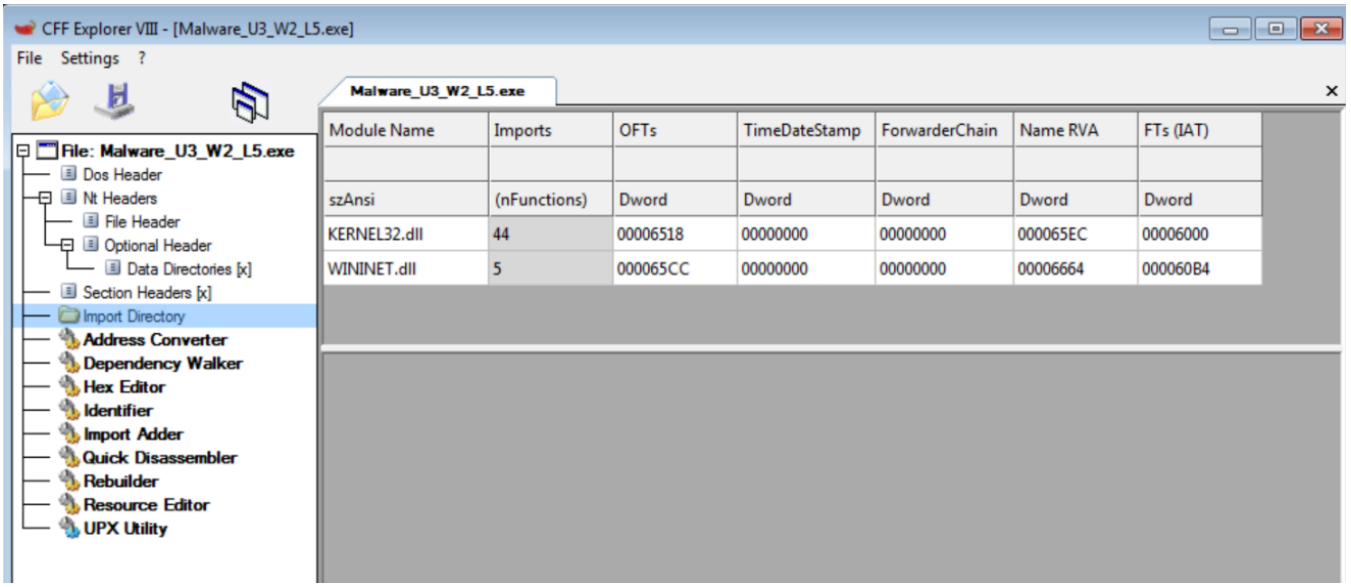
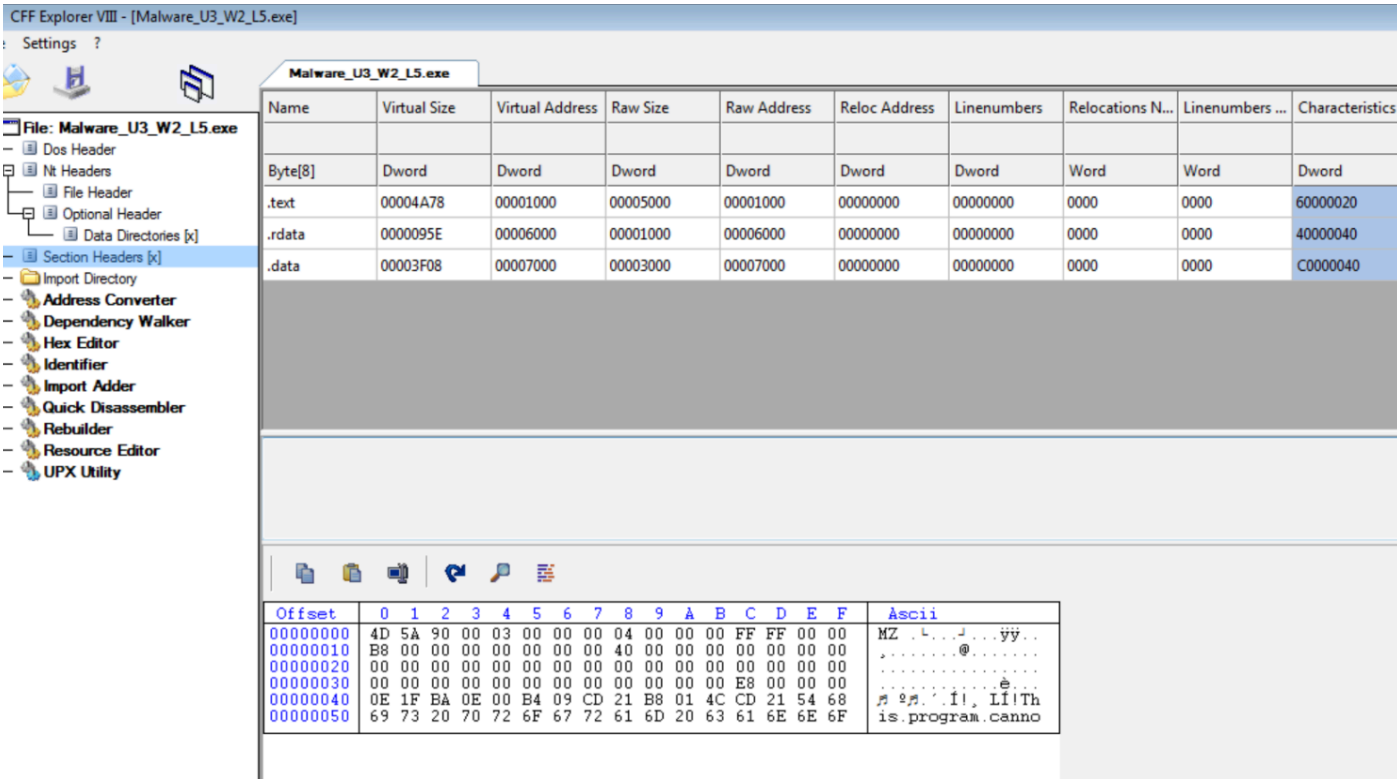
Devo analizzare un malware sulla macchina virtuale Windows 7.
Per prima cosa vado ad analizzare le librerie che vengono importate dal malware, quindi sulla macchina virtuale windows 7 apro il programma CFF Explorer e da li apro il malware che abbiamo in un'apposita cartella. Come primo passaggio vado ad analizzare se il malware ha importato delle librerie.



Vado su Import Directory e vedo che il malware ha importato le seguenti librerie:

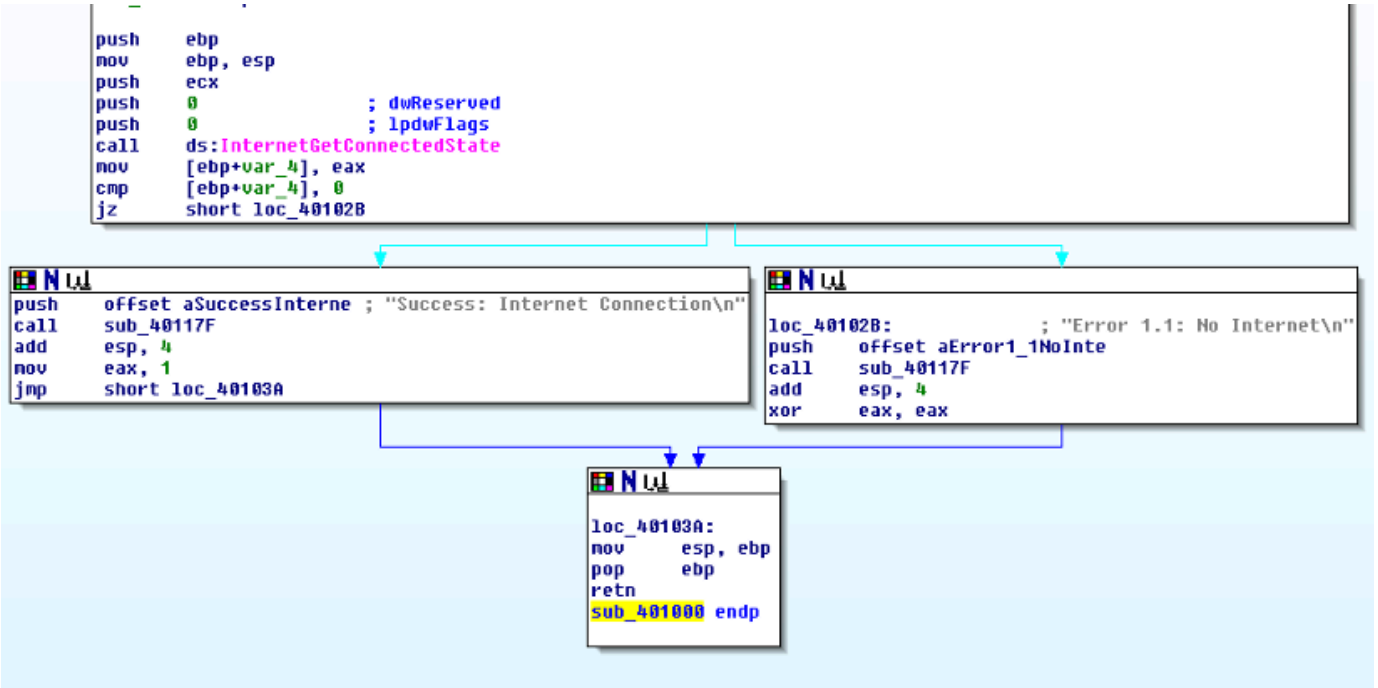
- **KERNEL32.dll** contiene le funzioni principali per interagire con il sistema operativo, tra cui la gestione della memoria, dei file e delle directory, come anche altre operazioni, gestioni dei processi e sincronizzazioni.
- **WININET.dll** contiene le funzioni per l'implementazione di alcuni protocolli di rete come HTTP, FTP, NTP e per svolgere operazioni di rete.

Dopo aver analizzato le librerie vado ad analizzare le sezioni del file malware, sul programma CFF Explorer nella sezione Section Headers.



Vedo che le sezioni del malware sono le seguenti:

- **.text** contiene le righe di codice che la CPU eseguirà una volta avviato il software.
- **.rdata** include le informazioni delle librerie e le funzioni importate ed esportate dal malware, contenente principalmente dati di sola lettura (read-only).
- **.data** contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Essendo una variabile globale è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile.



Vado ad identificare i costrutti noti dell'immagine mostrata.

Dall'analisi del codice, sembra che questo programma stia chiamando la funzione InternetGetConnectedState per verificare se la macchina è connessa a Internet.
Se il valore di ritorno di questa funzione è zero, il programma salta a loc_40102B, altrimenti continua l'esecuzione sequenziale.

Posso ipotizzare che analizzando il codice, il programma chiama la funzione "InternetGetConnectedState" per verificare la connessione ad internet della macchina, se il valore restituito è pari a zero, il programma salta alla locazione "loc_40102B" altrimenti continua l'esecuzione.

BONUS

- push ebp e mov ebp, esp: Queste istruzioni vengono utilizzate per creare un frame di stack per la funzione, salvando l'indirizzo del frame precedente (ebp) e spostando il puntatore dell'attuale frame (ebp) all'indirizzo attuale dello stack (esp).
- push ecx: Viene salvato il valore del registro ecx nello stack. Questo è un salvataggio temporaneo del registro ecx.ù
- push 0 e push 0: Due zeri vengono pushati nello stack. Questi valori saranno usati come argomenti per la chiamata alla funzione ‘InternetGetConnectedState’.
- call ds:’InternetGetConnectedState’: Viene chiamata la funzione InternetGetConnectedState, che probabilmente è una funzione della libreria di sistema di Windows. Gli zeri pushati prima vengono utilizzati come argomenti.
- mov [ebp+var_4], eax: Il risultato della chiamata a ‘InternetGetConnectedState’ (presumibilmente indicando lo stato della connessione) viene memorizzato nella variabile locale [ebp+var_4].
- cmp [ebp+var_4], 0: Viene effettuato un confronto tra il valore memorizzato in [ebp+var_4] e zero.
- jz short loc_40102B: Se il valore memorizzato in [ebp+var_4] è zero (il confronto è uguale a zero), salta a loc_40102B. Questo può indicare che non c'è connessione Internet.
- push offset asuccessInterne e call sub_40105F: Se il confronto è diverso da zero (indicando una connessione Internet), viene chiamata una subroutine sub_40105F con un argomento che punta alla stringa "Succes Internet Connection\n".
- add esp, 4: Viene ripristinato lo stack, rimuovendo gli argomenti pushati prima della chiamata alla subroutine.
- mov eax, 1: Il registro eax viene impostato a 1.