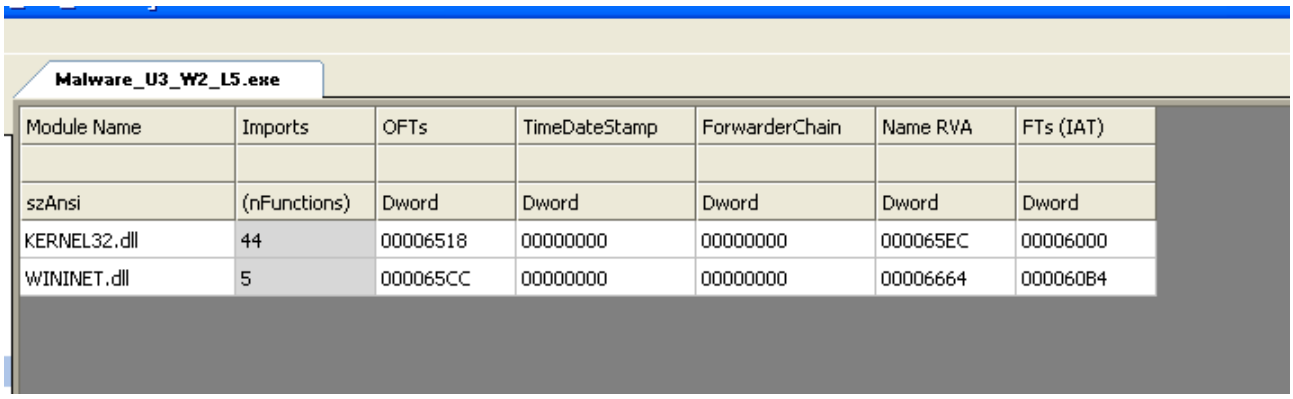
MALWARE ANALYSIS

- **LIBRERIE IMPORTATE DAL MALWARE**

Tramite CFF EXPLORER, apro il malware per analizzarlo.



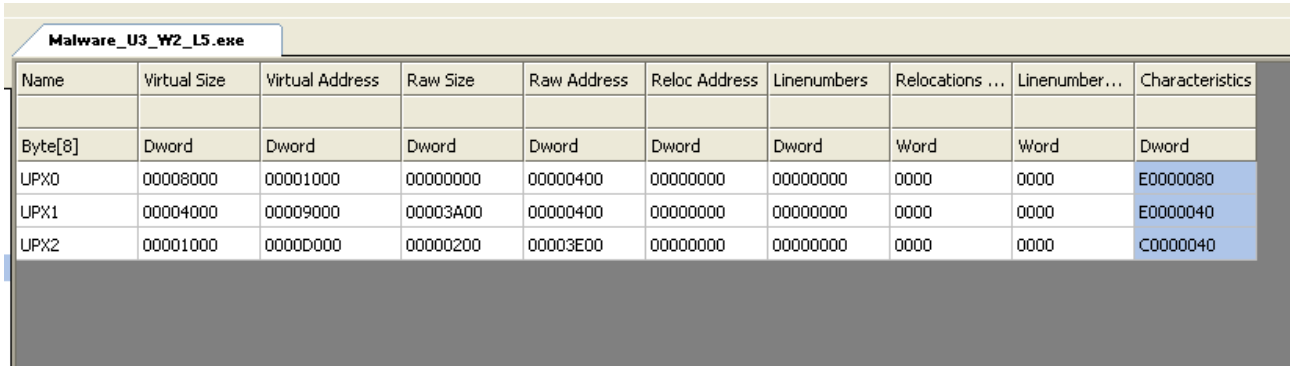
Malware_U3_W2_L5.exe

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

Come possiamo vedere dallo screenshot qui sopra le librerie importate dal Malware_U3_W_L5.exe sono le seguenti:

- KERNEL32.DLL: permette di interagire con il sistema operativo
- WINNET.dll: permette di implementare protocolli di rete come http,ecc

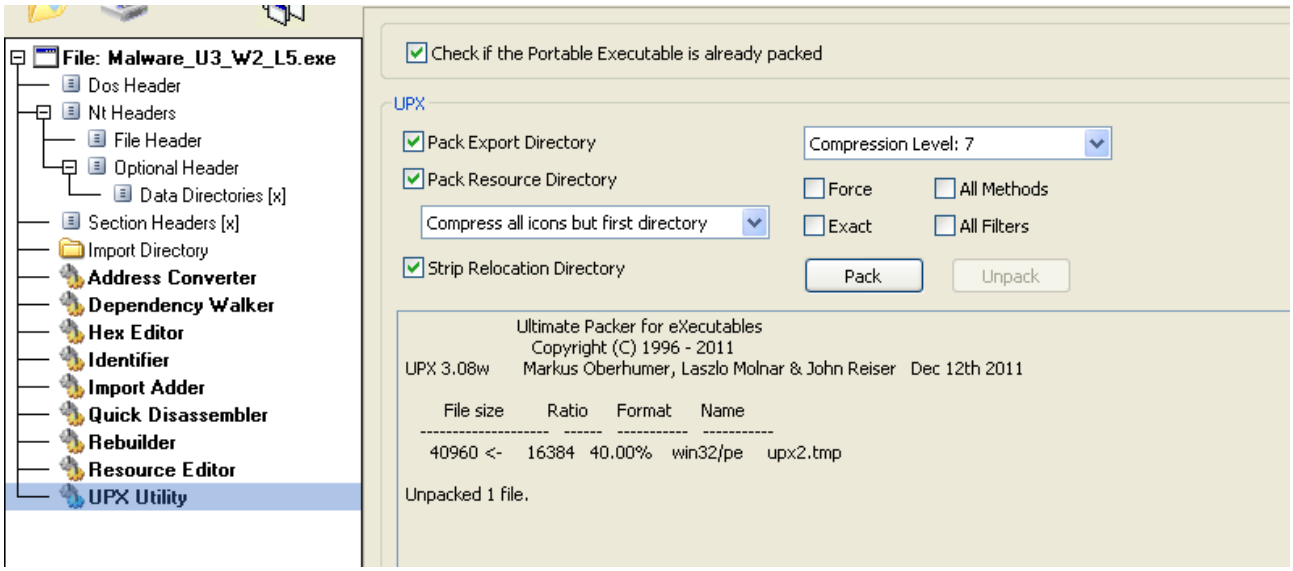
- **SEZIONI DI CUI E' COMPOSTO IL MALWARE**



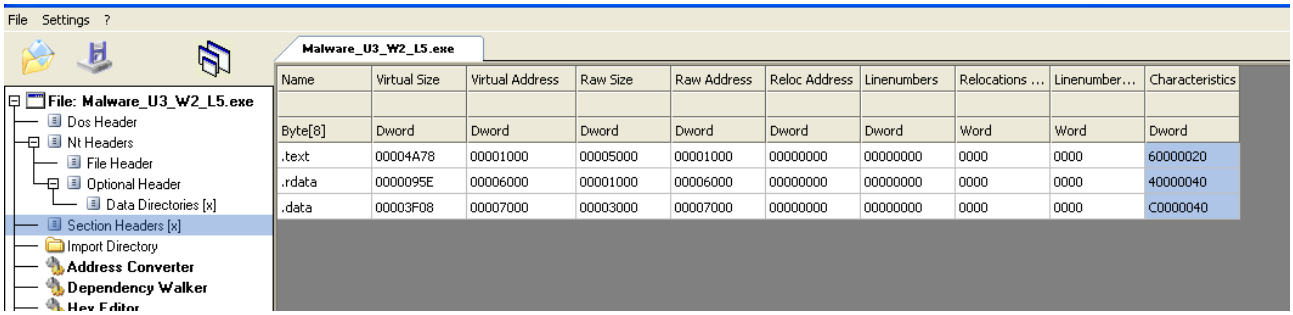
Malware_U3_W2_L5.exe

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
UPX0	00008000	00001000	00000000	00000400	00000000	00000000	0000	0000	E0000080
UPX1	00004000	00009000	00003A00	00000400	00000000	00000000	0000	0000	E0000040
UPX2	00001000	0000D000	00000200	00003E00	00000000	00000000	0000	0000	C0000040

Notiamo che il file è compresso , di conseguenza per vedere le sezione effettuiamo la seguente operazione:



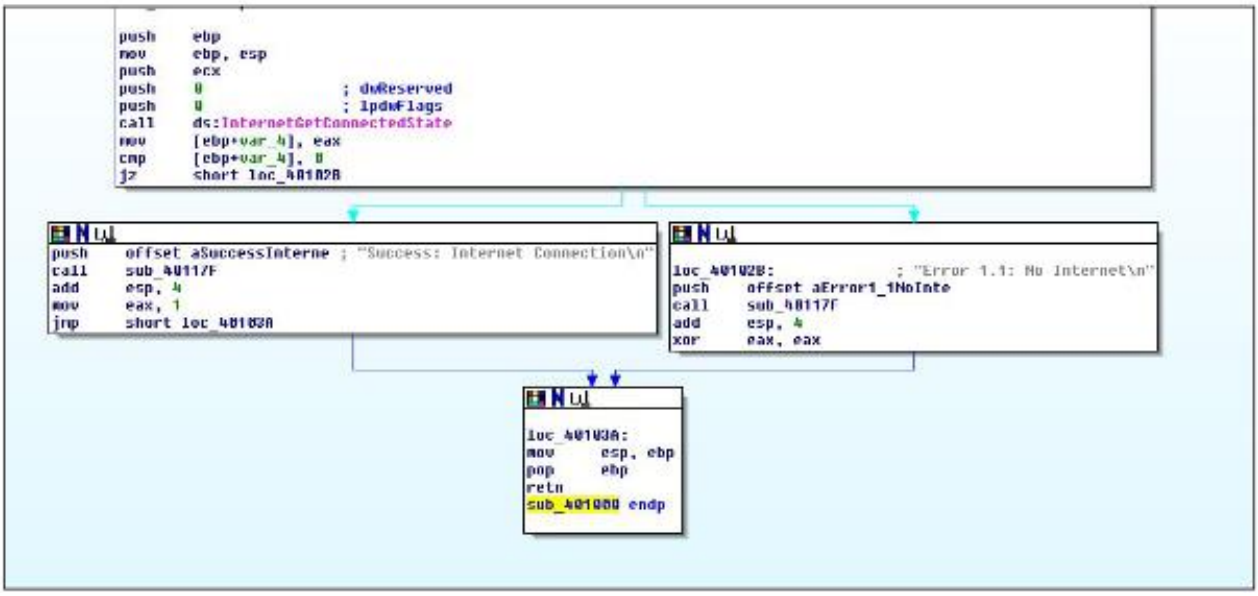
Otteniamo le seguenti sezioni:



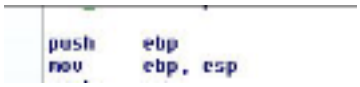
- SEZIONE.TEXT: contiene le informazioni che la CPU eseguirà una volta che il software viene avviato.
- SEZIONE .RDATA: include le informazioni sulle librerie e le funzioni importate ed esportate dall'eseguibile
- SEZIONE .DATA: contiene i dati e le variabili globali del programma eseguibile, quindi accessibili da qualsiasi funzione.

- **ANALISI CODICE ASSEMBLY**

L'immagine di seguito riportata mostra un frammento di codice assembly x86.



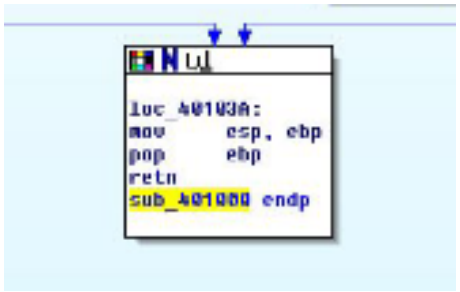
- **PARTE UNO: IDENTIFICAZIONE COSTRUTTI**

  - 

= creazione dello stack

  - 

= viene chiamata la funzione “INTERNETGETCONNECTEDSTATE” tramite le istruzioni PUSH

  - 

= parte finale del codice. Di base il seguente codice pulisce lo stack e ritorna alla funzione. Nel dettaglio: “mov esp, ebp” riporta il puntatore ESP al punto originale, prima della funziona chiamata; “pop ebp” ripristina il puntatore della funzione chiamante alla fine di una funzione e quindi il valore in cima allo stack viene recuperato. “retn” salta all'indirizzo di memoria di ritorno salvato

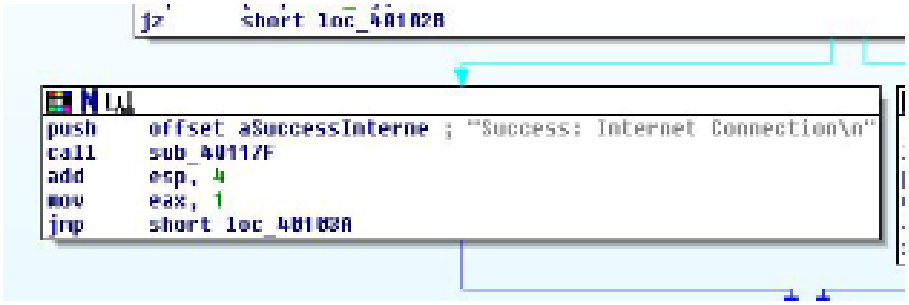
sullo stack cioè quando la funzione è stata chiamata, quindi termina l'esecuzione di una funzione e ritorna al punto in cui è stata chiamata.

- 

= viene settato il ciclo IF tramite le istruzioni CMP e JZ; in questo caso la condizione if è:
se il valore di ritorno della funzione è diverso da 0 , allora la connessione viene attivata .

DISTINGUO I DUE CASI:

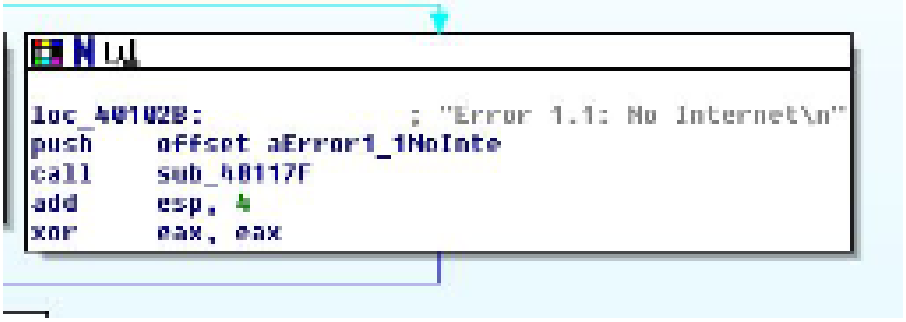
- SE IL RISULTATO E' DIVERSO DA ZERO



Una volta ottenuta la connessione ad internet:

- Vengono rimossi i parametri dello stack tramite ADD esp, 4
- Il valore 1 viene assegnato alla variabile eax e quindi avviene il salto di locazione alla memoria specificata tramite l'istruzione JMP

- SE IL RISULTATO E' UGUALE A ZERO



- In questo caso la connessione non viene attivata
- ADD esp, 4 elimina sempre i parametri dello stack
- l'istruzione xor eax, eax imposta il valore della variabile eax a zero

- **PARTE DUE: FUNZIONE IMPLEMENTATA**

Viene implementata la funzione internetgetconnectedstate nel malware che verifica se il computer è connesso a Internet o meno. Comprende il ciclo if e ne controlla appunto il valore di ritorno. Infatti tramite l'istruzione cmp , compara il valore di ritorno che se è diverso da zero stampa un messaggio di successo che indica che la connessione a Internet è disponibile. Nel caso opposto invece, stampa un messaggio di connessione ad internet fallita.