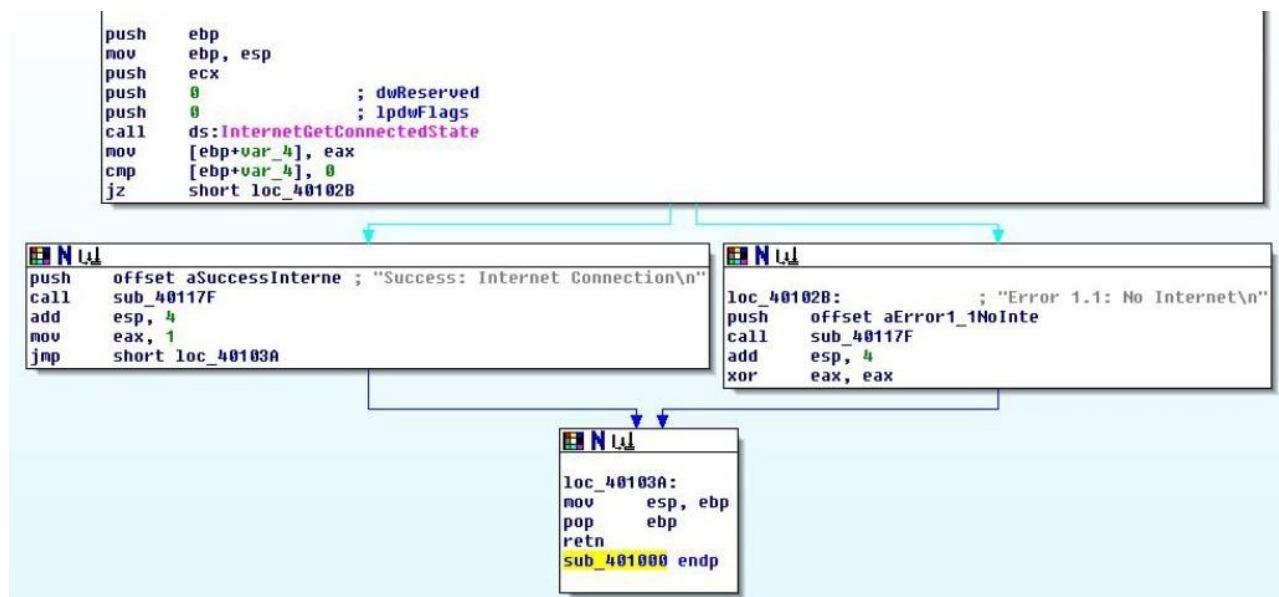


MALWARE ANALYSIS

Analisi del malware *Malware_U3_W2_L5.exe*

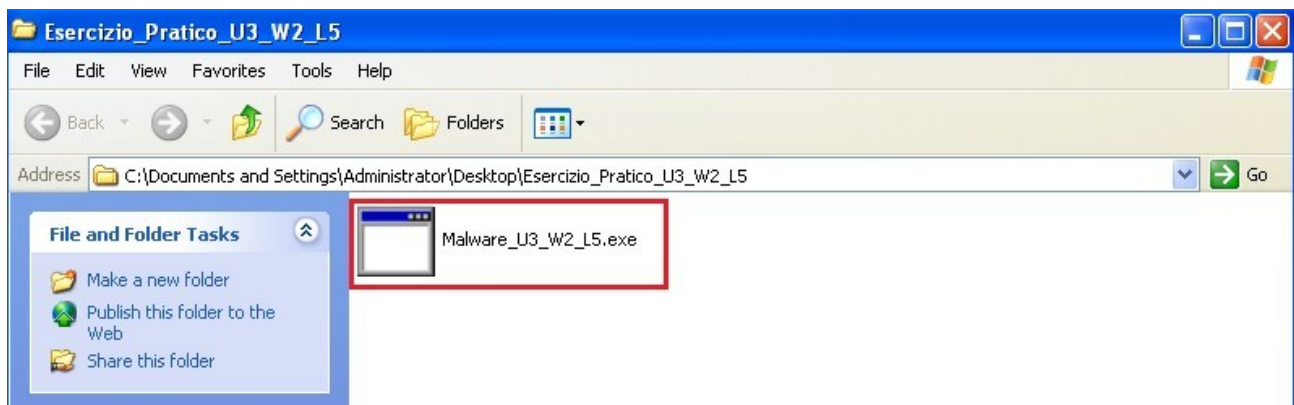
Traccia

1. Quali librerie vengono importate dal file eseguibile? Fare anche una descrizione
2. Quali sono le sezioni di cui si compone il file eseguibile dal malware? Fare anche una descrizione.
3. Identifica i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
4. Ipotizzare il comportamento della funzionalità implementata
5. Bonus



1. Identificazione delle librerie importate dal file eseguibile

Il file eseguibile di test **Malware_U3_W2_L5.exe**; lo rintracciamo all'interno del percorso **C:\Documents and Settings\Administrator\Desktop\Esercizio_Pratico_U3_W2_L5 Analysis_Final** c



Analisi Statica Basica

L'**analisi dei malware** è l'insieme di competenze e tecniche che permettono ad un professionista della sicurezza informatica di studiare accuratamente il comportamento del suddetto Malware, al fine di rimuoverlo correttamente dal sistema.

Questo tipo di analisi consiste nell'esaminare il file senza tener conto altri fattori. Lo scopo di questa analisi è confermare la natura malevola di un file e fornire informazioni generiche e le sue funzionalità.

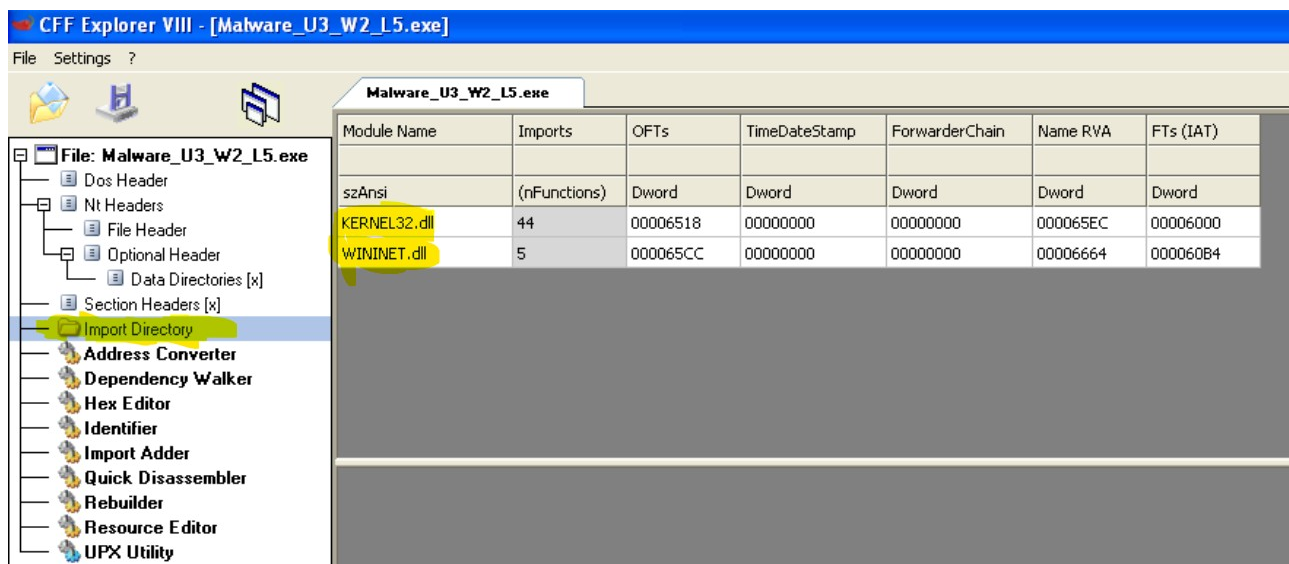
Per l'analisi utilizziamo CFF Explorer nella versione VIII, programma si occupa di l'header del formato PE. Difatti Windows utilizza questo formato per i file eseguibili, e all'interno troviamo le informazioni necessarie per gestire i vari file.

All'interno dell'header troviamo

- L'elenco delle librerie più importate e le funzioni eseguibili
- Eventuali funzioni esportate
- Frammenti del sistema operativo

Il programma restituisce alcune informazioni di riepilogo tra cui le dimensioni la data e l'hash nei formati MD5 e SHA-1, che si rivelano utili ai fini di una successiva ricerca su un database online come quello di VirusTotal, per ottenere più informazioni circa il comportamento del malware in oggetto.

Spostandoci nella sezione "**Import Directory**", possiamo identificare le librerie importate dal malware:



KERNEL32.dll : libreria piuttosto comune che contiene le funzioni principali per il sistema operativo, es. manipolazione del file, gestione della memori

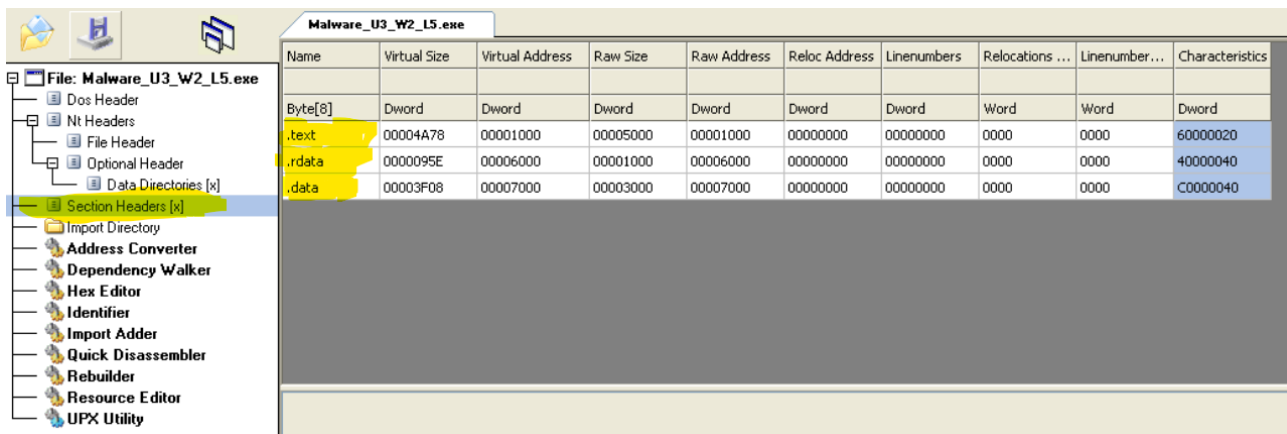
WININET.dll : libreria che contiene le funzioni per l'implementazione di alcuni protocolli di rete come HTTP, FTP, NTP ecc..

00006864	00006864	0131	GetOEMCP
00006870	00006870	02BB	VirtualAlloc
00006880	00006880	01A2	HeapReAlloc
0000688E	0000688E	013E	GetProcAddress
000068A0	000068A0	01C2	LoadLibraryA
000068B0	000068B0	011A	GetLastError
000068C0	000068C0	00AA	FlushFileBuffers
000068D4	000068D4	026A	SetFilePointer
00006950	00006950	001B	CloseHandle

Tra le funzioni richieste, notiamo la presenza di **InternetGetConnectedState**, che ha lo scopo di verificare se una macchina ha accesso ad Internet: da ciò si può ipotizzare che il malware svolga alcune operazioni tramite una connessione Internet.

Esercizio n2

Quali sono le sezioni di cui si compone il file eseguibile dal malware? Fare anche una descrizione.



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
.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

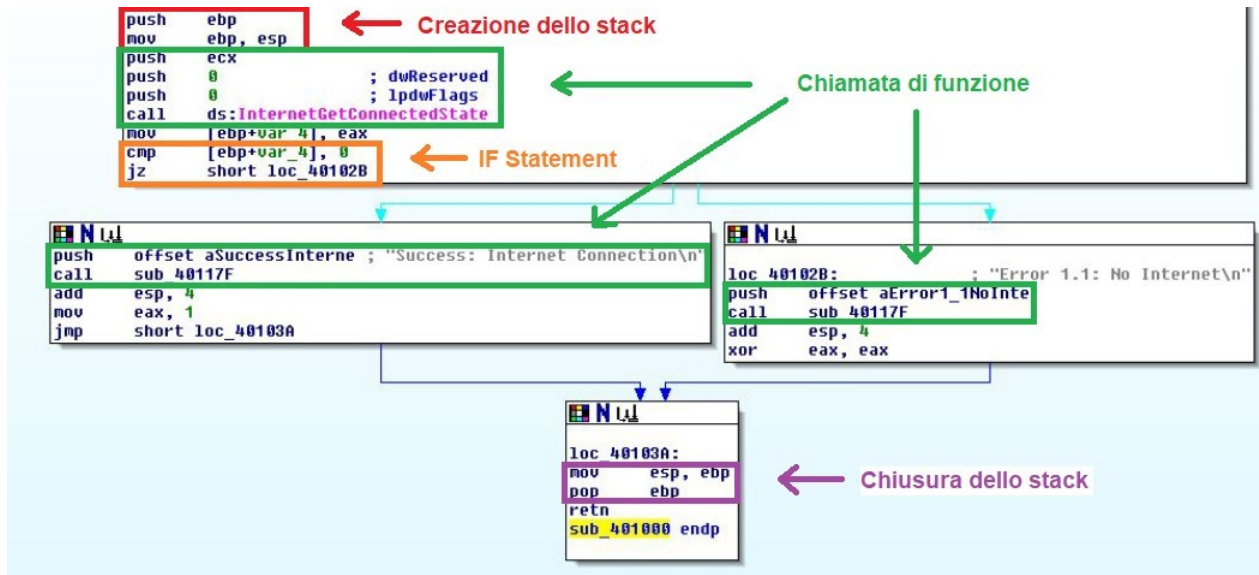
.text contiene istruzioni che la CPU esegue quando il software viene avviato.

.rdata contiene informazioni sulle librerie e le funzioni importate ed esportate dall'eseguibile.

.data contiene dati/**variabili globali** del programma eseguibile, che devono essere quindi disponibili dal programma. (una variabile è globale quando non è definita all'interno di una funzione, ma è globalmente dichiarata.)

Esercizio n3

Identifica i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti) All'interno delle istruzioni in linguaggio Assembly, è possibile identificare i costrutti rappresentati nella seguente immagine:



Creazione dello stack in rosso
Chiamata di funzione in verde
Istruzioni IF. In arancione
Chiusura dello stack in viola

Esercizio n4

Il linguaggio Assembly mostrano che il programma si occupa della creazione di uno stack per le variabili locali difatti notiamo la presenza dei due puntatori, EBP ed ESP che puntano rispettivamente alla base ed alla cima dello stack.

Successivamente, il programma si occupa di verificare, attraverso la chiamata della funzione **InternetGetConnectedState**, se la macchina vittima ha accesso ad Internet il malware può usufruire della connessione Internet per eseguire alcune operazioni dannose nei confronti dell'utente

Esercizio n5

Push ebp: Viene "spinto" il registro Extended Base Pointer sulla cima dello stack

mov ebp, esp : Viene assegnato il valore del registro dell'Extended, Stack pointer al registro dell'Extended Base Pointer

push ecx : Tramite l'istruzione push, viene posto il valore inserito nel registro "ecx" in cima allo stack

push 0 Mette il valore 0

push 0 Mette il valore 0

call ds: InternetGetConnectedState : Esegue una chiama alla funzione "InternetGetConnectedState" che verifica lo stato di connettività del sistema locale.

mov [ebp+var_4], eax Confronta il valore di memoria [EBP+var_4]

jmp short loc_40103A : Salta a loc_40103A

loc_40102B "Error 1.1: No Internet\n"

push offset aError1_1NoInte

call sub 40117F Viene chiamata la funzione all'indirizzo di memoria 40117F

add esp, 4 : Somma il valore 4 a quello contenuto nel registro ESP

xor eax, eax : Viene usata l'istruzione XOR per iniziare da 0 il registro EAX

loc_40103A Indica la location di memoria 40103A

mov esp, ebp Viene copiato il contenuto del registro EBP nel registro ESP

pop ebp : Viene rimosso il contenuto dal registro EBP dallo stack

retn : Indica un ritorno del programma al termine della procedura

sub 401000 endp : Indica la fine della procedura all'indirizzo di memoria 401000

