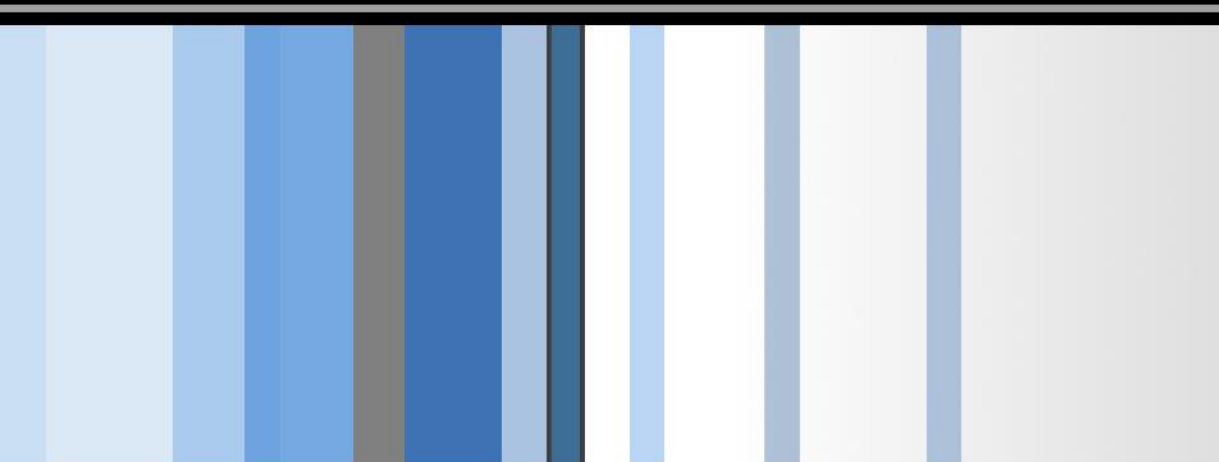


CSC 471 Modern Malware Analysis

Static Analysis & Dynamic Analysis (2) :

(De)Obfuscation

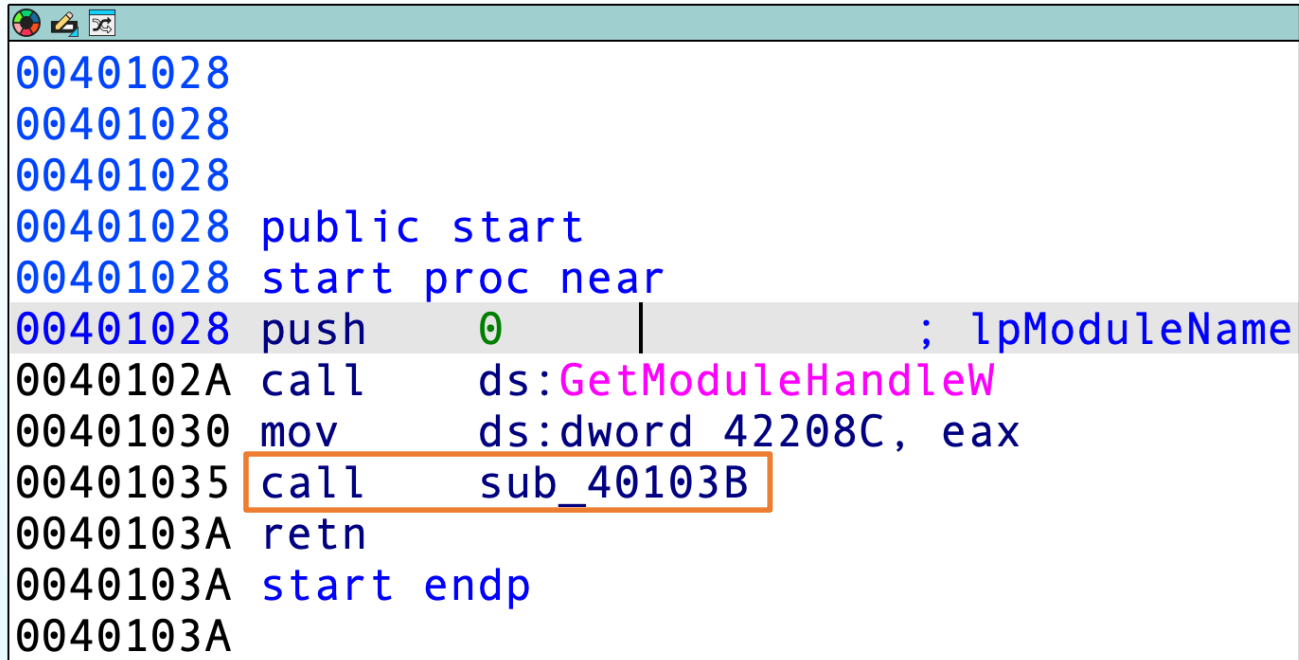
Si Chen (schen@wcupa.edu)



4298F9DDA63C3C1B17FEF433C082107A

(Trojan.Win32.Agent.b)

Load 4298F9DDA63C3C1B17FEF433C082107A into IDA




```
00401028
00401028
00401028
00401028 public start
00401028 start proc near
00401028 push     0                ; lpModuleName
0040102A call     ds:GetModuleHandleW
00401030 mov     ds:dword 42208C, eax
00401035 call     sub_40103B
0040103A retn
0040103A start endp
0040103A
```

Here, we can observe that after obtaining its own module handle, the program assigns the return value (stored in `eax`) to an address, followed by a `call`. Let's follow this `call` to see what it does

First, let's jump into the first **call** to examine its content

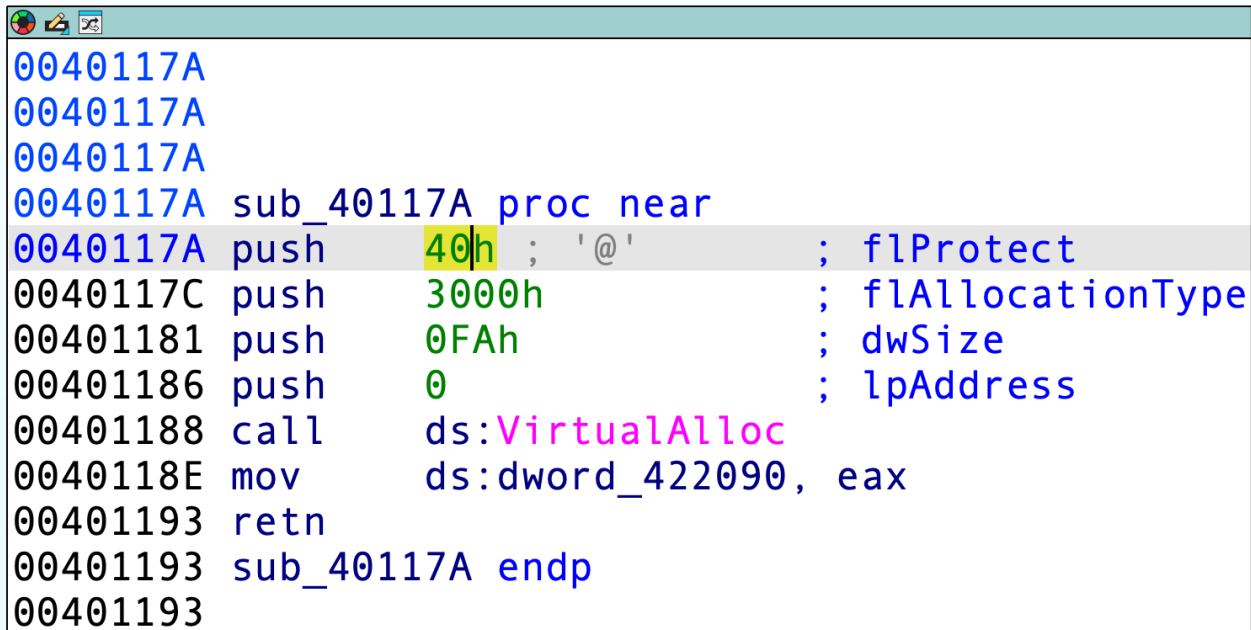
```

0040103B
0040103B
0040103B sub_40103B proc near
0040103B pminsw xmm0, xmm1
0040103F pminsw xmm3, xmm4
00401043 call sub_40117A
00401048 call sub_4010E6
0040104D jmp short loc_40106C
  
```



```

0040106C
0040106C loc_40106C:
0040106C push offset LibFileName ; "user32.dll"
00401071 call ds:LoadLibraryA
00401077 push offset ProcName ; "user_api_function"
0040107C push eax ; hModule
0040107D call ds:GetProcAddress
00401083 call ds:GetLastError
00401089 add eax, offset unk_422011
0040108F
  
```

```
0040117A
0040117A
0040117A
0040117A sub_40117A proc near
0040117A push 40h ; '@' ; flProtect
0040117C push 3000h ; flAllocationType
00401181 push 0FAh ; dwSize
00401186 push 0 ; lpAddress
00401188 call ds:VirtualAlloc
0040118E mov ds:dword_422090, eax
00401193 retn
00401193 sub_40117A endp
00401193
```

As we can see, the purpose of this **call** is to allocate memory space using the **VirtualAlloc** function. It is reasonable to believe that the decrypted code will likely be stored here.

```
0040103B
0040103B
0040103B sub_40103B proc near
0040103B pminsw xmm0, xmm1
0040103F pminsw xmm3, xmm4
00401043 call sub_40117A
00401048 call sub_4010E6
0040104D jmp short loc_40106C
```



```
0040106C
0040106C loc_40106C:
0040106C push offset LibFileName ; "user32.dll"
00401071 call ds:LoadLibraryA
00401077 push offset ProcName ; "user_api_function"
0040107C push eax ; hModule
0040107D call ds:GetProcAddress
00401083 call ds:GetLastError
00401089 add eax, offset unk_422011
```

Returning to the previous level, let's examine the content of the second call

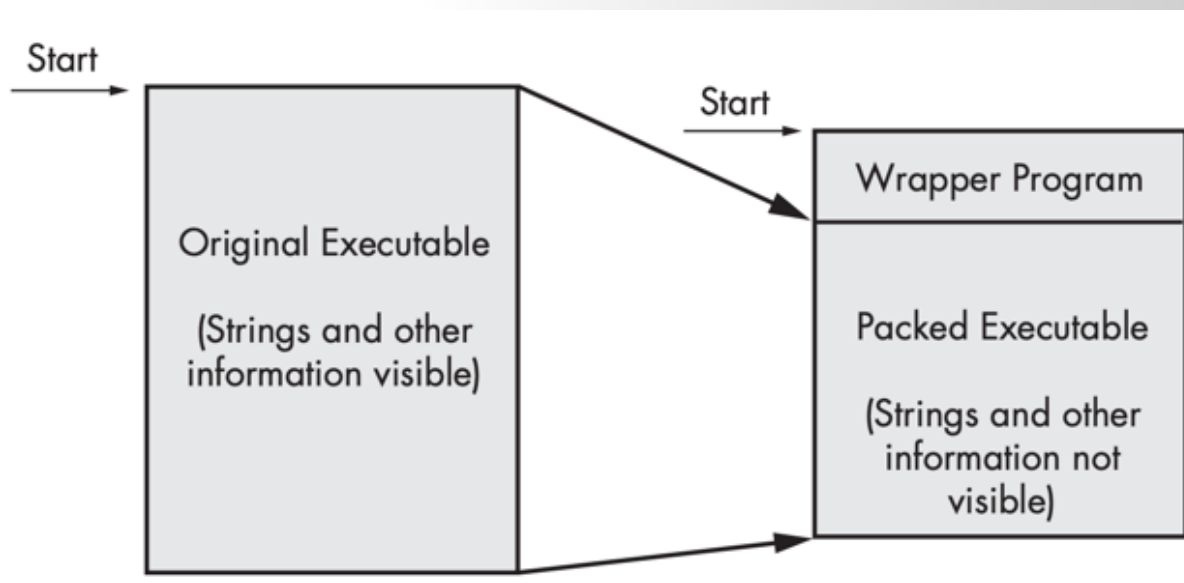
```
004010E6
004010E6
004010E6
004010E6 sub_4010E6 proc near
004010E6 mov     ebx, ds:dword_422090
004010EC or      dx, 0CB26h
004010F1 and     eax, ebx
004010F3 mov     edi, ebx
004010F5 not     ax
004010F8 not     dx
004010FB mov     esi, offset unk_445359
00401100 xor     edx, 0CB55h
00401106 and     edx, edx
00401108 mov     ecx, 0FAh
0040110D xor     ax, di
00401110 or      dx, 1Dh
00401114 rep movsb
00401116 retn
00401116 sub_4010E6 endp
00401116
```

In fact, this is a self-protection mechanism used by viruses, known as obfuscation, or it can also be understood as a "shell" written by the virus author for their malicious program.

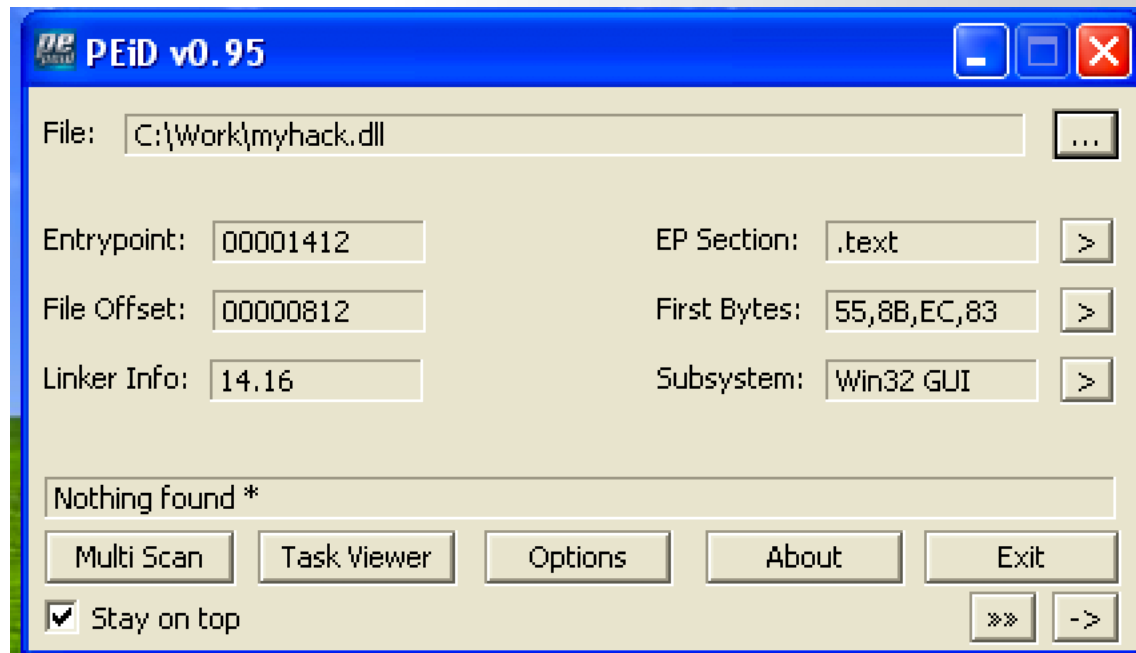
Here, we can see that operations such as **and**, **not**, and **xor** are used for decryption. This is something that should not appear in a normal program, so we can directly flag it as malicious: **Trojan.Win32.Agent.c**.

Packed and Obfuscated Malware

- Malware writers often use **packing or obfuscation** to make their files more difficult to detect or analyze.
- **Obfuscated** programs are ones whose execution the malware author has attempted to hide.
- **Packed** programs are a subset of obfuscated programs in which the malicious program is compressed and cannot be analyzed.
- Both techniques will severely limit your attempts to statically analyze the malware.



Packed and Obfuscated Malware



Packers and Cryptos

```
→ ~ upx -o myhack_packed.dll myhack.dll
      Ultimate Packer for eXecutables
      Copyright (C) 1996 - 2018
UPX 3.95      Markus Oberhumer, Laszlo Molnar & John Reiser      Aug 26th 2018

      File size      Ratio      Format      Name
      -----
      75264 ->      39424      52.38%      win32/pe      myhack_packed.dll

Packed 1 file.
```

4fafbfd2e560778f11beb8f736e80bb1

(Trojan.Win32.Agent.b)

IDA jump to 0x00613B50, which is the location of the `main` function. This is where the actual code of the sample is executed and is the focus of our analysis.


```
00613B98 cmp    word ptr [ebp+nShowCmd+2], 0 ; Compare Two Operands
00613B9D jnz    short loc_613BA6 ; Jump if Not Zero (ZF=0)
```

```
00613B9F call    loc_53FA40 ; Call Procedure
00613BA4 jmp     short loc_613BB5 ; Jump
```

```
00613BA6 loc_613BA6:
00613BA6 mov     eax, [ebp+hInstance]
00613BA9 push    esi
00613BAA push    edi
00613BAB push    ebx
00613BAC push    eax
00613BAD call    sub_6E8010 ; Call Procedure
00613BB2 add     esp, 10h ; Add
```

In this **main** function, we primarily analyze the **call** instructions. For example, let's look at the **call** located at 0x00613BAD.

```

006E8010 sub     esp, 834h          ; Integer Subtraction
006E8016 push    ebx
006E8017 xor     ebx, ebx      ; Logical Exclusive OR
006E8019 push    esi
006E801A mov     [esp+83Ch+var_1], bl
006E8021 mov     [esp+83Ch+var_B], bl
006E8028 mov     [esp+83Ch+var_2], 6Ch ; 'l'
006E8030 call    ds:GetEnvironmentStrings ; Indirect Call Near Procedure
006E8036 mov     ds:dword_76046C, eax
006E803B call    ds:GetProcessHeap ; Indirect Call Near Procedure
006E8041 mov     ds:dword_7603D0, eax
006E8046 mov     [esp+83Ch+var_C], 72h ; 'r'
006E804E call    sub_46B1C0      ; Call Procedure
006E8053 mov     al, [esp+83Ch+var_2]
006E805A mov     dl, [esp+83Ch+var_C]
006E8061 mov     cl, 73h ; 's'
006E8063 push    offset Buffer    ; lpString2
006E8068 push    offset Buffer    ; lpString1
006E806D mov     [esp+844h+var_11], cl
006E8074 mov     [esp+844h+var_3], al
006E807B mov     [esp+844h+var_4], 64h ; 'd'
006E8083 mov     [esp+844h+var_F], dl
006E808A mov     [esp+844h+var_E], cl
006E8091 mov     [esp+844h+var_5], 2Eh ; '.'
006E8099 mov     [esp+844h+var_6], al
006E80A0 mov     [esp+844h+var_7], al

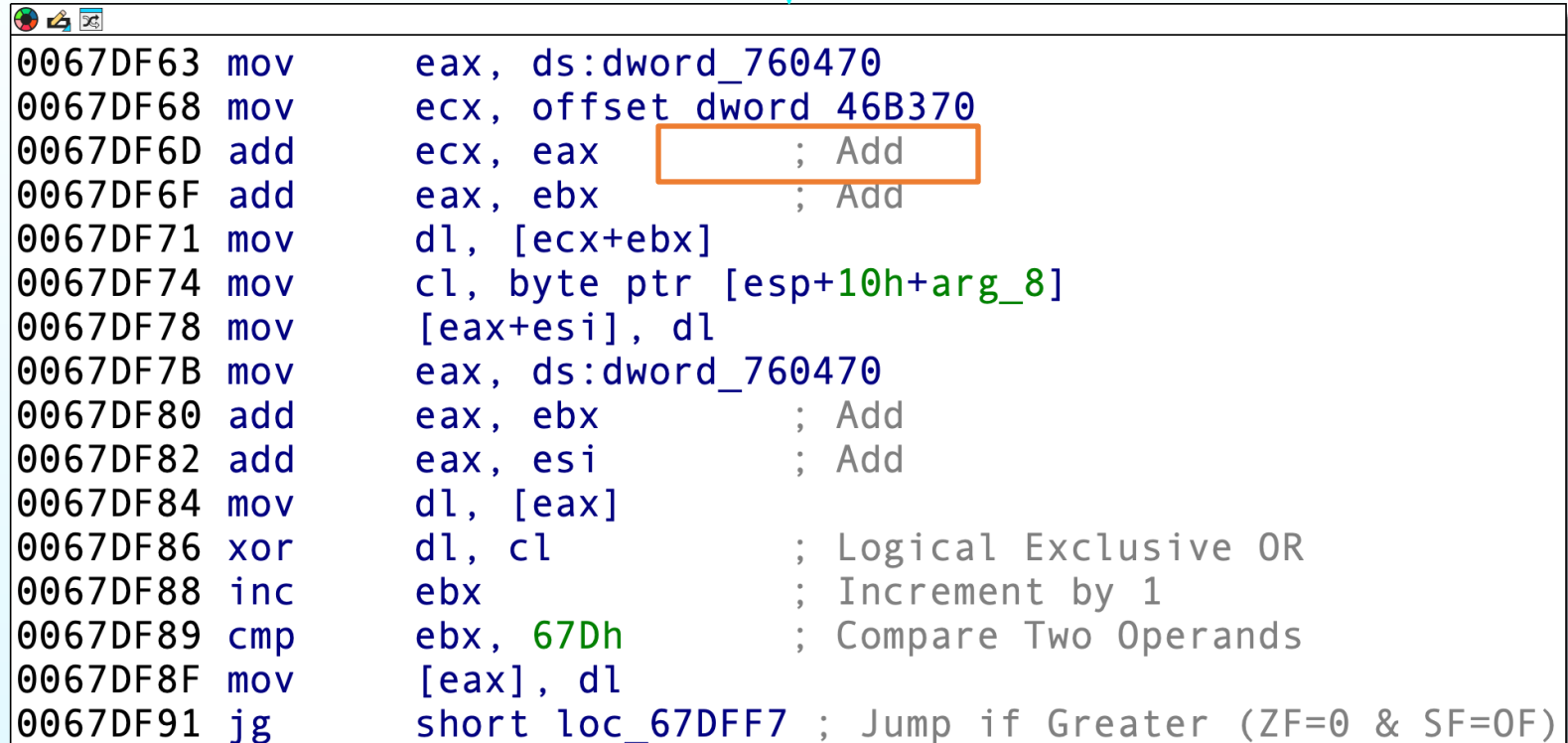
```

- In this code, we can see many instances of single characters being moved into memory. As mentioned earlier, this is highly suspicious.

```

006E81E0 mov     cl, [esp+83Ch+var_C]
006E81E7 mov     [esp+83Ch+var_834], edx
006E81EB mov     dl, [esp+83Ch+var_11]
006E81F2 mov     [esp+83Ch+var_E], dl
006E81F9 mov     edx, offset dword_53FA20
006E81FE add     edx, 10h ; Add
006E8201 mov     [esp+83Ch+var_F], cl
006E8208 mov     cl, [esp+83Ch+var_2]
006E820F push    edx ; int
006E8210 push    eax ; int
006E8211 push    ebx ; nIndex
006E8212 mov     [esp+848h+var_82C], offset loc_4010B0
006E821A mov     [esp+848h+var_808], offset unk_756180
006E8222 mov     ds:pcbBuffer, ebx
006E8228 mov     ds:dword_759350, ebx
006E822E mov     ds:dword_7609B8, eax
006E8233 mov     [esp+848h+var_4], 64h ; 'd'
006E823B mov     [esp+848h+var_5], 2Eh ; '.'
006E8243 mov     [esp+848h+var_6], cl
006E824A call     sub_67DF30 ; Call Procedure
006E824F mov     al, [esp+848h+var_2]
006E8256 mov     cl, [esp+848h+var_4]
006E825D lea     edx, [esp+848h+var_834] ; Load Effective Address
006E8261 mov     [esp+848h+var_7], al
006E8268 push    edx
006E8269 mov     [esp+84Ch+var_8], cl
006E8270 mov     [esp+84Ch+var_9], 74h ; 't'

```



```
0067DF63 mov     eax, ds:dword_760470
0067DF68 mov     ecx, offset dword 46B370
0067DF6D add     ecx, eax ; Add
0067DF6F add     eax, ebx ; Add
0067DF71 mov     dl, [ecx+ebx]
0067DF74 mov     cl, byte ptr [esp+10h+arg_8]
0067DF78 mov     [eax+esi], dl
0067DF7B mov     eax, ds:dword_760470
0067DF80 add     eax, ebx ; Add
0067DF82 add     eax, esi ; Add
0067DF84 mov     dl, [eax]
0067DF86 xor     dl, cl ; Logical Exclusive OR
0067DF88 inc     ebx ; Increment by 1
0067DF89 cmp     ebx, 67Dh ; Compare Two Operands
0067DF8F mov     [eax], dl
0067DF91 jg     short loc_67DFF7 ; Jump if Greater (ZF=0 & SF=0F)
```

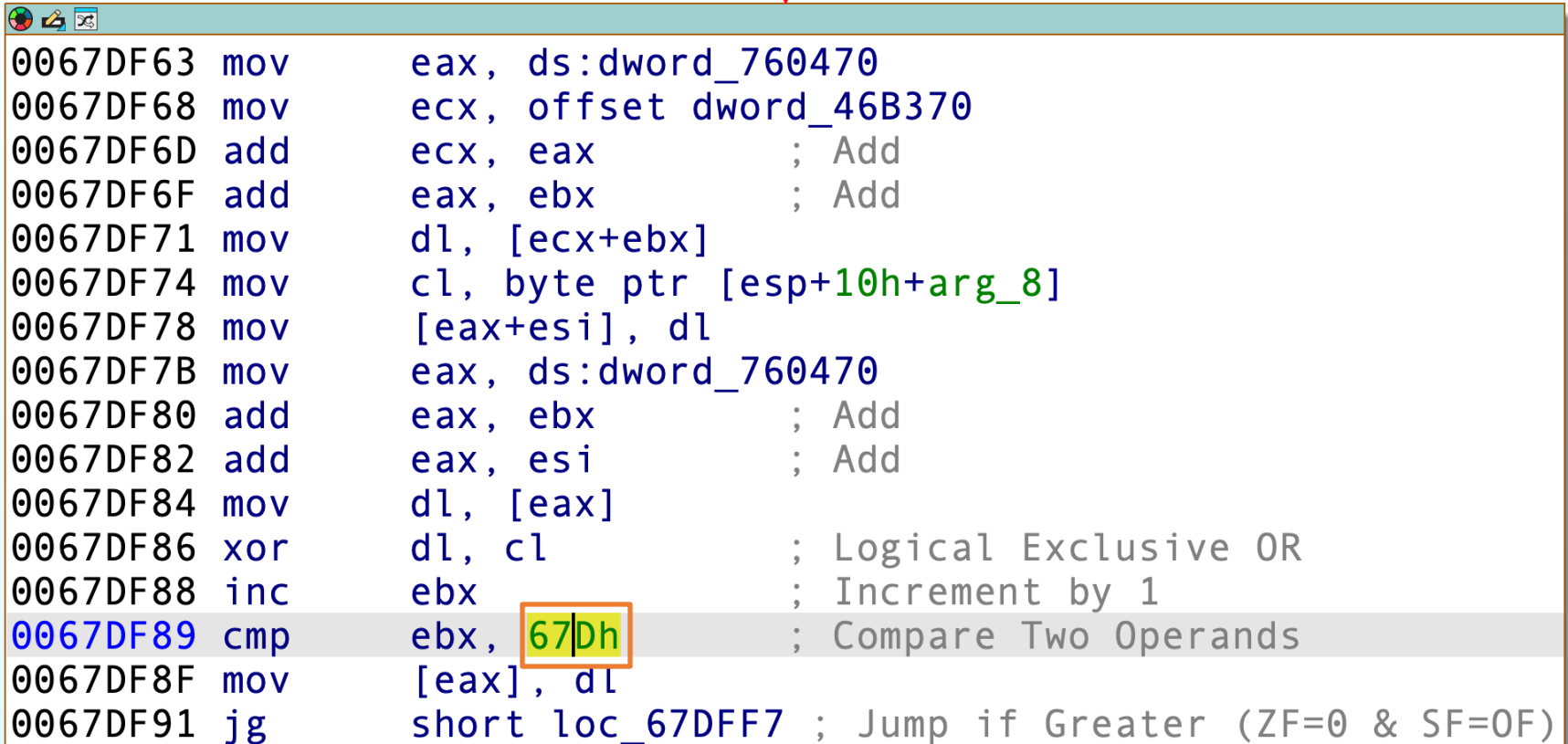
Starting from 0x0067DF63, this is actually a decryption process. Why do we say that? First, at 0x0067DF68, there is a `mov` assignment statement, which assigns the content at address 0x0046B370 to `ecx`. Let's take a look at the content at this address

4fafbfd2e560778f11beb8f736e80bb1

```
0046B36A sub_46B1C0 endp
0046B36A
0046B36A ; -----
0046B36B align 10h
0046B370 dword_46B370 dd 4D3217F0h, 0D12B6A4Ah, 8FC96873h, 1372219Dh, 452AC12Ch
0046B370 ; DATA XREF: sub_67DF30+38↓o
0046B384 dd 784BCAE3h, 19B671ADh, 42F911CDh, 0D29C17A9h, 0F6F3362Dh
0046B398 dd 0E7BD5C2Ch, 0F8287871h, 8B55CCA5h, 0A4A0ED52h, 0DFC954ECh
0046B3AC dd 0D7AA7171h, 1E0D5F7Dh, 5F20F47Ah, 4DB5D5E6h, 0FE92DFD8h
0046B3C0 dd 30A31FC3h, 6F7D4BA6h, 924F4A50h, 0AB8DA0C2h, 47DE9CAEh
0046B3D4 dd 460ACADBh, 0A17E5AAFh, 213EAA7Ah, 45109FDDh, 47FD26E7h
0046B3E8 dd 9D9E49ADh, 1372F303h, 0DE211FA5h, 0C3BE3BD5h, 1942C019h
0046B3FC dd 7FF01C4h, 0FCDA171Dh, 8B486A2Ch, 40C6CA27h, 2BF215F0h
0046B410 dd 4EF1E9F3h, 0F2113FDEh, 983A8C0Dh, 935154EAh, 0BBBD97DDh
0046B424 dd 0A329B5BBh, 1F5E55AFh, 7F95B91h, 313C1D97h, 0A783E59h
0046B438 dd 86C99F30h, 9AA422B7h, 9E5478F0h, 0F2AABAA8h, 0C47A8C4Ah
0046B44C dd 6BF20EF0h, 0E7ABF5A4h, 0E4744626h, 0EAC22F51h, 0E075588h
0046B460 dd 45FACD2Eh, 93F522FFh, 0A056875Eh, 0EC8DC304h, 0DA539CA5h
0046B474 dd 0B56CB535h, 0C7CBDA27h, 1D22AB6Dh, 0FDAB4E21h, 0F2340626h
0046B488 dd 933A8C0Ah, 935A54EAh, 0BBA690DDh, 8A00BE84h, 195E7965h
0046B49C dd 7FF05C4h, 0FCDA17DDh, 0A2646F2Dh, 9242F3DBh, 98F02508h
0046B4B0 dd 8DDACA1h, 0C987B2Fh, 0A905D061h, 3DF02E7Dh, 4521635Ah
0046B4C4 dd 0C2BF36D3h, 92420CD4h, 3F3CAA85h, 4ACEDD2Fh, 0ED4B4A66h
0006A770 0046B370: .0003:dword_46B370
```

As you can see, this is a bunch of garbled data, likely encrypted.

4fafbfd2e560778f11beb8f736e80bb1



```
0067DF63 mov     eax, ds:dword_760470
0067DF68 mov     ecx, offset dword_46B370
0067DF6D add     ecx, eax           ; Add
0067DF6F add     eax, ebx           ; Add
0067DF71 mov     dl, [ecx+ebx]
0067DF74 mov     cl, byte ptr [esp+10h+arg_8]
0067DF78 mov     [eax+esi], dl
0067DF7B mov     eax, ds:dword_760470
0067DF80 add     eax, ebx           ; Add
0067DF82 add     eax, esi           ; Add
0067DF84 mov     dl, [eax]
0067DF86 xor     dl, cl           ; Logical Exclusive OR
0067DF88 inc     ebx               ; Increment by 1
0067DF89 cmp     ebx, 67Dh        ; Compare Two Operands
0067DF8F mov     [eax], dl
0067DF91 jg     short loc_67DFF7 ; Jump if Greater (ZF=0 & SF=0F)
```

Following this, there is a series of operations, including **add** (addition) and **xor** (exclusive OR). The **xor** operation, in particular, is a common decryption technique often used by malicious programs. From the final **inc** (increment) and **cmp** (compare) operations, we can deduce that **ebx** holds the number of binary codes to be decrypted, which is 0x67D in this case.

Dynamic Analysis

Dynamic Analysis

- Dynamic analysis is the process of executing malware in a monitored environment to observe its behaviors.

4fafbfd2e560778f11beb8f736e80bb1

(revisited)

(Trojan.Win32.Agent.b)

Packed and Obfuscated Malware

•Definition:

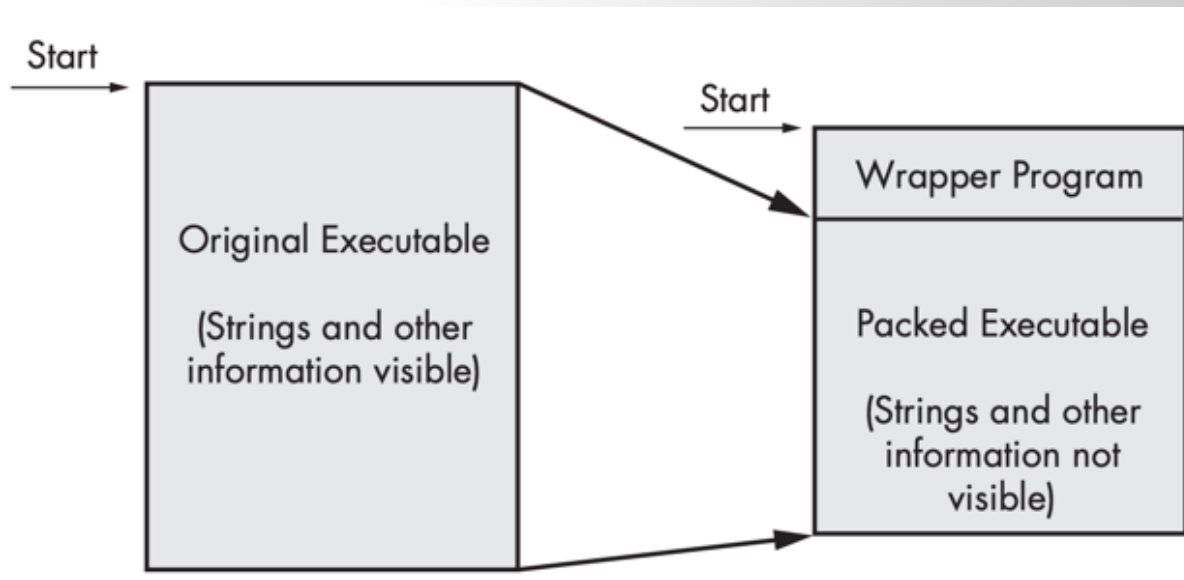
Obfuscation is a technique used by malware authors to hide malicious code.

•How it works:

- The PE (Portable Executable) file is encrypted and embedded within the program.
- During execution, the program decrypts and runs the hidden PE file.

•Goal:

To conceal the malicious payload and evade detection



How Obfuscation Works

1.Encryption:

The malicious PE file is encrypted and embedded in the program.

2.Execution:

1. The program allocates memory using functions like **VirtualAlloc**.
2. Decrypts the PE file into the allocated memory.
3. Executes the decrypted payload.

3.Result:

The malicious code runs hidden from detection tools.

Removing Obfuscation: Key Steps

1. Set Breakpoints:

Use a debugger (e.g., OllyDbg) to set breakpoints on memory allocation functions like **VirtualAlloc**.

2. Monitor Memory Allocation:

1. Track the starting address of allocated memory.
2. Set hardware breakpoints to detect writes to this memory.

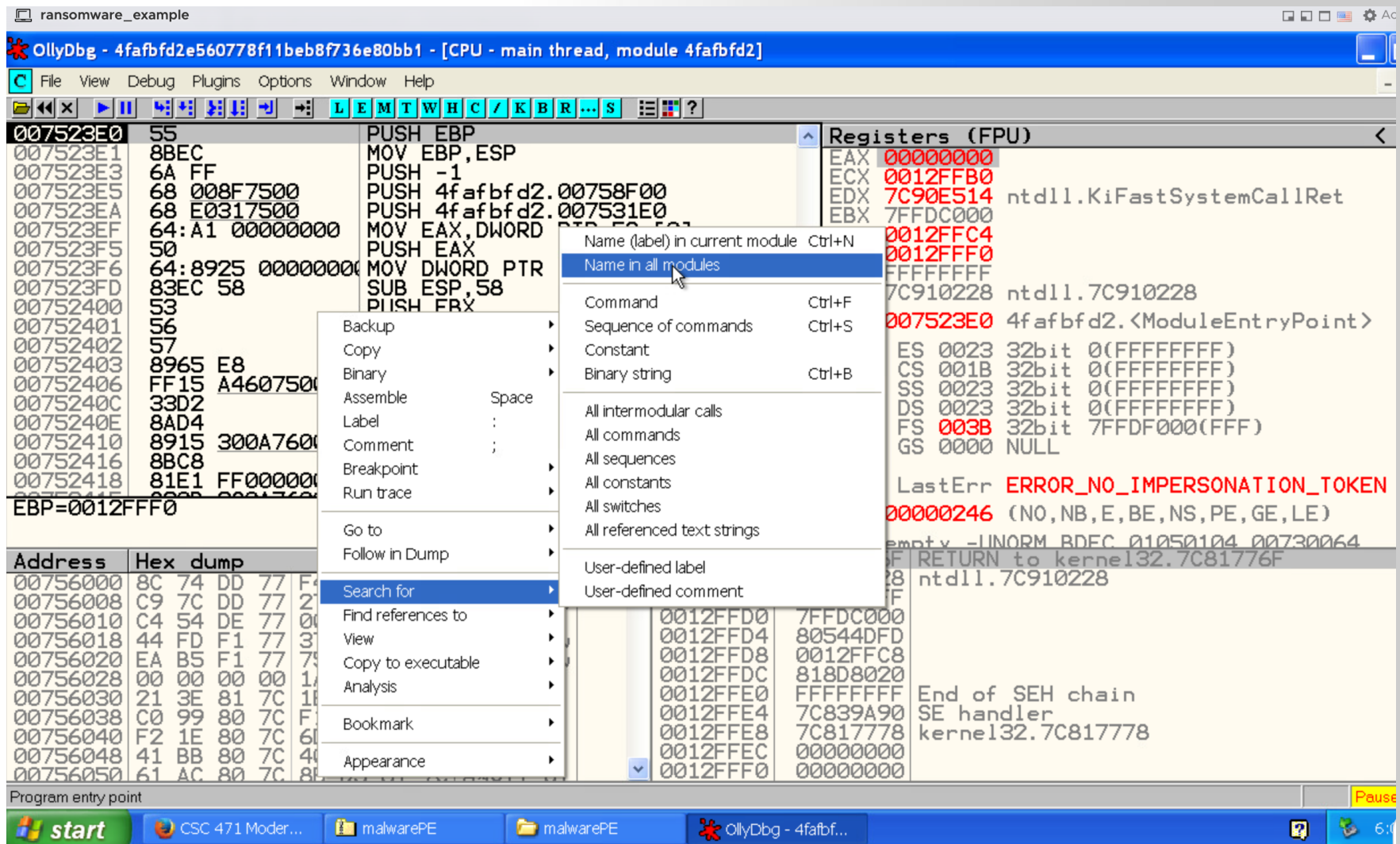
3. Analyze Decryption:

1. Observe the decryption process in memory.
2. Dump the decrypted PE file for further analysis.

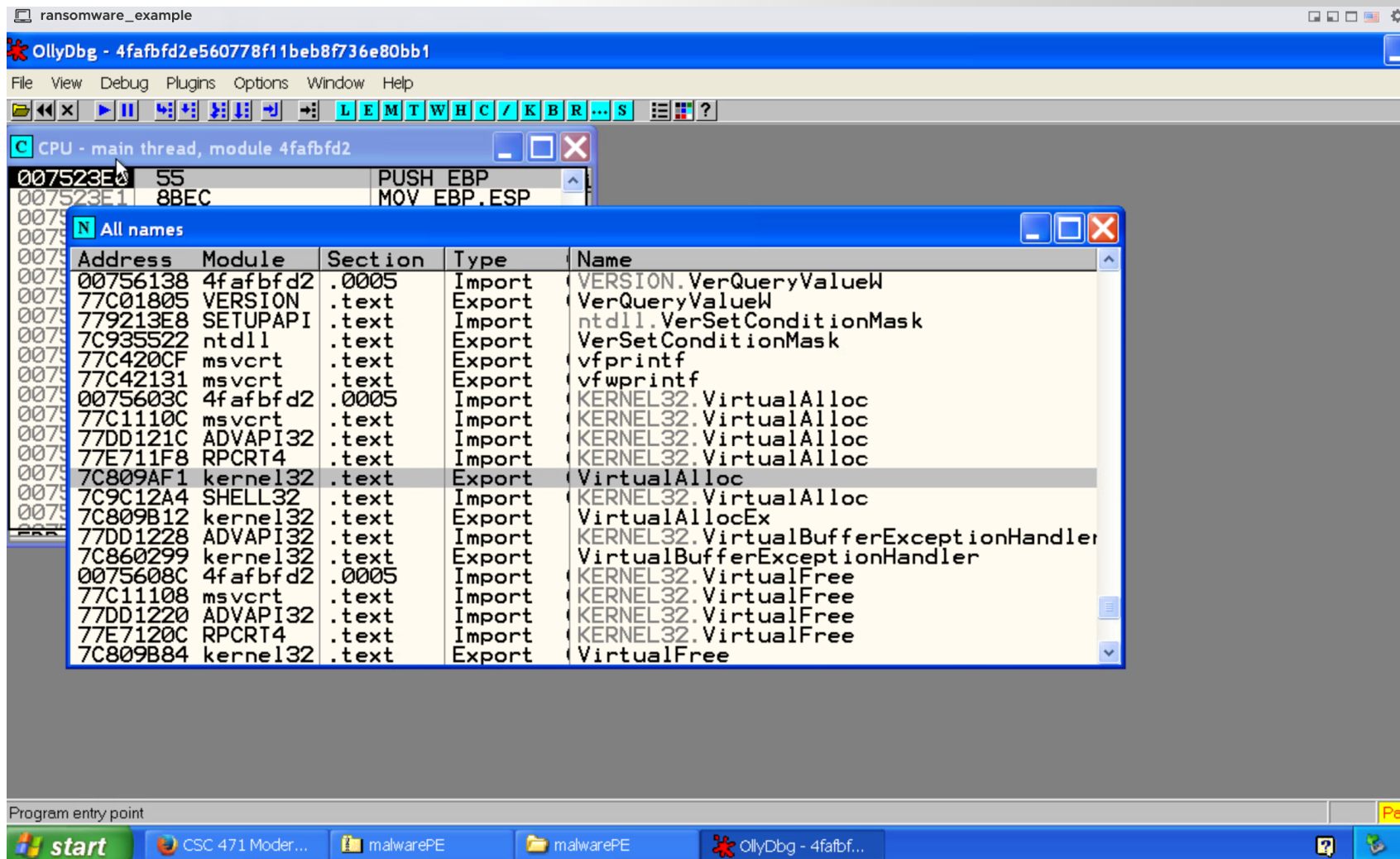
Load target file into OllyDBG

The screenshot displays the OllyDbg interface with the following components:

- Assembly View:** Shows assembly instructions for the main thread. The instruction at address 007523E0 is `PUSH EBP`. Subsequent instructions include `MOV EBP, ESP`, `PUSH -1`, `PUSH 4fafbfd2.00758F00`, `PUSH 4fafbfd2.007531E0`, `MOV EAX, DWORD PTR FS:[0]`, `PUSH EAX`, `MOV DWORD PTR FS:[0], ESP`, `SUB ESP, 58`, `PUSH EBX`, `PUSH ESI`, `PUSH EDI`, `MOV DWORD PTR SS:[EBP-18], ESP`, `CALL DWORD PTR DS:[<KERNEL32.Ge`, `XOR EDX, EDX`, `MOV DL, AH`, `MOV DWORD PTR DS:[760A30], EDX`, `MOV ECX, EAX`, and `AND ECX, 0FF`. The instruction at 00752415 is partially visible as `MOV DWORD PTR DS:[760A30], EDX`.
- Registers (FPU):** Displays the state of CPU registers. EAX is 00000000, ECX is 0012FFB0, EDX is 7C90E514 (labeled `ntdll.KiFastSystemCallRet`), EBX is 7FFDC000, ESP is 0012FFC4, EBP is 0012FFFF, ESI is FFFFFFFF, EDI is 7C910228 (labeled `ntdll.7C910228`), and EIP is 007523E0 (labeled `4fafbfd2.<ModuleEntryPoint>`). Control flags (C, P, A, Z, S, T, D, O) are shown with their respective values and bit positions. The LastErr register is highlighted in red and contains `ERROR_NO_IMPERSONATION_TOKEN`. EFL is 0000246 (labeled `(NO, NB, E, BE, NS, PE, GE, LE)`). The status of the instruction queue is shown as `ST0 empty - UNORM RDEC 01050104 00730064`.
- Memory Dump:** Shows a hex dump of memory starting at address 00756000. The dump includes hex values and their corresponding ASCII representations. For example, at address 00756000, the hex is `8C 74 DD 77 F4 E9 DD 77` and the ASCII is `it|w|l|w`. The dump continues with more hex and ASCII data.
- Program Entry Point:** Located at the bottom of the interface, it shows the start of the program.



Right click → “Search for” → Name in all modules



Looking for “VirtualAlloc” func → double click

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread, module kernel32]

File View Debug Plugins Options Window Help

LEMTWHC / KBR... S

Address	Hex dump	ASCII
00756000	8C 74 DD 77 F4 E9 DD 77	it w f w
00756008	C9 7C DD 77 27 6C DD 77	fl w l w
00756010	C4 54 DE 77 00 00 00 00	-T w . . .
00756018	44 FD F1 77 37 D8 F1 77	D² ±w7±w
00756020	EA B5 F1 77 75 76 F2 77	Ω±wuv±w
00756028	00 00 00 00 1A 98 80 7C	. . . ±w
00756030	21 3E 81 7C 1B 77 81 7C	!>ü ±wü
00756038	C0 99 80 7C F1 9A 80 7C	LöÇ ±üÇ
00756040	F2 1E 80 7C 6D 0C 81 7C	≥±Ç ±ü
00756048	41 BB 80 7C 40 AE 80 7C	AüÇ @«Ç
00756050	61 AC 80 7C 8B D3 81 7C	a%Ç iüü

Program entry point

Paused

Registers (FPU)

Register	Value
ES	0023 32bit 0(FFFFFFFF)
CS	001B 32bit 0(FFFFFFFF)
SS	0023 32bit 0(FFFFFFFF)
DS	0023 32bit 0(FFFFFFFF)
FS	003B 32bit 7FFDF000(FFF)
GS	0000 NULL

LastErr ERROR_NO_IMPERSONATION_TOKEN (000000246) (NO,NB,E,BE,NS,PE,GE,LE)

empty -INORM BDEC 01050104 00730064

Return to kernel32.7C81776F

ntdll.7C910228

End of SEH chain

SE handler

kernel32.7C817778

Here we go into the VirtualAlloc function.

ransomware_example

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread, module kernel32]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

7C809AF1 8BFF MOV EDI, EDI

7C809AF3 55 Backup

7C809AF4 8BE0 Copy

7C809AF6 FF79 Binary

7C809AF9 FF79 Assemble Space

7C809AFC FF79 Label

7C809AFF FF79 Comment

7C809B02 6A Comment

7C809B04 E8 Breakpoint

7C809B09 5D Run trace

7C809B0A C2 Conditional F2

7C809B0D 90 Conditional log Shift+F2

7C809B0E 90 Conditional log Shift+F4

7C809B0F 90 Run to selection F4

7C809B10 90 Memory, on access

7C809B11 90 Memory, on write

7C809B12 6A Search for

7C809B14 68 Find references to

7C809B19 E8 View

Copy to executable

Analysis

Bookmark

Appearance

Address Hex

00756000 8C 74

00756008 C9 70

00756010 C4 54

00756018 44 FD F1 77 37 D8 F1 77 D2 ±w7±w

00756020 EA B5 F1 77 75 76 F2 77 Ω ±wuv±w

00756028 00 00 00 00 1A 98 80 7C ... →üÇ|

00756030 21 3E 81 7C 1B 77 81 7C !>ü|+wü|

00756038 C0 99 80 7C F1 9A 80 7C LÖÇ|±üÇ|

00756040 F2 1E 80 7C 6D 0C 81 7C ≥△Ç|m.ü|

00756048 41 BB 80 7C 40 AE 80 7C A7Ç|@<<Ç|

00756050 61 AC 80 7C 8B D3 81 7C a7Ç|iü|

Registers (FPU)

00000000

0012FFB0

7C90E514 ntdll.KiFastSystemCallRet

7FFDC000

0012FFC4

0012FFF0

FFFFFFFF

7C910228 ntdll.7C910228

007523E0 4fafbfd2.<ModuleEntryPoint>

ES 0023 32bit 0(FFFFFFFF)

CS 001B 32bit 0(FFFFFFFF)

SS 0023 32bit 0(FFFFFFFF)

DS 0023 32bit 0(FFFFFFFF)

FS 003B 32bit 7FFDF000(FFF)

GS 0000 NULL

LastErr ERROR_NO_IMPERSONATION_TOKEN (00000000246 (NO,NB,E,BE,NS,PE,GE,LE)

empty -LINORM BDEC 01050104 00730064

0012FFC4 7C81776F RETURN to kernel32.7C81776F

0012FFC8 7C910228 ntdll.7C910228

0012FFCC FFFFFFFF

0012FFD0 7FFDC000

0012FFD4 80544DFD

0012FFD8 0012FFC8

0012FFDC 818D8020

0012FFE0 FFFFFFFF

0012FFE4 7C839A90 End of SEH chain

0012FFE8 7C817778 SE handler

0012FFEC 00000000 kernel32.7C817778

0012FFF0 00000000

Program entry point

start CSC 471 Moder... malwarePE malwarePE OllyDbg - 4fafbf... 6:

Right click the first line (MOV EDI, EDI) → Breakpoint → Toggle

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread, module kernel32]

File View Debug Plugins Options Window Help

LEMTWHC / KBR... S

Address	Hex dump	ASCII
00756000	8C 74 DD 77 F4 E9 DD 77	it w f w
00756008	C9 7C DD 77 27 6C DD 77	fl w l w
00756010	C4 54 DE 77 00 00 00 00	-T w...
00756018	44 FD F1 77 37 D8 F1 77	D z w7 z w
00756020	EA B5 F1 77 75 76 F2 77	z z wuv z w
00756028	00 00 00 00 1A 98 80 7C	... -y C l
00756030	21 3E 81 7C 1B 77 81 7C	! > u l + w u l
00756038	C0 99 80 7C F1 9A 80 7C	L o C l z u C l
00756040	F2 1E 80 7C 6D 0C 81 7C	z A C l m. u l
00756048	41 BB 80 7C 40 AE 80 7C	A q C l w << C l
00756050	61 AC 80 7C 8B D3 81 7C	a % C l i l u l

Breakpoint at kernel32.VirtualAlloc

Registers (FPU)

Register	Value
ESI	00000362
EDI	00756180 4fafbfd2.00756180
EAX	0000442E
ECX	00002654
EDX	0012F64C
EBX	0012F6C0
ESP	0012F6E0
EBP	000003BA
7C809AF1	kernel32.VirtualAlloc
ES	0023 32bit 0(FFFFFFFF)
CS	001B 32bit 0(FFFFFFFF)
SS	0023 32bit 0(FFFFFFFF)
DS	0023 32bit 0(FFFFFFFF)
FS	003B 32bit 7FFDF000(FFF)
GS	0000 NULL
LastErr	ERROR_SUCCESS (00000000)
00000202	(NO,NB,NE,A,NS,PO,GE,G)
bad	-NaN FFFF C0000000 00000000
0012F64C	0012F88F
0012F650	00000000
0012F654	000000E0
0012F658	00003000
0012F65C	00000004
0012F660	0000000A
0012F664	7C80AE40
0012F668	00000000
0012F66C	0000067D
0012F670	0000002C
0012F674	0012F6AC
0012F678	7E4296F7

CALL to VirtualAlloc from 0012F889
Address = NULL
Size = E0 (224.)
AllocationType = MEM_COMMIT|MEM_RESERVE
Protect = PAGE_READWRITE
kernel32.GetProcAddress
RETURN to 0012F6AC
RETURN to USER32.7E4296F7 from USER32

Click the “play” button (F9) and the program will run and hit the break point
Check out the allocation size → 224 Bytes → too small

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread, module kernel32]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

Address	Hex dump	ASCII
00756000	8C 74 DD 77 F4 E9 DD 77	it w f w
00756008	C9 7C DD 77 27 6C DD 77	rl w l w
00756010	C4 54 DE 77 00 00 00 00	-T w . . .
00756018	44 FD F1 77 37 D8 F1 77	D z ± w 7 ± w
00756020	EA B5 F1 77 75 76 F2 77	± w u v ± w
00756028	00 00 00 00 1A 98 80 7C	. . . ± y Ç l
00756030	21 3E 81 7C 1B 77 81 7C	! ± ü ± w ü l
00756038	C0 99 80 7C F1 9A 80 7C	L ö Ç l ± ü Ç l
00756040	F2 1E 80 7C 6D 0C 81 7C	± ± Ç l m . ü l
00756048	41 BB 80 7C 40 AE 80 7C	A 7 Ç l @ ± Ç l
00756050	61 AC 80 7C 8B D3 81 7C	a 7 Ç l i ± ü l

Breakpoint at kernel32.VirtualAlloc

EDX=00000000

Registers (FPU)

Register	Value
0067F000	4fafbfd2.0067F000
0046CA00	4fafbfd2.0046CA00
000DCA00	
0053FA20	4fafbfd2.0053FA20
0012F64C	
0012F6C0	
0012F6E0	
00000000	

7C809AF1 kernel32.VirtualAlloc

Register	Value
ES	0023 32bit 0(FFFFFFFF)
CS	001B 32bit 0(FFFFFFFF)
SS	0023 32bit 0(FFFFFFFF)
DS	0023 32bit 0(FFFFFFFF)
FS	003B 32bit 7FFDF000(FFF)
GS	0000 NULL

LastErr ERROR_PROC_NOT_FOUND (0000007F)

00000206 (NO,NB,NE,A,NS,PE,GE,G)

had -NAN FEEF C0000000 00000000

CALL to VirtualAlloc from 0012F9B9

Register	Value
0012F64C	0012F9BC
0012F650	00000000
0012F654	000DCA00
0012F658	00003000
0012F65C	00000004
0012F660	0000000A
0012F664	7C80AE40
0012F668	00000000
0012F66C	004020B0
0012F670	004D6A10
0012F674	00540A40
0012F678	005AAB40

kernel32.GetProcAddress

4fafbfd2.004020B0

4fafbfd2.004D6A10

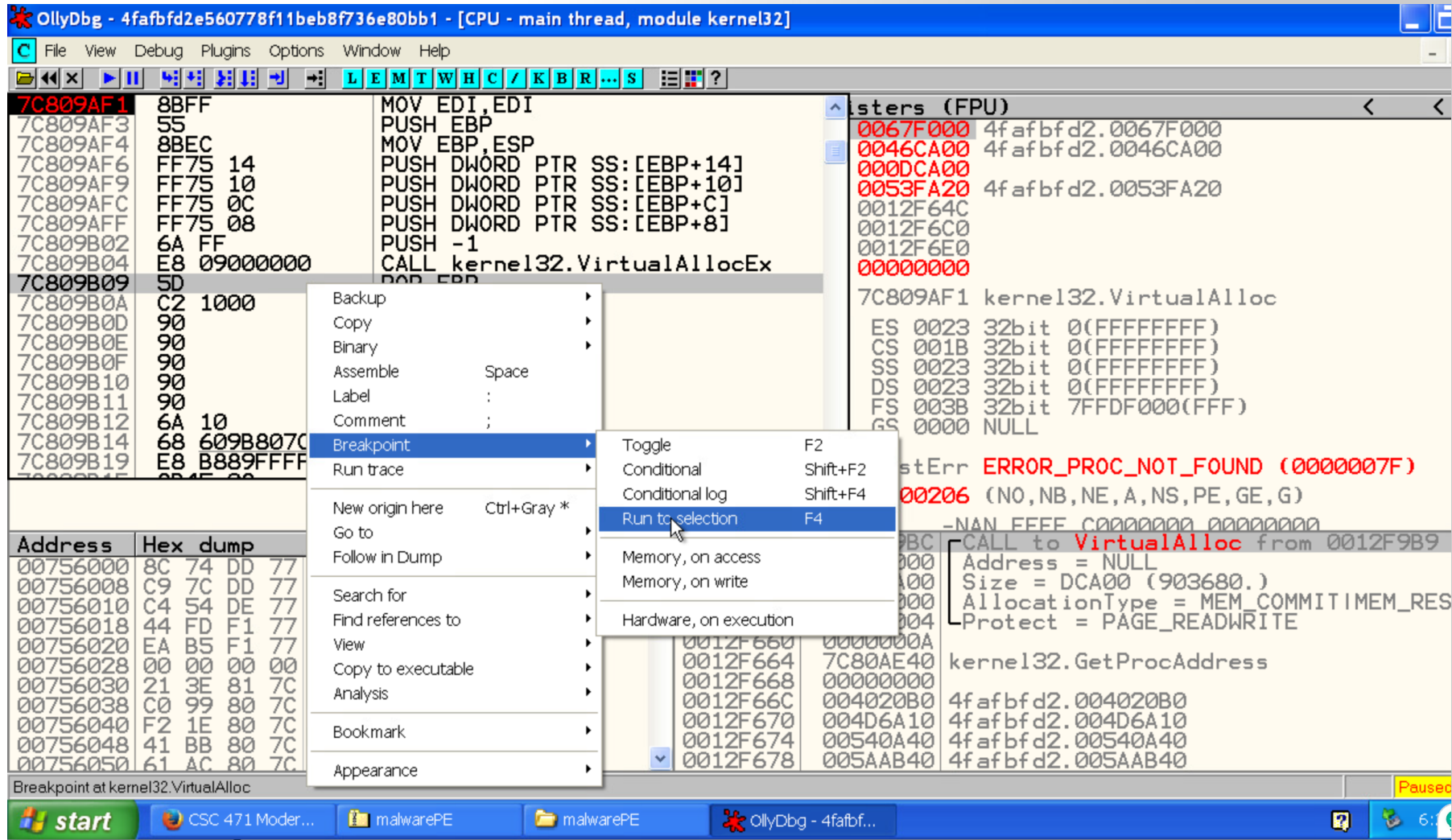
4fafbfd2.00540A40

4fafbfd2.005AAB40

Paused

Click the “play” button (F9) again.

Check out the allocation size → 903680 Bytes → Good!



Right click "POP EBP" link (below CALL VirtualAllocEx)
→ "Breakpoint" → Run to selection (F4)

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread, module kernel32]

File View Debug Plugins Options Window Help

LEMTWHCKBR...S

7C809AF1	8BFF	MOV EDI,EDI
7C809AF3	55	PUSH EBP
7C809AF4	8BEC	MOV EBP,ESP
7C809AF6	FF75 14	PUSH DWORD PTR SS:[EBP+14]
7C809AF9	FF75 10	PUSH DWORD PTR SS:[EBP+10]
7C809AFC	FF75 0C	PUSH DWORD PTR SS:[EBP+C]
7C809AFF	FF75 08	PUSH DWORD PTR SS:[EBP+8]
7C809B02	6A FF	PUSH -1
7C809B04	E8 09000000	CALL kernel32.VirtualAllocEx
7C809B09	5D	POP EBP
7C809B0A	C2 1000	RETN 10
7C809B0D	90	NOP
7C809B0E	90	NOP
7C809B0F	90	NOP
7C809B10	90	NOP
7C809B11	90	NOP
7C809B12	6A 10	PUSH 10
7C809B14	68 609B807C	PUSH kernel32.7C809B60
7C809B19	E8 B889FFFF	CALL kernel32.7C8024D6

Stack [0012F648]=0012F6C0 (0012F6C0)
EBP=0012F648

Address	Hex dump	ASCII
00756000	8C 74 DD 77 F4 E9 DD 77	it w f w
00756008	C9 7C DD 77 27 6C DD 77	rl w l w
00756010	C4 54 DE 77 00 00 00 00	-T w . . .
00756018	44 FD F1 77 37 D8 F1 77	D² ±w7±w
00756020	EA B5 F1 77 75 76 F2 77	±wuv±w
00756028	00 00 00 00 1A 98 80 7C	. . . ±üÇl
00756030	21 3E 81 7C 1B 77 81 7C	!>ü ±wü
00756038	C0 99 80 7C F1 9A 80 7C	LüÇl ±üÇl
00756040	F2 1E 80 7C 6D 0C 81 7C	±üÇl m. ü
00756048	41 BB 80 7C 40 AE 80 7C	AüÇl @±üÇl
00756050	61 AC 80 7C 8B D3 81 7C	aüÇl iüü

Registers (FPU)

00CB0000

7C809B59 kernel32.7C809B59
7C90E514 ntdll.KiFastSystemCallRet
0053FA20 4fafbfd2.0053FA20
0012F648
0012F648
0012F6E0
00000000

7C809B09 kernel32.7C809B09

ES 0023 32bit 0(FFFFFFFF)
CS 001B 32bit 0(FFFFFFFF)
SS 0023 32bit 0(FFFFFFFF)
DS 0023 32bit 0(FFFFFFFF)
FS 003B 32bit 7FFDF000(FFF)
GS 0000 NULL

LastErr ERROR_PROC_NOT_FOUND (0000007F)
00000246 (NO,NB,E,BE,NS,PE,GE,LE)
bad -NaN FFFF C0000000 00000000

Address	Hex dump	ASCII
0012F648	0012F6C0	RETURN to 0012F9BC
0012F64C	0012F9BC	00000000
0012F650	00000000	000DCA00
0012F654	000DCA00	00003000
0012F658	00003000	00000004
0012F65C	00000004	0000000A
0012F660	0000000A	7C80AE40 kernel32.GetProcAddress
0012F664	7C80AE40	004020B0
0012F668	00000000	004020B0 4fafbfd2.004020B0
0012F66C	004020B0	004D6A10 4fafbfd2.004D6A10
0012F670	004D6A10	00540A40 4fafbfd2.00540A40
0012F674	00540A40	

start CSC 471 Moder... malwarePE malwarePE OllyDbg - 4fafb...

Check EAX → 00CB0000



OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread, module kernel32]

File View Debug Plugins Options Window Help

LEMTWHC / KBR ... S

7C809AF1 8BFF MOV EDI,EDI
 7C809AF3 55 PUSH EBP
 7C809AF4 8BEC MOV EBP,ESP
 7C809AF6 FF75 14 PUSH DWORD PTR SS:[EBP+14]
 7C809AF9 FF75 10 PUSH DWORD PTR SS:[EBP+10]
 7C809AFC FF75 0C PUSH DWORD PTR SS:[EBP+C]
 7C809AFF FF75 08 PUSH DWORD PTR SS:[EBP+8]
 7C809B02 6A FF PUSH -1
 7C809B04 E8 09000000 CALL kernel32.VirtualAllocEx
 7C809B09 5D POP EBP
 7C809B0A C2 1000 RETN 10
 7C809B0D 90 NOP
 7C809B0E 90 NOP
 7C809B0F 90 NOP
 7C809B10 90 NOP
 7C809B11 90 NOP
 7C809B12 6A 10 PUSH 10
 7C809B14 68 609B807C PUSH kernel32.7C809B60
 7C809B19 E8 B889FFFF CALL kernel32.7C8024D6
 7C809B1F 0B 45 00 MOV EBX,EBP
 7C809B20 0B 45 00 MOV EBX,EBP

Stack [0012F648]=0012F6C0 (0012F6C0)
 EBP=0012F648

Registers (FPU)

00CB0000
 7C809B59 kernel32.7C809B59
 7C90E514 ntdll.KiFastSystemCallRet
 0053FA20 4fafbfd2.0053FA20
 0012F648
 0012F6E0
 00000000
 7C809B09 kernel32.7C809B09
 ES 0023 32bit 0(FFFFFFFF)
 CS 001B 32bit 0(FFFFFFFF)
 SS 0023 32bit 0(FFFFFFFF)
 DS 0023 32bit 0(FFFFFFFF)
 FS 003B 32bit 7FFDF000(FFF)
 GS 0000 NULL
 LastErr ERROR_PROC_NOT_FOUND (0000007F)
 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
 had -NAN FFFF C0000000 00000000

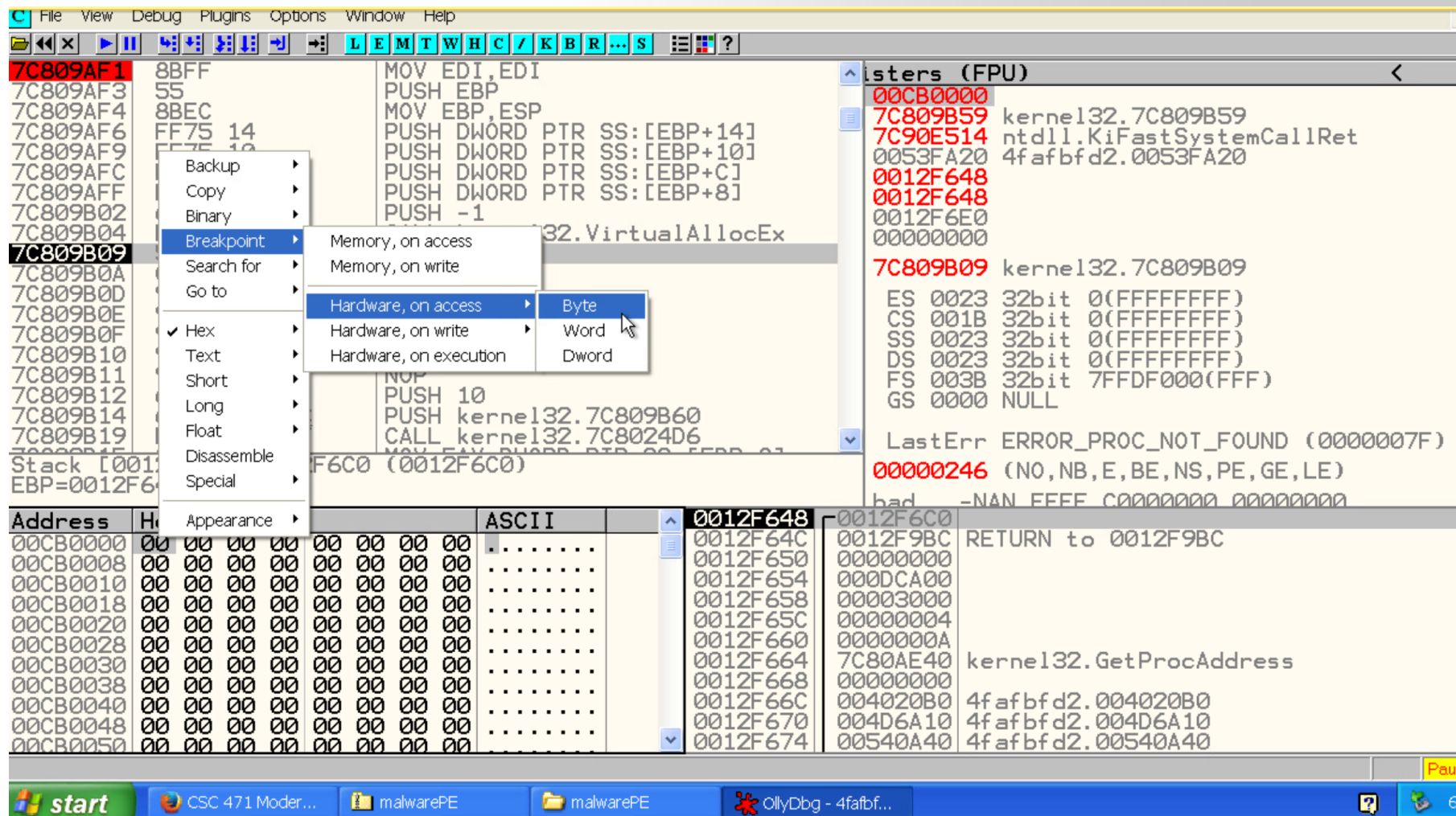
Address	Hex dump	ASCII
00CB0000	00 00 00 00 00 00 00 00
00CB0008	00 00 00 00 00 00 00 00
00CB0010	00 00 00 00 00 00 00 00
00CB0018	00 00 00 00 00 00 00 00
00CB0020	00 00 00 00 00 00 00 00
00CB0028	00 00 00 00 00 00 00 00
00CB0030	00 00 00 00 00 00 00 00
00CB0038	00 00 00 00 00 00 00 00
00CB0040	00 00 00 00 00 00 00 00
00CB0048	00 00 00 00 00 00 00 00
00CB0050	00 00 00 00 00 00 00 00

0012F648 0012F6C0
 0012F64C 0012F9BC RETURN to 0012F9BC
 0012F650 00000000
 0012F654 000DCA00
 0012F658 00003000
 0012F65C 00000004
 0012F660 0000000A
 0012F664 7C80AE40 kernel32.GetProcAddress
 0012F668 00000000
 0012F66C 004020B0 4fafbfd2.004020B0
 0012F670 004D6A10 4fafbfd2.004D6A10
 0012F674 00540A40 4fafbfd2.00540A40

start CSC 471 Moder... malwarePE malwarePE OllyDbg - 4fafbf...

We follow this address in the data window

Set a hardware access breakpoint at 0x00CB0000.



Right click the first byte at 00CB0000 → “Breakpoint”
→ “Hardware on access” → “Word”

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread]

File View Debug Plugins Options Window Help

LEMTWHC/KBR...S

Address	Hex	dump	ASCII
00CB0000	E8 00 00 00 00 00 00 00
00CB0008
00CB0010
00CB0018
00CB0020
00CB0028
00CB0030
00CB0038
00CB0040
00CB0048
00CB0050

Stack SS:[0012F6B8]=00000000
ECX=004020B0 (4fafbfd2.004020B0)

Registers (FPU)

00CB0000	4fafbfd2.004020B0
004020B0	4fafbfd2.0053FA20
000DCAE8	4fafbfd2.0053FA20
0053FA20	4fafbfd2.0053FA20
0012F660	4fafbfd2.0053FA20
0012F6C0	4fafbfd2.0053FA20
0012F6E0	4fafbfd2.0053FA20
00000000	4fafbfd2.0053FA20
0012F9D6	4fafbfd2.0053FA20
ES 0023	32bit 0(FFFFFFFF)
CS 001B	32bit 0(FFFFFFFF)
SS 0023	32bit 0(FFFFFFFF)
DS 0023	32bit 0(FFFFFFFF)
FS 003B	32bit 7FDF000(FFF)
GS 0000	NULL

LastErr ERROR_PROC_NOT_FOUND (0000007F)

00000206 (NO,NB,NE,A,NS,PE,GE,G)

bad -NAN FFFF C0000000 00000000

0012F660 0000000A kernel32.GetProcAddress

0012F664 7C80AE40

0012F668 00000000

0012F66C 004020B0 4fafbfd2.004020B0

0012F670 004D6A10 4fafbfd2.004D6A10

0012F674 00540A40 4fafbfd2.00540A40

0012F678 005AAB40 4fafbfd2.005AAB40

0012F67C 00614BE0 4fafbfd2.00614BE0

0012F680 006E9330 4fafbfd2.006E9330

0012F684 0067F000 4fafbfd2.0067F000

0012F688 0046CA00 4fafbfd2.0046CA00

0012F68C 74696445

Hardware breakpoint 1 at 0012F9D6 - EIP points to next instruction

start CSC 471 Moder... malwarePE malwarePE OllyDbg - 4fafbf...

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

Address	Hex dump	ASCII
00CB0000	E8 C7 00 00 00 00 00 00	CALL EBX
00CB0008	00 00 00 00 00 00 00 00
00CB0010	00 00 00 00 00 00 00 00
00CB0018	00 00 00 00 00 00 00 00
00CB0020	00 00 00 00 00 00 00 00
00CB0028	00 00 00 00 00 00 00 00
00CB0030	00 00 00 00 00 00 00 00
00CB0038	00 00 00 00 00 00 00 00
00CB0040	00 00 00 00 00 00 00 00
00CB0048	00 00 00 00 00 00 00 00
00CB0050	00 00 00 00 00 00 00 00

Stack SS:[0012F6B8]=00000001
ECX=004D6A10 (4fafbfd2.004D6A10)

Registers (FPU)

Register	Value
00CB0000	004D6A10 4fafbfd2.004D6A10
000DCAC7	0053FA20 4fafbfd2.0053FA20
0012F660	0012F6C0
0012F6C0	0012F6E0
00000001	0012F9D6
ES 0023	32bit 0(FFFFFFFF)
CS 001B	32bit 0(FFFFFFFF)
SS 0023	32bit 0(FFFFFFFF)
DS 0023	32bit 0(FFFFFFFF)
FS 003B	32bit 7FFDF000(FFF)
GS 0000	NULL

LastErr ERROR_PROC_NOT_FOUND (0000007F)
00000283 (NO,B,NE,BE,S,PO,L,LE)
had -NAN FFFF C0000000 00000000

Hardware breakpoint 1 at 0012F9D6 - EIP points to next instruction

start CSC 471 Moder... malwarePE malwarePE OllyDbg - 4fafbf... 6

Click Play button (F9) again → Check the data window

Debugger window showing assembly code and registers.

Assembly Code:

```

0012FA11 32CA XOR CL,DL
0012FA13 47 INC EDI
0012FA14 83FF 0F CMP EDI,0F
0012FA17 897D F8 MOV DWORD PTR SS:[EBP-8],EDI
0012FA1A 76 09 JBE SHORT 0012FA25
0012FA1C 81E7 00010000 AND EDI,100
0012FA22 897D F8 MOV DWORD PTR SS:[EBP-8],EDI
0012FA25 880C10 MOV BYTE PTR DS:[EAX+EDX],CL
0012FA28 8B4B 10 MOV ECX,DWORD PTR DS:[EBX+10]
0012FA2B 42 INC EDX
0012FA2C 3BD1 CMP EDX,ECX
0012FA2E 72 D8 JB SHORT 0012FA08
0012FA30 8B36 MOV ESI,DWORD PTR DS:[ESI]
0012FA32 8D4D F8 LEA ECX,DWORD PTR SS:[EBP-8]
0012FA35 51 PUSH ECX
0012FA36 6A 40 PUSH 40
0012FA38 8B46 3C MOV EAX,DWORD PTR DS:[ESI+3C]
0012FA3B 8975 08 MOV DWORD PTR SS:[EBP+8],ESI
0012FA3E 8B4430 50 MOV EAX,DWORD PTR DS:[EAX+ESI+50]

```

Registers (FPU):

```

ES 0023 32bit 0(FFFFFFFF)
CS 001B 32bit 0(FFFFFFFF)
SS 0023 32bit 0(FFFFFFFF)
DS 0023 32bit 0(FFFFFFFF)
FS 003B 32bit 7FFDF000(FFF)
GS 0000 NULL

```

Registers:

```

DL=00
CL=4D ('M')

```

Memory Dump:

Address	Hex dump	ASCII
00CB0000	E8 C7 4C CD A1 3B 7E 80	IL=i:~Ç
00CB0008	28 05 48 99 76 D3 7C 1C	(Hövl
00CB0010	0D 8D CC DD B2 2B 6E 90	.iH+nE
00CB0018	7C 15 58 89 99 3C 6C 0C	isXëö<1.
00CB0020	85 BD FC ED 82 1B 5E A0	à"øé+^â
00CB0028	0C 25 68 B9 A9 0C 5C 3C	.:h Γ.\\<
00CB0030	95 AD EC FD 92 0B 4E B0	òï∞²ÆoN
00CB0038	1C 35 78 A9 B1 1D 4C 2C	LSxΓ+L,
00CB0040	EB C2 26 83 E2 CF 37 0D	ST&âΓ±7.
00CB0048	4D FD 09 95 04 4D 68 34	M².ò+Mh4
00CB0050	9C BF AC FD 80 04 49 A2	£¼C+T6

Registers (FPU) (Continued):

```

LastErr ERROR_PROC_NOT_FOUND (0000007F)
00000206 (NO,NB,NE,A,NS,PE,GE,G)
bad -NAN FEEF C0000000 00000000

```

Registers (Continued):

```

0012F660 0000000A
0012F664 7C80AE40 kernel32.GetProcAddress
0012F668 00000000
0012F66C 0041D9F0 4fafbfd2.0041D9F0
0012F670 004F2350 4fafbfd2.004F2350
0012F674 0055C380 4fafbfd2.0055C380
0012F678 005C6480 4fafbfd2.005C6480
0012F67C 00630520 4fafbfd2.00630520
0012F680 00704C70 4fafbfd2.00704C70
0012F684 0069A940 4fafbfd2.0069A940
0012F688 00488340 4fafbfd2.00488340
0012F68C 74696445

```

Hardware breakpoint 1 at 0012FA11 - EIP points to next instruction

Paused

Click Play button (F9) again → Check the data window

ransomware_example

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

Address	Hex	dump	ASCII
0012FA11	32CA		
0012FA13	47		
0012FA14	83FF 0F		
0012FA17	897D F8		
0012FA1A	76 09		
0012FA1C	81E7 00010000		
0012FA22	897D F8		
0012FA25	880C10		
0012FA28	8B4B 10		
0012FA2B	42		
0012FA2C	3BD1		
0012FA2E	72 D8		
0012FA30	8B36		
0012FA32	8D4D F8		
0012FA35	51		
0012FA36	6A 40		
0012FA38	8B46 3C		
0012FA3B	8975 08		
0012FA3E	8B4430 50		

DS:[0053FA30]=000DCA00
ECX=000DCA4D

XOR CL,DL
INC EDI
CMP EDI,0F
MOV DWORD PTR SS:[EBP-8],EDI
JBE SHORT 0012FA25
AND EDI,100
MOV DWORD PTR SS:[EBP-8],EDI
MOV BYTE PTR DS:[EAX+EDX],CL
MOV ECX,DWORD PTR DS:[EBX+10]
INC EDX
CMP EDX,ECX
JB SHORT 0012FA08
MOV ESI,DWORD PTR DS:[ESI]
LEA ECX,DWORD PTR SS:[EBP-8]
PUSH ECX
PUSH 40
MOV EAX,DWORD PTR DS:[ESI+3C]
MOV DWORD PTR SS:[EBP+8],ESI
MOV EAX,DWORD PTR DS:[EAX+ESI+50]
PUSH EAX

Registers (FPU)

Register	Type	Value
00CB0000		
000DCA4D		
00000000		
0053FA20	4f afbfd2.0053FA20	
0012F660		
0012F6C0		
0012F6E0		
00000001		
0012FA28		
ES 0023	32bit	0(FFFFFFFF)
CS 001B	32bit	0(FFFFFFFF)
SS 0023	32bit	0(FFFFFFFF)
DS 0023	32bit	0(FFFFFFFF)
FS 003B	32bit	7FDF000(FFF)
GS 0000		NULL

LastErr ERROR_PROC_NOT_FOUND (0000007F)
0000293 (N0,B,NE,BE,S,P0,L,LE)
bad -NAN FFFF C0000000 00000000

Address	Hex	dump	ASCII
00CB0000	4D		
00CB0008	28 05 48 99 76 D3 7C 1C		
00CB0010	0D 8D CC DD B2 2B 6E 90		
00CB0018	7C 15 58 89 99 3C 6C 0C		
00CB0020	85 BD FC ED 82 1B 5E A0		
00CB0028	0C 25 68 B9 A9 0C 5C 3C		
00CB0030	95 AD EC FD 92 0B 4E B0		
00CB0038	1C 35 78 A9 B1 1D 4C 2C		
00CB0040	EB C2 26 83 E2 CF 37 0D		
00CB0048	4D FD 09 95 04 4D 68 34		
00CB0050	9C BF AC FD 80 04 49 A2		

Hardware breakpoint 1 at 0012FA28 - EIP points to next instruction

0012F660 0000000A
0012F664 7C80AE40 kernel32.GetProcAddress
0012F668 00000000
0012F66C 0041D9F0 4fafbfd2.0041D9F0
0012F670 004F2350 4fafbfd2.004F2350
0012F674 0055C380 4fafbfd2.0055C380
0012F678 005C6480 4fafbfd2.005C6480
0012F67C 00630520 4fafbfd2.00630520
0012F680 00704C70 4fafbfd2.00704C70
0012F684 0069A940 4fafbfd2.0069A940
0012F688 00488340 4fafbfd2.00488340
0012F68C 74696445

start CSC 471 Moder... malwarePE malwarePE OllyDbg - 4fafbf...

Click Play button (F9) again → Check 00CB0000

OllyDbg - 4fafbfd2e560778f11beb8f736e80bb1 - [CPU - main thread]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

0012FA0B	8A0C1F	MOV CL, BYTE PTR DS:[EDI+EBX]
0012FA0E	320C10	XOR CL, BYTE PTR DS:[EAX+EDX]
0012FA11	32CA	XOR CL, DL
0012FA13	47	INC EDI
0012FA14	83FF 0F	CMP EDI, 0F
0012FA17	897D F8	MOV DWORD PTR SS:[EBP-8], EDI
0012FA1A	76 09	JBE SHORT 0012FA25
0012FA1C	81E7 00010000	AND EDI, 100
0012FA22	897D F8	MOV DWORD PTR SS:[EBP-8], EDI
0012FA25	880C10	MOV BYTE PTR DS:[EAX+EDX], CL
0012FA28	8B4B 10	MOV ECX, DWORD PTR DS:[EBX+10]
0012FA2B	42	INC EDX
0012FA2C	3BD1	CMP EDX, ECX
0012FA2E	72 D8	JB SHORT 0012FA08
0012FA30	8B36	MOV ESI, DWORD PTR DS:[ESI]
0012FA32	8D4D F8	LEA ECX, DWORD PTR SS:[EBP-8]
0012FA35	51	PUSH ECX
0012FA36	6A 40	PUSH 40
0012FA38	8B46 3C	MOV EAX, DWORD PTR DS:[ESI+3C]

DS:[0053FA30]=000DCA00
ECX=000DCA5A

Registers (FPU)

00CB0000
000DCA5A
00000001
0053FA20 4fafbfd2.0053FA20
0012F660
0012F6C0
0012F6E0
00000002
0012FA28
ES 0023 32bit 0(FFFFFFFF)
CS 001B 32bit 0(FFFFFFFF)
SS 0023 32bit 0(FFFFFFFF)
DS 0023 32bit 0(FFFFFFFF)
FS 003B 32bit 7FFDF000(FFF)
GS 0000 NULL
LastErr ERROR_PROC_NOT_FOUND (0000007F)
00000297 (NO, B, NE, BE, S, PE, L, LE)
bad -NAN FFFF C0000000 00000000

Address	Hex dump	ASCII
00CB0000	4D 5A 4C CD A1 3B 7E 80	MZL=i;~
00CB0008	28 03 48 99 76 D3 7C 1C	(HöVl
00CB0010	0D 8D CC DD B2 2B 6E 90	.iH nE
00CB0018	7C 15 58 89 99 3C 6C 0C	isXëö<1.
00CB0020	85 BD FC ED 82 1B 5E A0	à"é+^a
00CB0028	0C 25 68 B9 A9 0C 5C 3C	.%h r.\<
00CB0030	95 AD EC FD 92 0B 4E B0	ò i ∞ A o N
00CB0038	1C 35 78 A9 B1 1D 4C 2C	LSx r +L
00CB0040	EB C2 26 83 E2 CF 37 0D	S T & a Γ ± 7.
00CB0048	4D FD 09 95 04 4D 68 34	M z . ò M h 4
00CB0050	9C BE AC FD 80 04 A9 A2	Ed % & C I 6

0012F660 0000000A kernel32.GetProcAddress
0012F664 7C80AE40
0012F668 00000000
0012F66C 0041D9F0 4fafbfd2.0041D9F0
0012F670 004F2350 4fafbfd2.004F2350
0012F674 0055C380 4fafbfd2.0055C380
0012F678 005C6480 4fafbfd2.005C6480
0012F67C 00630520 4fafbfd2.00630520
0012F680 00704C70 4fafbfd2.00704C70
0012F684 0069A940 4fafbfd2.0069A940
0012F688 00488340 4fafbfd2.00488340
0012F68C 74696445

Hardware breakpoint 1 at 0012FA28 - EIP points to next instruction

start CSC 471 Moder... malwarePE malwarePE OllyDbg - 4fafbf...

Click Play button (F9) again → Check the data window

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

0012FA0E	320C10	XOR CL, BYTE PTR DS:[EAX+EDX]
0012FA11	32CA	XOR CL, DL
0012FA13	47	INC EDI
0012FA14	83FF 0F	CMP EDI, 0F
0012FA17	897D F8	MOV DWORD PTR SS:[EBP-8], EDI
0012FA1A	76 09	JBE SHORT 0012FA25
0012FA1C	81E7 00010000	AND EDI, 100
0012FA22	897D F8	MOV DWORD PTR SS:[EBP-8], EDI
0012FA25	880C10	MOV BYTE PTR DS:[EAX+EDX], CL
0012FA28	8B4B 10	MOV ECX, DWORD PTR DS:[EBX+10]
0012FA2B	42	INC EDX
0012FA2C	3BD1	CMP EDX, ECX
0012FA2E	72 D8	JB SHORT 0012FA08
0012FA30	8B36	MOV ESI, DWORD PTR DS:[ESI]
0012FA32	8D4D F8	LEA ECX, [EBP-8]
0012FA35	51	PUSH EAX
0012FA36	6A 40	PUSH 40
0012FA38	8B46 3C	MOV EAX, DWORD PTR DS:[EBX+3C]
0012FA3B	8975 08	MOV DWORD PTR DS:[EAX], EDI

Address Hex dump

00CB0000	4D 5A 90 00 03 00 00 00
00CB0008	04 00 00 00 FF FF 00 00
00CB0010	B8 00 00 00 00 00 00 00
00CB0018	40 00 00 00 00 00 00 00
00CB0020	00 00 00 00 00 00 00 00
00CB0028	00 00 00 00 00 00 00 00
00CB0030	00 00 00 00 00 00 00 00
00CB0038	00 00 00 00 08 01 00 00
00CB0040	0E 1F BA 0E 00 B4 09 CD
00CB0048	21 B8 01 4C CD 21 54 68
00CB0050	69 73 20 70 72 6F 67 72

Hardware breakpoint 1 at 0012FA75 - EIP points to next instruction

Registers (FPU)

00CB0108	ASCII "PE"
00400000	4fafbfd2.00400000
7C90E54D	ntdll.7C90E54D
0053FA20	4fafbfd2.0053FA20
0012F660	
0012F6C0	
00000000	
008B0000	
0012FA75	
ES 0023	32bit 0(FFFFFFFF)
CS 001B	32bit 0(FFFFFFFF)
SS 0023	32bit 0(FFFFFFFF)
DS 0023	32bit 0(FFFFFFFF)
FS 003B	32bit 7FFDF000(FFF)
GS 0000	NULL

LastErr ERROR_PROC_NOT_FOUND (0000007F)

00000206 (NO, NB, NE, A, NS, PE, GE, G)

FF C0000000 00000000

32. GetProcAddress

002.0041D9F0	
002.004F2350	
002.0055C380	
002.005C6480	
002.00630520	
002.00704C70	
002.0069A940	
00488340	4fafbfd2.00488340
74696445	

start CSC 471 Moder... malwarePE malwarePE OllyDbg - 4fafbf... 6

Using OllyDbg to Remove Obfuscation

1.Set Breakpoint on VirtualAlloc:

1. Run the program and pause at VirtualAlloc.
2. Monitor the return value (memory address) and allocation size.

2.Identify Large Allocations:

1. Focus on large memory allocations (e.g., > 900,000 bytes) typical for PE files.

3.Set Hardware Breakpoint:

1. Set a hardware breakpoint at the allocated memory address.
2. Detect when data is written to this memory.

Q & A

