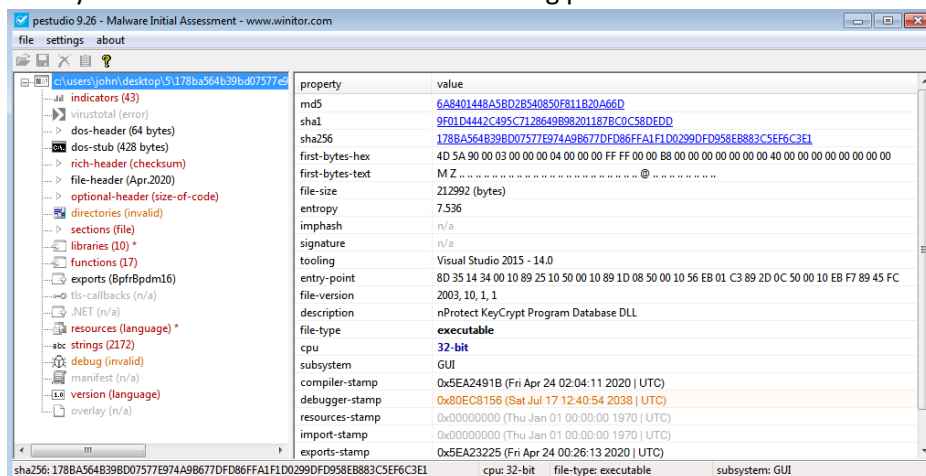


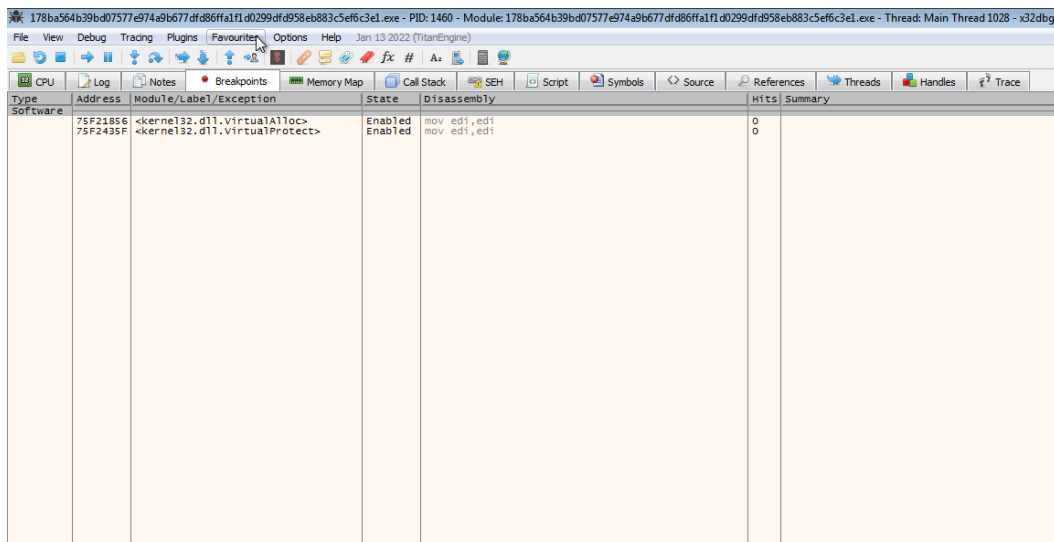
Initially we calculated the hash of the file using pestudio.




The highest entropy value can be 8, but here it is 7.538 which is close to 8 so we can assume that it is packed.

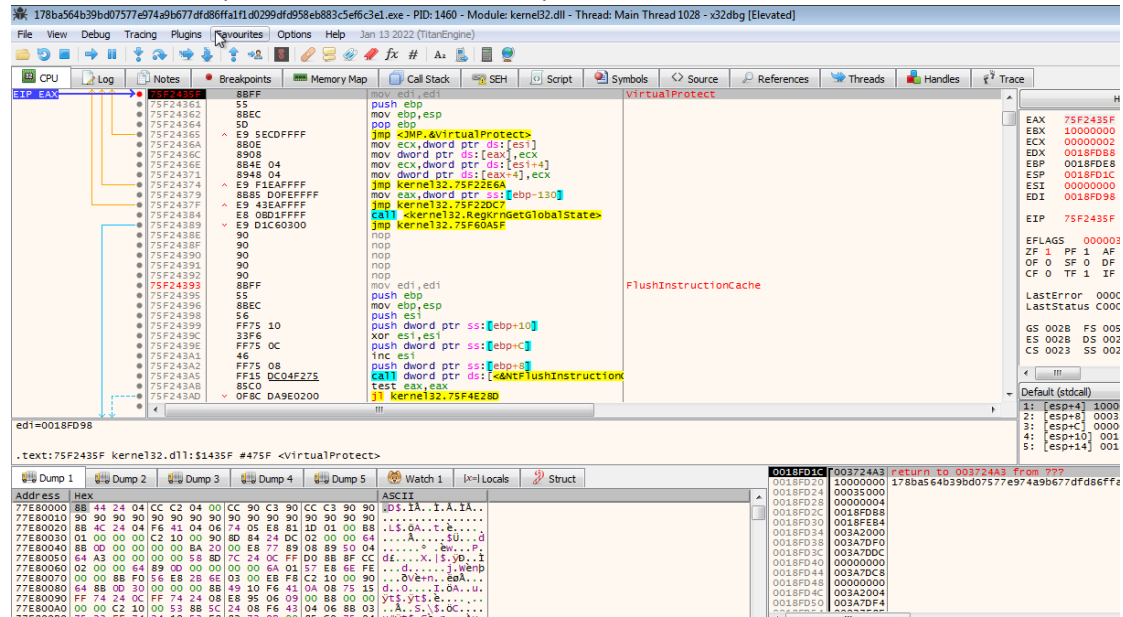
The other method to note that a file can be packed by either using detect it easy or by loading in IDA in IDA it will show us warning like Corrupted Fixup Table or it will have less imports or function name.

Now we load the program in x32dbg. Here after loading we will put two breakpoints VirtualAlloc and VirtualProtect.

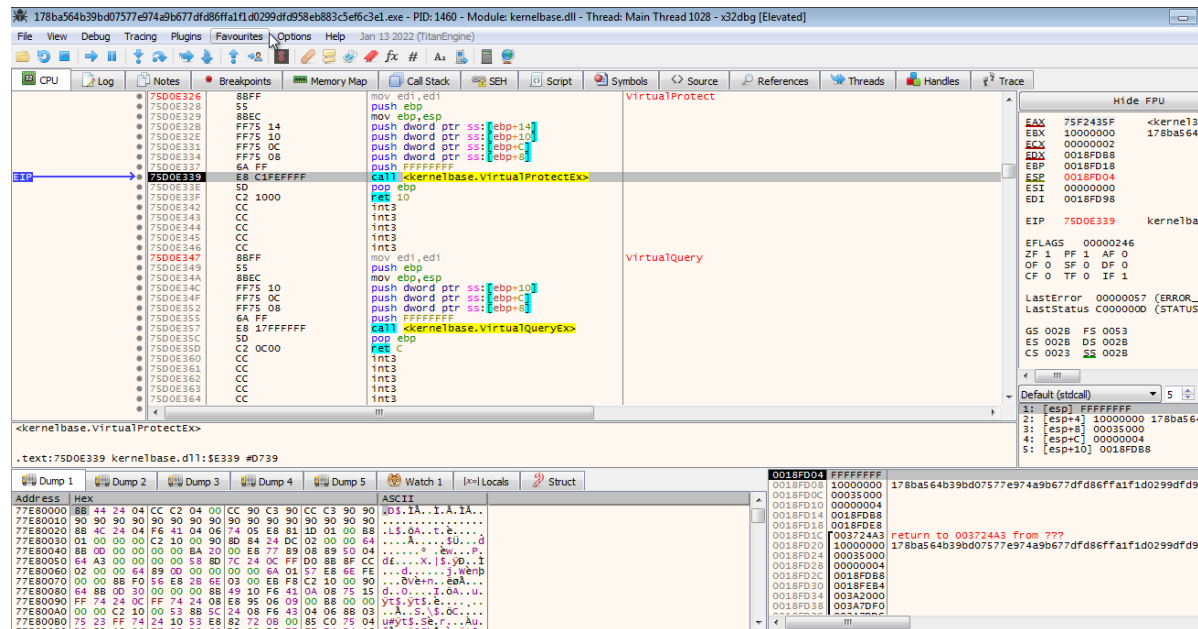


Initially run the sample we got breakpoints at VirtualAlloc, one can check out the memory allocated by VirtualAlloc by pressing the upward arrow  (i.e. execute till return) and then on eax we can right click and dump the memory returned by VirtualAlloc, but here we didn't find anything useful using VirtualAlloc, after 3 calls of VirtualAlloc, we got VirtualProtect.

Now we hit breakpoint at VirtualProtect ,step over it.



Now in this below image we can see the second parameter (i.e. 10000000) this is the location the VirtualProtect has changed the permission.



Now here we again hit the VirtualProtect, now here dump the second parameter i.e. 10000000 in dump1, if we view this address in memory map previously it was read,write and after call it become

[illegible][illegible]

178ba564b39bd07577e74a9b677df86ffa1fd0299df958eb883c5ef6c3e1.exe - PID: 1460 - Module: kernelbase.dll - Thread: Main Thread 1028 - x32dbg [Elevated]

File View Debug Tracing Plugins Favourites Options Help Jan 13 2022 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace

75D0E326 8BFF mov edi,edi  
75D0E327 55 push ebp  
75D0E328 8BEC mov ebp,esp  
75D0E329 FF75 14 push dword ptr esi[ebp+14]  
75D0E32E FF75 0C push dword ptr esi[ebp+C]  
75D0E331 FF75 08 push dword ptr esi[ebp+8]  
75D0E334 6A FF push ffffffff  
75D0E337 E8 C1FEFFFF call <kernelbase.VirtualProtectEx>  
75D0E33E 50 pop ebp  
75D0E33F C2 1000 ret 10  
75D0E342 CC int3  
75D0E343 CC int3  
75D0E344 CC int3  
75D0E345 CC int3  
75D0E346 CC int3  
75D0E347 8BFF mov edi,edi  
75D0E348 55 push ebp  
75D0E349 8BEC mov ebp,esp  
75D0E34A FF75 10 push dword ptr esi[ebp+10]  
75D0E34B FF75 0C push dword ptr esi[ebp+C]  
75D0E34E 6A FF push ffffffff  
75D0E351 E8 17FFFFF call <kernelbase.VirtualQueryEx>  
75D0E354 50 pop ebp  
75D0E355 C2 0C00 ret C  
75D0E356 CC int3  
75D0E357 CC int3  
75D0E358 CC int3  
75D0E359 CC int3  
75D0E35A CC int3  
75D0E35B CC int3  
75D0E35C CC int3  
75D0E35D CC int3  
75D0E35E CC int3  
75D0E35F CC int3  
75D0E360 CC int3  
75D0E361 CC int3  
75D0E362 CC int3  
75D0E363 CC int3  
75D0E364 CC int3

VirtualProtect

VirtualQuery

EAX 0018FD08  
EBX 10000000  
ECX 10000000  
EDX 75F2435F  
ESP 0018FD18  
EBP 0018FD04  
ESI 10000000  
EDI 0018FD9E  
EIP 75D0E339 kernelbase.dll:SE339

EF 00000206  
ZF 0 PF 1 AF 0  
OF 0 SF 0 DF 0  
CF 0 TF 0 IF 1

LastError 00000057 (ERROR\_INVALID\_PARAMETER)  
LastStatus C0000000 (STATUS\_INVALID\_PARAMETER)

GS 0028 FS 0053  
ES 0028 DS 0028  
CS 0023 SS 0028

Default (stdcall) 5

1: [esp] FFFFFFFF  
2: [esp+4] 10000000  
3: [esp+8] 00000000  
4: [esp+C] 00000000  
5: [esp+10] 0018FD08

Address Hex  
10000000 5A E0 3E ED 51 00 00 04 00 00 00 FF FF 00 00  
10000010 B8 00 00 00 00 00 00 40 00 00 00 00 00 00  
10000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
10000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
10000040 0E 1F 8A 0E 04 09 CD 21 B8 01 4C CD 21 54 68  
10000050 69 73 20 70 72 6F 67 72 61 60 20 63 61 6E 6F  
10000060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  
10000070 6D 6F 64 65 2E 00 0A 24 00 00 00 00 00 00 00  
10000080 7E 58 5E B5 3A 3A 30 E6 3A 3A 30 E6 3A 3A 30 E6  
10000090 33 42 83 E6 3B 3A 30 E6 3B 3A 30 E6 3B 3A 30 E6  
100000A0 3A 3A 31 E6 36 3A 30 E6 36 3A 30 E6 36 3A 30 E6  
100000B0 21 A7 9B E6 3A 3A 30 E6 21 A7 9B E6 3A 3A 30 E6  
100000C0 21 A7 9B E6 3A 3A 30 E6 21 A7 9B E6 3A 3A 30 E6

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

Now we again hit the breakpoint at VirtualProtect and note the second parameter mention is continuation to the previously change bit region 10022000.

178ba564b39bd07577e74a9b677df86ffa1fd0299df958eb883c5ef6c3e1.exe - PID: 1460 - Module: kernelbase.dll - Thread: Main Thread 1028 - x32dbg [Elevated]

File View Debug Tracing Plugins Favourites Options Help Jan 13 2022 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace

75D0E326 8BFF mov edi,edi  
75D0E327 55 push ebp  
75D0E328 8BEC mov ebp,esp  
75D0E329 FF75 14 push dword ptr esi[ebp+14]  
75D0E32E FF75 0C push dword ptr esi[ebp+C]  
75D0E331 FF75 08 push dword ptr esi[ebp+8]  
75D0E334 6A FF push ffffffff  
75D0E337 E8 C1FEFFFF call <kernelbase.VirtualProtectEx>  
75D0E33E 50 pop ebp  
75D0E33F C2 1000 ret 10  
75D0E342 CC int3  
75D0E343 CC int3  
75D0E344 CC int3  
75D0E345 CC int3  
75D0E346 CC int3  
75D0E347 8BFF mov edi,edi  
75D0E348 55 push ebp  
75D0E349 8BEC mov ebp,esp  
75D0E34A FF75 10 push dword ptr esi[ebp+10]  
75D0E34B FF75 0C push dword ptr esi[ebp+C]  
75D0E34E 6A FF push ffffffff  
75D0E351 E8 17FFFFF call <kernelbase.VirtualQueryEx>  
75D0E354 50 pop ebp  
75D0E355 C2 0C00 ret C  
75D0E356 CC int3  
75D0E357 CC int3  
75D0E358 CC int3  
75D0E359 CC int3  
75D0E35A CC int3  
75D0E35B CC int3  
75D0E35C CC int3  
75D0E35D CC int3  
75D0E35E CC int3  
75D0E35F CC int3  
75D0E360 CC int3  
75D0E361 CC int3  
75D0E362 CC int3  
75D0E363 CC int3  
75D0E364 CC int3

VirtualProtect

VirtualQuery

EAX 75F2435F  
EBX 00005EFE  
ECX 10022000  
EDX 0018FD08  
ESP 0018FD04  
EBP 0018FD08  
ESI 0018FD08  
EDI 00000000  
EIP 75D0E339 kernelbase.dll:SE339

EF 00000202  
ZF 0 PF 0 AF 0  
OF 0 SF 0 DF 0  
CF 0 TF 0 IF 1

LastError 00000057 (ERROR\_INVALID\_PARAMETER)  
LastStatus C0000000 (STATUS\_INVALID\_PARAMETER)

GS 0028 FS 0053  
ES 0028 DS 0028  
CS 0023 SS 0028

Default (stdcall) 5

1: [esp] FFFFFFFF  
2: [esp+4] 10022000  
3: [esp+8] 00005EFE  
4: [esp+C] 00000000  
5: [esp+10] 0018FD08

Address Hex  
10000000 5A E0 3E ED 51 00 00 04 00 00 00 FF FF 00 00  
10000010 B8 00 00 00 00 00 00 40 00 00 00 00 00 00  
10000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
10000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
10000040 0E 1F 8A 0E 04 09 CD 21 B8 01 4C CD 21 54 68  
10000050 69 73 20 70 72 6F 67 72 61 60 20 63 61 6E 6F  
10000060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  
10000070 6D 6F 64 65 2E 00 0A 24 00 00 00 00 00 00 00  
10000080 7E 58 5E B5 3A 3A 30 E6 3A 3A 30 E6 3A 3A 30 E6  
10000090 33 42 83 E6 3B 3A 30 E6 3B 3A 30 E6 3B 3A 30 E6  
100000A0 3A 3A 31 E6 36 3A 30 E6 36 3A 30 E6 36 3A 30 E6  
100000B0 21 A7 9B E6 3A 3A 30 E6 21 A7 9B E6 3A 3A 30 E6  
100000C0 21 A7 9B E6 3A 3A 30 E6 21 A7 9B E6 3A 3A 30 E6

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

Paused INT3 breakpoint at <kernel32.VirtualProtect> (75F2435F)

Time Wasted Debug

Till now we have hit the VirtualAlloc 3 times and VirtualProtect 5 times.

178ba564b39bd07577e74a9b677df86ffa1fd0299df958eb883c5ef6c3e1.exe - PID: 1460 - Module: kernel32.dll - Thread: Main Thread 1028 - x32dbg [Elevated]

File View Debug Tracing Plugins Favourites Options Help Jan 13 2022 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References

Type	Address	Module/Label/Exception	State	Disassembly	Hits	Summ
Software	75F2185F	<kernel32.dll.VirtualAlloc>	Enabled	mov edi,edi	3	
Software	75F2435F	<kernel32.dll.VirtualProtect>	Enabled	mov edi,edi	5	

Now we again hit the VirtualProtect.

The screenshot shows the debugger interface with the following details:

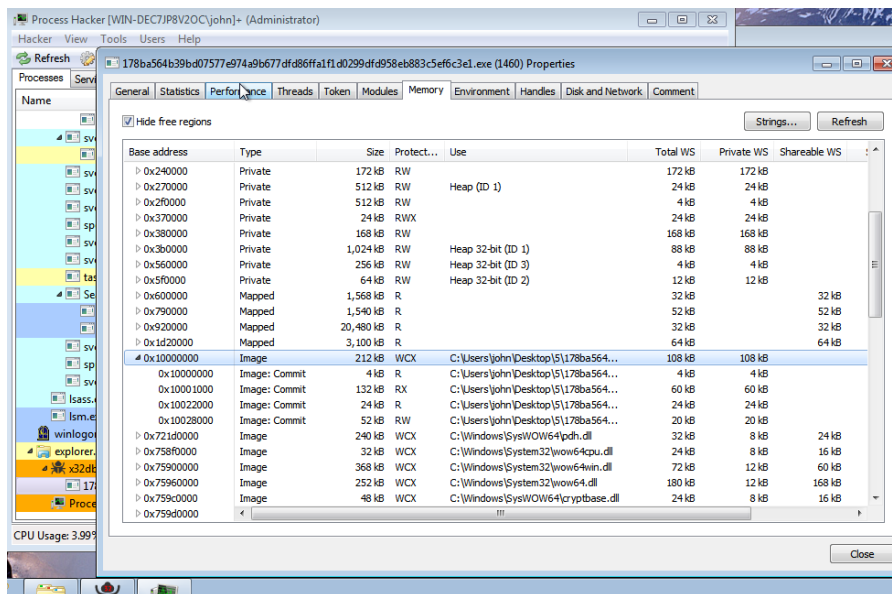
- File View:** 178ba54b39bd07577e974a9b677df86ffa1f1d0299df958eb883c5ef6c3e1.exe - PID: 1460 - Module: kernel32.dll - Thread: Main Thread 1028 - x32dbg [Elevated]
- Log:** CPU window showing assembly instructions. Key instructions include:
  - 75F2435F: `mov edi,edi`
  - 75F24361: `push ebp`
  - 75F24362: `mov ebp,esp`
  - 75F24363: `pop ebp`
  - 75F24365: `jmp <JMP.&VirtualProtect>`
  - 75F2436A: `mov ecx,dword ptr ds:[esi]`
  - 75F2436C: `mov dword ptr ds:[ecx],ecx`
  - 75F2436E: `mov ecx,dword ptr ds:[esi+4]`
  - 75F24371: `mov dword ptr ds:[ecx+4],ecx`
  - 75F24374: `jmp kernel32.75F22E6A`
  - 75F24379: `mov eax,dword ptr ds:[ebp-130]`
  - 75F2437F: `call <kernel32.RegOpenGlobalState>`
  - 75F24384: `jmp kernel32.75F60A5F`
  - 75F24388: `nop`
  - 75F2438F: `nop`
  - 75F24390: `nop`
  - 75F24391: `nop`
  - 75F24392: `nop`
  - 75F24393: `push esi`
  - 75F24395: `push ebp`
  - 75F24396: `mov ebp,esp`
  - 75F24398: `push esi`
  - 75F24399: `push dword ptr ss:[ebp+10]`
  - 75F2439C: `xor esi,esi`
  - 75F2439E: `push dword ptr ss:[ebp+C]`
  - 75F243A1: `inc esi`
  - 75F243A2: `push dword ptr ss:[ebp+8]`
  - 75F243A5: `call dword ptr ds:[&NtFlushInstructionCache]`
  - 75F243AB: `test eax,ecx`
  - 75F243AD: `jmp kernel32.75F4E2D0`
- Registers:** EAX: 75F2435F, ECX: 00000EC8, EDX: 10029000, EBP: 0018FDB8, ESP: 0018FD1C, ESI: 0018FDA8, EDI: 00000000.
- Memory Dump:** Address 10000000. Hex: 5A E0 3E ED 51 00 00 04 00 00 00 FF FF 00 00 B2 a 1 Q. ASCII: . . . . .

Now here as we know we will get an exception after VirtualProtect has been called 6 times, that means it had finished the unpacking now we can dump it .

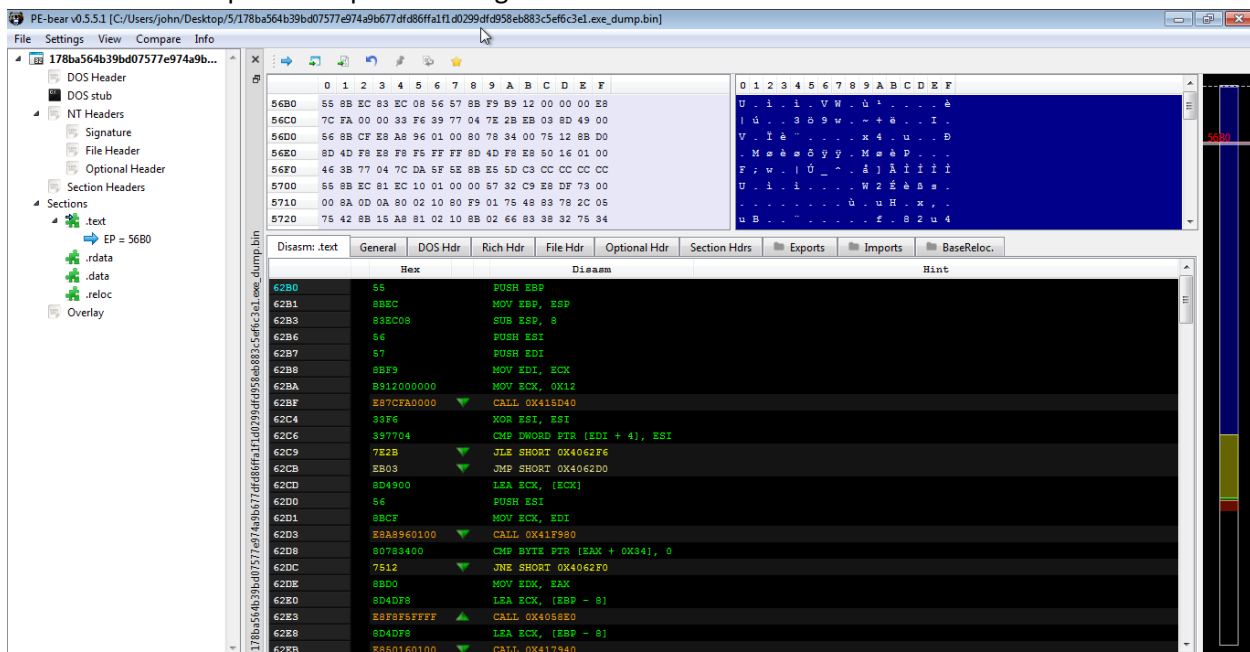
The screenshot shows the debugger interface with the following details:

- File View:** 178ba54b39bd07577e974a9b677df86ffa1f1d0299df958eb883c5ef6c3e1.exe - PID: 1460 - Module: kernelbase.dll - Thread: Main Thread 1028 - x32dbg [Elevated]
- Log:** CPU window showing assembly instructions. Key instructions include:
  - 75F21856: `<kernel32.dll.VirtualAlloc>`
  - 75F2435F: `<kernel32.dll.VirtualProtect>`
- Registers:** EAX: 75F2435F, ECX: 00000EC8, EDX: 10029000, EBP: 0018FDB8, ESP: 0018FD1C, ESI: 0018FDA8, EDI: 00000000.
- Memory Dump:** Address 10000000. Hex: 5A E0 3E ED 51 00 00 04 00 00 00 FF FF 00 00 B2 a 1 Q. ASCII: . . . . .

This malware is self injecting, so that means it will overwrite it process , so Now we can dump this using process hacker , double click on dridex and open memory tab and dump it by right clicking on memory location at 10000000.

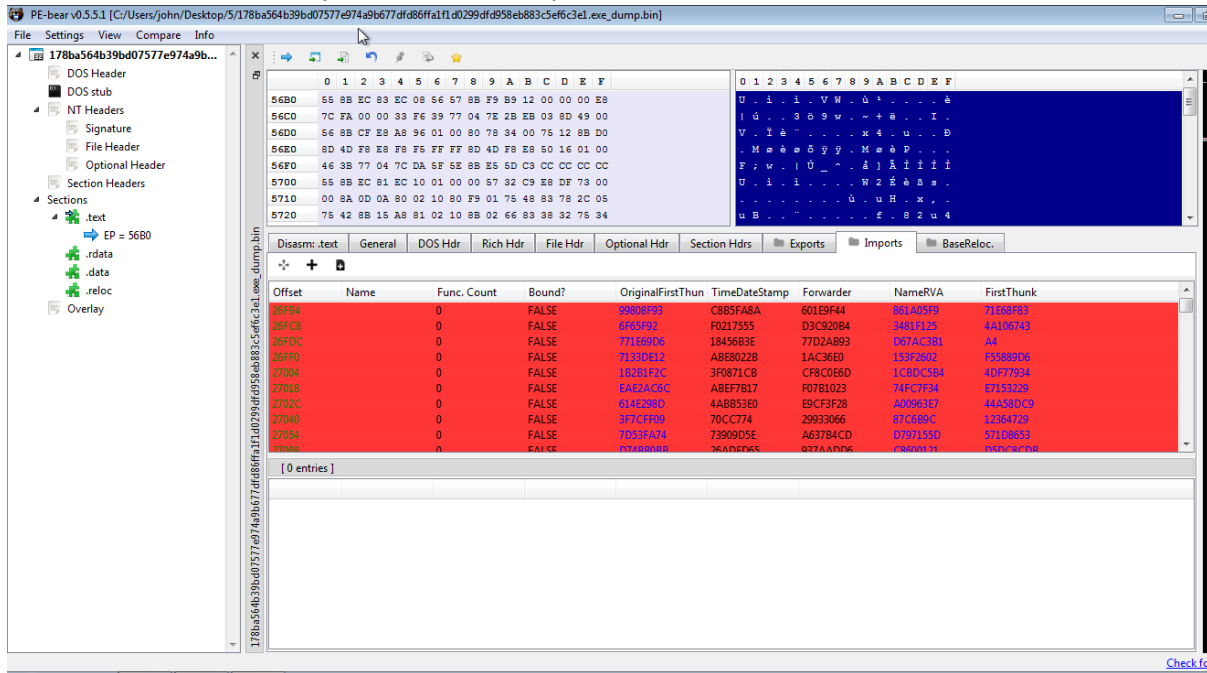


Now we will unmap the dumped file using PE-bear.

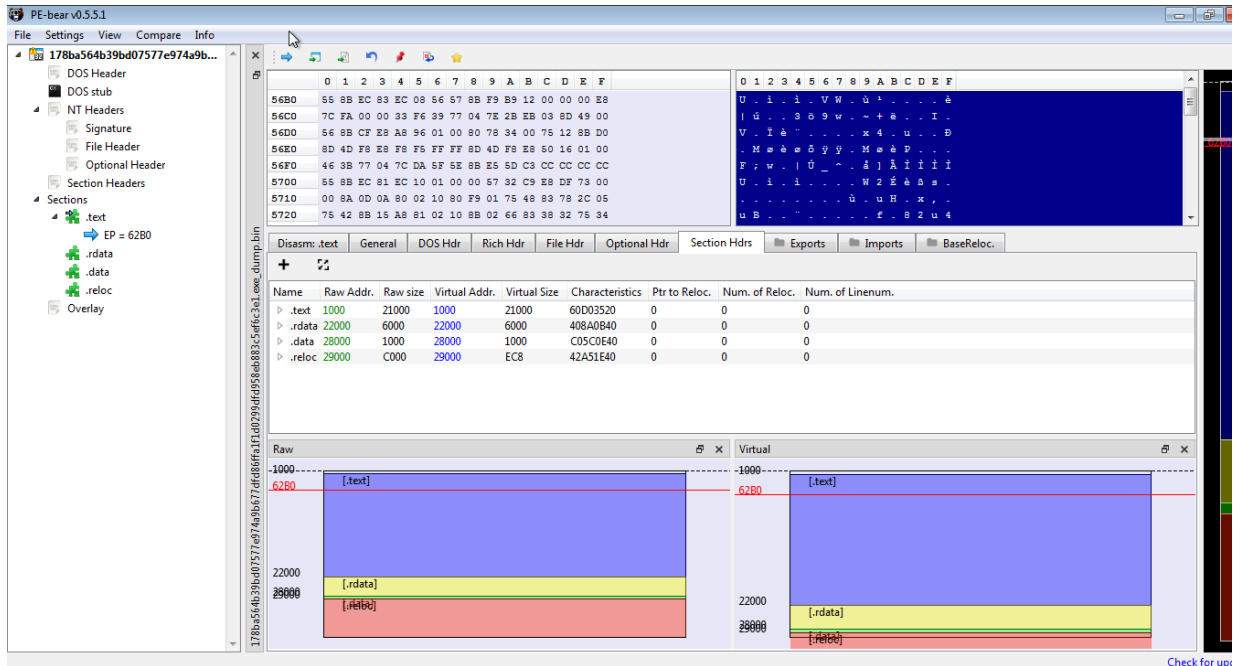




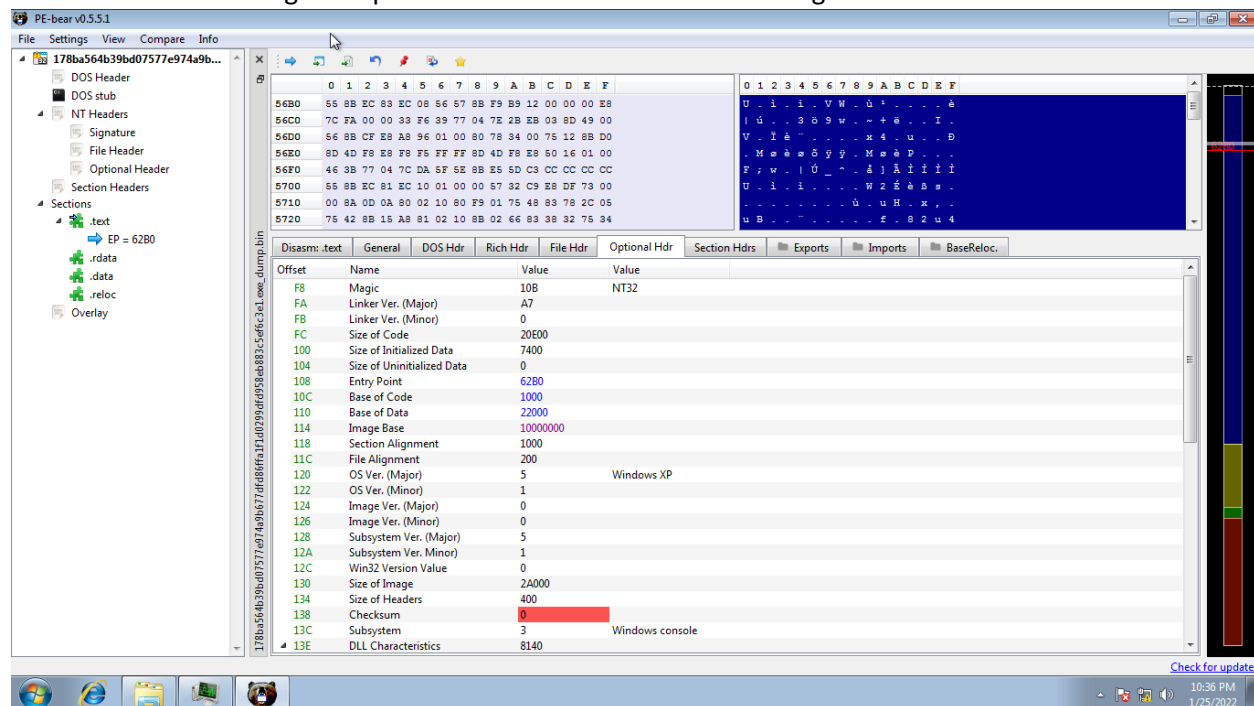
Here we can see all the imports are messed up.



So now here we will fix the section header, initially we will make raw addr. Same as virtual addr. And then calculate the raw size and then make the virtual size same as raw size.



Now after that we will go to Optional Hdr. Tab and then fix the image base to 10000000.



Now here we can check the name in exe i.e. ldr.exe which is loader for dridex, now we can dump it.

