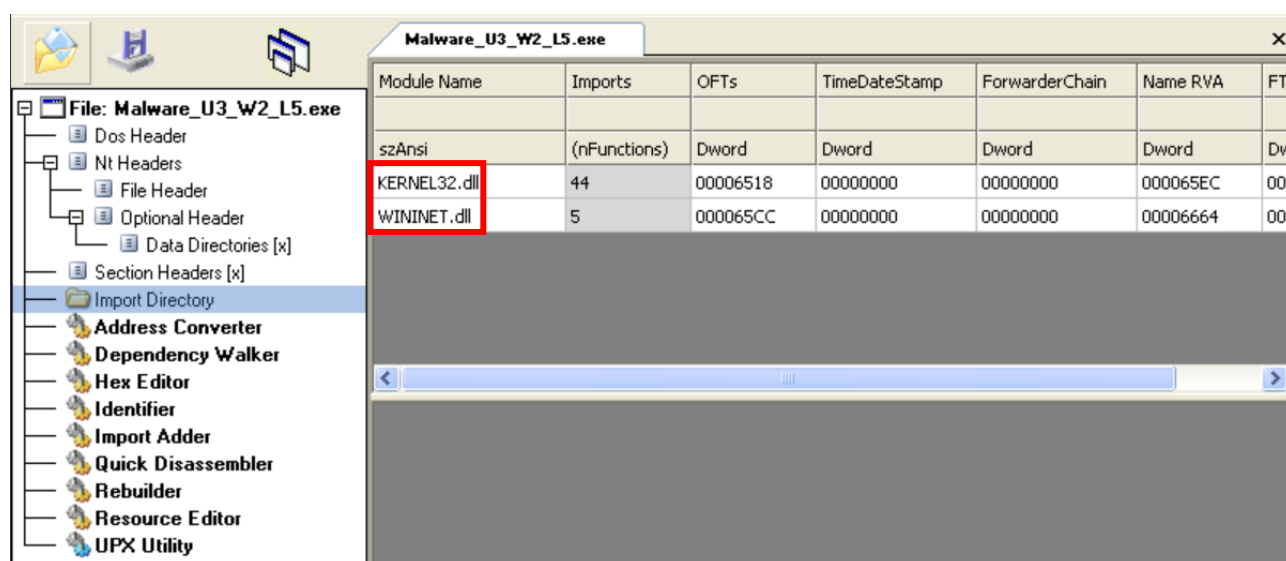


# ANALISI STATICA E DINAMICA: UN APPROCCIO PRATICO

Con riferimento al file Malware\_U3\_W2\_L5 presente all'interno della macchina virtuale dedicata, rispondere ai seguenti quesiti.

## 1) Quali librerie vengono importate dal file eseguibile?



Tramite l'utilizzo di CFF Explorer, spostandosi nella tab "Import Directory", si può notare che le librerie importate dal malware siano due:

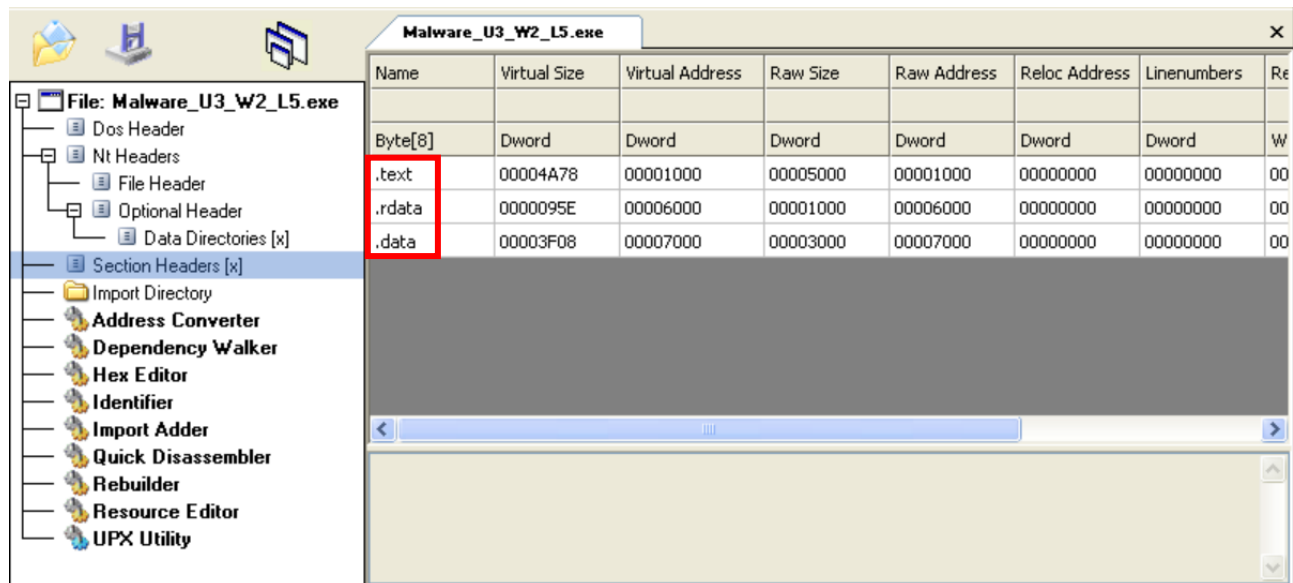
a) KERNEL32.dll: è una libreria di sistema essenziale nei sistemi operativi Windows. Si tratta di una delle principali librerie di Windows e il suo nome deriva dall'insieme di funzioni principali che fornisce per la gestione del kernel del sistema operativo. Si può notare come il malware utilizzi 44 funzioni della libreria KERNEL32.dll.

b) WININET.dll: è una libreria di sistema di Microsoft Windows che fornisce funzionalità per la gestione delle operazioni di rete e della connettività Internet. Il nome "WININET" deriva da "Windows Internet," ovvero il ruolo principale della libreria nella gestione delle operazioni di rete e Internet.

Questa libreria consente di eseguire operazioni di rete, come l'accesso a risorse su Internet, il download e l'upload di file, la gestione di connessioni FTP, HTTP e altre attività di rete.

Si può notare come il malware utilizzi 5 funzioni della libreria WININET.dll.

## 2) Quali sono le sezioni di cui si compone il file eseguibile del malware?



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Re
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	W
.text	00004A78	00001000	00005000	00001000	00000000	00000000	00
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	00
.data	00003F08	00007000	00003000	00007000	00000000	00000000	00

Spostandosi nella tab “Section Headers”, si possono notare le seguenti sezioni:

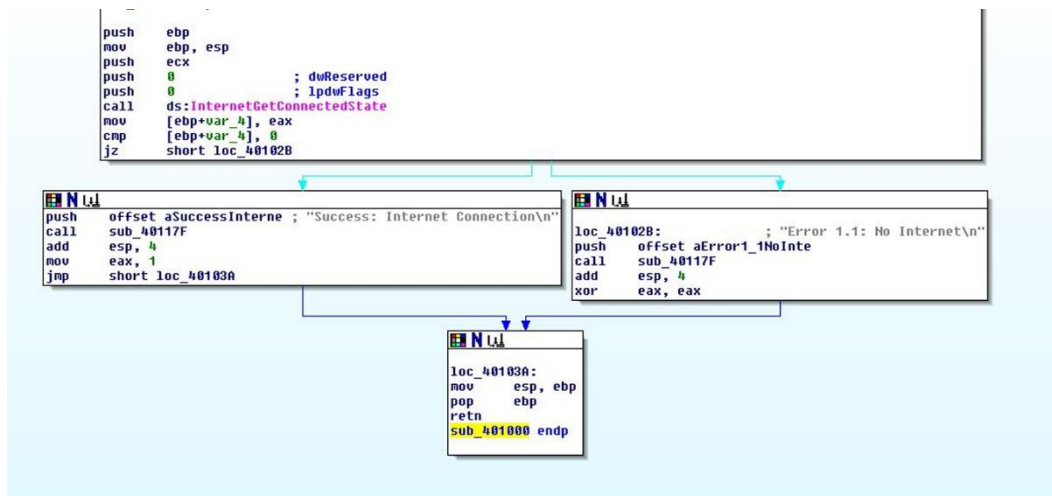
- a) .TEXT: Contiene il codice eseguibile del programma. È la sezione in cui risiedono le istruzioni che vengono eseguite dal processore durante l'esecuzione del programma.
- b).RDATA: Contiene dati di sola lettura (read-only data). Può includere costanti, stringhe e altri dati che il programma deve utilizzare.
- c).DATA: Contiene dati che possono essere modificati durante l'esecuzione del programma. Variabili globali e variabili statiche inizializzate vengono spesso allocate in questa sezione.

In un file eseguibile, le Section Headers sono porzioni specifiche di dati che svolgono ruoli diversi nell'esecuzione di un programma.

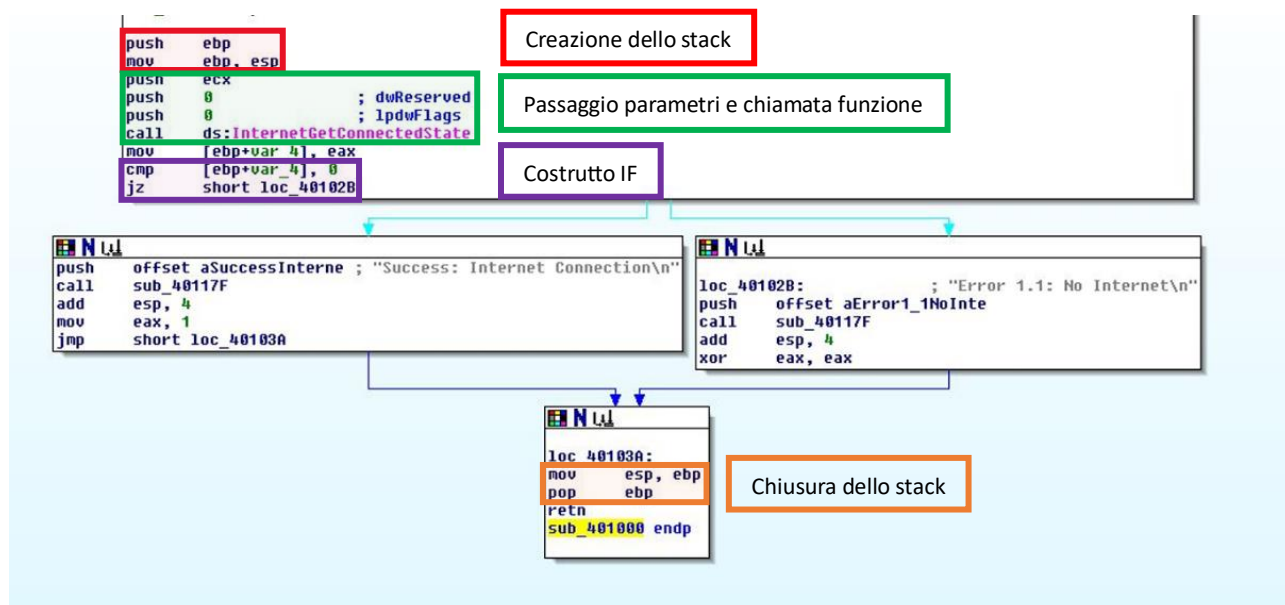
Ad esempio, una sezione può contenere il codice eseguibile, un'altra i dati inizializzati e così via.

Le Section Headers forniscono informazioni dettagliate su ciascuna di queste sezioni.

Con riferimento alla figura seguente, rispondere ai seguenti quesiti.



3) Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti).



Si identificano i seguenti costrutti:

- Creazione dello Stack.
- Passaggio dei parametri e chiamata della funzione InternetGetConnectedState.
- Costrutto IF.
- Chiusura dello Stack.

#### 4) Ipotizzare il comportamento della funzionalità implementata

Questo codice in Assembly indica che attraverso la funzione InternetGetConnectedState, si determina se su una macchina sia presente una connessione Internet.

Con il costrutto IF, avviene un controllo sulla funzione, che a seconda del parametro restituito (ovvero lo ZF settato a 0 o 1), indica a schermo la presenza o meno di una connessione internet sulla macchina. Come si può notare, se c'è la presenza di una connessione Internet, verrà restituito il messaggio "Success: Internet Connection", altrimenti verrà restituito "Error 1.1: No Internet".

#### 5) Fare una tabella con significato delle singole righe di codice Assembly

push ebp	sposta il valore di ebp nello stack.
mov ebp, esp	copia il valore di esp in ebp.
push ecx	sposta il valore di ecx nello stack.
push 0 ; dwReserved	sposta il valore 0 (var. dwReserved) nello stack.
push 0 ; lpdwFlags	sposta il valore 0 (var. lpdwFlags) nello stack.
call ds:InternetGetConnectedStatus	chiama la funzione InternetGetConnectedStatus usando i parametri precedenti.
mov [ebp+var_4], eax	copia il valore di eax nella variabile locale [ebp+var_4]
cmp [ebp+var_4], 0	compara il valore 0 con il valore in [ebp+var_4] / ris. = ZF
jz short loc_40102B	'salta' alla zona 40102B se ZF=1

push offset aSuccessInternet ; "Success:InternetConn\n"	sposta nello stack l'offset della stringa aSuccessInternet
call sub_40117F	chiama la funzione nell'indirizzo l'indirizzo sub_40117F
add esp, 4	libera lo spazio nello stack
mov eax, 1	copia il valore 1 in eax
jmp short loc_40103A	'salta' alla locazione 40103A (ultima parte del codice)

loc_40102B: "Error 1.1: No Internet\n"	destinazione dello jz con ZF = 1
push offset aError1_1NoInte	sposta nello stack l'offset della stringa di errore
call sub_40117F	chiama la funzione nell'indirizzo l'indirizzo sub_40117F
add esp, 4	libera lo spazio nello stack
xor eax, eax	esegue un'operazione di XOR tra il registro EAX e sé stesso, il che imposta EAX a zero Viene utilizzato per cancellare il registro

loc_40103A:	destinazione dello jmp short
mov esp, ebp	ripristina lo stack pointer al suo valore originale
pop ebp	ripristina il valore originale del registro di base
retn	ritorno dalla funzione
sub_401000 endp	fine del blocco di codice