



PROGETTO S10/L5



MALWARE ANALYSIS

L'esercizio della settimana prevede di effettuare l'analisi statica basica di un malware e l'analisi di un codice assembly. L'analisi statica basica sarà effettuata con l'utilizzo di CFF explorer,

La traccia da seguire è la seguente:

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, rispondere ai seguenti quesiti:

1. Quali librerie vengono importate dal file eseguibile?
2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

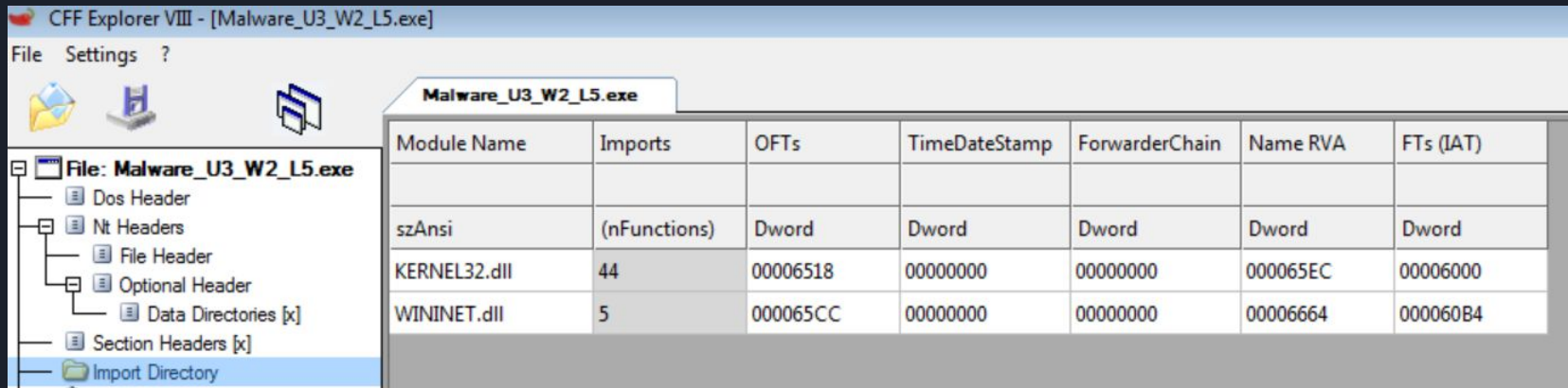
3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
4. Ipotezzare il comportamento della funzionalità implementata
5. BONUS : fare tabella con significato delle singole righe di codice assembly

LIBRERIE IMPORTATE

Per funzionare, il malware ha bisogno di importare delle librerie specifiche per raggiungere lo scopo dell'hacker che lo ha utilizzato. Apriamo il malware senza eseguirlo con CFF explorer per effettuare l'analisi statica del malware e trovare le informazioni di nostro interesse. Le librerie utilizzate come si vede nell'immagine sono:

KERNEL32.dll

WININET.dll



CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

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



LIBRERIE

Vediamo nel dettaglio a cosa servono queste librerie:

KERNEL32.dll : contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria. Kernel32.dll si trova nella cartella C:\Windows\System32 o talvolta nella cartella Temp di Windows.

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

SEZIONI DEL FILE

Oltre alle librerie, la traccia richiede le sezioni utilizzate dal file e una breve descrizione di ognuna. Le sezioni utilizzate sono

.text

.rdata

.data

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	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



SEZIONI UTILIZZATE

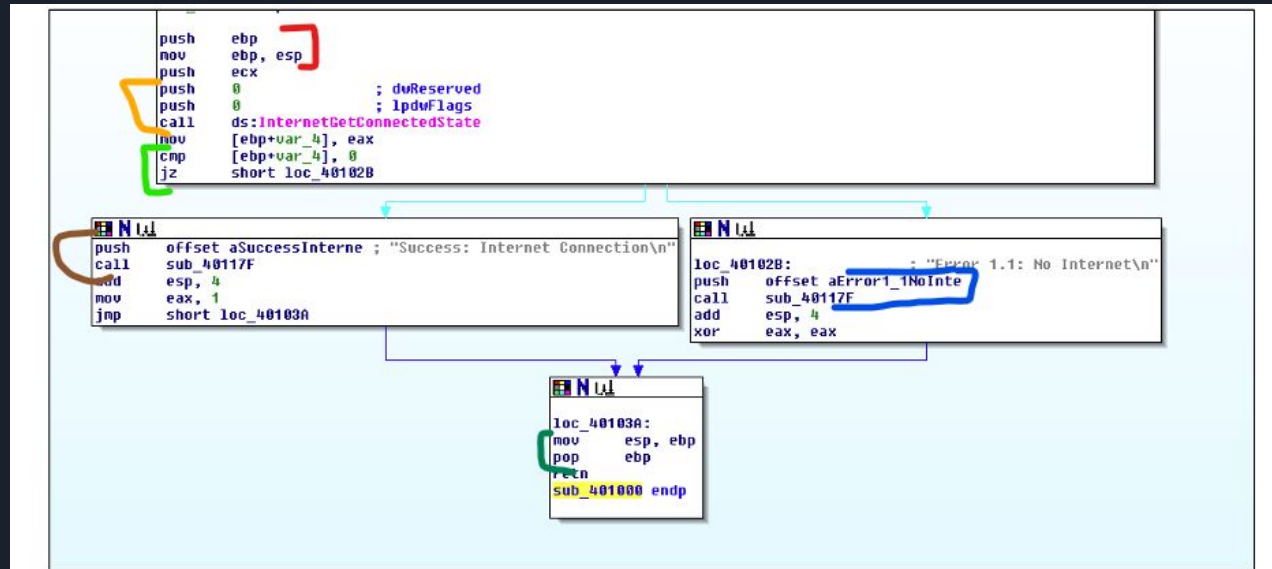
.text : contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.

.rdata : include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer.

.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.

COSTRUTTI UTILIZZATI IN ASSEMBLY

I costrutti individuati sono 6 e sono stati evidenziati nell'immagine.





COSTRUTTI

1. **Dichiarazione stack** : il comando “push ebp” aggiunge il registro EBP in cima allo stack e con “mov ebp, esp” sposta il registro ESP in EBP
2. **Chiamata di funzione**: I parametri sono passati sullo stack tramite le istruzioni push e successivamente chiama la funzione internetgetconnectedstate e ne controlla con un «if» il valore di ritorno. Se il valore è diverso da zero la connessione è attiva
3. **If** : La funzione “cmq” (compare) mette a confronto 2 valori e determina la condizione. evidenzia anche che JZ viene eseguita se [flag di Zero]=1
Salta nell’indirizzo short_loc40102B se il risultato di Jump zero è appunto 0
4. **Chiamata di funzione** : Il comando push “offset aSuccessInternet” serve caricare la variabile nello stack e call “sub_40117F” per chiamare la funzione specifica
5. **Chiamata di funzione** : Il comando push “offset aError1_NoInte” serve caricare la variabile nello stack e call “sub_40117F” per chiamare la funzione
6. **Chiusura dello stack** : il comando “mov esp, ebp” sposta il registro ebp nel registro esp. Con pop si elimina lo stack che ha finito il suo compito



SCOPO DEL PROGRAMMA ASSEMBLY

Si ipotizza che lo scopo del programma sia quello di cercare di stabilire una connessione con la macchina vittima.

Possiamo vedere quando il malware chiama `InternetGetConnectedState` e ne controlla con un «if» il valore di ritorno. Se il valore di ritorno (return) della funzione è diverso da 0, allora vuol dire che c'è una connessione attiva.