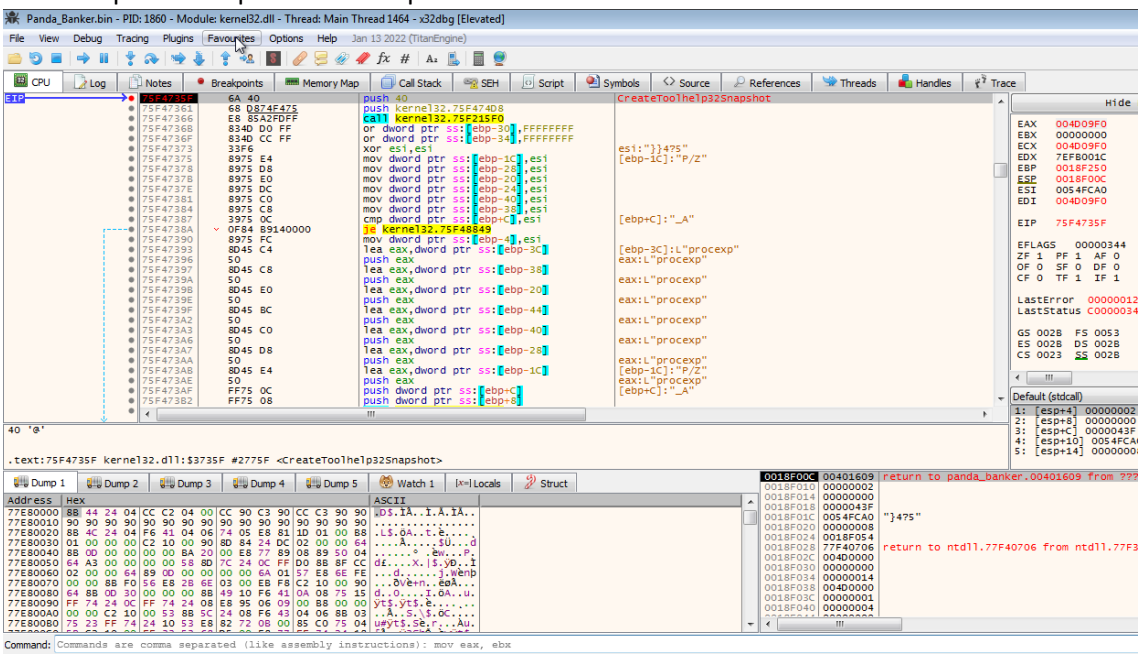


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the process hacker is running in memory since we did

not start process hacker we can press f9 here.

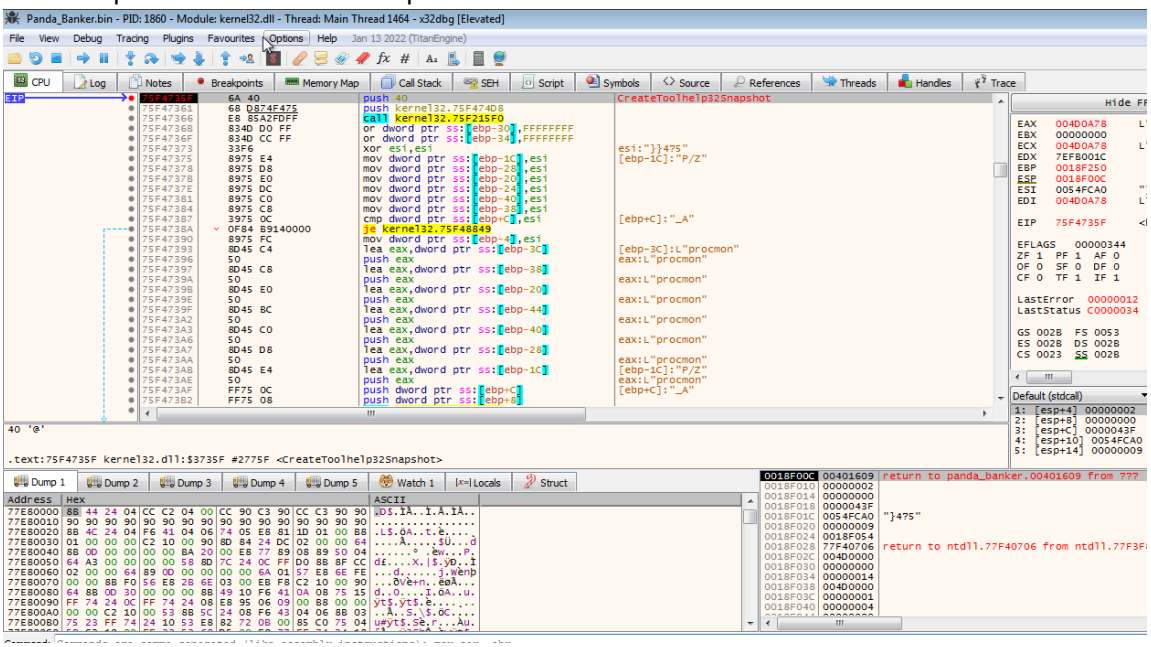


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the process explorer is running in memory since we did not start process explorer we can press f9 here

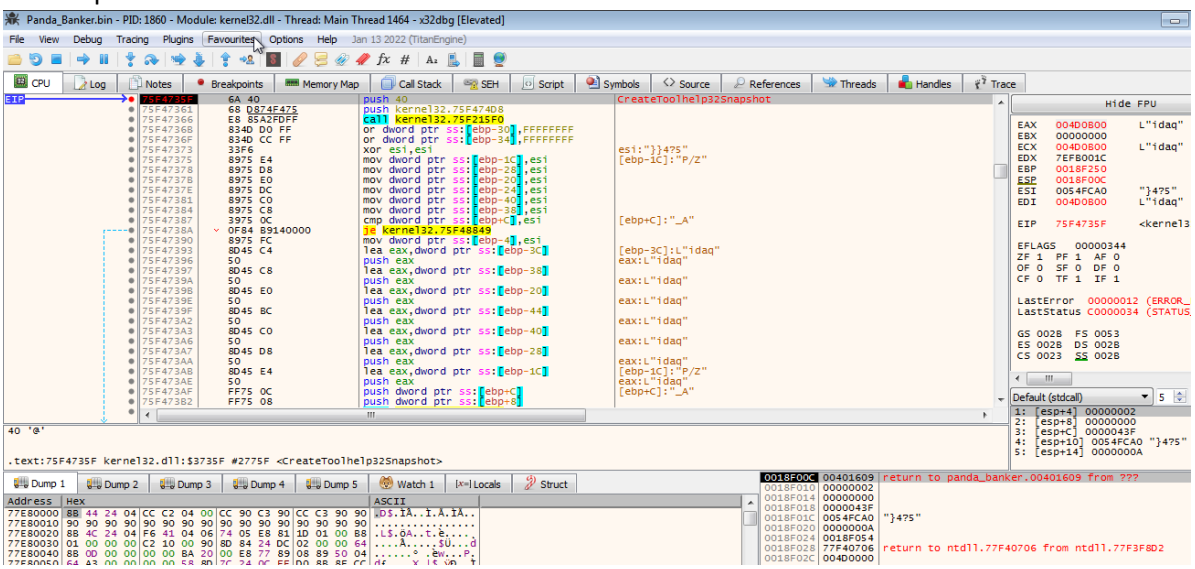


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the process monitor is running in memory since we did

not start process monitor we can press f9 here.

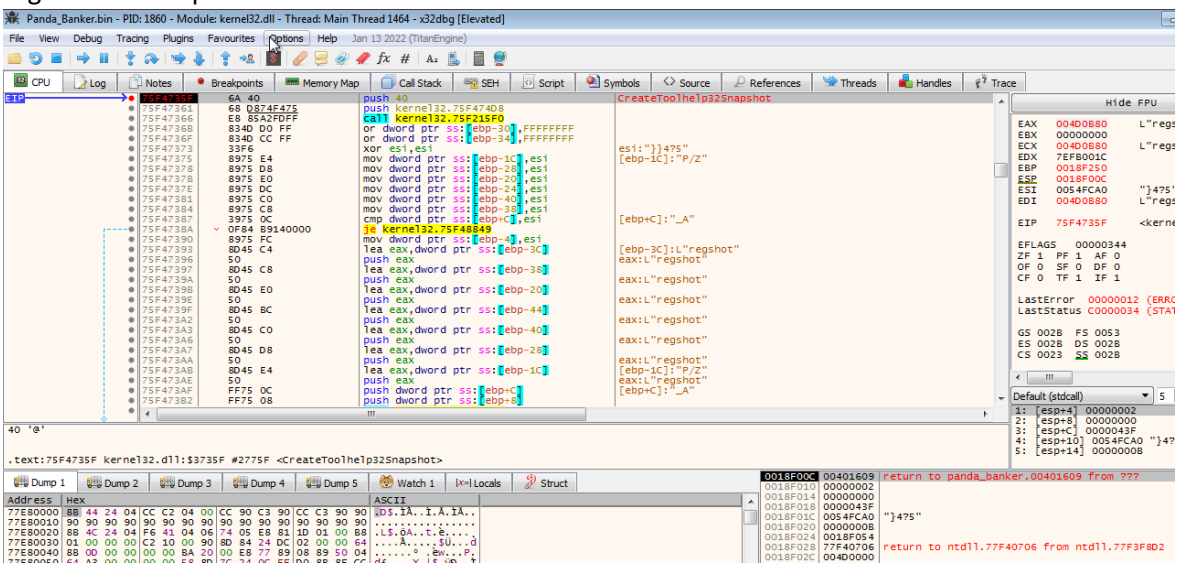


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the IDA is running in memory since we did not start IDA we can press f9 here.

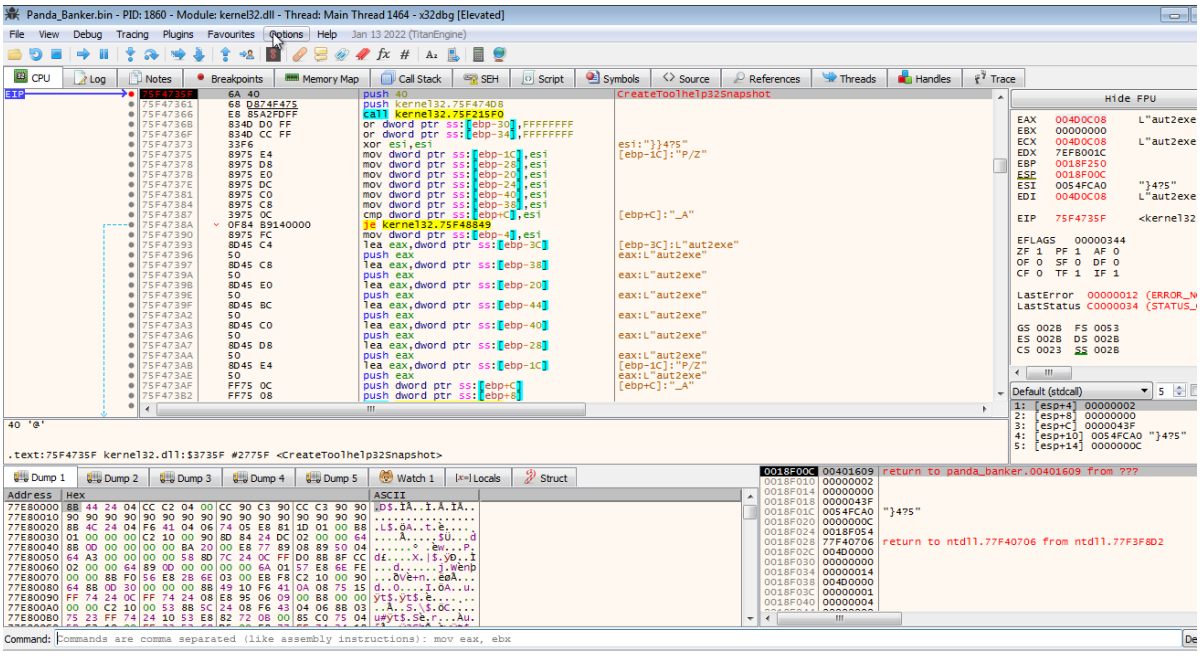


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the regshot is running in memory since we did not start

regshot we can press f9 here.

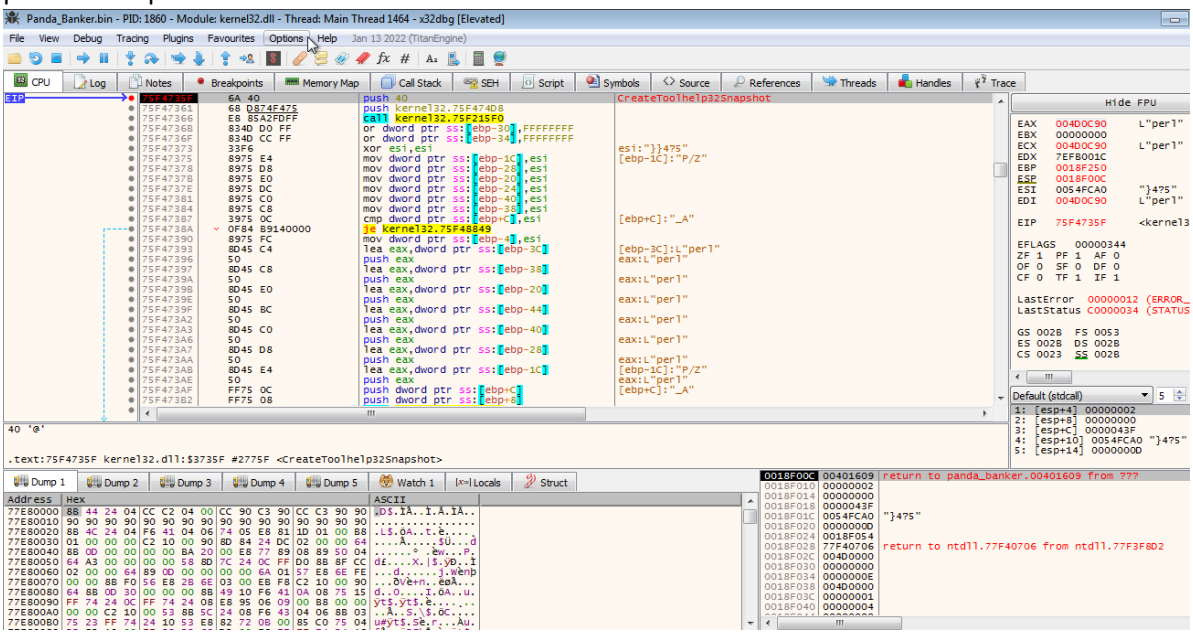


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the autoit is running in memory since we did not start autoit we can press f9 here.

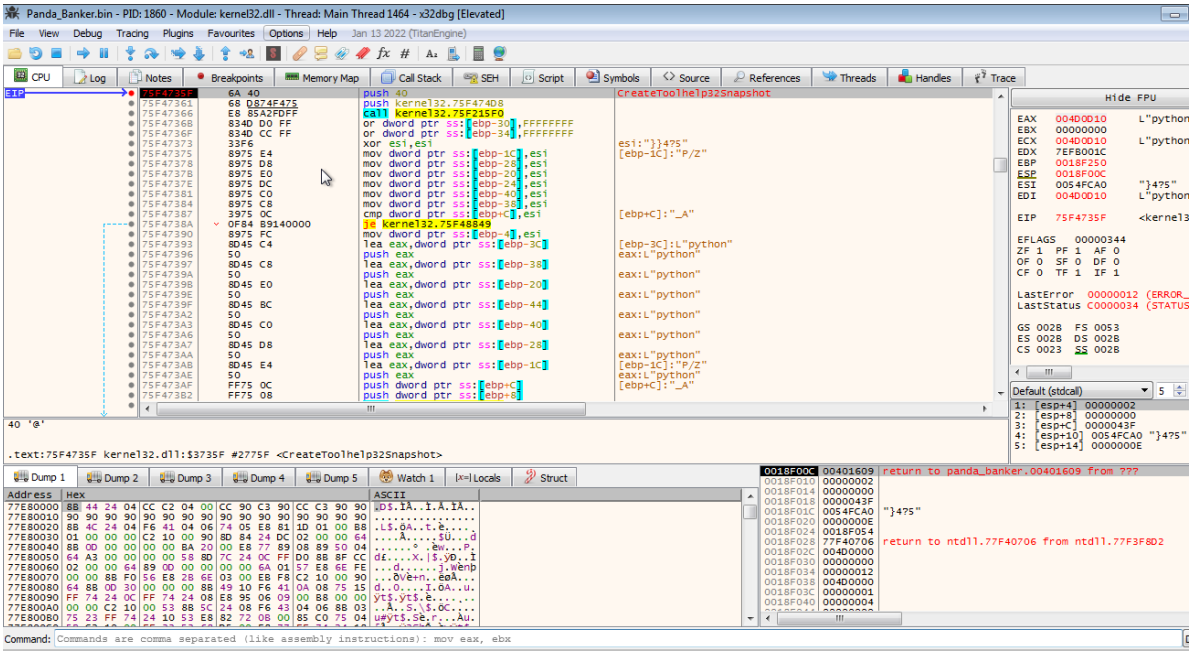


Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the perl is running in memory since we did not start

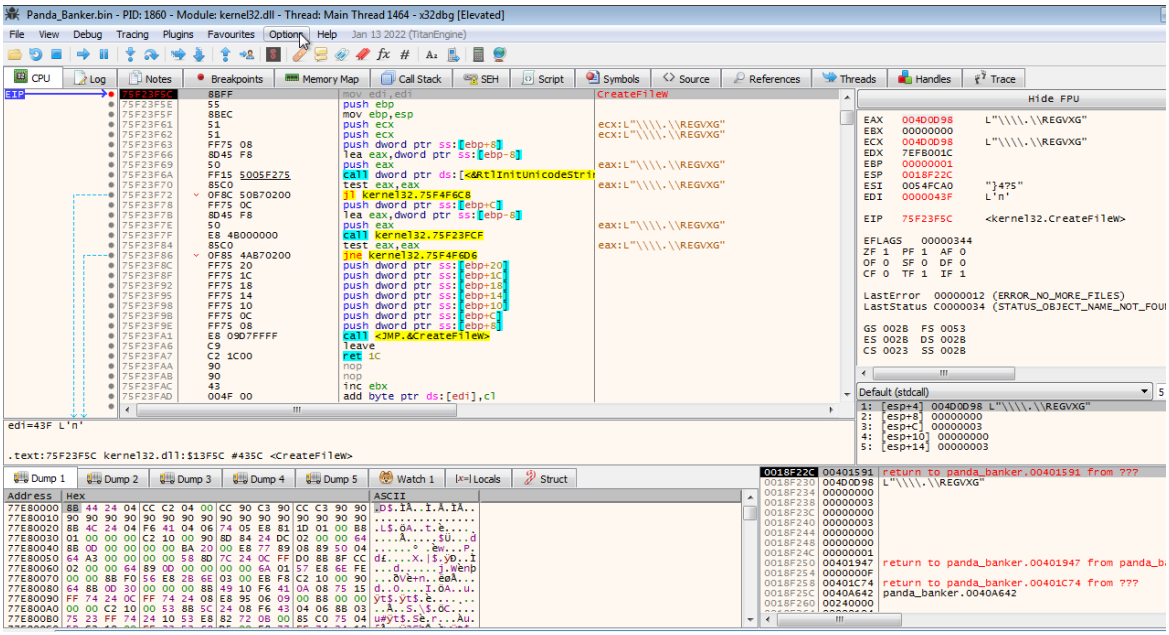
perl we can press f9 here.



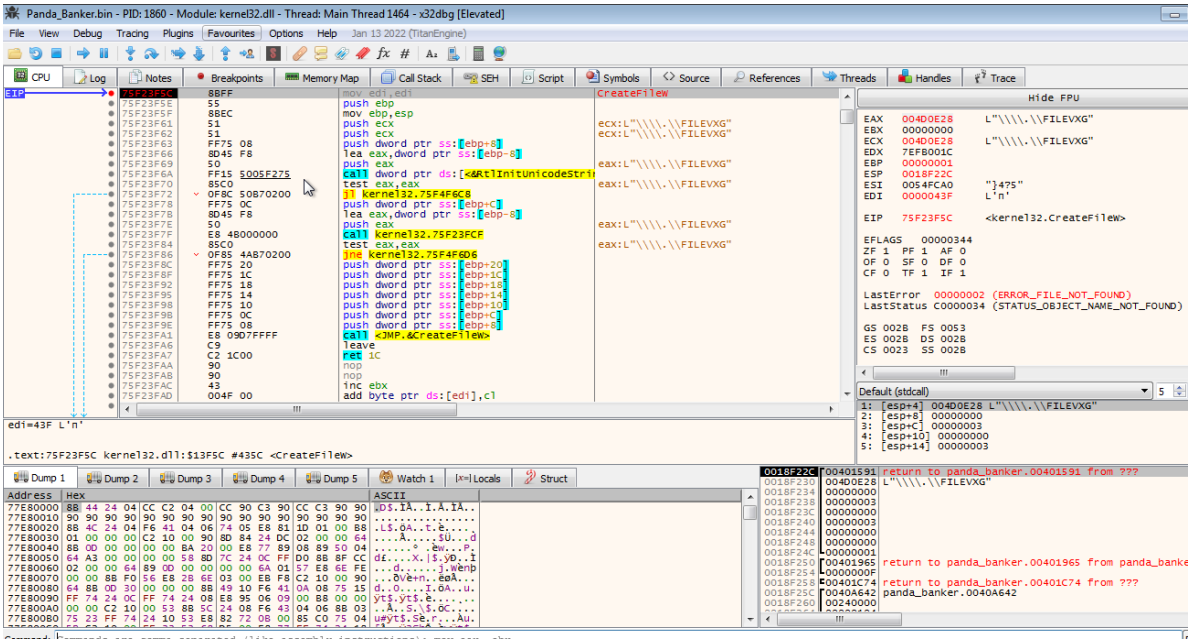
Here we hit breakpoint at CreateToolhelp32Snapshot (here it is checking what processes are already running in memory), here it is checking whether the python is running in memory since we did not start python we can press f9 here.



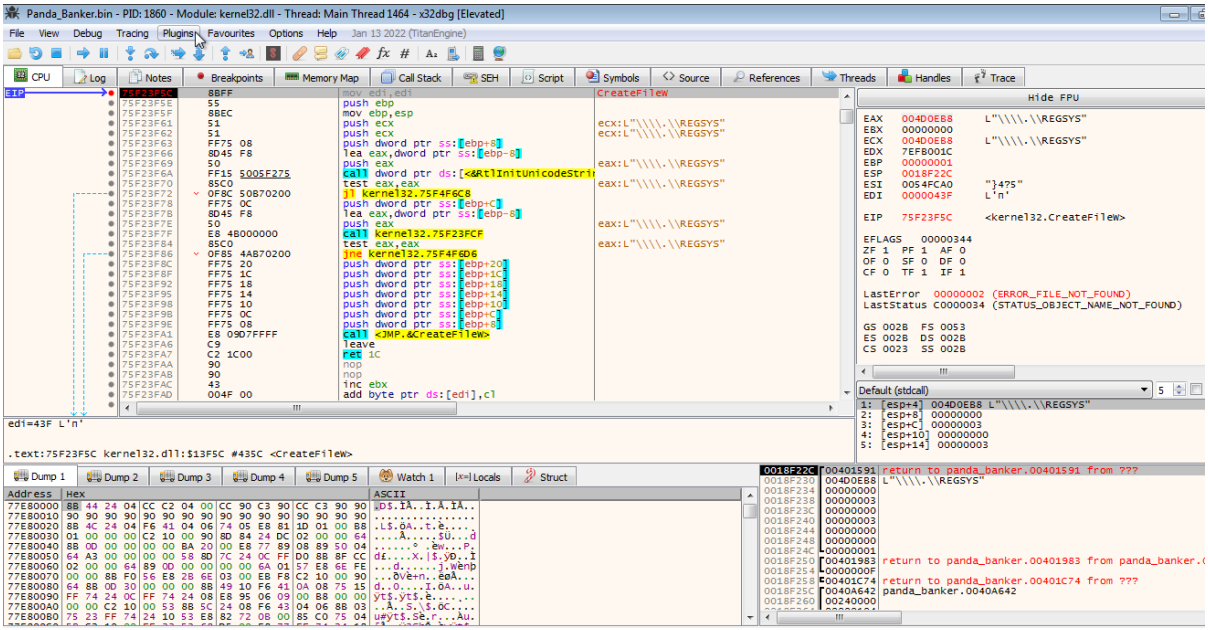
Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system.



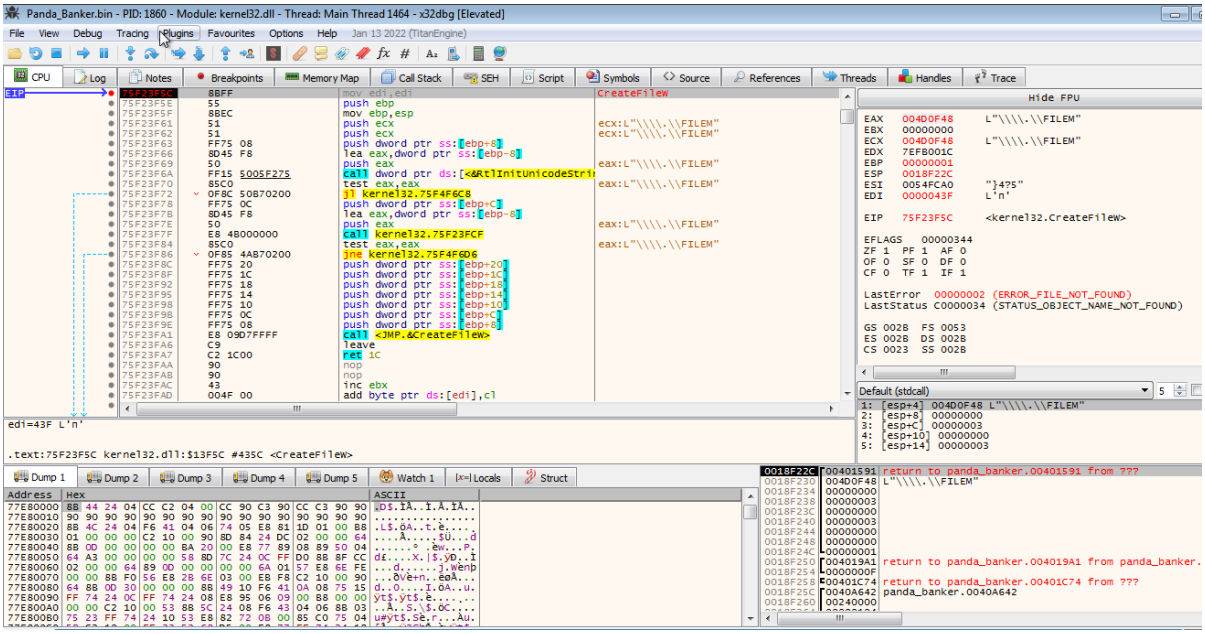
Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system.



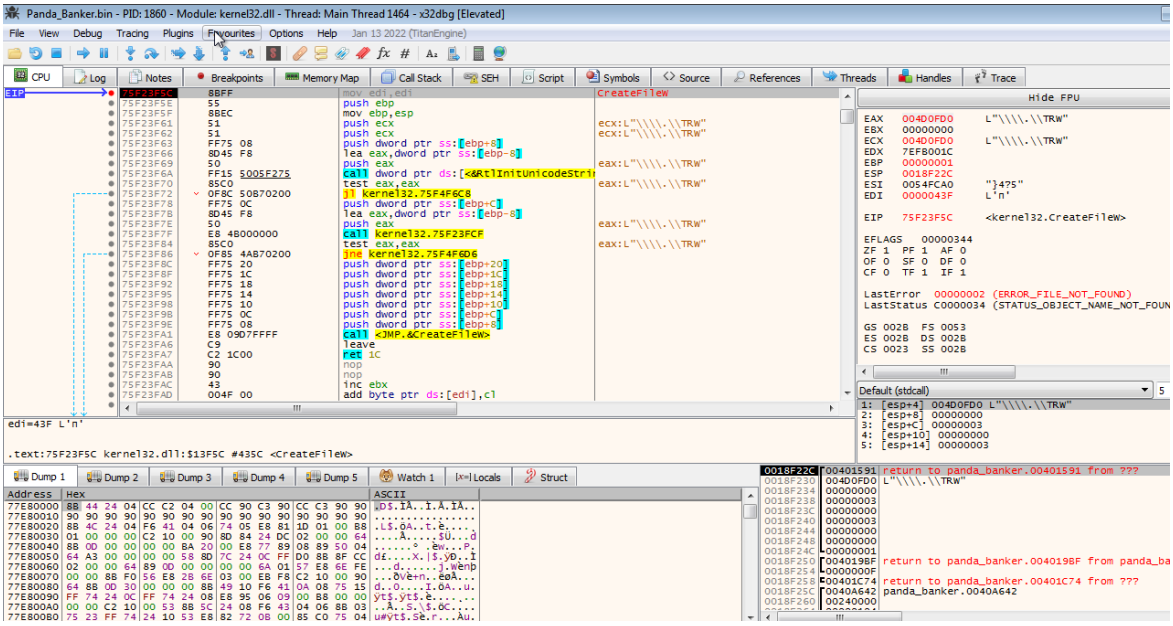
Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system.



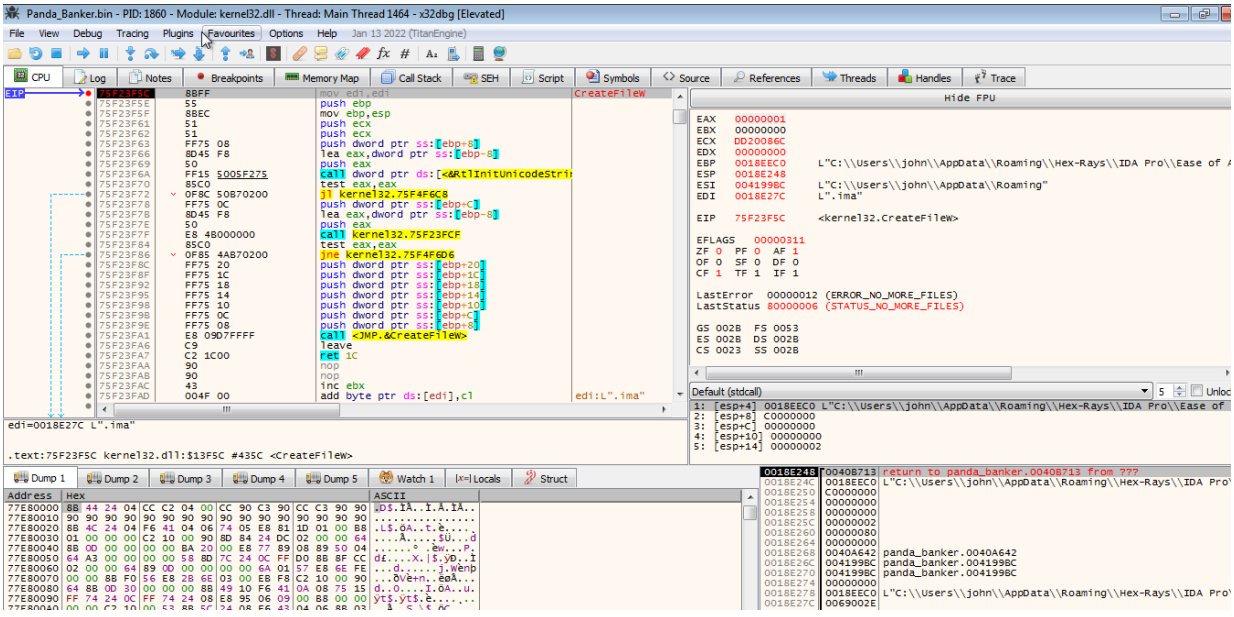
Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system.



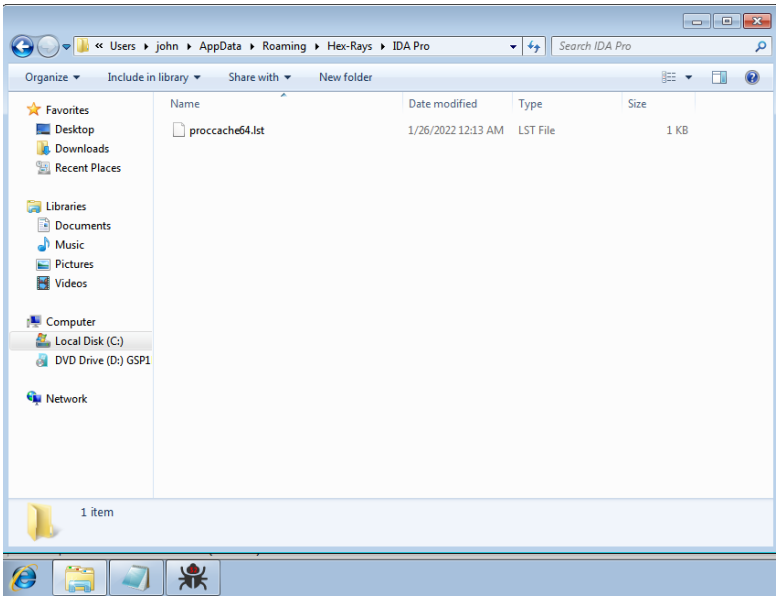
Here we again hit the breakpoint at CreateFileW , it is checking here whether the following files are present in our system .



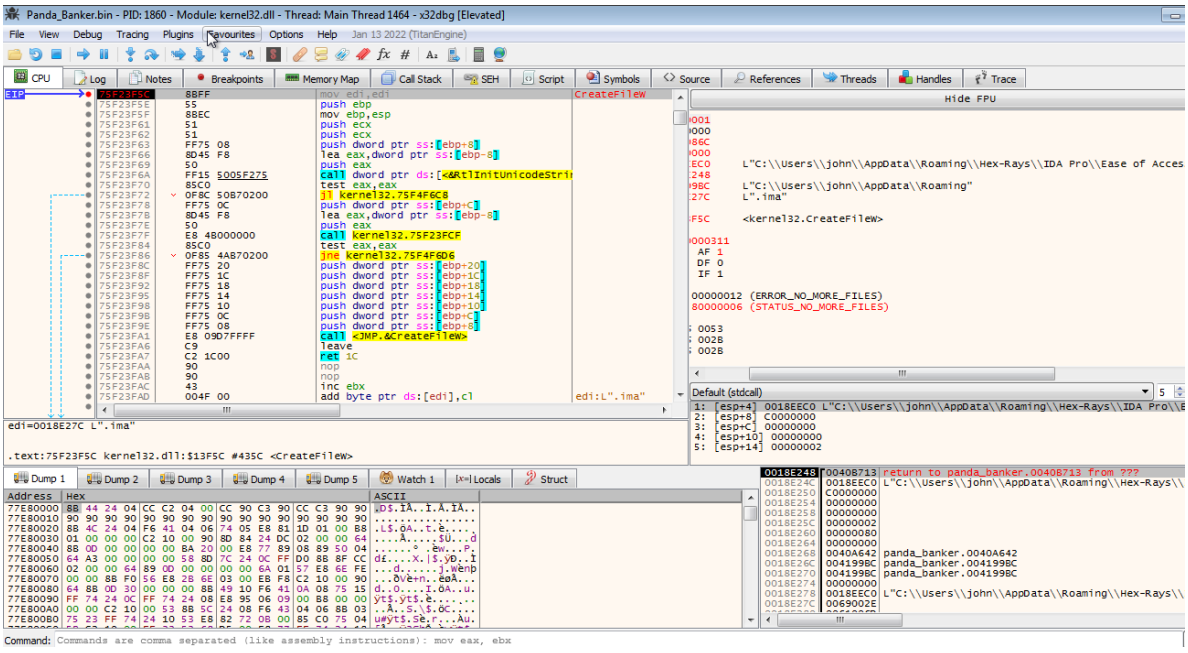
Here we again hit breakpoint at CreateFileW, but here it is accessing a location , might be trying to drop payload there , press f9 .



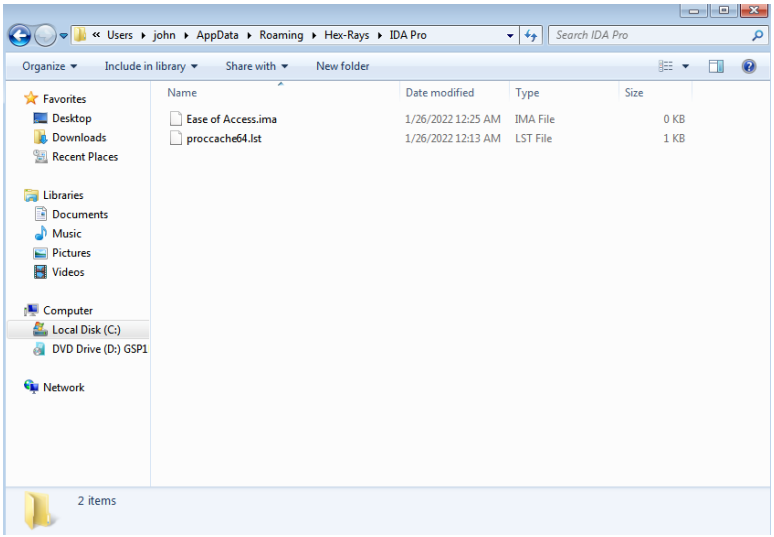
Here we view the location and till now it has drop the following file.



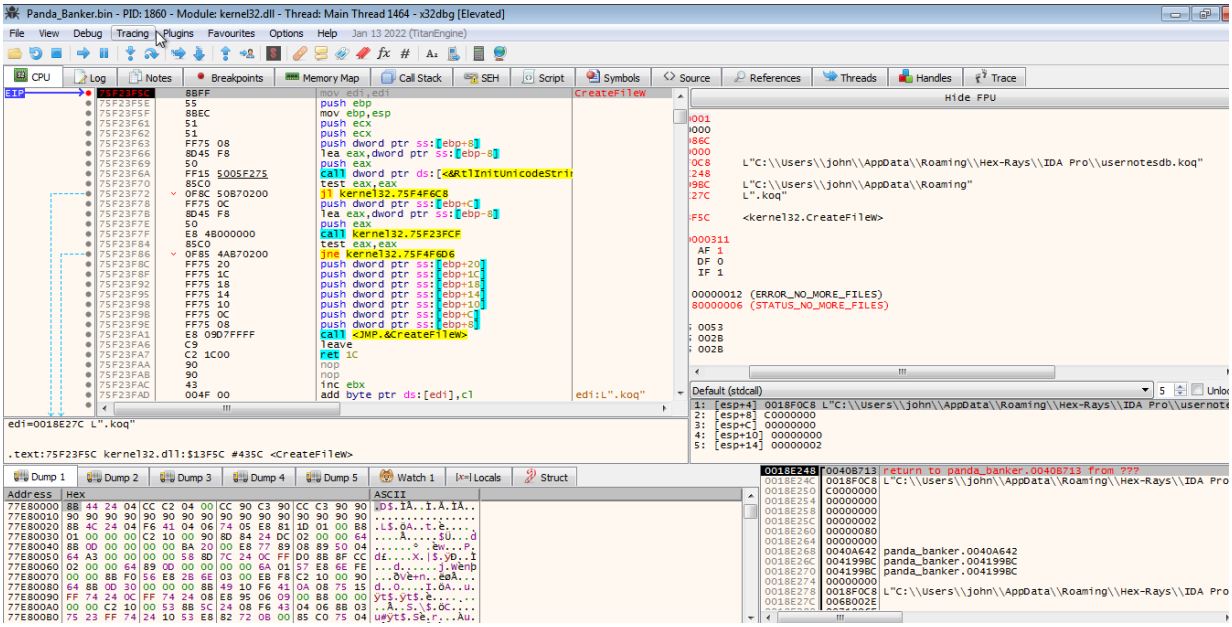
Here we again hit breakpoint at CreateFileW, but here it is accessing a location , might be trying to drop payload there , press f9 .



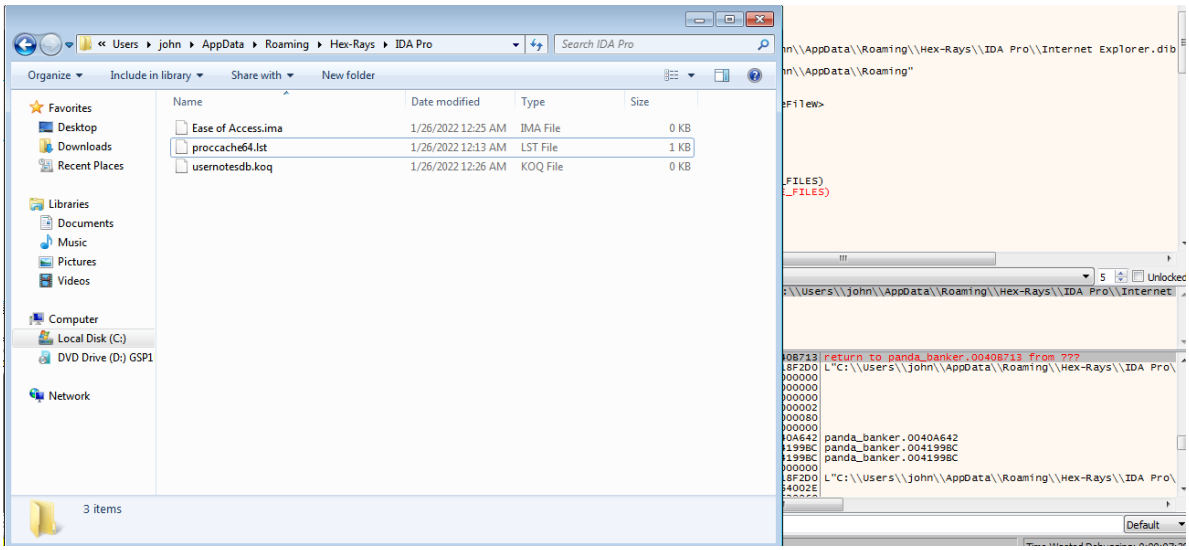
Here we view the location and till now it has drop the following files.



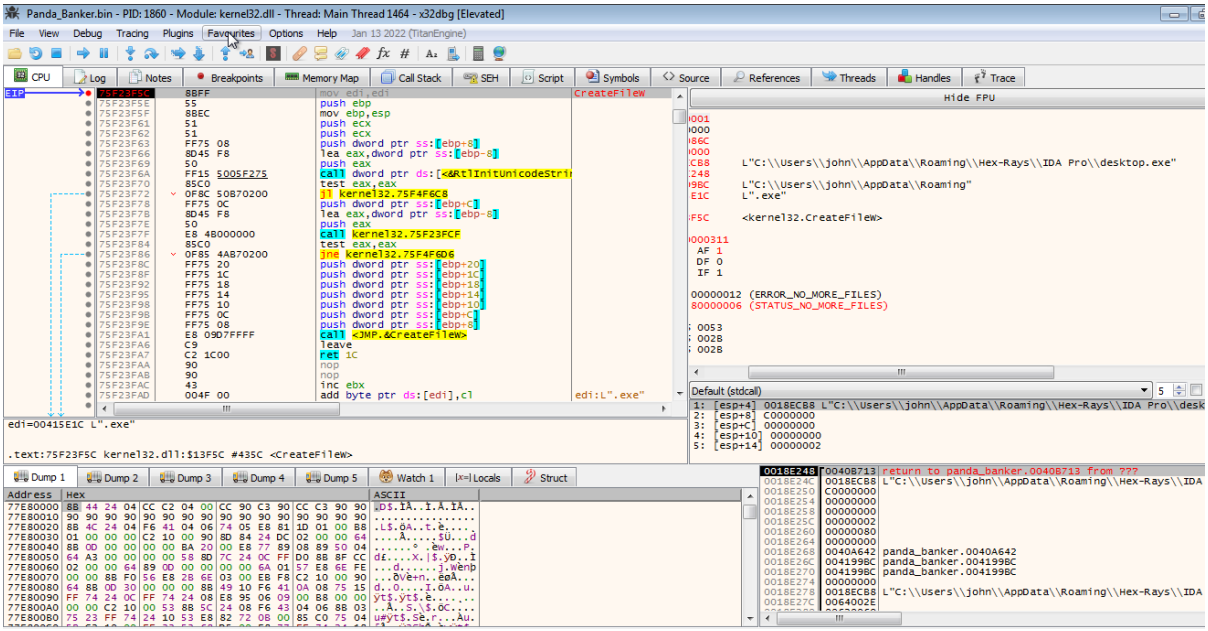
Here we again hit breakpoint at CreateFileW, but here it is accessing a location , might be trying to drop payload there , press f9.



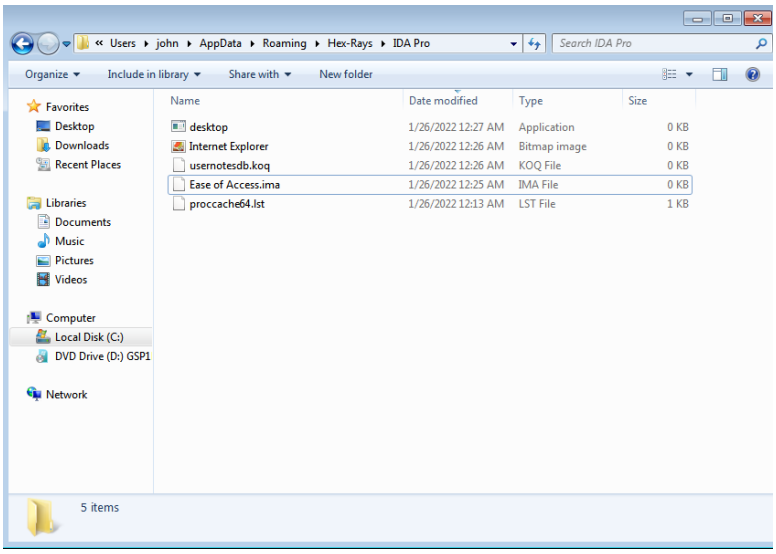
Here we view the location and till now it has drop the following files.



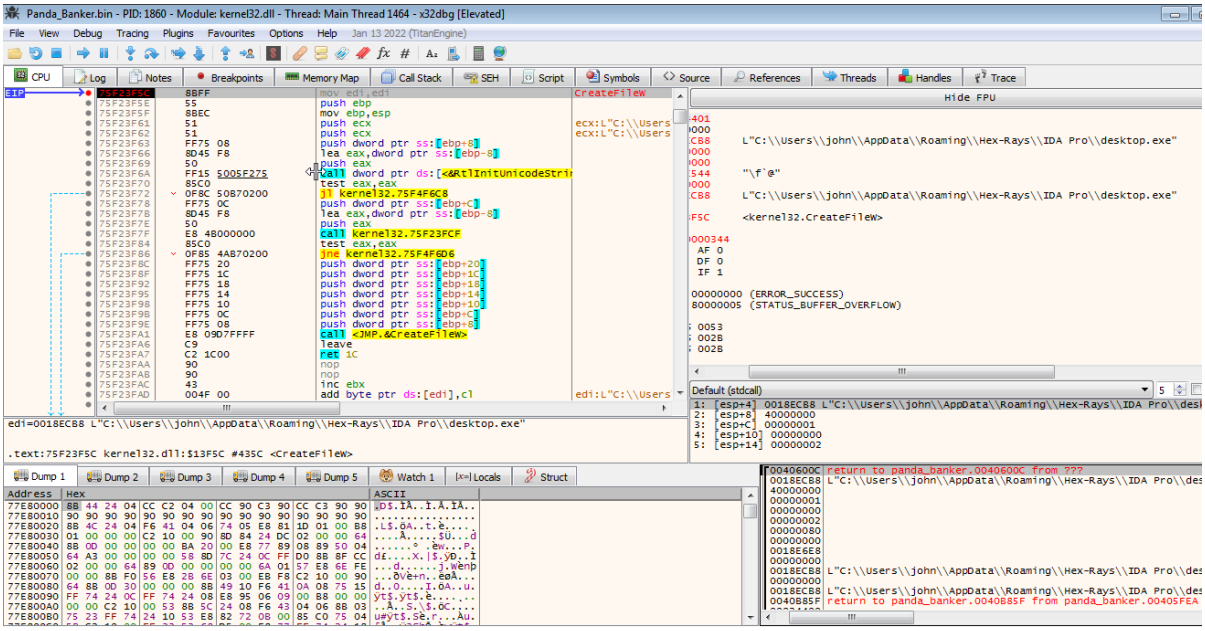
Here we again hit breakpoint at CreateFileW, but here it is accessing a location , might be trying to drop payload there , press f9.



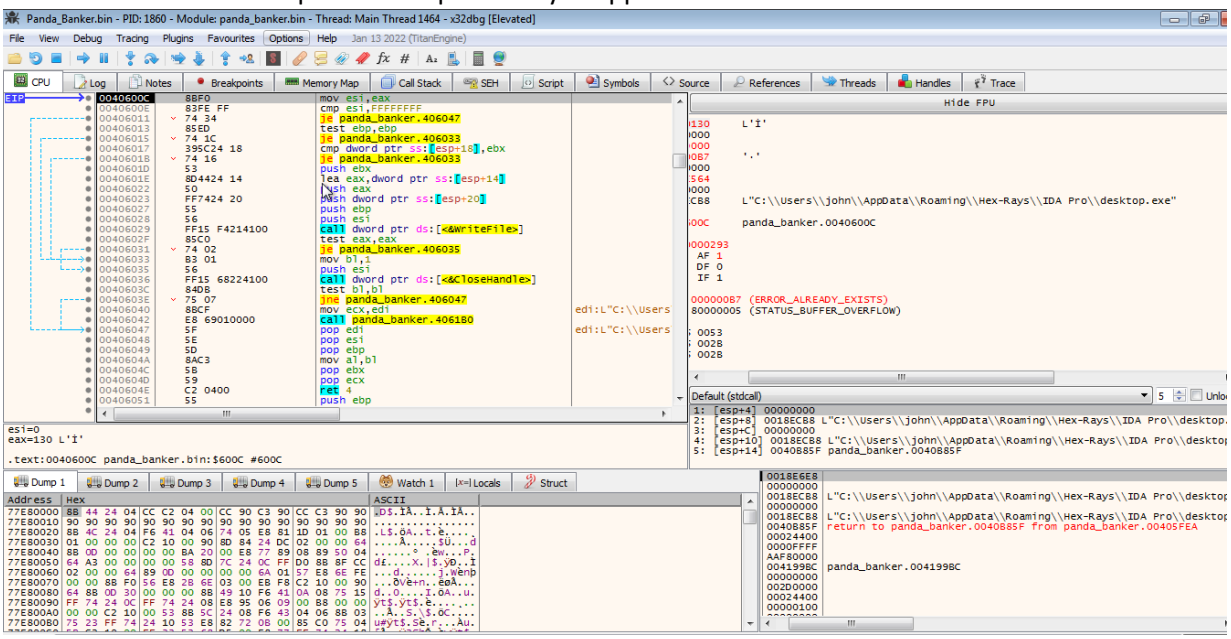
Here we view the location and till now it has drop the following files .



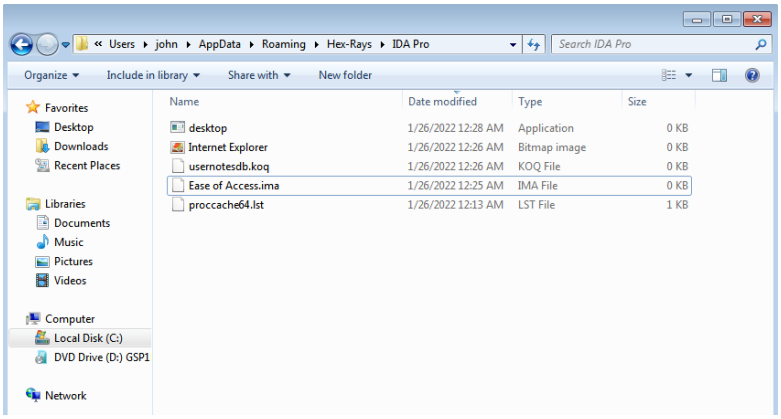
Here we again hit breakpoint at CreateFileW, but here it is trying to write in it , press f9.



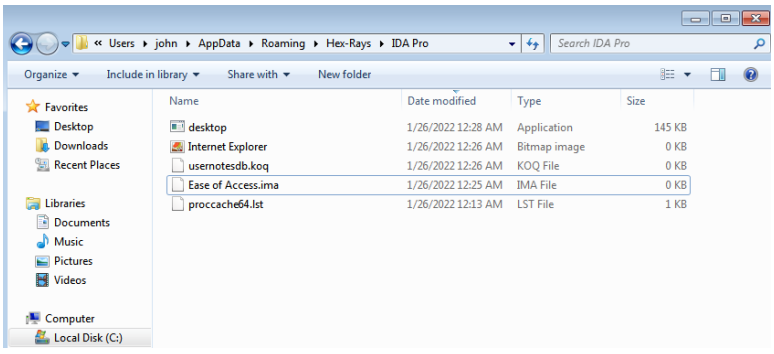
Here we can see the full path of the previously dropped exe.



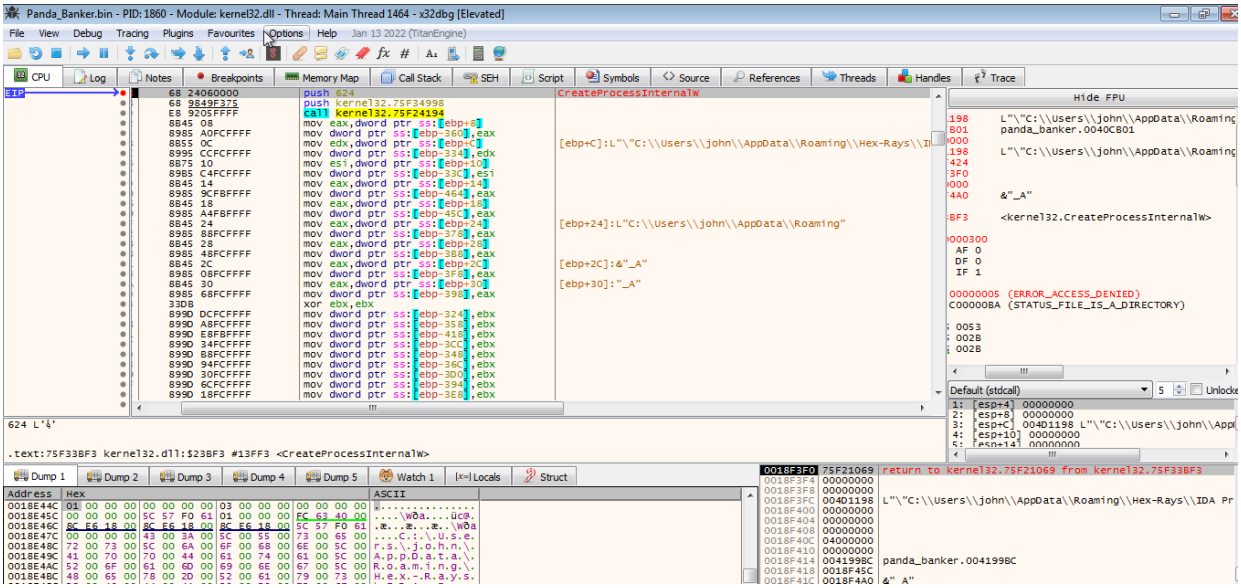
This is the dropped exe ,Note that here till now it has only created the empty exe, now press f9.



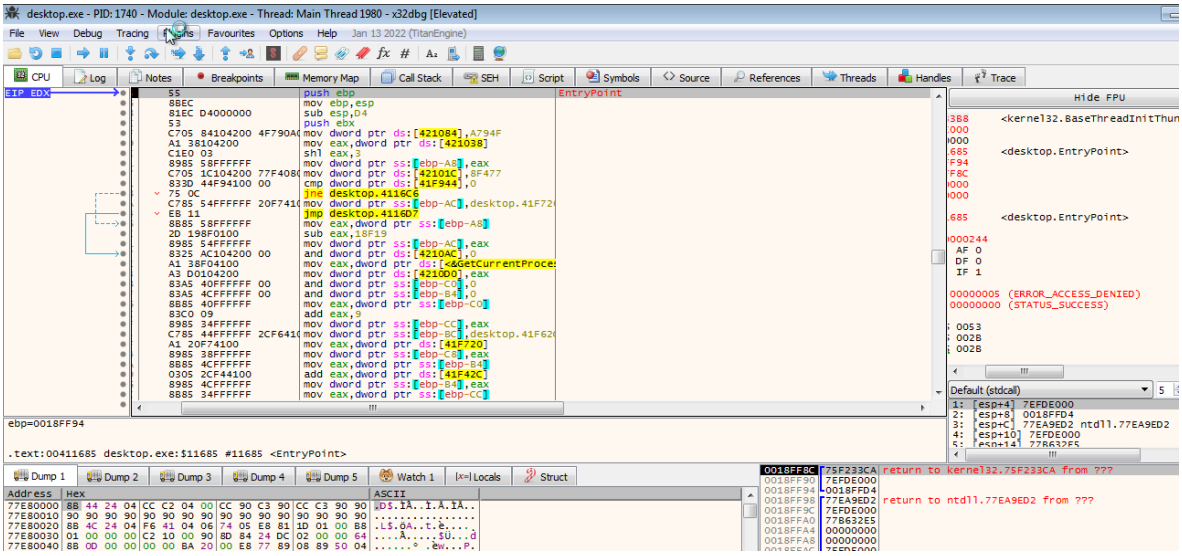
Here it has written to the exe file created and press f9 , now we can compare the hash of this exe and panda banker and they both are same.



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.

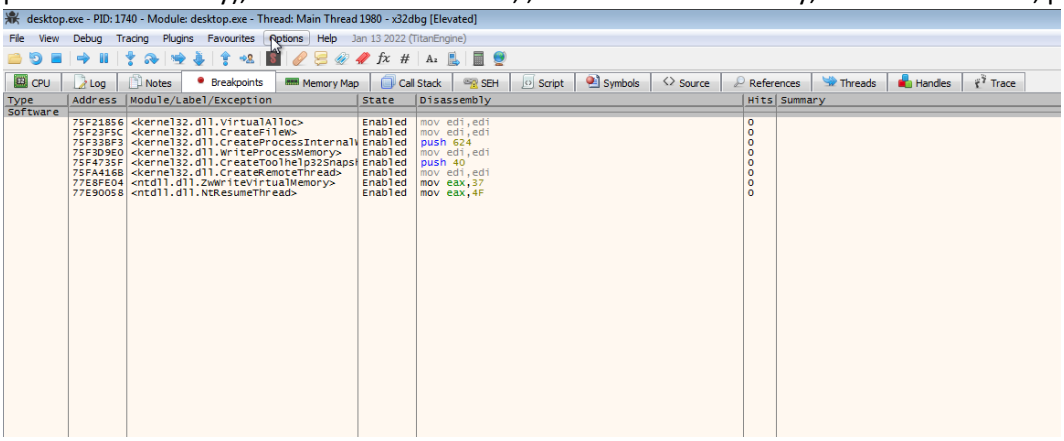


Here we load that exe .

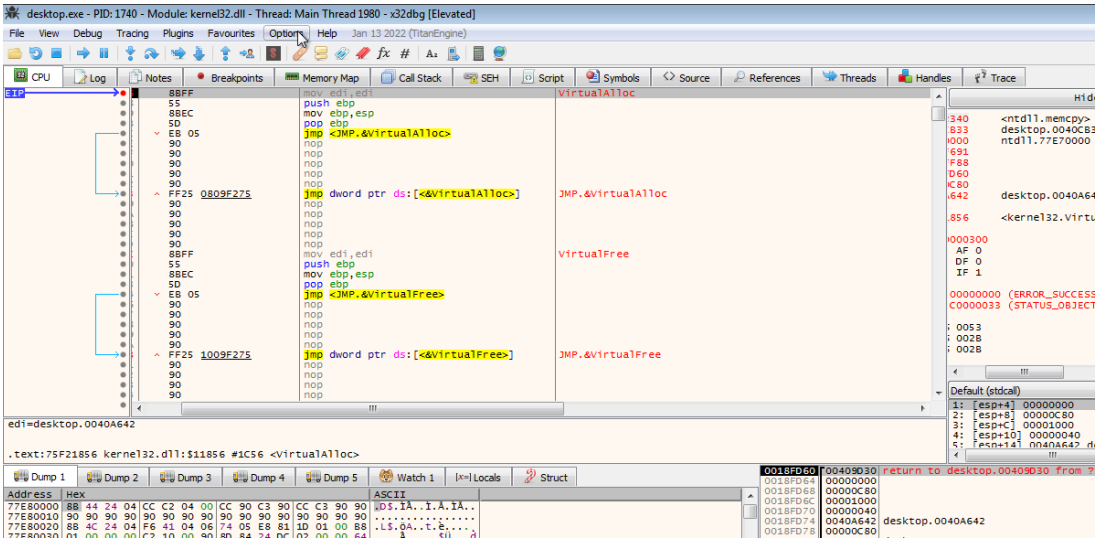


Now we put breakpoint then put breakpoints on VirtuaalAlloc, CreateFileW, CreateProcessInternalW,WriteProcessMemory, CreateToolhelp32Snapshot(enumerate list of running

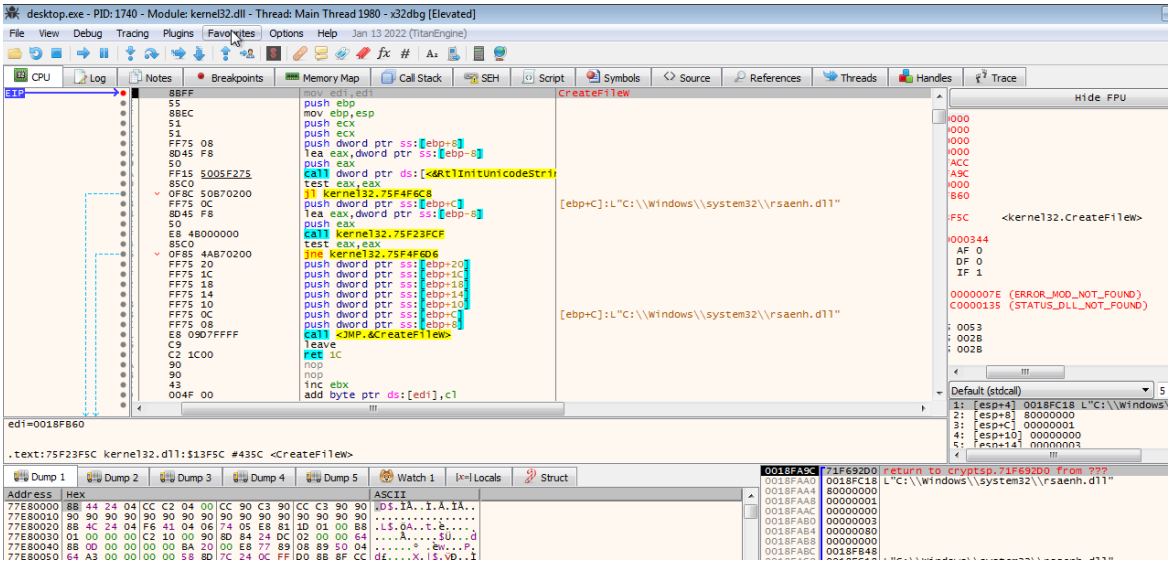
process in memory), CreateRemoteThread, ,ZwWriteVirualMemory,NtResumeThread , press f9.



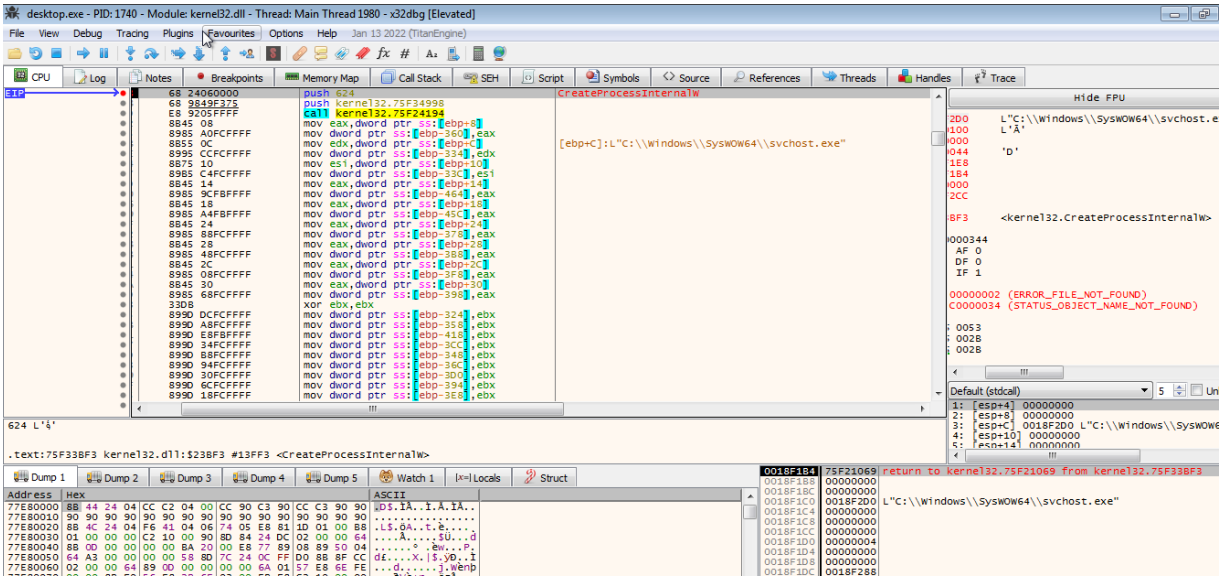
Here we hit breakpoint at VirtualAlloc just press f9.



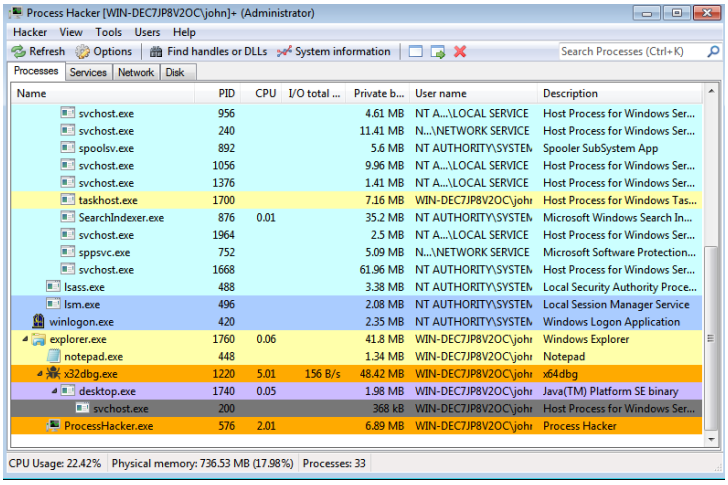
Here we hit breakpoint at CreateFileW ,here it is opening a dll just press f9.



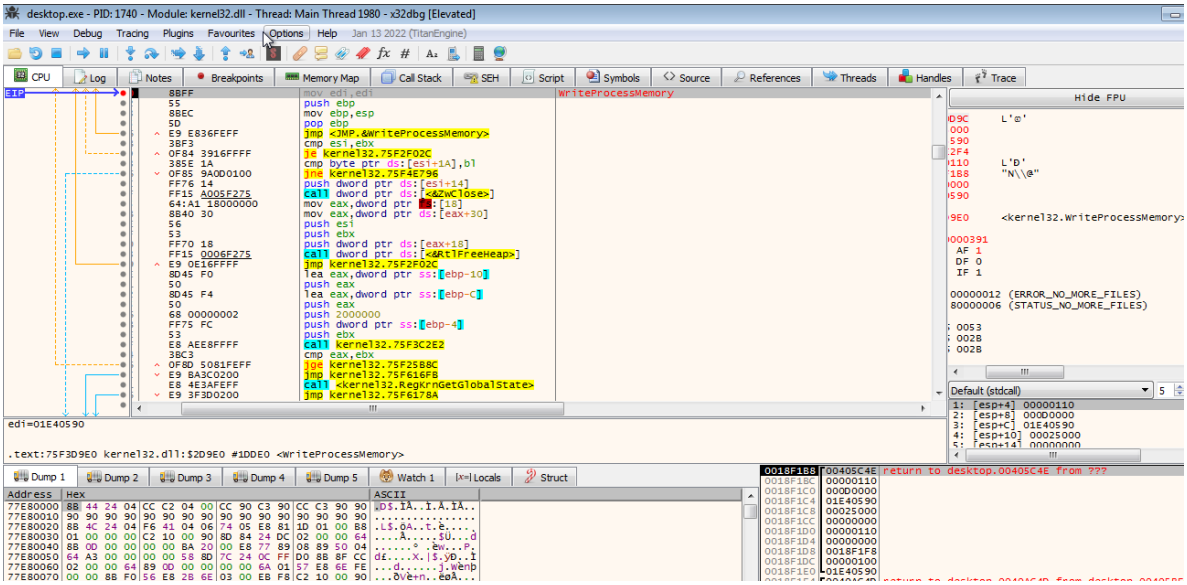
Now we hit breakpoint at CreateProcessInternalW,here it is probably writing in the memory of svchost.exe so we can use process hacker to verify it, press f9.



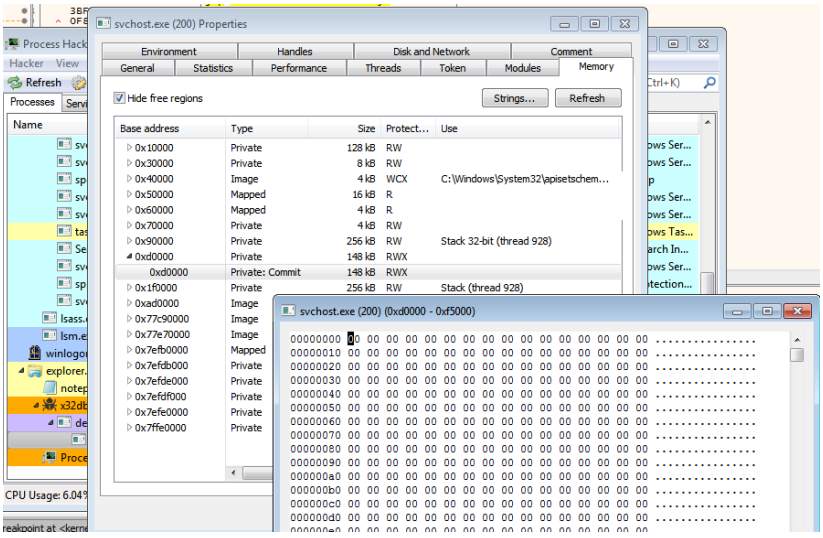
Here we can see now it has created a process svchost.exe in suspended state.



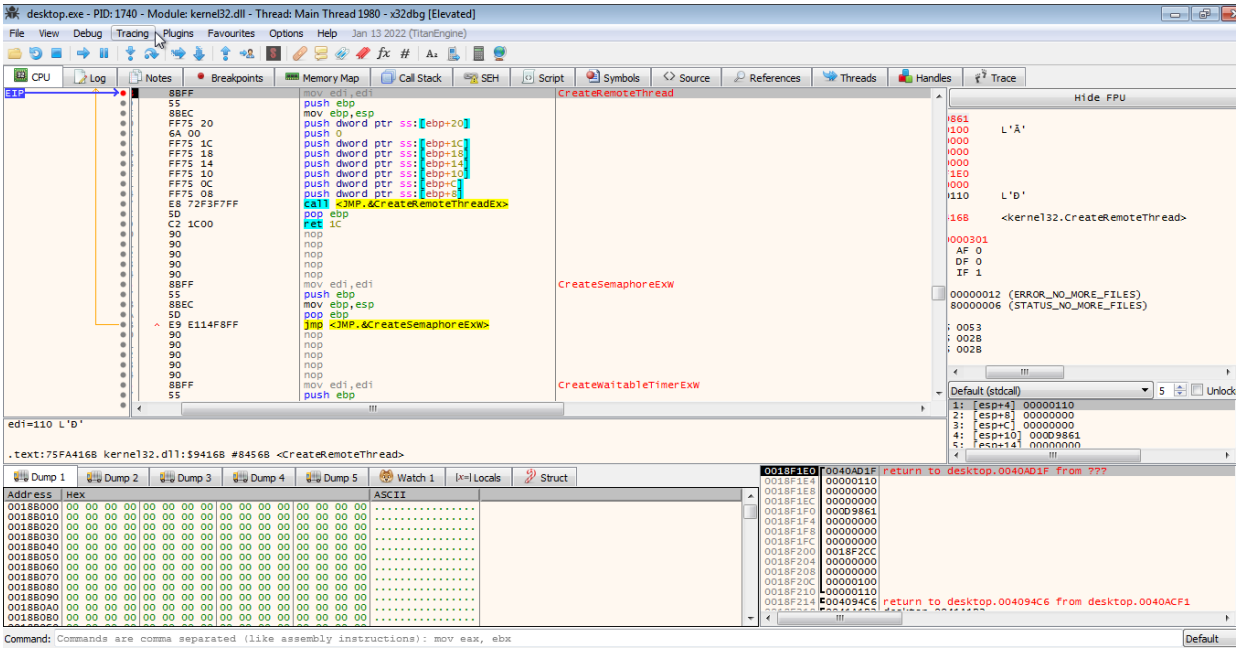
Now here we hit breakpoint at WriteProcessMemory here it is in memory of svchost.exe , here we can note the second parameter for this call i.e D0000 this is the region where it is writing to.



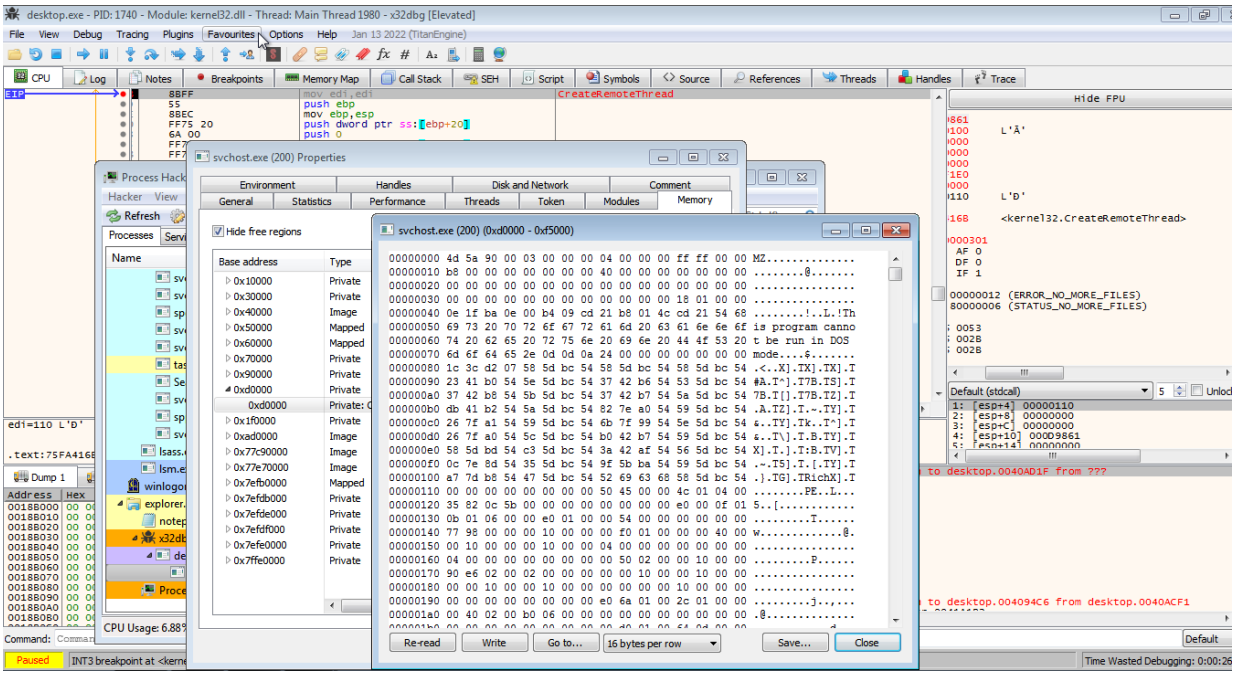
Here in process hacker we analysed memory for that address i.e d0000 we can also note here that permission of this region is read,write and execute, here this malware is trying to write to this location and then will execute it, here now just press f9.



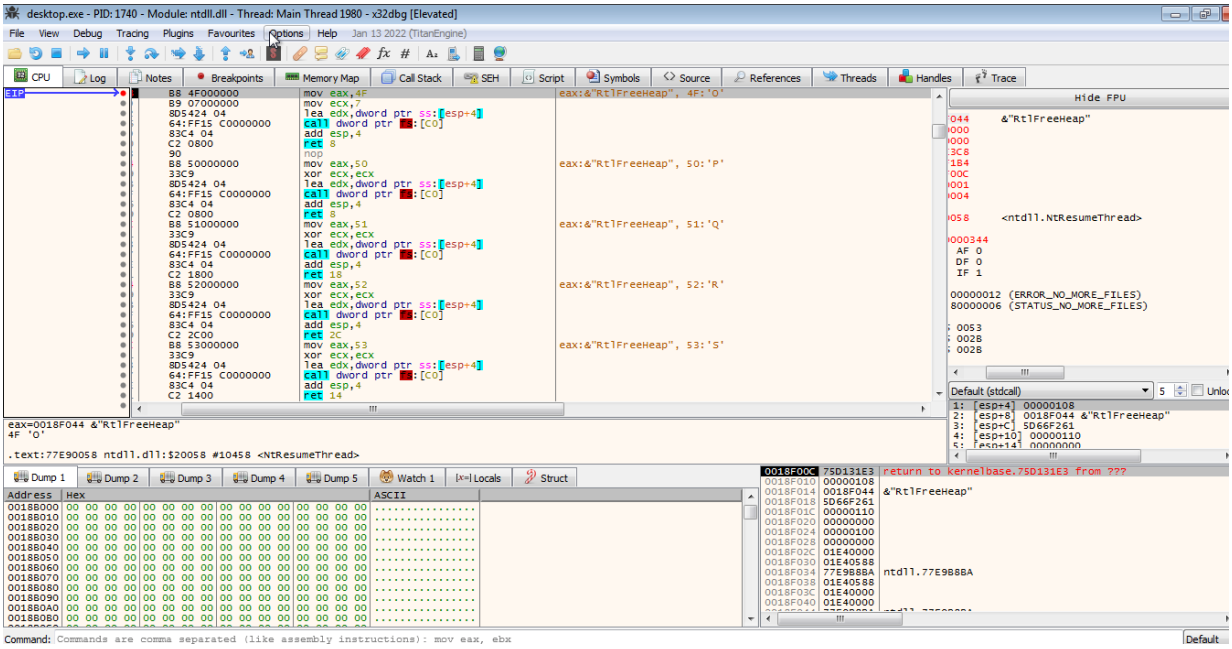
Now we hit breakpoint at CreateRemoteThread , and now it had written to svchost.exe we can verify it by viewing it in process hacker.



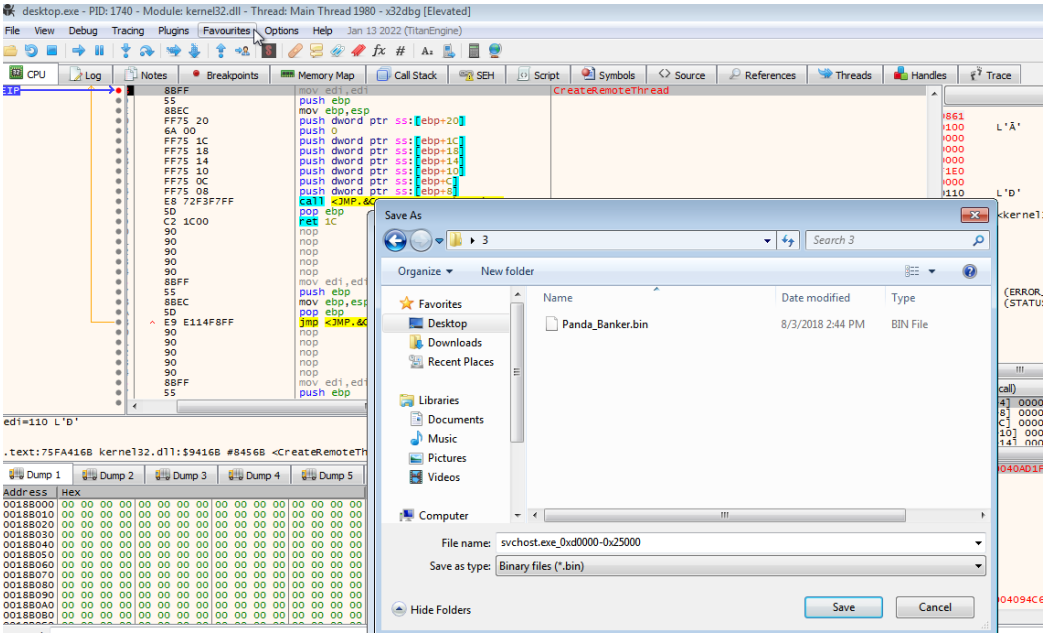
Here it has now written to that memory region and press f9.



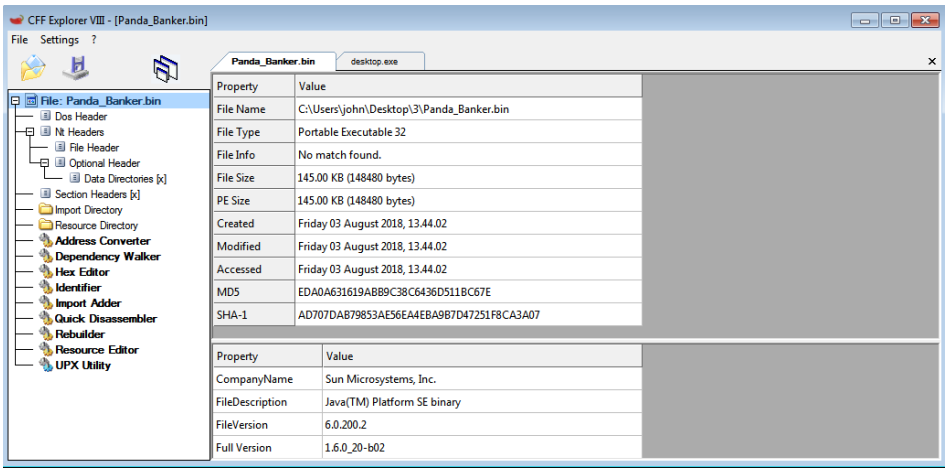
Now here we can see that now it is trying to resume thread created so now we can dump that memory region

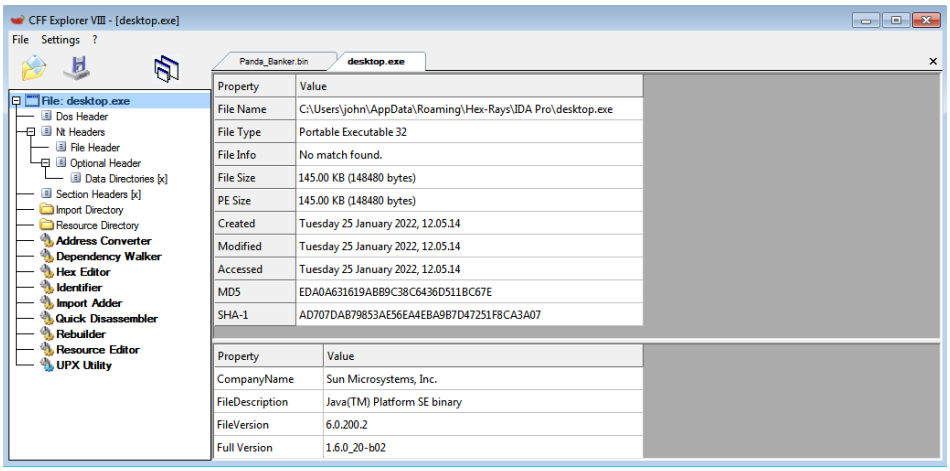


Here we dump that file.

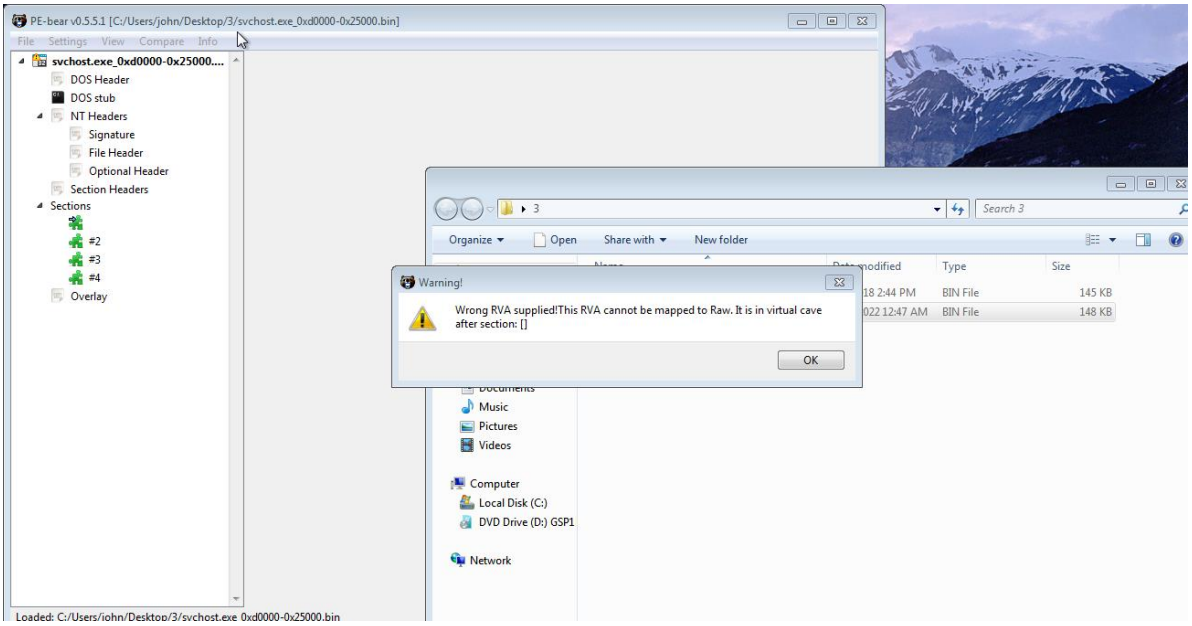


Now we load that file in CFF Explorer.

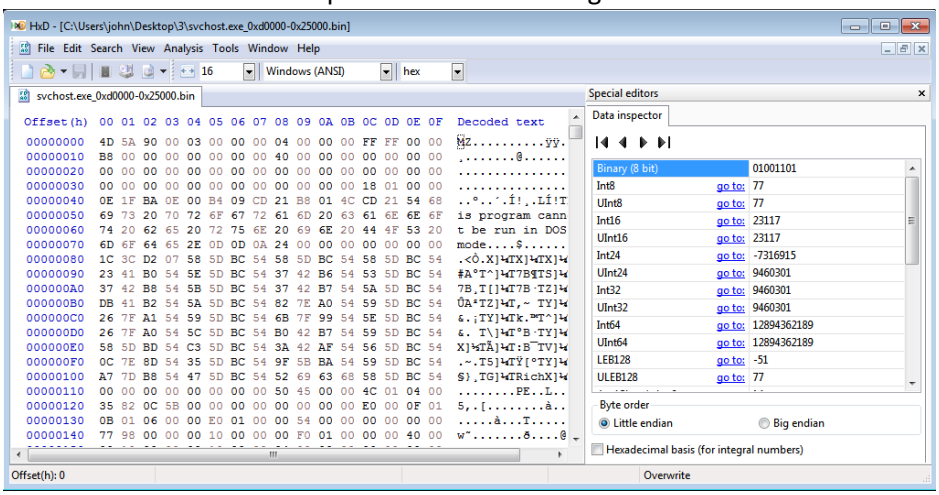




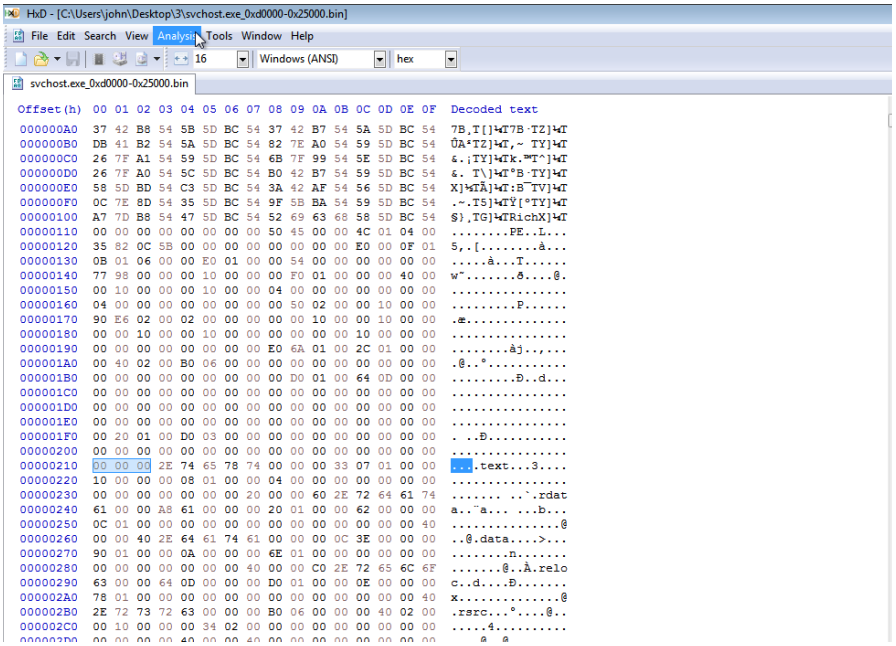
Here now we will fix the section alignments , section headers so we load the sample in PE-bear, here it is showing warning because file had been mapped to the memory therefore all the section header are out of alignment now close the pe-bear .



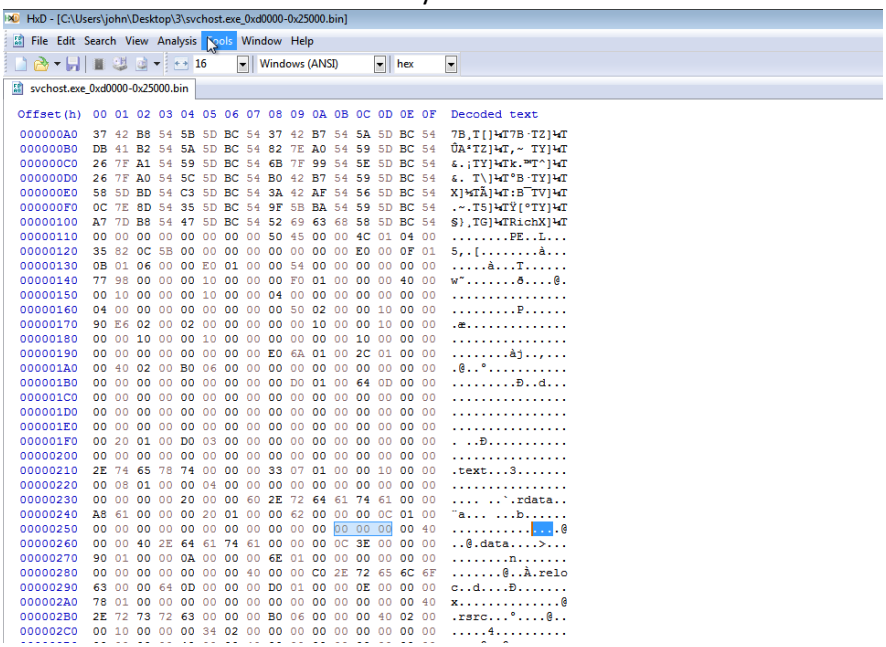
Now we have load the sample in hex editor to align the section header.



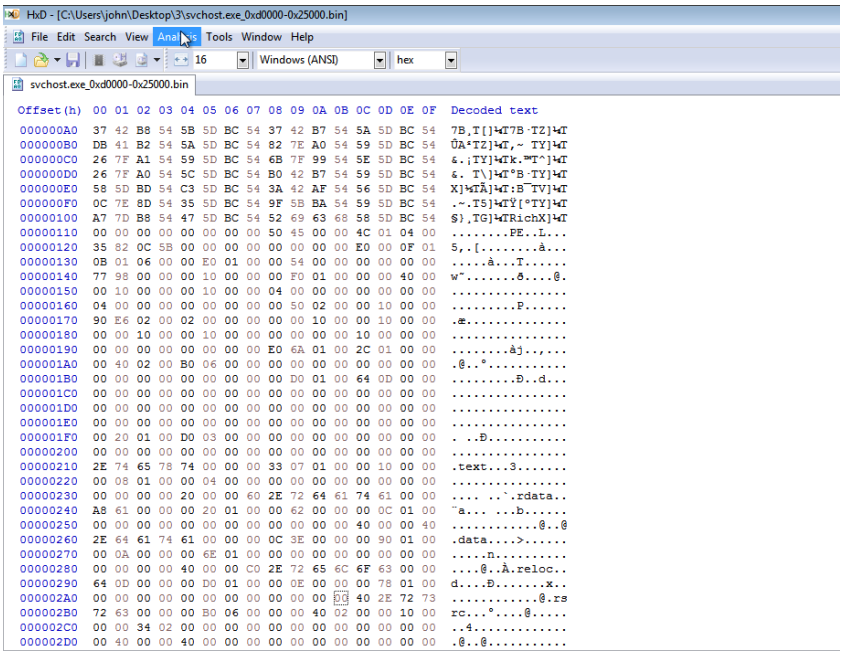
Here we can note that section header got three null bytes that means it is not aligned , so cut these three nul bytes and put them just before the end of section headers.



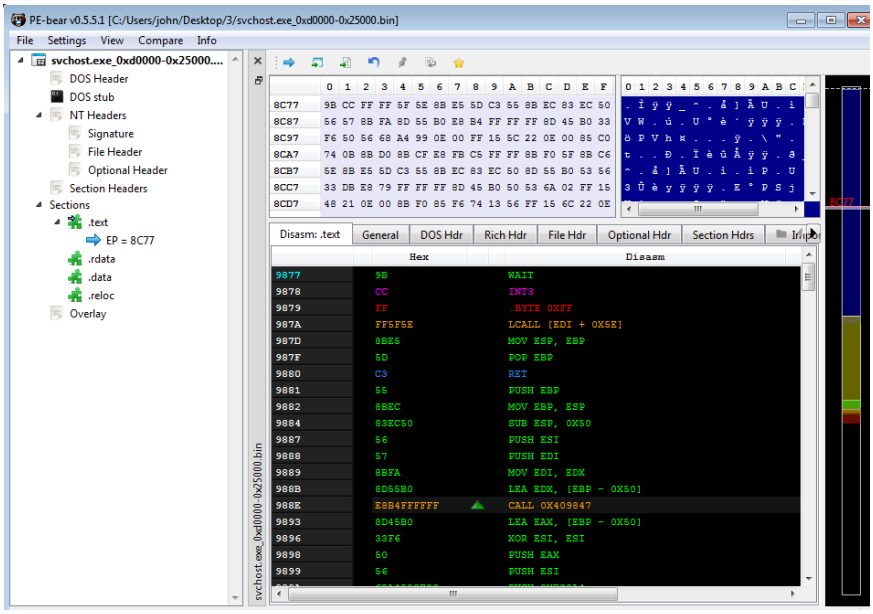
Here we have inserted those null bytes.



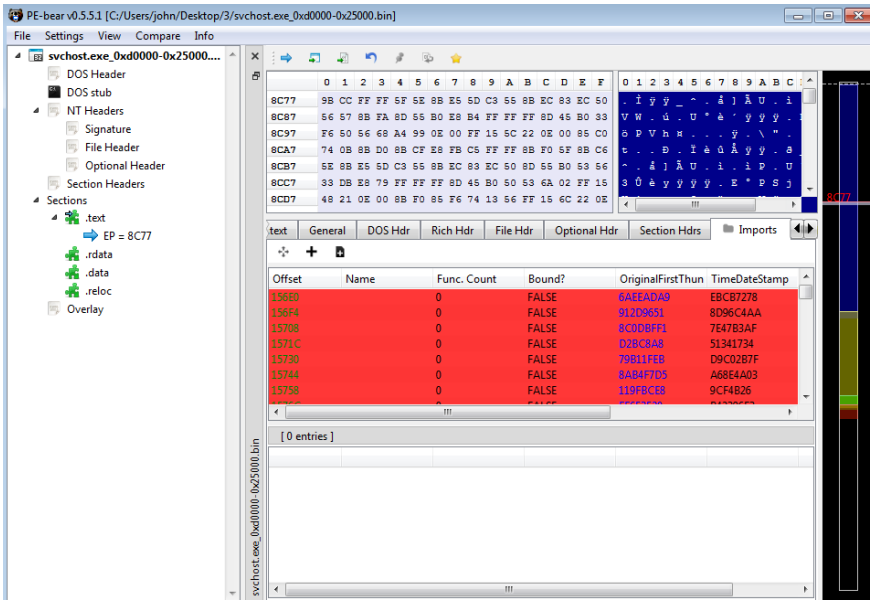
Here now we can see that it is properly aligned , now save it.



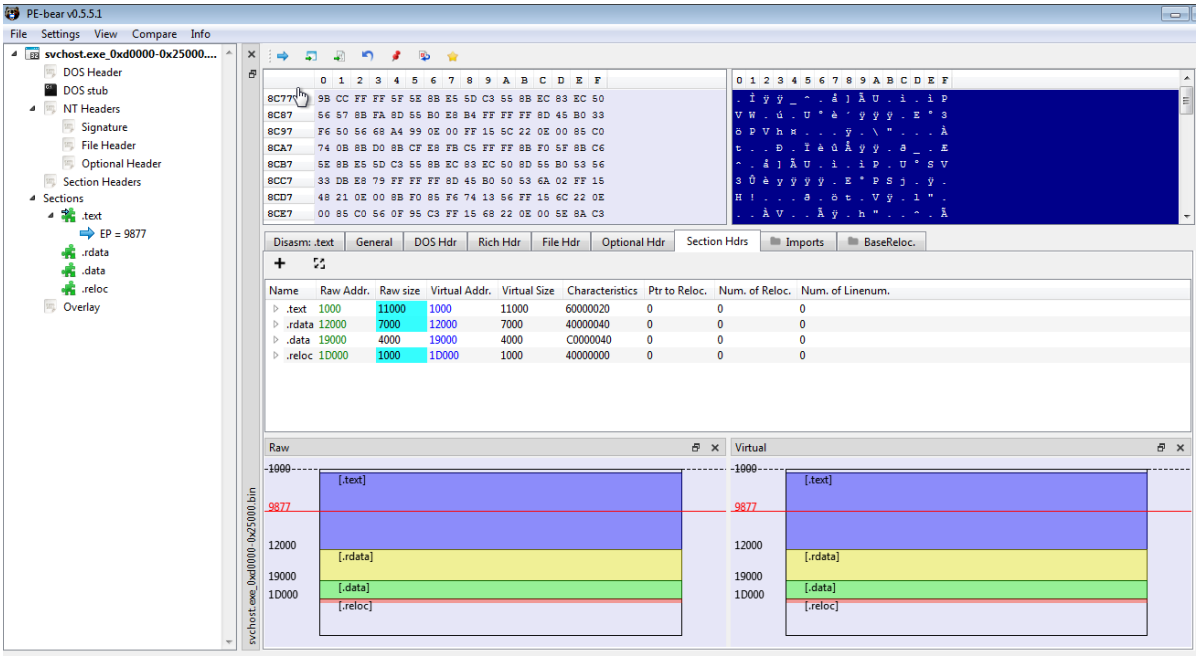
Now we load the sample in PE-bear , here we can note that now name of section headers are visible to us now we will unmap the section headers.



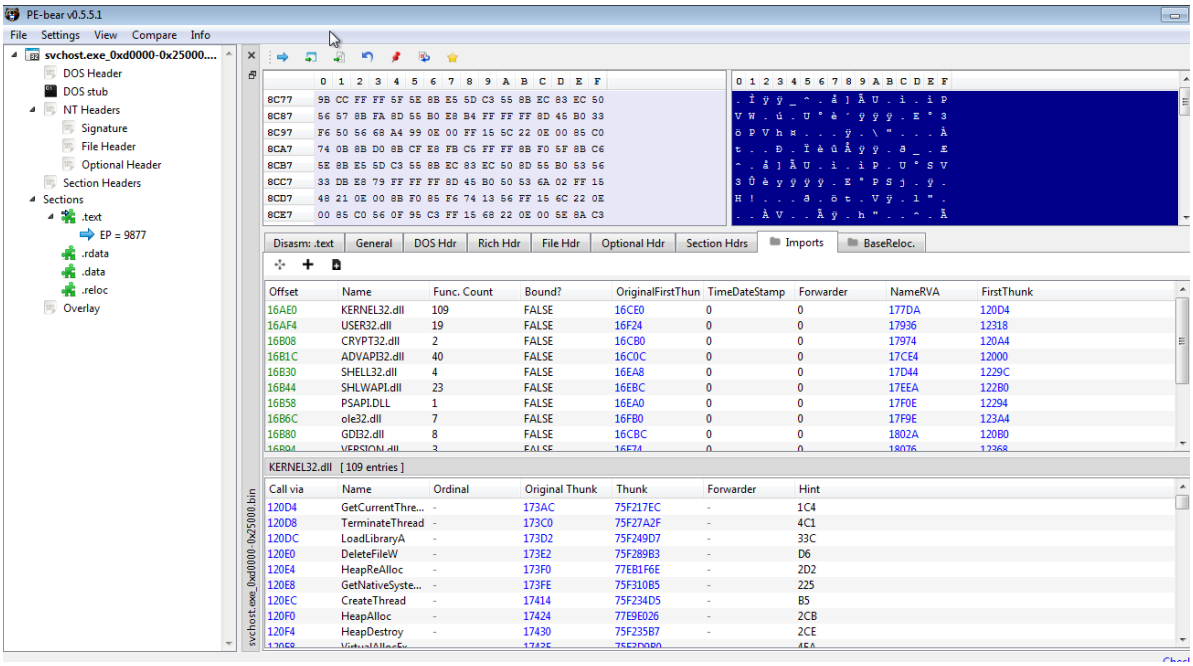
Here we can see that all the imports are distorted so now we need to fix them too.



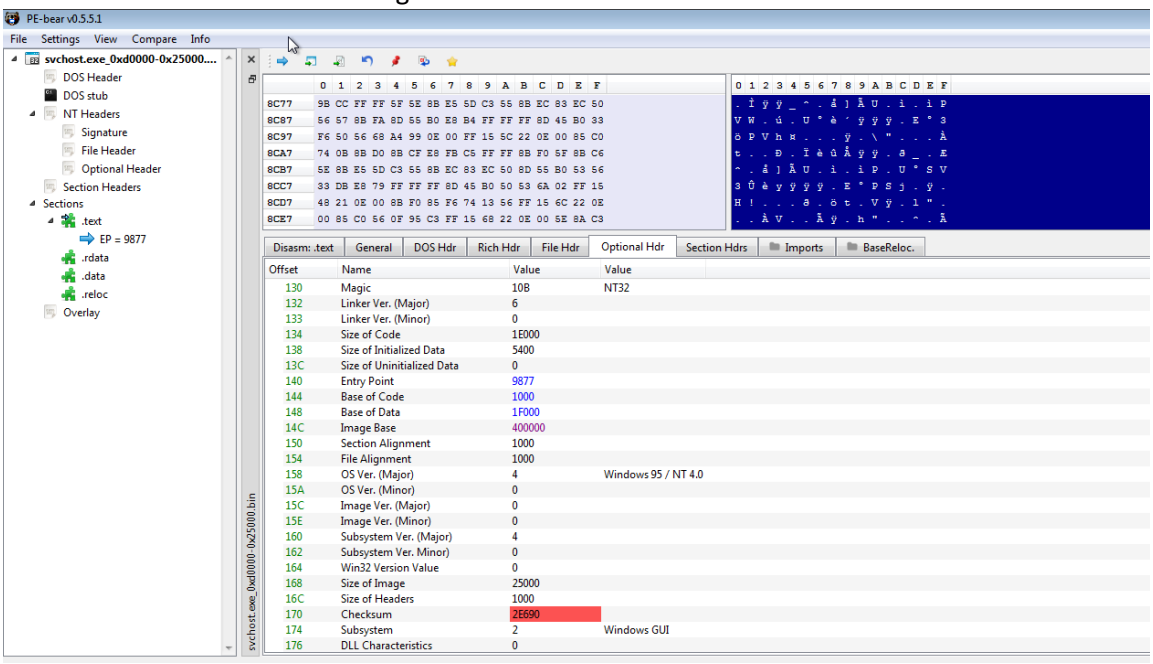
Unmap is necessary because all the raw address is mapped from memory so we need to make the raw address same as virtual address and then calculate the raw size and make virtual size same as raw size.



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.