

S10/L5

ANALISI STATICA : UN APPROCCIO PRATICO

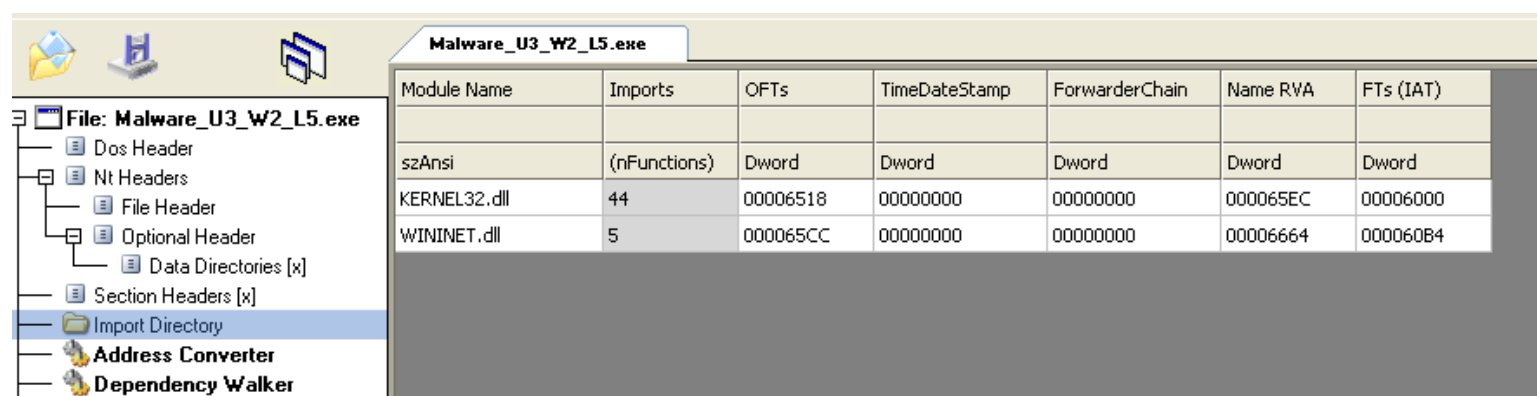
Presented By
COLOMBO FEDERICO

L'esercizio di oggi consiste nell'analizzare il Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5» tramite un'analisi di tipo statica basica, quindi senza avviare il malware.

L'esercizio prevede anche un'analisi di un codice in Assembly che vedremo più avanti.

ANALISI STATICA BASICA

Iniziamo l'esercizio aprendo il malware con il programma CFF Explorer e analizziamo quali librerie importa il nostro malware:



Module Name	Imports	OFs	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

Possiamo notare l'importazione di due librerie:

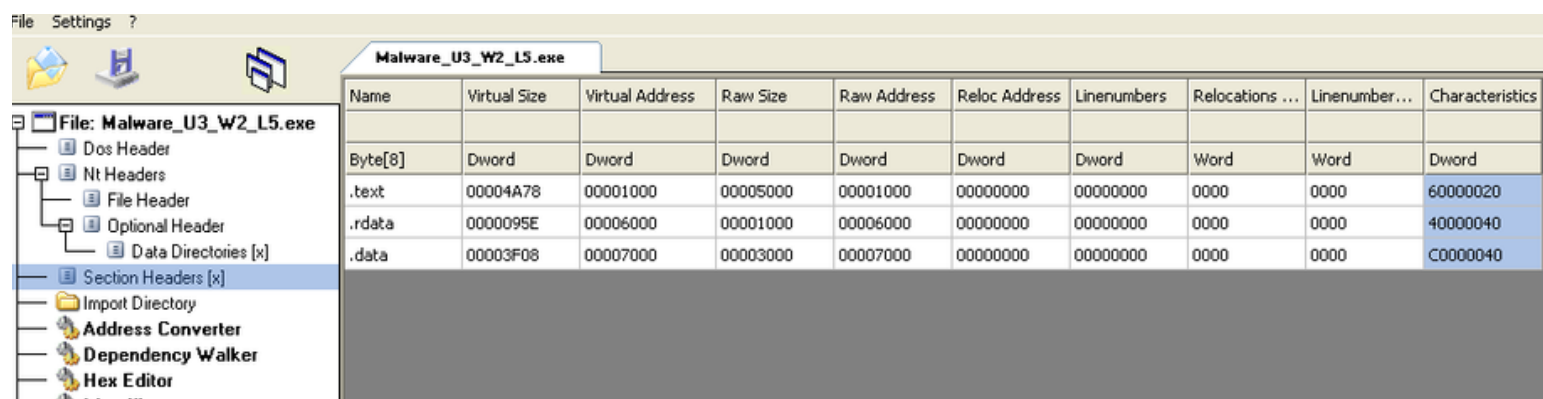
- **Kernel32.dll**

La libreria kernel32.dll è una parte fondamentale del sistema operativo Microsoft Windows. Contiene una serie di funzioni che sono utilizzate per gestire operazioni di sistema, gestire la memoria, controllare i file, creare processi e molto altro ancora.

- **Wininet.dll**

La libreria wininet.dll è una libreria di Microsoft Windows, si concentra principalmente sulla fornitura di funzionalità legate alla connettività di rete e alle operazioni Internet, come l'implementazione di protocolli di rete come HTTP, FTP ed NTP.

Passiamo ora all'analisi delle sezioni di cui si compone il file eseguibile del malware:



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

Le sezioni di cui è composto sono:

- **.text**

Contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato.

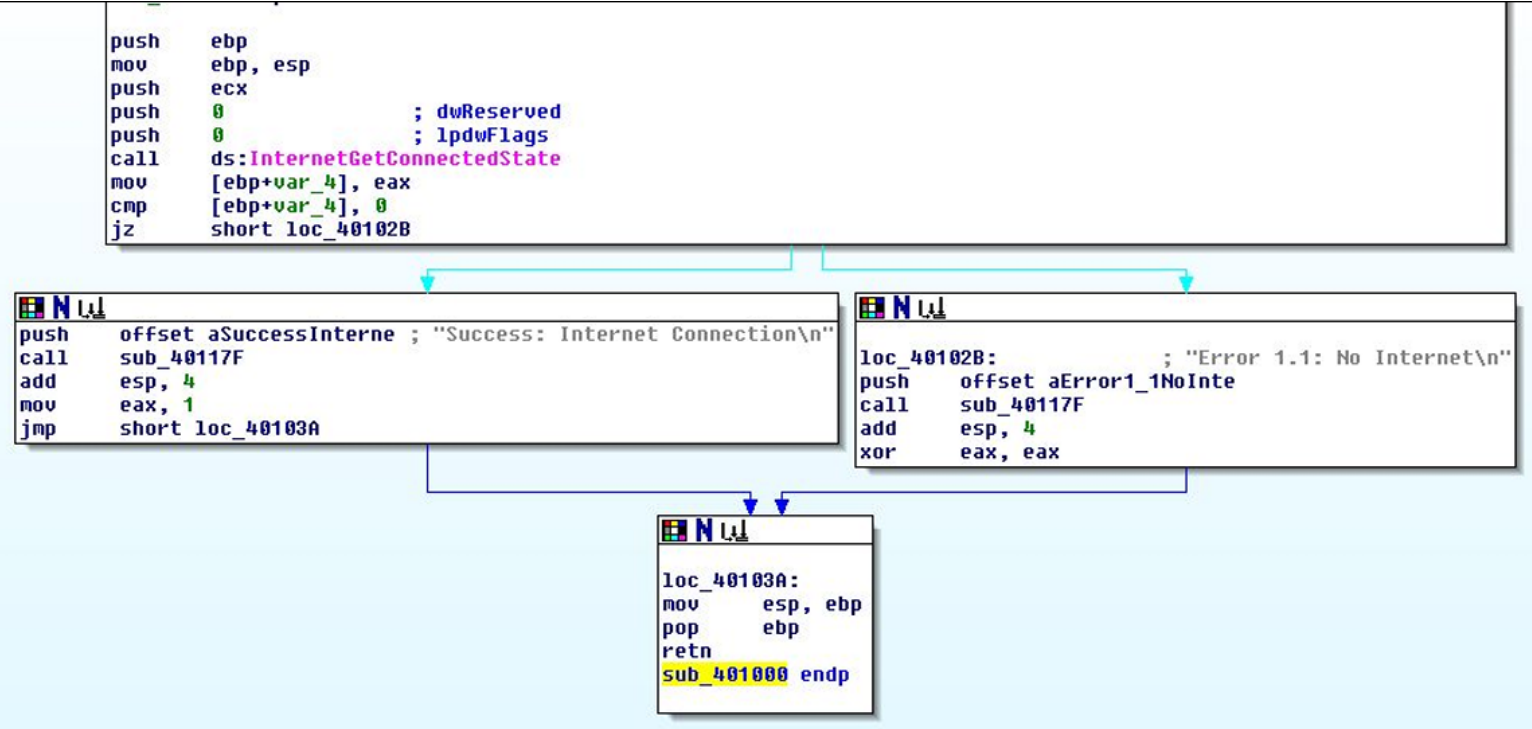
- **.rdata**

Contiene le informazioni delle librerie e le funzioni importate ed esportate dall'eseguibile.

- **.data**

Contiene i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.

Passiamo ora all'analisi del seguente codice Assembly:



```
push    ebp
mov     ebp, esp
```

In questo blocco vengono racchiuse le righe necessarie alla creazione dello stack.

```
push    ecx
push    0           ; dwReserved
push    0           ; lpdwFlags
call    ds:InternetGetConnectedState
```

Chiamata della funzione “InternetGetConnectedState” tramite l’istruzione push.

```
mov     [esp+var_4], 0  
cmp     [ebp+var_4], 0  
jz      short loc_40102B
```

In questo blocco inizia il ciclo IF-Style

```
mov     esp, ebp  
pop     ebp
```

Infine nell'ultimo blocco abbiamo la rimozione dello stack iniziale.

Con questo frammento di codice possiamo ipotizzare che il malware che stiamo andando ad analizzare cerca di avviare una connessione ad Internet tramite la funzione **InternetGetConnectedState**.

Attraverso poi un ciclo IF avviene il controllo sulla funzione, se c'è la presenza di una connessione Internet il malware avvierà il ciclo a sinistra, al contrario se non trova nessuna connessione avvierà il ciclo a destra.

Con le poche informazioni disponibili possiamo ipotizzare che questo malware sia una backdoor.