S10/L5- Analisi statica e dinamica

Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5» sul desktop della macchina virtuale dedicata per l'analisi dei malware:

identifica le librerie e i section headers del file eseguibile.

Premessa: stiamo per effettuare un analisi statica, questo vuol dire che sono solo supposizioni quelle che andremo a fare e ci baseremo sulle informazioni del malware senza eseguirlo, per farlo usufruiremo del tool **Cff Explorer:**

Cff explorer è un tool di esplorazione e analisi per file binari, come eseguibili e librerie di sistema. Fornisce dettagli sulla struttura interna dei file.

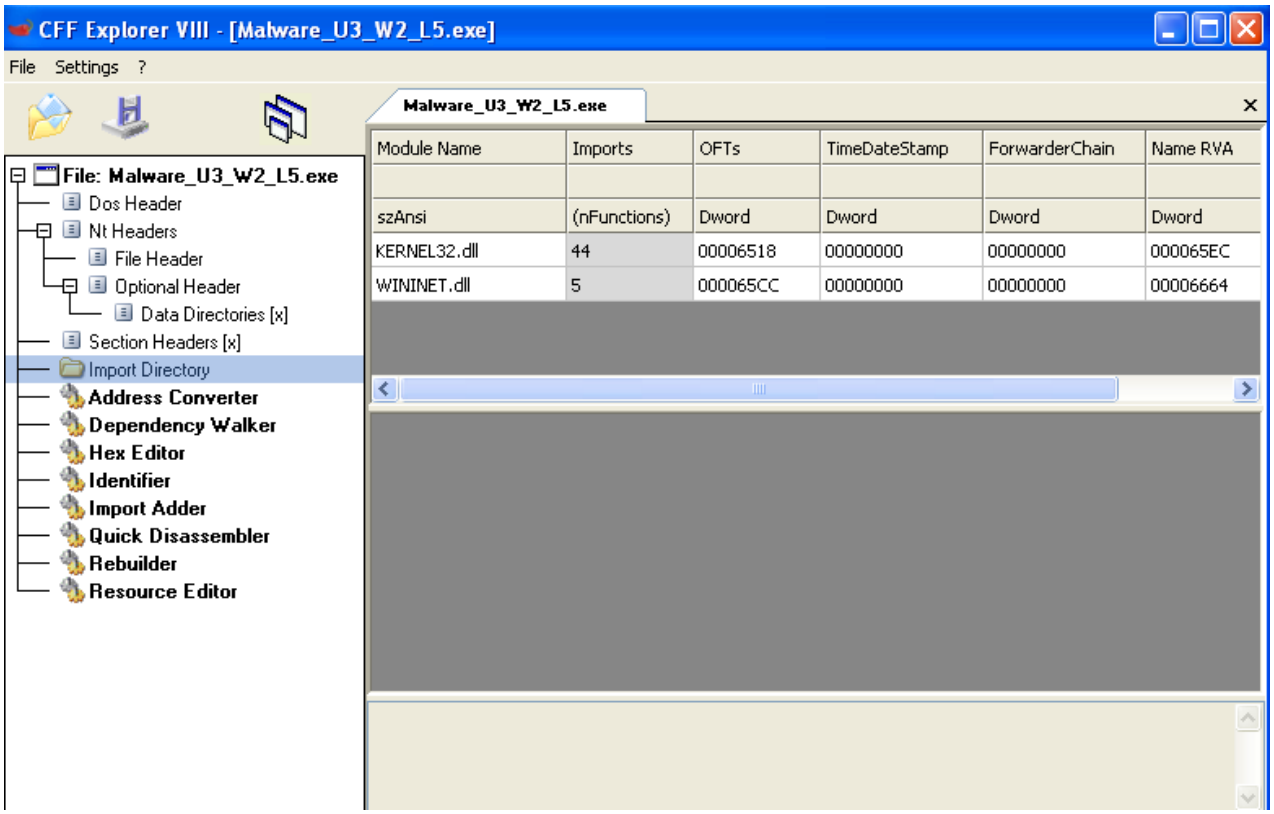
I file che andiamo ad analizzare sono in formato **PE (portable executable)** contengono informazioni per il sistema operativo e su come gestire il codice del file e le librerie.

Esecuzione del programma: è bastato aprire il programma selezionare la cartella con al suo interno il malware e caricarlo, da qui abbiamo accesso sia alle librerie che alle section headers. Di seguito vedremo cosa sono e cosa fanno:

Librerie: Le librerie di un malware sono file di codice o componenti software che vengono utilizzati per supportare le funzionalità principali del malware. Queste librerie possono includere codice per la comunicazione di rete, la crittografia, la manipolazione dei file e altro ancora.

Section headers: si tratta di informazioni contenute nell'intestazione (header) del file eseguibile del malware. Le sezioni includono solitamente il codice eseguibile, i dati iniziali, le risorse e altre informazioni cruciali per l'esecuzione del malware.

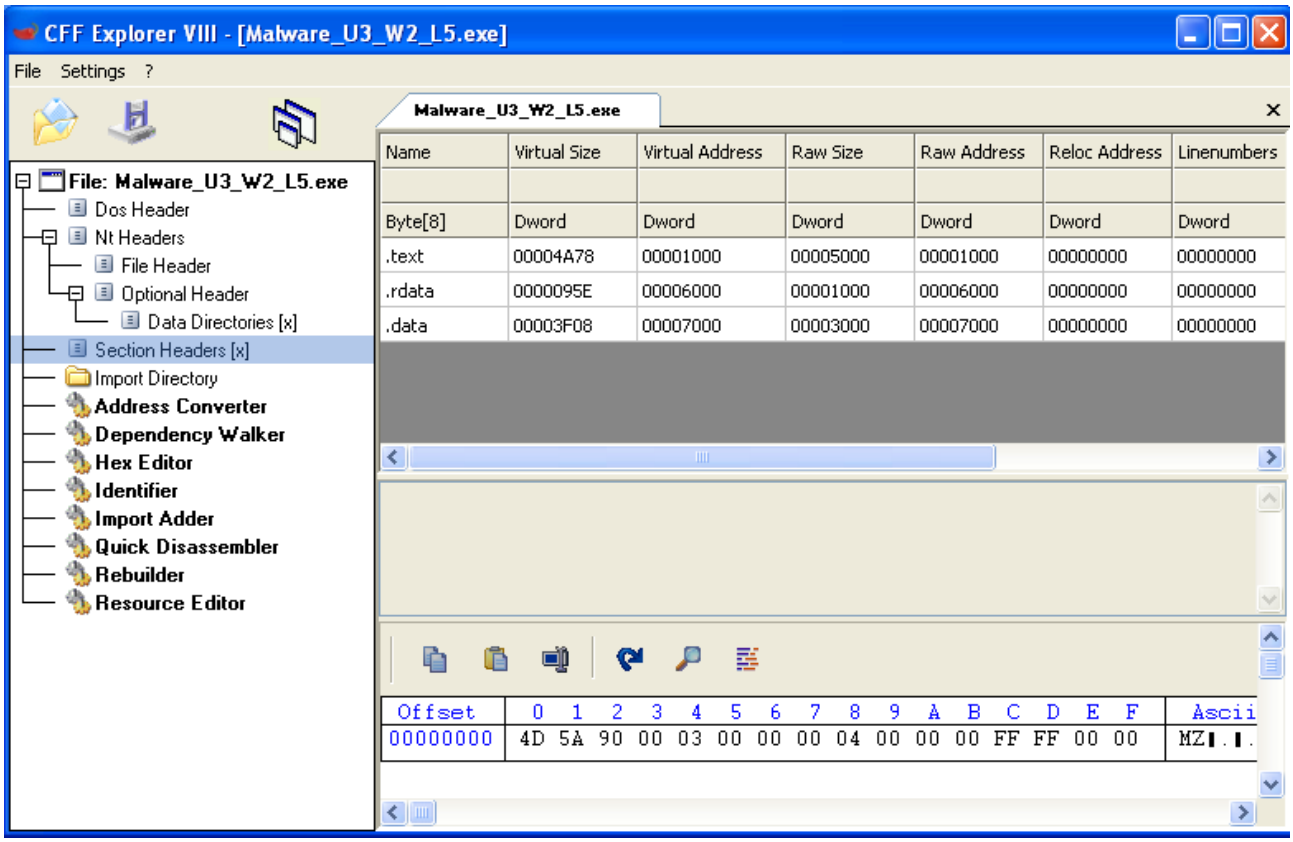
Nell’immagine di seguito vedremo le librerie presenti nel malware:



Kernel32.dll: contiene le funzioni principali per interagire con il sistema operativo, (manipolazione dei file, la gestione della memoria)

Wininet.dll: contiene le funzioni per l’implementazione di alcuni protocolli di rete come HTTP, FTP, NTP.

Nell'immagine di seguito vedremo i section headers e a cosa servono:



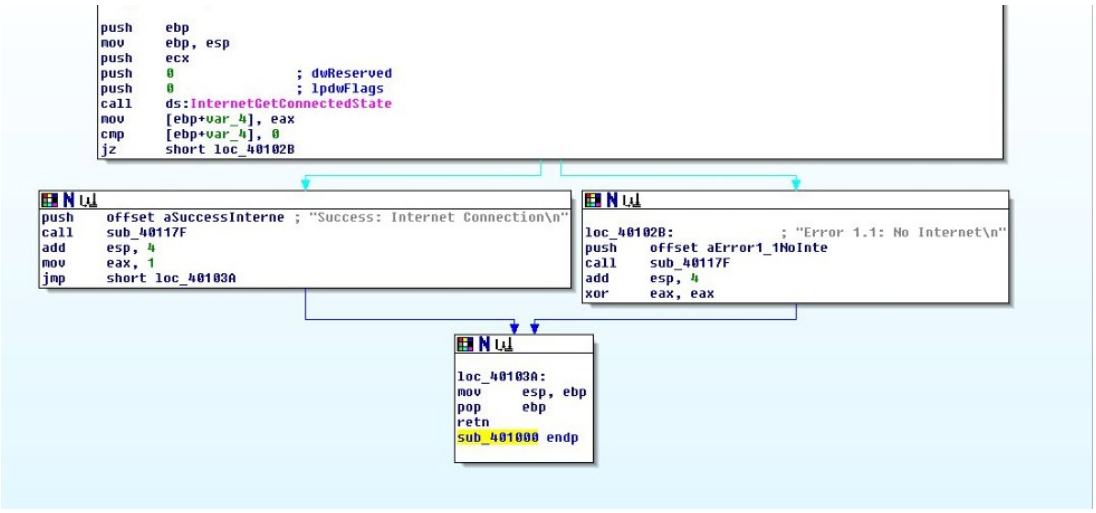
.text: contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. (Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU)

.rdata: include generalmente le informazioni sulle librerie e le funzioni importate ed esportate dall'eseguibile.

.data: contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.

(Una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile)

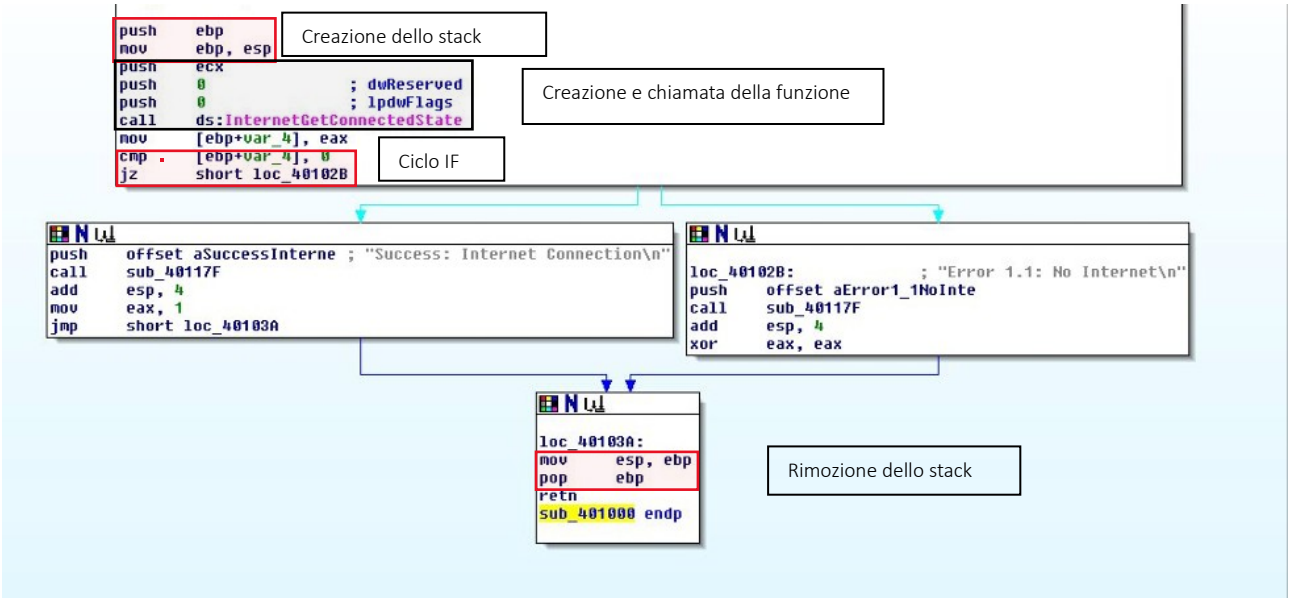
Parte2: Con riferimento alla figura di seguito, risponde ai seguenti quesiti:



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

2)Ipotizzare il comportamento della funzionalità implementata

1)



2) Il malware esegue la funzione `internetgetconnectedstate` e successivamente verifica il suo risultato attraverso una condizione IF. Se il compare setta lo zero falg a 0 il programma interpreta ciò come la presenza di una connessione attiva e risponde con "Success: Internet Connection\n". In caso contrario, se il valore di ritorno è 1, il programma verrà inidirizzato alla locazione 40102b la risposta del programma sarà "Error 1.1 No Internet\n",

Push ebp	Sposta il valore di ebp nello stack
Mov ebp, esp	Copia il valore esp nel registro ebp
Push ecx	Sposta il valore di ecx nello stack
Push 0	Sposta il valore 0 nello stack
Push 0	Sposta il valore 0 nello stack
Call ds: `internetgetconnectedstate`	Effettua una chiamata alla funzione InternetGetConnectedStatus:
Mov [ebp+var_4], eax	Copia il valore eax nella variabile [ebp+var_4]
Cmp [ebp+var_4],0	Compara il valore del registro con 0
Jz shortloc_40102b	Se la condizione precedente avviene effettuerà il jump in quella locazione specifica (40102b)
PUSH offset aSuccessInternet	Sposta nello stack l'indirizzo della stringa aSuccessInternet
CALL sub_40117F	Effettua una chiamata all'indirizzo sub_40117F
ADD ESP, 4:	addiziona il valore 4 a ESP
MOV EAX, 1:	copia il valore 1 nel registro EAX
JMP short loc_40103A:	salto incondizionato all'indirizzo 40103A
loc_40102B: "Error 1.1: No Internet\n":	punto di arrivo del jump precedente,il programma risponde con "Error 1.1: No Internet\n":
PUSH offset aError1_1NoInte:	sposta nello stack l'indirizzo di errore
CALL sub_40117F:	effettua una chiamata alla subroutine 40117F
ADD ESP, 4:	aggiunge il valore 4 nel registro ESP
XOR EAX, EAX:	imposta EAX a 0
loc_40103A:	punto di arrivo del Jz shortloc_40102b
MOV ESP, EBP:	ripristina lo stack
POP EBP:	ripristina il valore originale del registro di base
RETN:	ritorno dalla funzione
sub_401000 endp	fine del blocco di codice