TEST 13/01/2023

FACCILONGO DOMENICO

ANALISI STATICA E DINAMICA: UN APPROCCIO PRATICO

TASK

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:

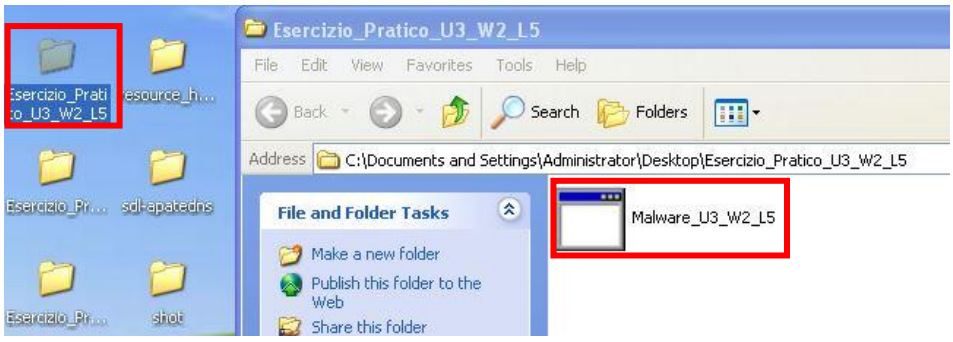
1. Quali **librerie** vengono importate dal file eseguibile?
2. Quali sono le **sezioni** di cui si compone il file eseguibile del malware?

Con riferimento alla figura in slide 3, risponde 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

ANALISI E VALUTAZIONI DEL MALWARE

Come da traccia andiamo ad esaminare come primo passo il **Malware_U3_W2_L5** che si troverà su Desktop della virtual box *Malware Analysis_Final* nella cartella *Eserizio_Pratico_U3_W2_L5*. Come da immagine seguente:

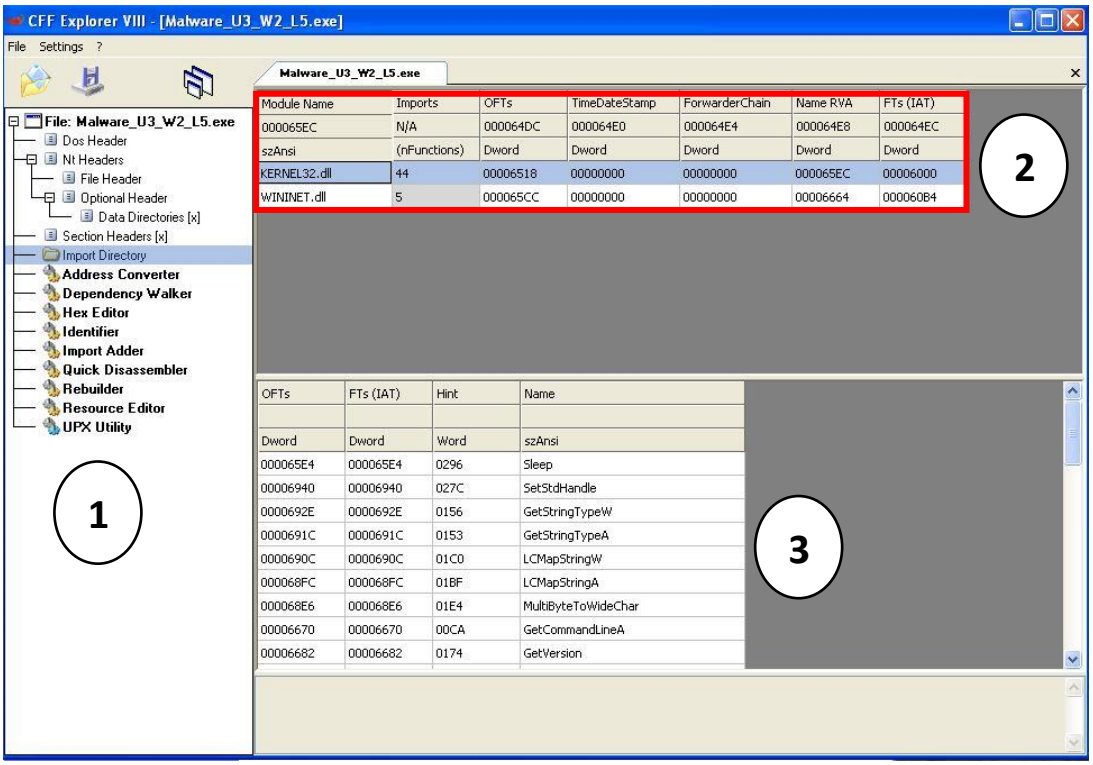


1. LIBRERIE IMPORTATE DAL FILE ESEGUIBILE

Per esaminare le librerie importate andiamo ad utilizzare il tool **CFF Explorer VIII.**

All'apertura del tool ritroviamo la schermata in figura, dal quale per caricare il file del malware selezioniamo l'icona della cartella.

Dopo averlo aperto andiamo nelle sezioni di sinistra cliccando su ***Import Directory***.



Module Name	Imports	OFTs	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

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

Come possiamo notare nella sezione **1** ci sono i vari settori che possiamo analizzare del file eseguibile. Nella sezione **2** ritroviamo ciò che interessa a noi ora, ovvero le librerie importate dal file. Infine nella sezione **3** troviamo le funzioni importate per ciascuna libreria. Le librerie trovate sono:

- **KERNELL32.dll**: libreria che contiene le funzioni per interagire con il sistema operativo. Le funzioni al suo interno (44) sono rispettivamente:

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
00006712	00006712	00B3	FreeEnvironmentStringsW

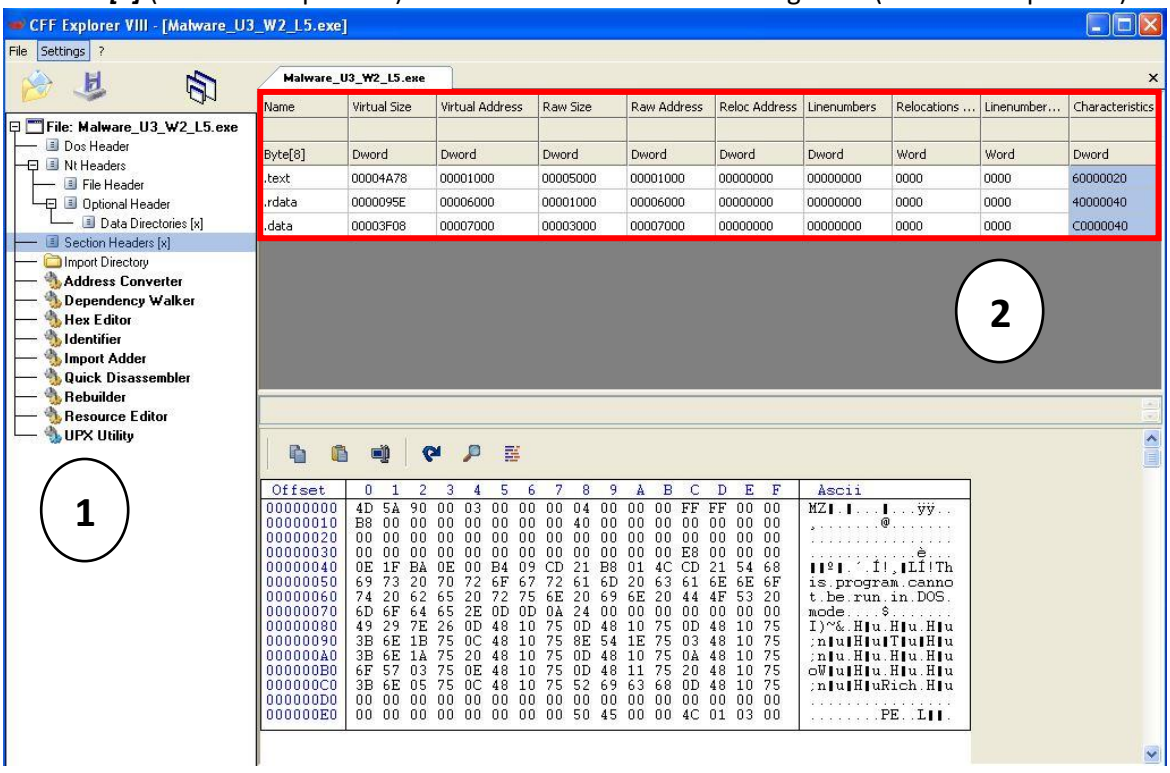
0000672C	0000672C	02D2	WideCharToMultiByte
00006742	00006742	0106	GetEnvironmentStrings
0000675A	0000675A	0108	GetEnvironmentStringsW
00006774	00006774	026D	SetHandleCount
00006786	00006786	0152	GetStdHandle
00006796	00006796	0115	GetFileType
000067A4	000067A4	0150	GetStartupInfoA
000067B6	000067B6	0126	GetModuleHandleA
000067CA	000067CA	0109	GetEnvironmentVariableA
000067E4	000067E4	0175	GetVersionExA
000067F4	000067F4	019D	HeapDestroy
00006802	00006802	0198	HeapCreate
00006810	00006810	02BF	VirtualFree
0000681E	0000681E	019F	HeapFree
0000682A	0000682A	022F	RtlUnwind
00006836	00006836	02DF	WriteFile
00006842	00006842	0199	HeapAlloc
0000684E	0000684E	00BF	GetCPInfo
0000685A	0000685A	00B9	GetACP
00006864	00006864	0131	GetOEMCP
00006870	00006870	02B8	VirtualAlloc
00006880	00006880	01A2	HeapReAlloc
0000688E	0000688E	013E	GetProcAddress
000068A0	000068A0	01C2	LoadLibraryA
000068B0	000068B0	011A	GetLastError
000068C0	000068C0	00AA	FlushFileBuffers
000068D4	000068D4	026A	SetFilePointer
00006950	00006950	0018	CloseHandle

- **WININET.dll**: libreria che contiene funzioni per implementazione di alcuni protocolli di rete HTPP, FTP, NTP. Le sue funzioni (5) sono:

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

2. SEZIONI DI CUI SI COMPONE IL FILE ESEGUIBILE

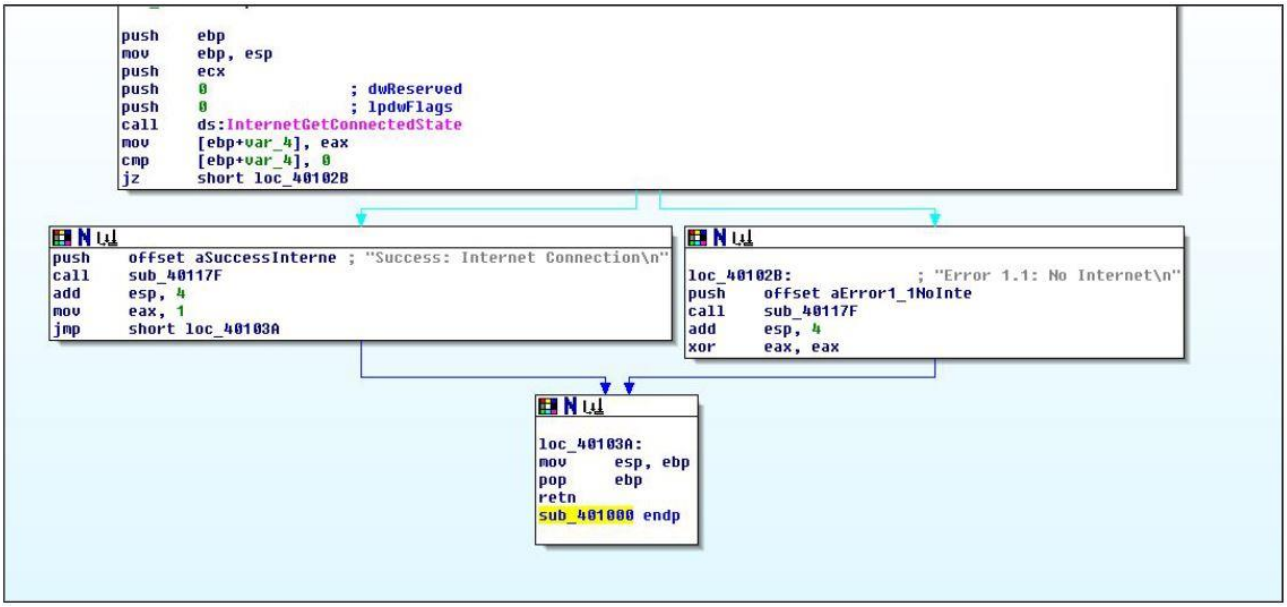
Per quanto riguarda le sezioni usufruiamo sempre di CFF Explorer. Andando nella sezione ***Section Headers [x]*** (indicata dal punto **1**) otteniamo le sezioni del file eseguibile (indicate dal punto **2**):



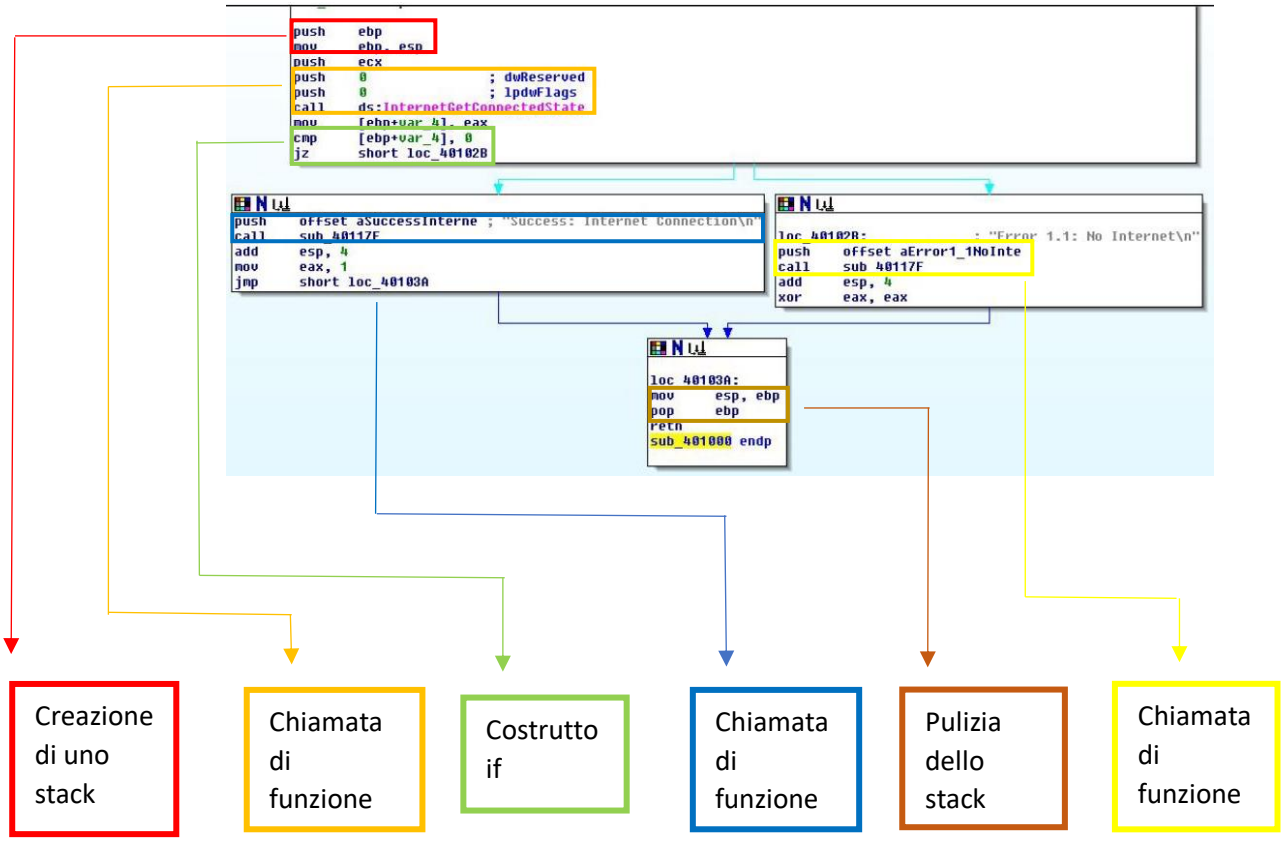
- **.text**: contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato.
- **.rdata**: contiene le informazioni come librerie e funzioni importate ed esportate dal malware.
- **.data**: contiene i dati/variabili globali del programma eseguibile.

ANALISI CODICE ASSEMBLY

In riferimento al codice in assembly nella figura sottostante andiamo ad esaminarlo al fine di rispondere ai quesiti dettati dalle task:



3. IDENTIFICARE I COSTRUTTI NOTI



4. IPOTIZZARE IL COMPORTAMENTO DELLA FUNZIONALITA' IMPLEMENTATA

In conclusione possiamo supporre, analizzando il codice in assembly, che, quest'ultimo, contiene istruzioni per il malware al fine analizzare se la macchina vittima abbia o meno la connessione ad internet, mandando in stampa il risultato finale dell'operazione.

5. BONUS: TABELLA CON SIGNIFCATO DI OGNI SINGOLA RIGA DEL CODICE ASSEMBLY

ISTRUZIONE	SIGNIFICATO
push ebp	Spinge il registro ebp in cima allo stack
mov ebp, esp	Copia il contenuto del registro sorgente (esp) nel registro destinatario (ebp)
push ecx	Spinge il registro ecx in cima allo stack
push 0 ; dwReserved	Passa il valore del parametro booleano in cima allo stack
push 0 ; lpdwFlags	Passa il valore del parametro booleano in cima allo stack
call ds:InternetGetConnectedState	Chiamata di funzione tramite il puntatore **ds** che crea un nuovo EIP per la funzione
mov [ebp+var_4], eax	Copia il contenuto del registro eax nella locazione destinatario
cmp [ebp+var_4], 0	Compara il valore del destinatario con la sorgente andando a modificare ZF e CF
jz short loc_40102B	Salto condizionale alla locazione di memoria indicata se lo ZF è 1
push offset aSuccessInterne	Spinge il contenuto della stringa in cima allo stack
call sub40117F	Chiamata di funzione alla locazione di memoria indicata
add esp, 4	Aggiunge il valore 4 al contenuto del registro esp
mov eax, 1	Copia 1 nel contenuto del registro eax
jmp short loc_40103A	Salto condizionale alla locazione di memoria indicata
loc40102B:	Locazione di memoria (istruzione successiva a jz short… se verificata)
push offset aError1_1NoInte	Passa il contenuto della stringa in cima allo stack
call sub40117F	Chiamata di funzione alla locazione di memoria indicata
add esp, 4	Aggiunge il valore 4 al contenuto del registro esp
xor eax, eax	Operatore logico che inizializza il valore dei registri eax (0 se identici)
loc_40103A:	Locazione di memoria (istruzione successiva a jmp short…)
mov esp, ebp	Copia il contenuto del registro ebp, nel registro esp
pop ebp	Rimuove il registro ebp dalla cima dello stack
retn	Istruzione utile per ritornare alla funzione chiamante (*****)
sub_401000 endp	Segna la fine del processo principale nominato sub_401000 (******)

***http://www.giobe2000.it/Tutorial/Schede/07-IstruzioniCpu/RETN.asp**

****https://developer.arm.com/documentation/101655/0961/Ax51-User-s-Guide/Control-Statements/Reference/ENDP-Assembler-Statement**