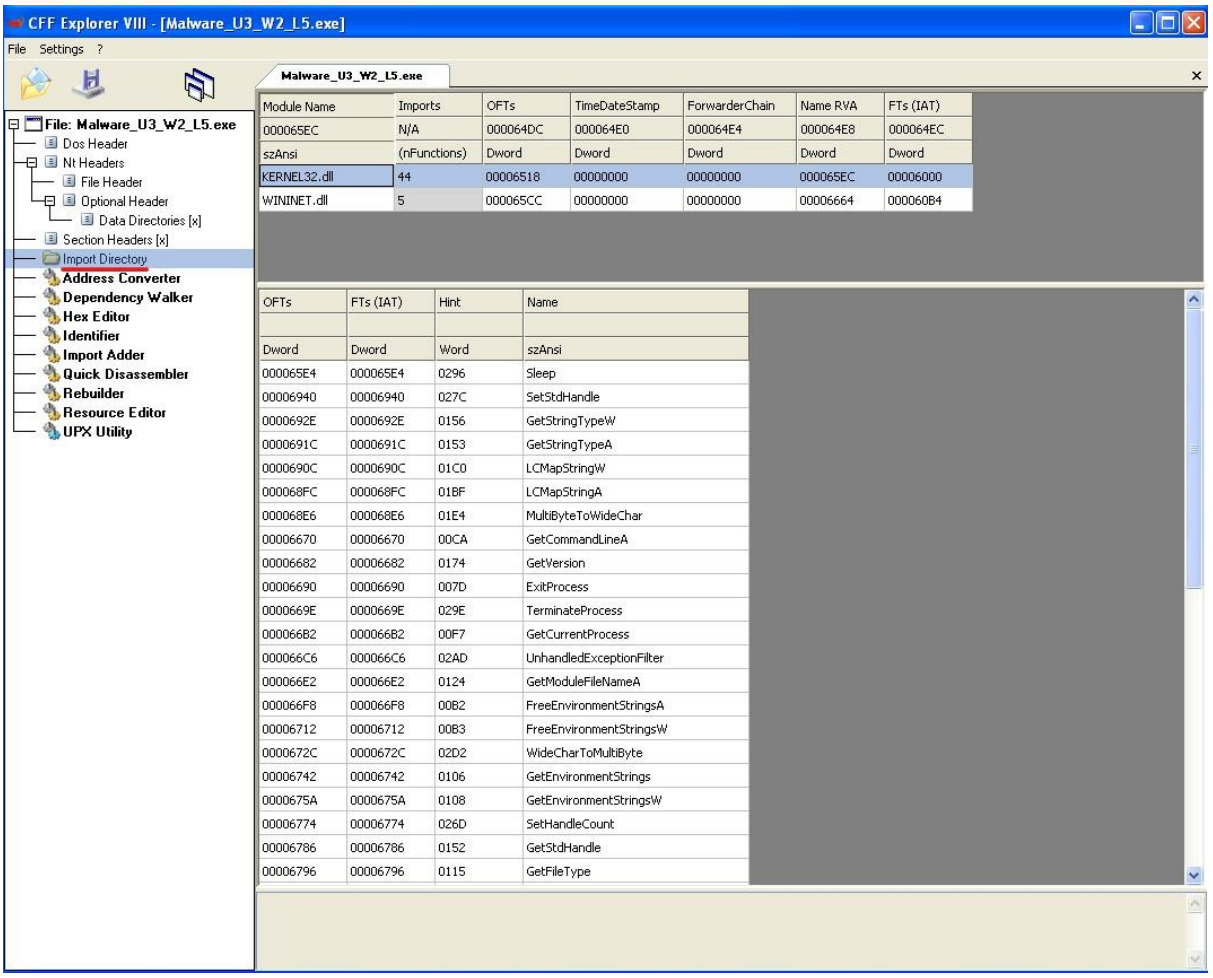
Rossella Amore - 07/07/2023

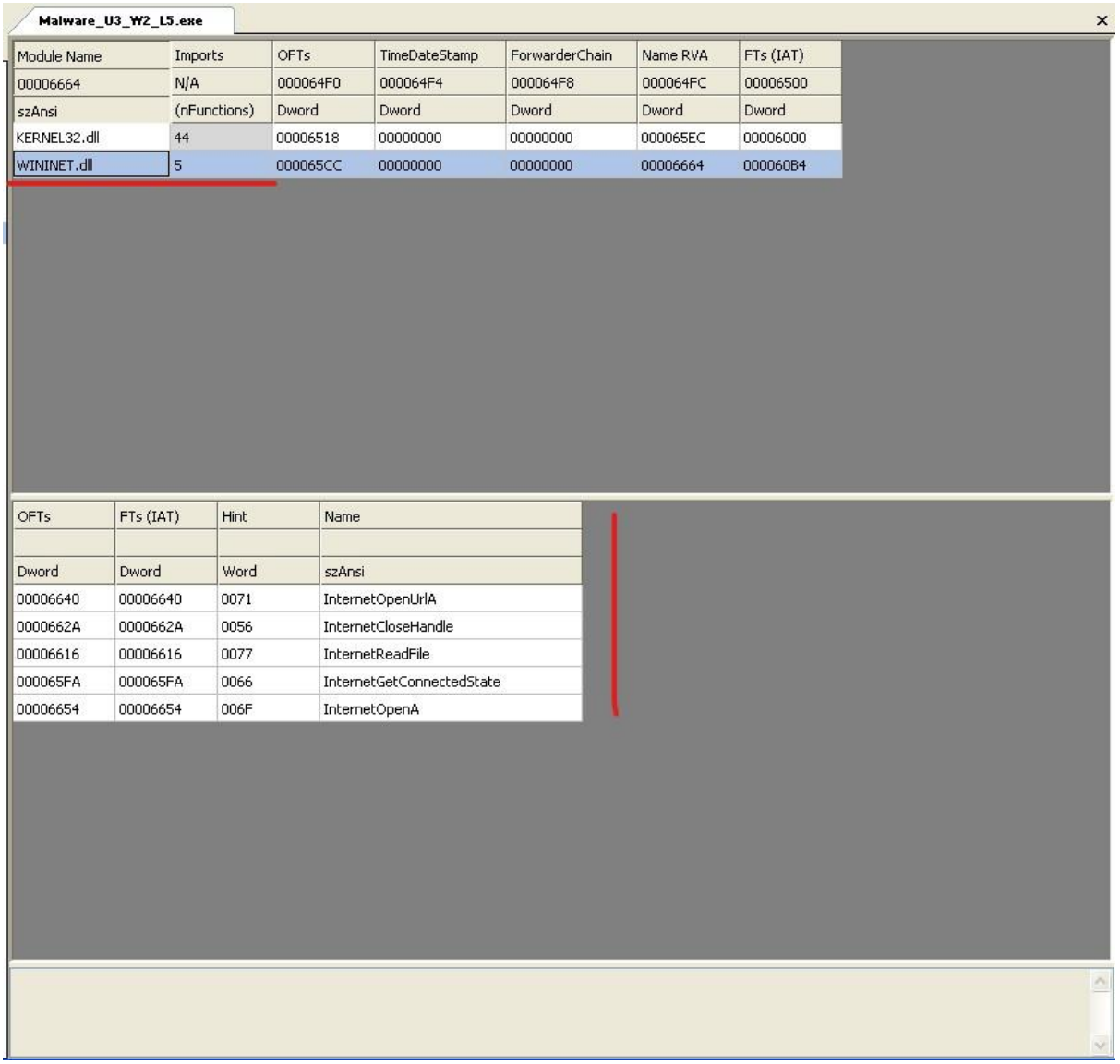
Report - Analisi statica e dinamica

Analizzo in maniera statica il malware **“Malware_U3_W2_L5.exe”** con lo strumento **CFF Explorer**.
Una volta aperto il file, navigo nel programma fino ad arrivare alla pagina *“Import Directory”*, la quale mi mostra le librerie e le funzioni importate dal file.



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
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

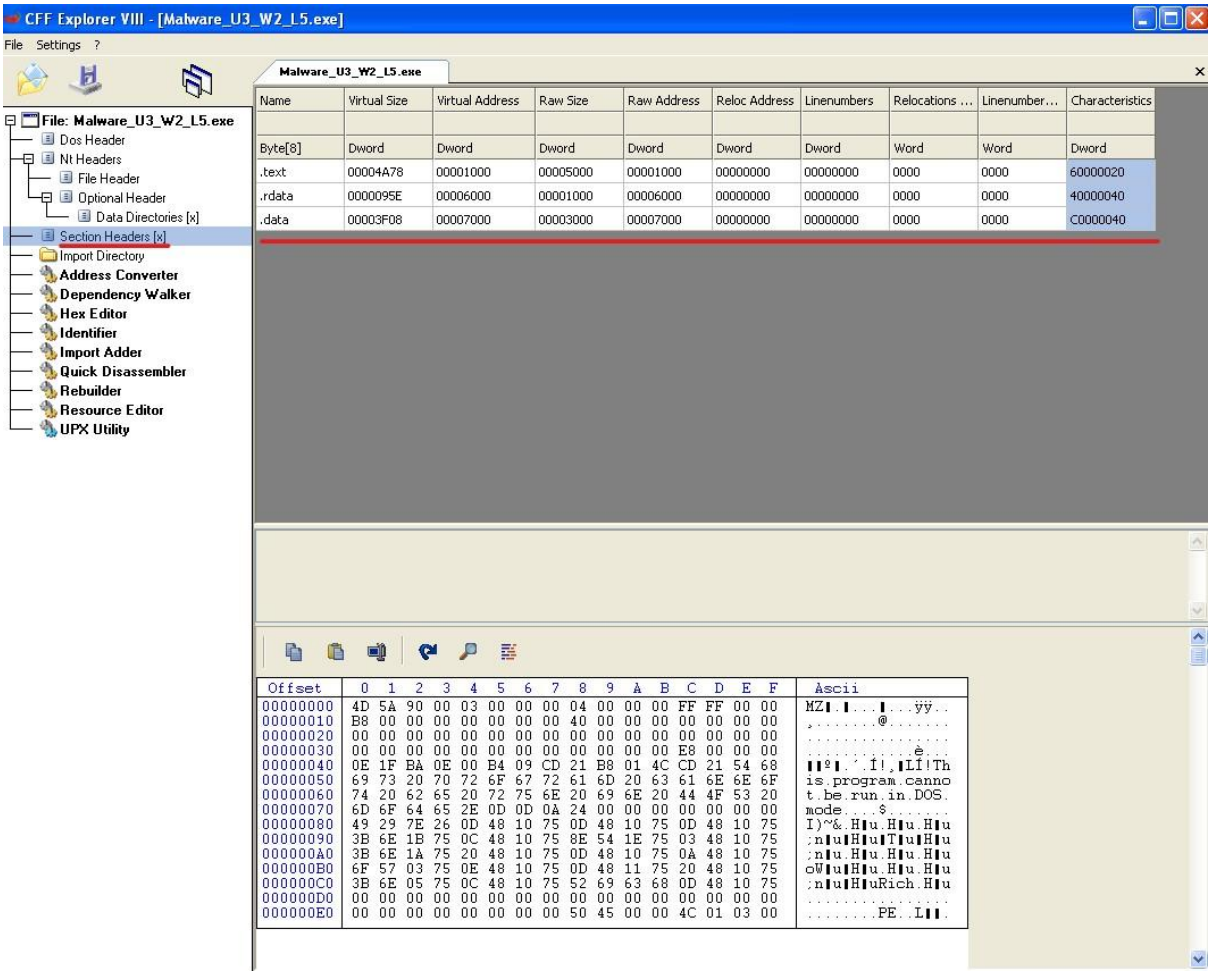


Le librerie importate, in questo caso, sono due: Kernel32.dll e Wininet.dll.

Kernel32.dll, nei sistemi operativi Windows, è un modulo che contiene le funzioni principali per interagire con il sistema operativo, gestire memoria, file e processi. Ci sono in tutto 44 funzioni su questa libreria.
Questa libreria permette al malware, per esempio, di creare e modificare file sensibili, far partire altri processi malevoli, importare altre librerie con funzioni potenzialmente malevole usando “LoadLibrary” e “GetProcAddress”, come in questo caso. Può anche utilizzare funzioni per ottenere maggiori informazioni sul sistema bersaglio. Una delle funzioni in questo malware, per esempio, è “GetVersion”**,** che riesce infatti a ottenere la versione di Windows della macchina bersaglio.

Wininet.dll, nei sistemi Windows, dà accesso a funzionalità e operazioni legate alla connessione internet e protocolli come FTP, HTTP, NTP (Network Time Protocol) e altri. Ci sono in tutto 5 funzioni in questa libreria.
È maggiormente utilizzato per il furto e l'invio di dati sensibili su rete, per esempio su server remoti, per eseguire un'enumerazione di rete e scaricare altri componenti malevoli. Può inoltre avere l'accesso a cookie di sessione, impostazioni dei browser, manipolazione dei certificati e bypassare o modificare proxy già esistenti.
Le funzioni presenti permettono al malware di connettersi a internet, aprire url e leggere file. (InternetGetConnectedState, InternetOpenUrlA, InternetReadFile ecc.)
La funzione InternetOpenUrlA viene usata per codice scritto in ANSI ed è obsoleta, ma in questo caso può essere utile su questa versione di Windows XP.

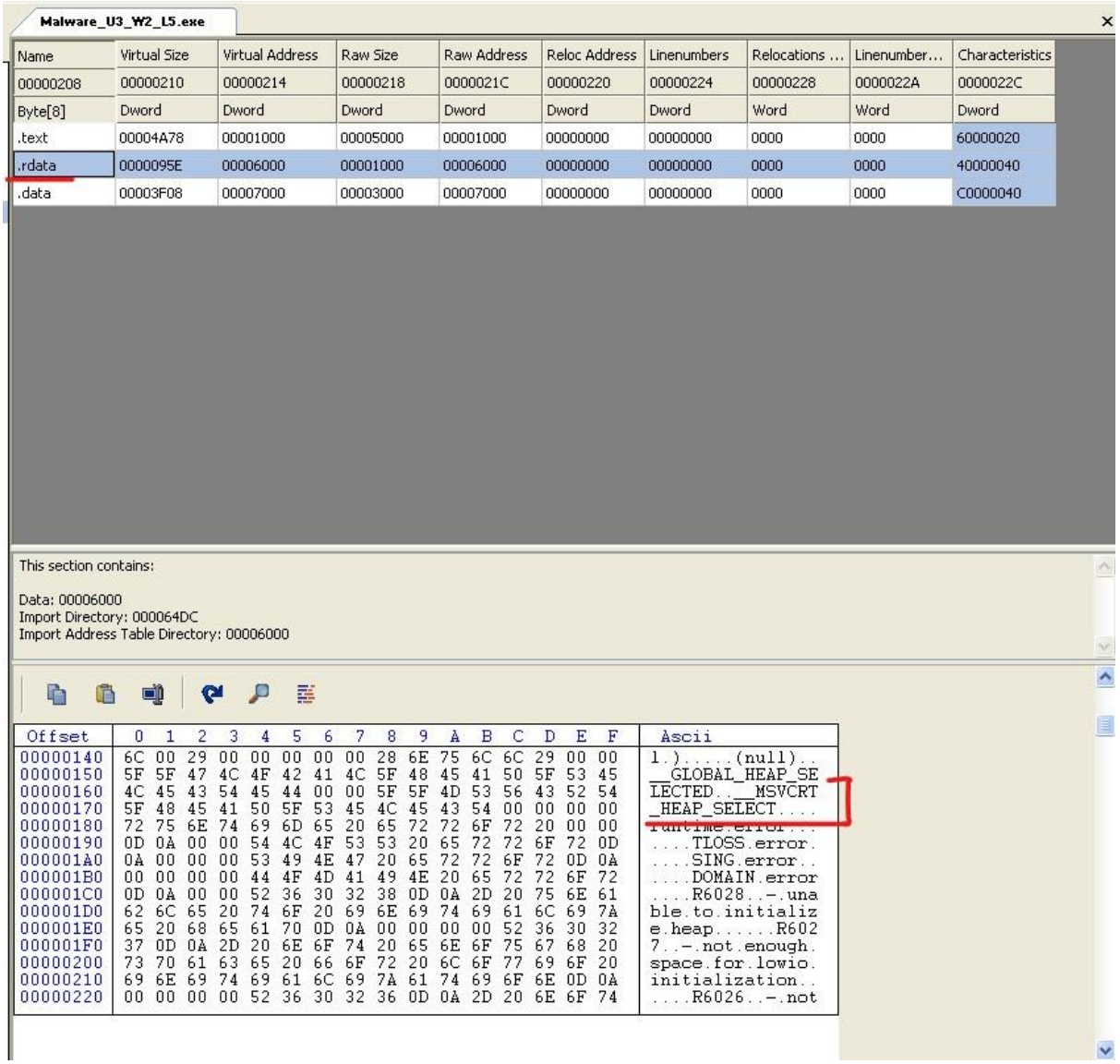
E' altamente probabile che il malware abbia altre funzioni importate con la funzione "LoadLibrary" e "GetProcAddress".
Potrebbe inoltre usufruire di librerie importate dinamicamente, ma non è possibile scoprirlo per certo con un'analisi statica.



Il file ha tre sezioni: .text, .rdata e .data.

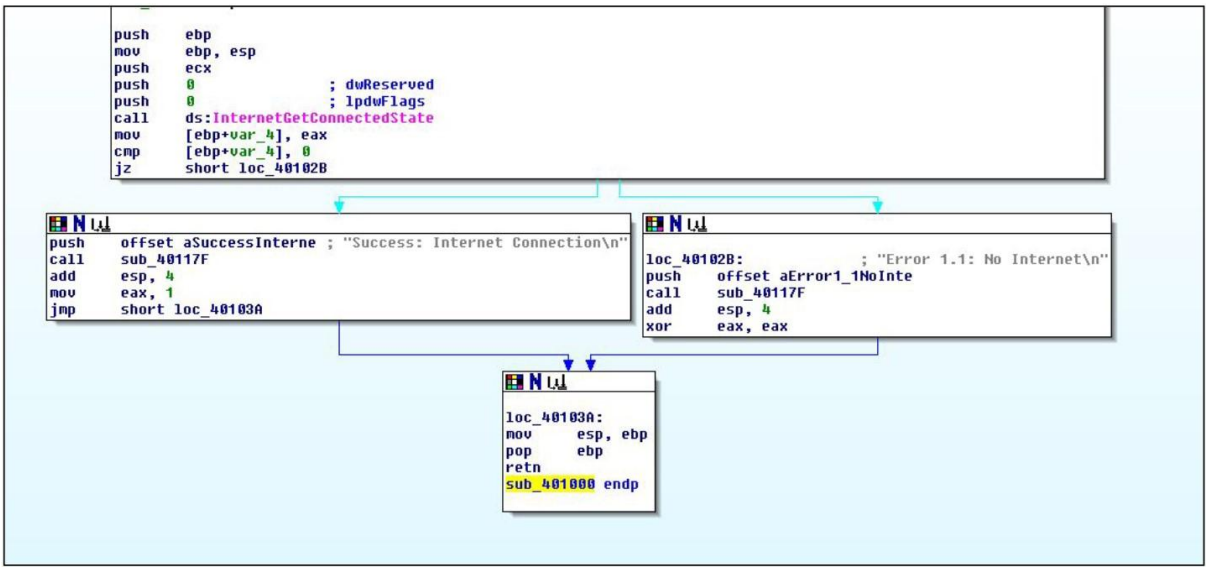
.text è la sezione più importante del programma che contiene le istruzioni e il codice da eseguire (tipicamente in C, C++ o assembly) dalla CPU una volta avviato l'eseguibile, con le sue funzioni e procedure. E' di solito supportato dai dati e informazioni contenuti nelle altre sezioni.

.rdata è la sezione che contiene informazioni read-only, ovvero dati che il programma legge senza modificare, come le informazioni sulle librerie e funzioni importate. Grazie a .rdata scopriamo che il malware, tramite "LoadLibrary" e "GetProcAddress", ha anche importato la libreria **MSVCRT.dll,** utile per la manipolazione di stringhe, allocazione di memoria e chiamate per input e output. Le sue funzioni saranno eseguite una volta avviato il file.

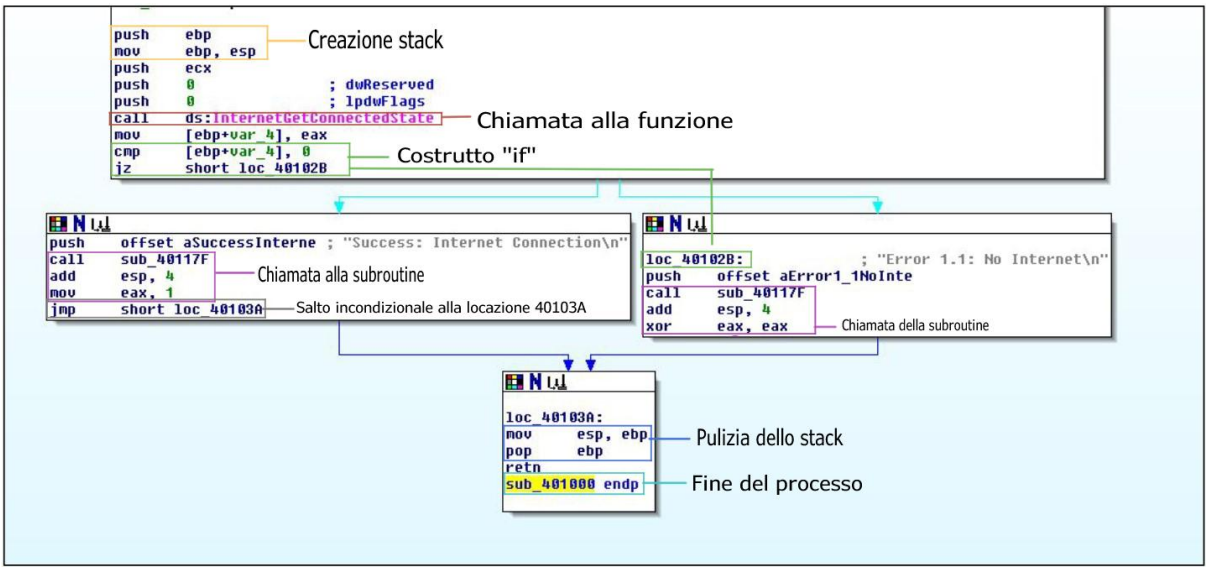


.data è la sezione che contiene variabili globali e statiche accessibili da ogni parte del programma, quindi le informazioni non sono read-only come nella sezione .rdata.

Codice assembly



Abbiamo qui come riferimento la figura con il codice assembly in un comodo schema ad albero. Andiamo quindi a individuare i suoi costrutti.



Dentro questo codice assembly troviamo la creazione dello stack (push ebp; mov ebp, esp) dove il valore dello stack pointer viene copiato in quello del base pointer.
La chiamata alla funzione “InternetGetConnectedState” serve a identificare lo stato della connessione internet. Il programma esegue poi un costrutto if (jz short loc_40102B) per i due possibili risultati.

Se la connessione è presente, uscirà in output la stringa “Success: Internet Connection\n”, verrà chiamata una subroutine (mini programma presente all’interno del programma principale accessibile in qualunque punto) e il programma farà un jump alla locazione 40103A.

Se la connessione non dovesse essere presente, quindi cmp = 0, la ZF sarà 1 e il programma farà un salto alla locazione 40102B, poiché il costrutto è jz, ovvero jump if zero.

Arrivato alla locazione 40103A, il programma eseguirà una pulizia dello stack e finirà il processo. (mov esp, ebp; pop ebp)

Si può ipotizzare che questa porzione di codice faccia parte di un malware più grande e complesso, e che questa parte si occupi di constatare la presenza o meno di connessione internet, dando in output errore in caso dell’assenza di connessione.