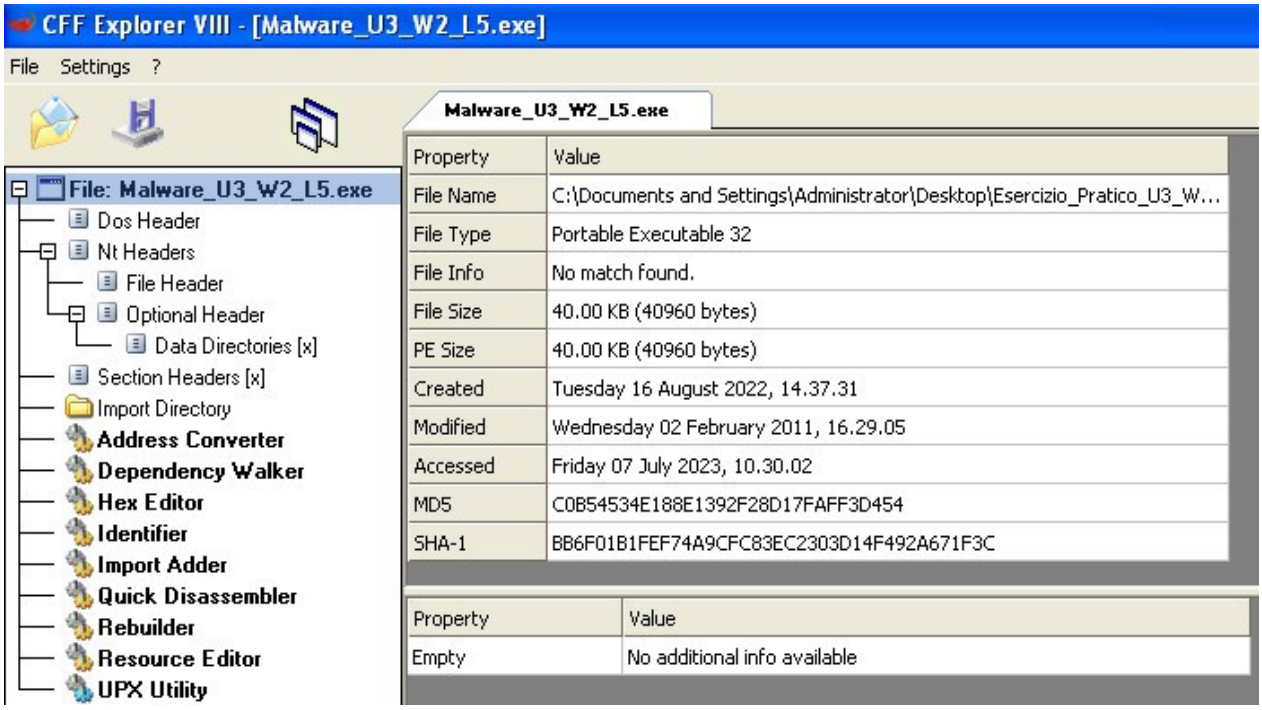
ESERCIZIO 5 U3 W2

Nell'esercitazione odierna,andremo ad analizzare il Malware_U3_W2_L5 all'interno della nostra macchina Win XP.

Per prima cosa prima di avviare la macchina,andiamo a creare un istantanea per non dover incorrere in un attacco da parte del malware in caso di errore del tecnico oppure per eseguire un analisi dinamica sulla macchina in questione e resettare Win XP alla forma originale.

Successivamente per iniziare la nostra analisi grazie al tool **CFF Explorer** andiamo a caricare il file eseguibile **Malware_U3_W2_L5** presente nella cartella Esercizio_Pratico_U3_W2_L5 sul Desktop.

Schermata iniziale:



Aperta la schermata di analisi andiamo a vedere nella colonna di sinistra ,la categoria **import directory** dove troveremo il tipo di librerie importate :

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

Come vediamo dall'immagine ,sono state importate due librerie ovvero:

KERNEL32.dll che contiene al suo interno le funzioni principali per interagire con il sistema operativo.

WININET.dll contiene al suo interno delle funzioni per l'mplementazione di protocolli di rete come FTP,HTTP e NTP.

All'interno delle due librerie troviamo alcune funzioni interessanti:

OFTs	FTs (IAT)	Hint	Name	OFTs	FTs (IAT)	Hint	Name
00006534	0000601C	00006670	00006672	00006534	0000601C	00006670	00006672
Dword	Dword	Word	szAnsi	Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep	000067F4	000067F4	019D	HeapDestroy
00006940	00006940	027C	SetStdHandle	00006802	00006802	019B	HeapCreate
0000692E	0000692E	0156	GetStringTypeW	00006810	00006810	02BF	VirtualFree
0000691C	0000691C	0153	GetStringTypeA	0000681E	0000681E	019F	HeapFree
0000690C	0000690C	01C0	LCMapStringW	0000682A	0000682A	022F	RtlUnwind
000068FC	000068FC	01BF	LCMapStringA	00006836	00006836	02DF	WriteFile
000068E6	000068E6	01E4	MultiByteToWideChar	00006842	00006842	0199	HeapAlloc
00006670	00006670	00CA	GetCommandLineA	0000684E	0000684E	00BF	GetCPInfo
00006682	00006682	0174	GetVersion	0000685A	0000685A	00B9	GetACP
00006690	00006690	007D	ExitProcess	00006864	00006864	0131	GetOEMCP
0000669E	0000669E	029E	TerminateProcess	00006870	00006870	02BB	VirtualAlloc
000066B2	000066B2	00F7	GetCurrentProcess	00006880	00006880	01A2	HeapReAlloc
000066C6	000066C6	02AD	UnhandledExceptionFilter	0000688E	0000688E	013E	GetProcAddress
000066E2	000066E2	0124	GetModuleFileNameA	000068A0	000068A0	01C2	LoadLibraryA
000066F8	000066F8	00B2	FreeEnvironmentStringsA	000068B0	000068B0	011A	GetLastError
00006712	00006712	00B3	FreeEnvironmentStringsW	000068C0	000068C0	00AA	FlushFileBuffers

possiamo andare a vedere ad esempio le funzioni **LoadLibraryA** e **GetProcAddress** dove la prima caricare il modulo specificato per il processo chiamante mentre la seconda recupera l'indirizio di una funzione esportata.

Andando avanti nell'analisi ci spostiamo nella categoria **section headers** dove al suo interno troviamo le sezioni :

.text,al suo interno contiene le istruzioni che la CPU eseguirà nel momento in cui il software viene avviato.

.rdata dove al suo interno troviamo le librerie e le funzioni importate dall'eseguibile.

.data che contiene le variabili globali del programma eseguibile.

Malware_U3_W2_L5.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
00000230	00000238	0000023C	00000240	00000244	00000248	0000024C	00000250	00000252	00000254
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

A questo punto terminata la parte di analisi statica basica ,passiamo alla parte avanzata del nostro scan,che consisterà di tradurre il costrutto nell'immagine linguaggio assembly in un linguaggio di alto livello.
Il tipo linguaggio di cui stiamo parlando è C .

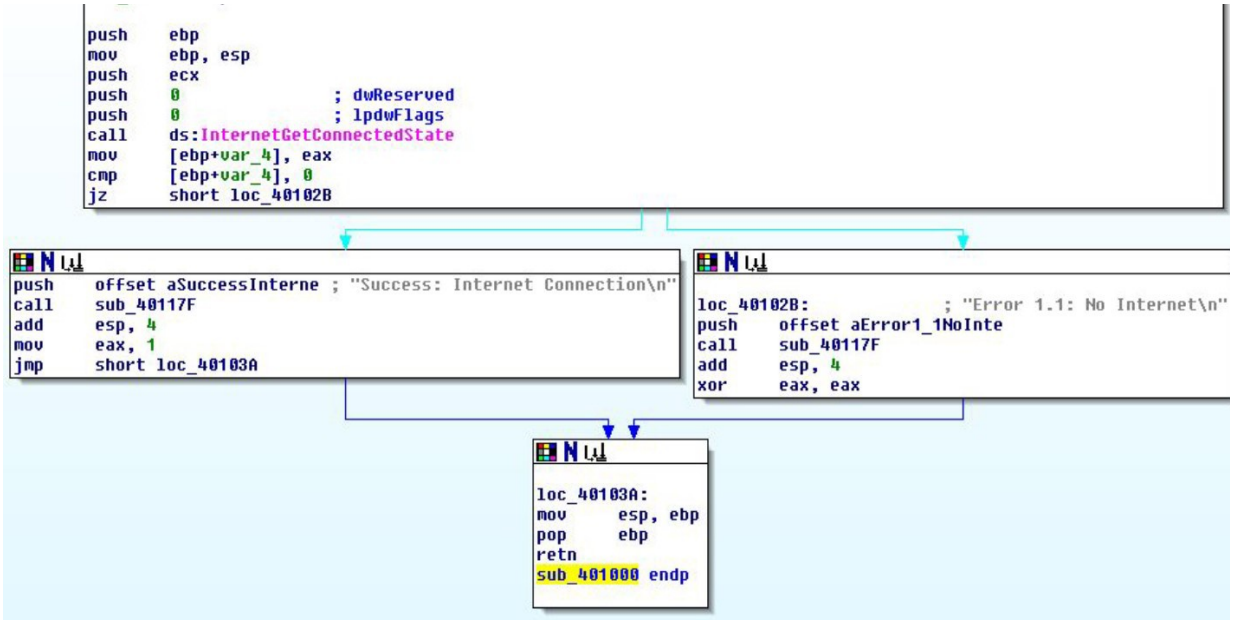
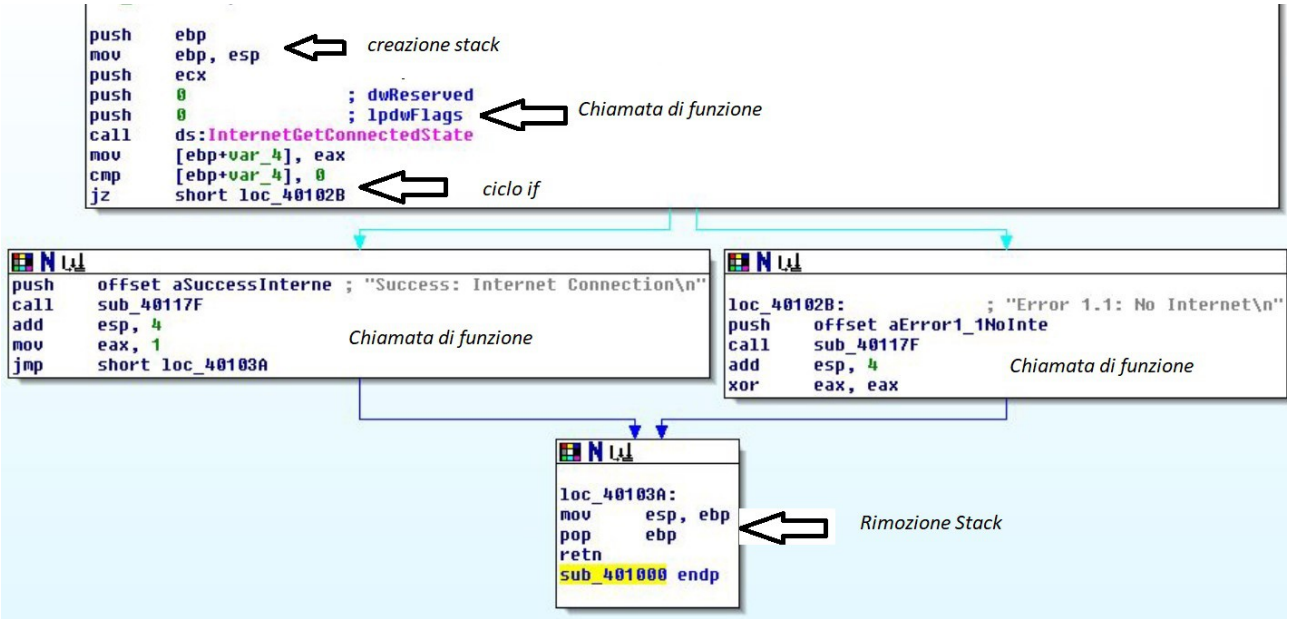


Immagine con specifica dei costrutti:



Per quanto riguarda la creazione dello stack EBP(extended base pointer) e ESP(extended stack pointer) sono 2 puntatori che delimitano la base e la cima dello stack necessari alla costruzione di un sistema LIFO (last in first out) ovvero che l'ultimo elemento aggiunto è il primo ad essere rimosso.

La chiamata di funzione INTENETGETCONNECTED STATE indica che i tre parametri ecx,0 e 0 vengono "pushati" e verificano se la macchina può connettersi ad internet.

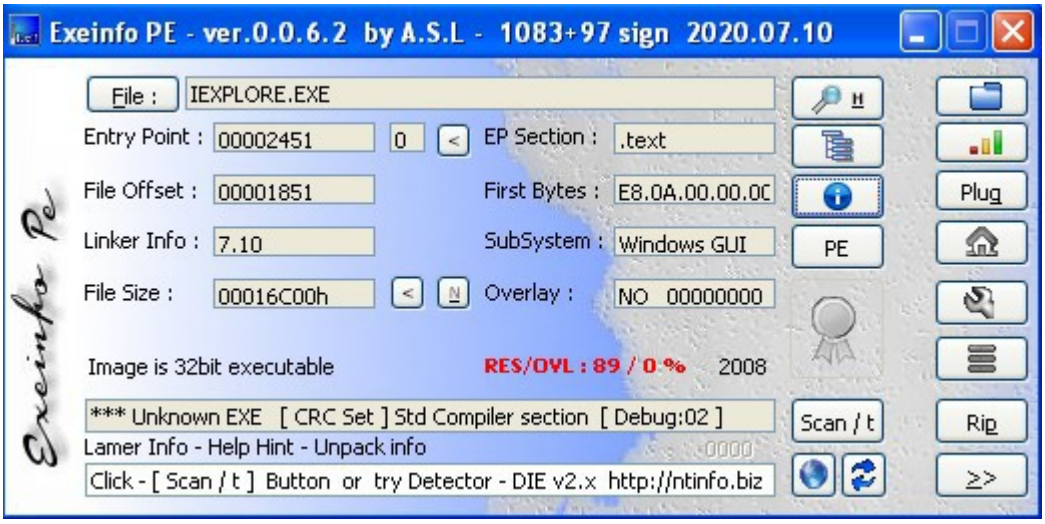
A questo punto interviene il ciclo if dove se il risultato è uguale a 0 effettua un salto alla locazione di memoria effettuando la connessione ad internet mentre se risulta diverso da 0 viene effettuato un salto alla locazione 40102B che darà un errore e rifiuterà di connettersi.

Infine avendo appurato se la connessione è stata rifiutata o stabilita,alla locazione 40103A viene effettuata la rimozione dello stack precedentemente creata nel primo costrutto.

ESERCIZIO BONUS:

Il bonus dell'esercizio richiedeva di convincere un dipendente neo assunto dalla nostra compagnia che aveva segnalato un file sospetto chiamato IEXPLORE.EXE .Per analizzarlo abbiamo utilizzato il tool EXEinfoPE caricando l'eseguibile in questione.Da gli screen si tratta di Internet Explorer,e quindi di un eseguibile non malevolo di Microsoft con annesso copyright.

Screen EXEinfoPE:



Info PE: