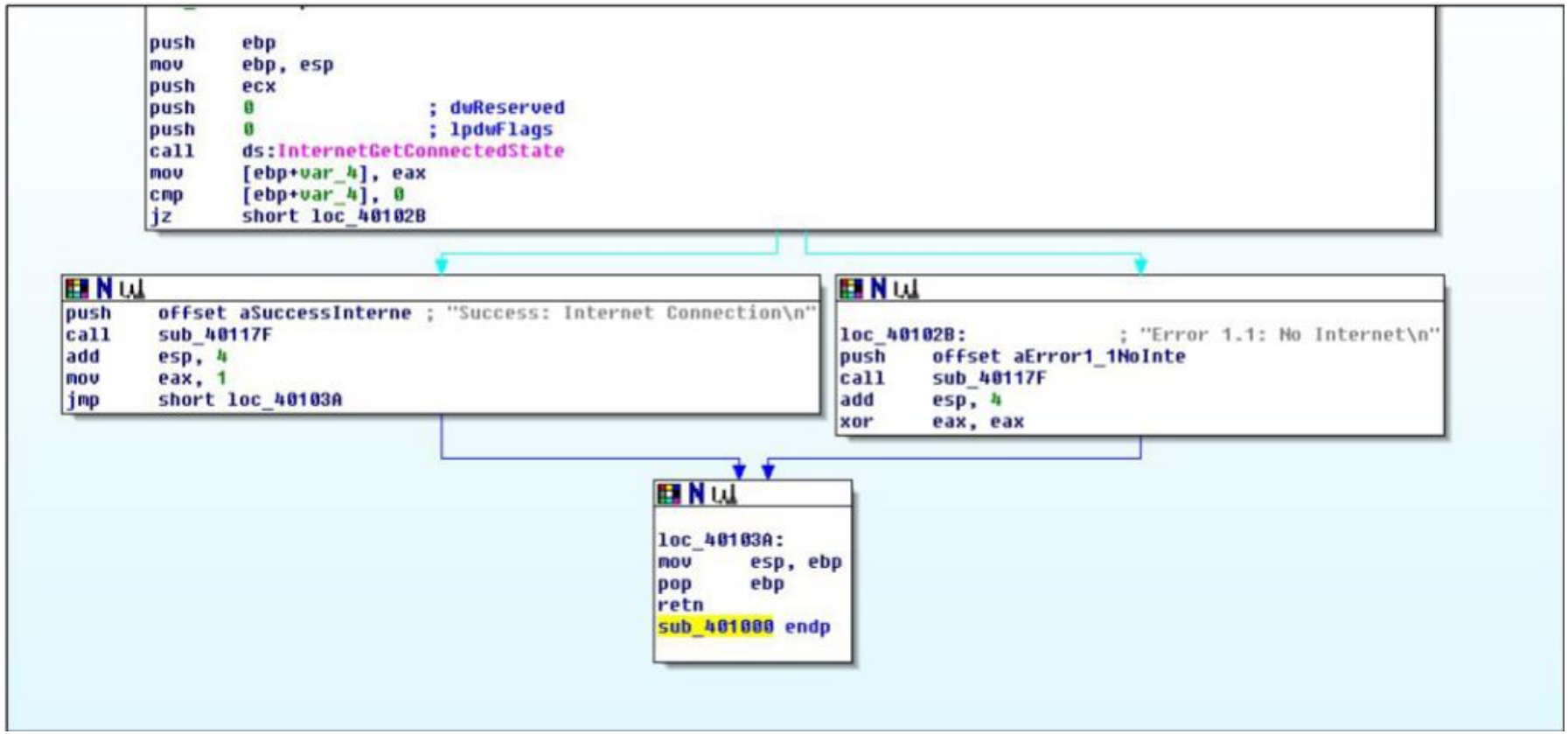
Esercizio 31-03

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:

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

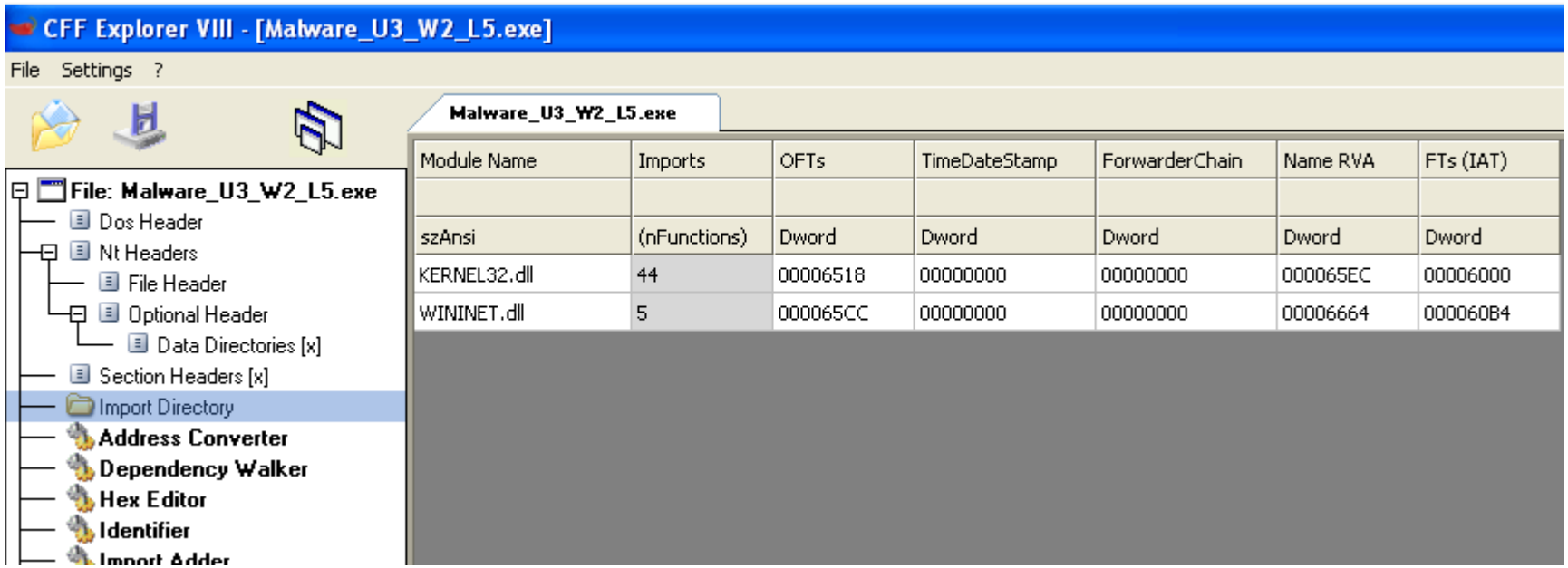
Con riferimento alla figura sotto, rispondere ai seguenti quesiti:

- Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)
- Ipotizzare il comportamento della funzionalità implementata



Analisi statica dell'eseguibile "Malware_U3_W2_L5"

Utilizzando CFF Explorer si identificano due librerie importate dal malware: KERNEL32.dll e WININET.dll



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

Module Name	Imports	OFTs	TimeDateStamp	ForwarderC
000065EC	N/A	000064DC	000064E0	000064E4
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000
WININET.dll	5	000065CC	00000000	00000000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW
000068FC	000068FC	01BF	LCMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar
00006670	00006670	00CA	GetCommandLineA
00006682	00006682	0174	GetVersion
00006690	00006690	007D	ExitProcess
0000669E	0000669E	029E	TerminateProcess
000066B2	000066B2	00F7	GetCurrentProcess
000066C6	000066C6	02AD	UnhandledExceptionFilter
000066E2	000066E2	0124	GetModuleFileNameA
000066F8	000066F8	00B2	FreeEnvironmentStringsA

KERNEL32.dll è una librerie che contiene funzioni che permettono di interagire con il sistema operativo: modificare file, gestire la memoria...

<= queste sono le funzioni della libreria richiamate dal malware

WININET.dll	5	000065CC	00000000	00000000

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 è una libreria che contiene le funzioni necessarie all'utilizzo di alcuni protocolli di rete come HTTP, FTP e NTP.

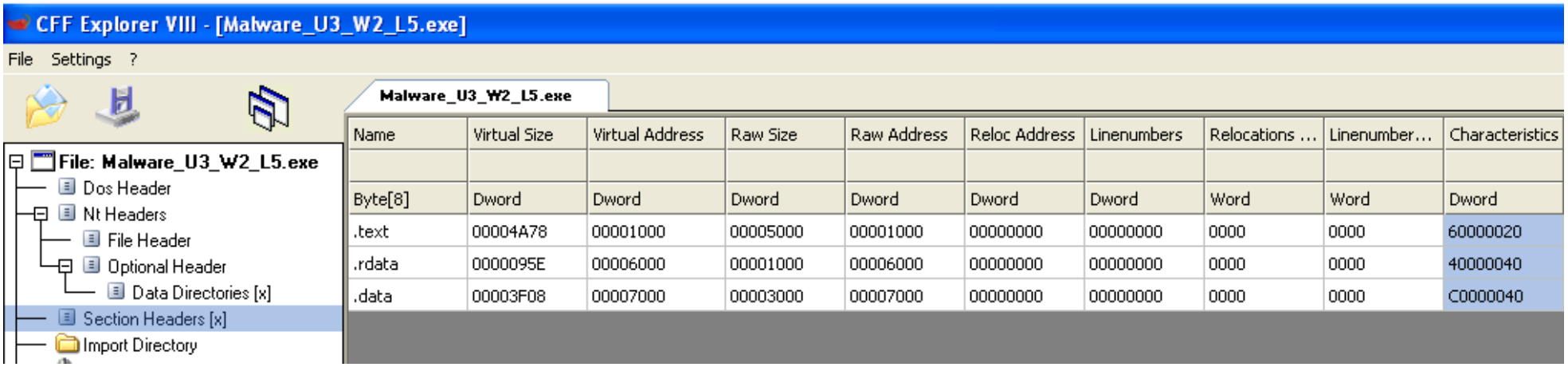
<= queste sono le funzioni della libreria chiamate dal malware

il malware è composto da tre sezioni:

.text: si tratta della sezione che contiene le righe di codice che il malware deve far eseguire alla CPU “vittima”

.rdata: contiene informazioni riguardanti le librerie e le funzioni che userà il malware

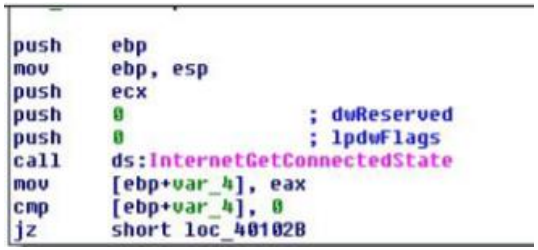
.data: contiene le variabili globali necessarie al funzionamento del codice del malware



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.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

Analisi del codice Assembly

Dopo aver creato lo stack ebp e avergli assegnato il valore contenuto in esp, il programma utilizza “push” per fornire 3 parametri (ecx,0,0) alla funzione InternetGetConnectedState.



Questa funzione verifica la connettività alla rete Internet restituendo un valore:

1 = connessione riuscita

0 = connessione fallita

Questo valore viene automaticamente allocato nel registro eax.

Il programma copia il valore di eax in var_4 di ebp e viene comparato con il valore “0”.

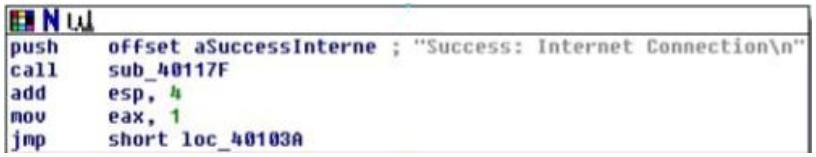
l’istruzione cmp paragonerà i due valori designati con un ragionamento simile alla sottrazione di valori: 0 - 0 = 0/0 - 1 = -1; questo porterà nel primo caso all’attivazione dello Zero Flag (ZF = 1) mentre nel secondo caso no (ZF = 0).

l’istruzione jz permette un salto alla locazione loc_40102B, ma solo in maniera condizionale: significa che se il valore richiesto all’attivazione di jz non è fornito, l’istruzione non si attiverà. Il valore necessario all’attivazione di jz è ZF = 1.

Costrutto if

Questa differenza di comportamento del programma, che esegue o “skippa” un’istruzione in base ad una certa condizione, ci fa capire di star guardando un costrutto **if** che, in condizione “prevista”, il codice segue il suo normale flusso, altrimenti (**else**) passa ad un’altra porzione di codice.

il codice prosegue con:



qualora il jz non si sia attivato e sia mancato il jump alla loc_40102B, il programma prosegue chiamando la funzione sub_40117F che stamperà a schermo un messaggio di successo con scritto “Success: Internet Connection”, e dopo aver aggiunto 4 unità al valore di esp e copiata in eax il valore “1” effettua un jump alla loc_41103A.

Se invece il jz è andato a buon fine:

Viene effettuato il salto condizionale alla loc_40102B, dove verrà chiamata la funzione sub_40117F che stamperà a schermo un messaggio di fallimento con scritto “Error 1.1: No Internet”, vengono aggiunte 4 unità al valore di esp e reso “0” il valore di eax (xor restituisce falso ovvero “0” quando ha due input uguali, in questo eax = eax quindi xor = 0).

il codice termina con:

La locazione loc_40103A è la locazione che sarebbe stata raggiunta a seguito della riuscita connessione a Internet (pagina precedente) e che serviva a “saltare” la porzione di codice riguardante la mancata connessione che, seguendo il suo normale flusso di esecuzione, arriva a questa locazione senza necessità di jmp.

Il valore di ebp viene copiato in esp e il registro ebp viene eliminato con l’istruzione pop.

L’istruzione retn dichiara la fine della funzione.

Sub_401000 endp indica la fine della funzione chiamata.

Conclusione

Riassumendo, il programma si serve di un costrutto if che permette di conoscere se la macchina bersaglio si connette o meno a Internet.