

Progetto settimana 10 lezione 5 Daniele D'Esposito

Traccia:

Con riferimento al file `Malware_U3_W2_L5` presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5» sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

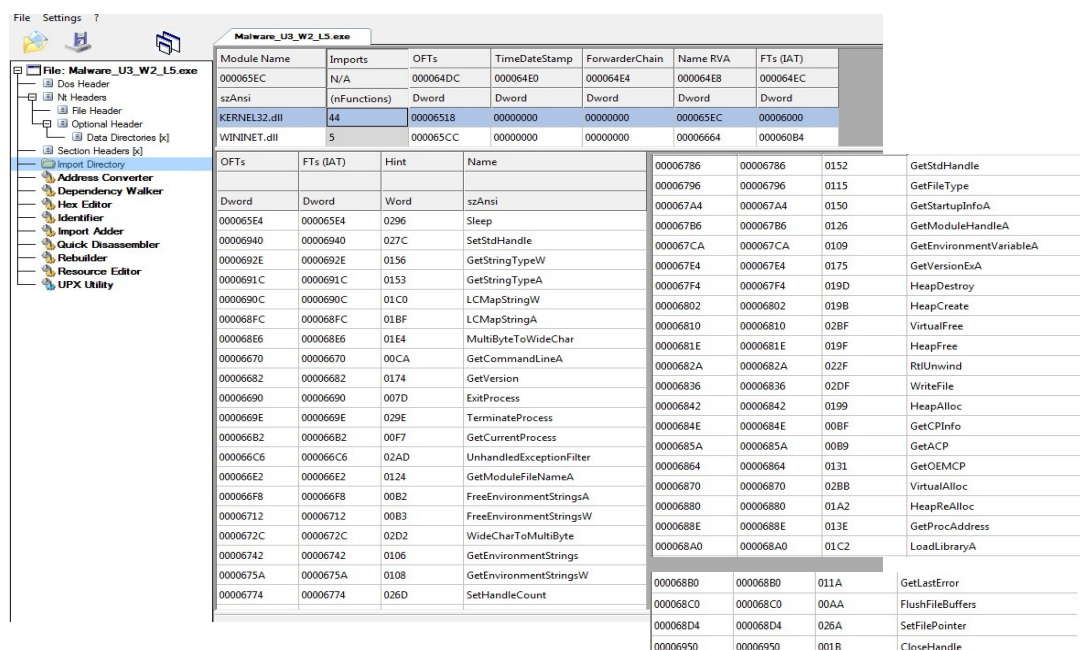
1. Quali librerie vengono importate dal file eseguibile?
2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

Con riferimento alla figura in Figura 1, rispondere ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
4. Ipotizzare il comportamento della funzionalità implementata
5. BONUS fare tabella con significato delle singole righe di codice assembly

1- LIBRERIE

Avviando il programma **CFF Explorer** e' possibile analizzare l' header di un file eseguibile; nell' header e' possibile sapere tante informazioni, come le librerie o funzioni che importa o esporta il programma e le sezioni di cui si compone. Nell'immagine si nota che sono state importate solo 2 librerie ma sono state utilizzate 49 funzioni appartenenti ad esse.



Module Name	Imports	OFFs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	000064EC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFFs	FTs (IAT)	Hint	Name	00006786	00006786	0152	GetStdHandle
				00006796	00006796	0115	GetFileType
Dword	Dword	Word	szAnsi	000067A4	000067A4	0150	GetStartupInfoA
000065E4	000065E4	0296	Sleep	000067B6	000067B6	0126	GetModuleHandleA
00006940	00006940	027C	SetStdHandle	000067CA	000067CA	0109	GetEnvironmentVariableA
0000692E	0000692E	0156	GetStringTypeW	000067E4	000067E4	0175	GetVersionExA
0000691C	0000691C	0153	GetStringTypeA	000067F4	000067F4	019D	HeapDestroy
0000690C	0000690C	01C0	LCMapStringW	00006802	00006802	019B	HeapCreate
000068FC	000068FC	01BF	LCMapStringA	00006810	00006810	02BF	VirtualFree
000068E6	000068E6	01E4	MultiByteToWideChar	0000681E	0000681E	019F	HeapFree
00006670	00006670	00CA	GetCommandLineA	0000682A	0000682A	022F	RtlUnwind
00006682	00006682	0174	GetVersion	00006836	00006836	02DF	WriteFile
00006690	00006690	007D	ExitProcess	00006842	00006842	0199	HeapAlloc
0000669E	0000669E	029E	TerminateProcess	0000684E	0000684E	00BF	GetCPInfo
000066B2	000066B2	00F7	GetCurrentProcess	0000685A	0000685A	00B9	GetACP
000066C6	000066C6	02AD	UnhandledExceptionFilter	00006864	00006864	0131	GetOEMCP
000066E2	000066E2	0124	GetModuleFileNameA	00006870	00006870	02BB	VirtualAlloc
000066F8	000066F8	00B2	FreeEnvironmentStringsA	00006880	00006880	01A2	HeapReAlloc
00006712	00006712	00B3	FreeEnvironmentStringsW	0000688E	0000688E	013E	GetProcAddress
0000672C	0000672C	02D2	WideCharToMultiByte	000068A0	000068A0	01C2	LoadLibraryA
00006742	00006742	0106	GetEnvironmentStrings				
0000675A	0000675A	0108	GetEnvironmentStringsW	000068B0	000068B0	011A	GetLastError
00006774	00006774	026D	SetHandleCount	000068C0	000068C0	00AA	FlushFileBuffers
				000068D4	000068D4	026A	SetFilePointer
				00006950	00006950	001B	CloseHandle

KERNEL32.DLL

Questa libreria contiene le funzioni principali per interagire con il sistema operativo come la manipolazione dei file e la gestione della memoria; si puo' intuire dai nomi delle varie funzioni che sono processi basilari e indispensabili al funzionamento della macchina; nella stragrande maggioranza dei casi, questa libreria viene caricata dai malware che ne utilizzano almeno una funzione. Si puo' notare che tra le funzioni richiamate sono presenti anche **GetProcAddress** e **LoadLibraryA** che sono molto usate per richiamare altre funzioni da altre librerie durante l'esecuzione stessa del programma(in runtime) senza doverle caricare in anticipo e nascondendo quindi all'analisi statica quali altre funzioni vengono utilizzate.

Module Name	Imports	OFts	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
00006664	N/A	000064F0	000064F4	000064F8	000064FC	00006500
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFts	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

WININET.DLL

Questa libreria contiene le funzioni per l'implementazione di alcuni protocolli di rete come **HTTP** (HyperText Transfer Protocol) e **FTP**(File Transfer Protocol).

InternetOpenUrlA apre una risorsa specificata da un URL, FTP o HTTP;

InternetOpenA inizializza un'applicazione all'uso della libreria wininet;

InternetGetConnectedState recupera lo stato di connessione del sistema locale;

sapendo cosa svolgono queste funzioni e' facile intuire che il malware carica o scarica qualcosa da o su un determinato link.

2-SEZIONI

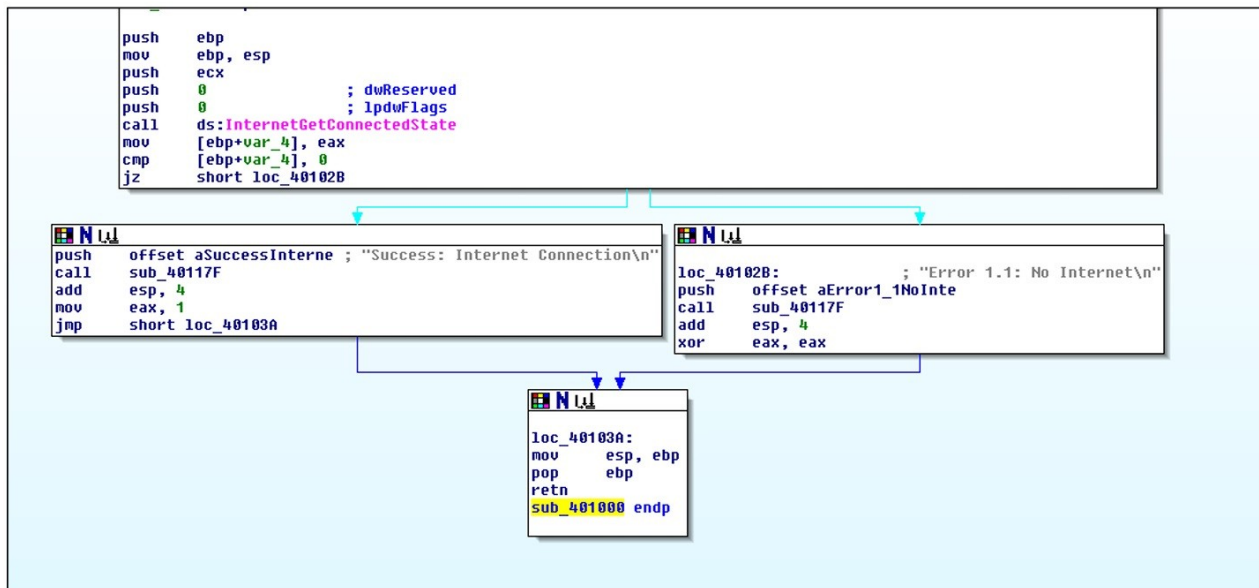
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000150	5F	5F	47	4C	4F	42	41	4C	5F	48	45	41	50	5F	53	45	GLOBAL_HEAP_SE
00000160	4C	45	43	54	45	44	00	00	5F	5F	4D	53	56	43	52	54	TEXTED_HEAPCT
00000170	5F	48	45	41	50	5F	53	45	4C	45	43	54	00	00	00	00	HEAP_SELECT
00000180	72	75	6E	74	69	6D	65	20	65	72	72	6F	72	20	00	00	runtime error ...
00000190	0D	0A	00	00	54	4C	4F	53	53	20	65	72	72	6F	72	0D	RTOS error:
000001A0	0A	00	00	00	53	49	4E	47	20	65	72	72	6F	72	0D	0A	SING_error
000001B0	0D	00	00	00	44	4F	4D	41	49	4E	20	65	72	72	6F	72	DOMAIN error
000001C0	0D	0A	00	00	52	36	30	32	38	0D	0A	2D	20	75	6E	61	R6028 - una
000001D0	62	6C	65	20	74	6F	20	69	6E	69	74	69	61	6C	69	7A	ble to initializ
000001E0	65	20	68	65	61	70	0D	0A	00	00	00	52	36	30	32		e.heap... R602
000001F0	37	0D	0A	2D	20	6E	6F	74	20	65	6E	6F	75	67	68	20	7 - not enough
00000200	73	70	61	63	65	20	66	6F	72	20	6C	6F	77	69	6F	20	space for lowio
00000210	69	6E	69	74	69	61	6C	69	7A	61	74	69	68	6E	0D	0A	initialization
00000220	00	00	00	00	52	36	30	32	38	0D	0A	2D	20	6E	6F	74	R6026 - not
00000230	20	65	6E	6F	75	67	68	20	73	70	61	63	65	20	66	6F	enough space fo
																	e. stdio initiali

Le sezioni sono le parti che costituiscono un programma, nell'immagine sopra si puo' vedere che questo malware e' costituito da 3 sezioni:

1. **.text** e' la sezione che contiene le righe di codice che vengono eseguite dalla CPU quando il programma viene avviato.
2. **.rdata** e' la sezione che contiene le librerie e le funzioni importate ed esportate dal programma
3. **.data** e' la sezione che contiene i dati e le variabili globali che devono essere disponibili da qualsiasi parte del programma

Figura 1



3-COSTRUTTI NOTI

```

push    ebp
mov     ebp, esp

```

CREAZIONE STACK

```

push    ecx
push    0          ; dwReserved
push    0          ; lpdwFlags
call    ds: InternetGetConnentedState

```

CHIAMATA FUNZIONE

```

mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B

```

COSTRUTTO IF

```

push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A

```

COSTRUTTO ELSE

```

loc_40102B:
push    offset aError1_1NoInte ; "Error 1.1:No Internet\n"
call    sub_40117F
add     esp, 4
xor     eax, eax

```

COSTRUTTO IF

```

loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp

```

RIMOZIONE STACK

4-COMPORTAMENTO

E' facile ipotizzare che questa parte di codice del malware si occupa di controllare se e' disponibile una connessione internet e di salvare il valore 1 al registro eax in caso positivo(mov eax, 1) e valore 0 in caso negativo(xor eax, eax); rimangono dubbi sul richiamo alla funzione sub_040117F in quanto non e' presente in figura ma si puo' supporre che serva a stampare su schermo il risultato della verifica di connessione in quanto e' presente in entrambe le condizioni.

5-BONUS

push ebp inserisce il registro ebp(extended base pointer) nello stack

mov ebp, esp aggiunge allo stack il registro esp(extended stack pointer) sopra lo ebp

push ecx aggiunge allo stack il registro ad uso generico ecx

push 0 ; dwReserved aggiunge allo stack il valore 0 preso dalla chiave di registro dwreserved

push 0 ; lpdwFlags aggiunge allo stack il valore 0 preso dalla chiave di registro lpdwflags

call ds: InternetGetConnentedState richiama la funzione InternetGetConnentedState che analizza lo stato di connessione della macchina e inserisce un valore binario nel registro eax

mov [ebp+var_4], eax inserisce il valore di eax dentro il registro ebp+var_4

cmp [ebp+var_4], 0 fa la sottrazione tra il valore di ebp+var4 e 0 e se la destinazione e' maggiore della sorgente allora setta la zeroflag a 0 e se invece e' uguale alla sorgente lo setta a 1

jz short loc_40102B salta alla locazione 40102B soltanto se il valore di zeroflag e' 1

push offset aSuccessInterne ; "Success: Internet Connection/n" inserisce la stringa nello stack

call sub_40117F chiama la funzione allocata in 40117F

add esp, 4 aggiunge 4 al registro esp per rimuovere il push offset aSuccessInterne

mov eax, 1 inserisce il valore 1 nel registro eax

jmp short loc_40103A salta alla locazione 40103A

loc_40102B: la locazione dove si finisce in caso il salto jz viene effettuato

push offset aError1_1NoInte; "Error 1.1:No Internet/n" inserisce la stringa nello stack

call sub_40117F chiama la funzione allocata in 40117F

add esp, 4 aggiunge 4 al registro esp per rimuovere il push offset aError1_1NoInte

xor eax, eax usa l'operatore logico xor per inizializzare a 0 il registro eax

loc_40103A:

mov esp, ebp rimuove lo stack

pop ebp rimuove lo stack

retn

sub_401000 endp