

Progetto assembly analisi malware

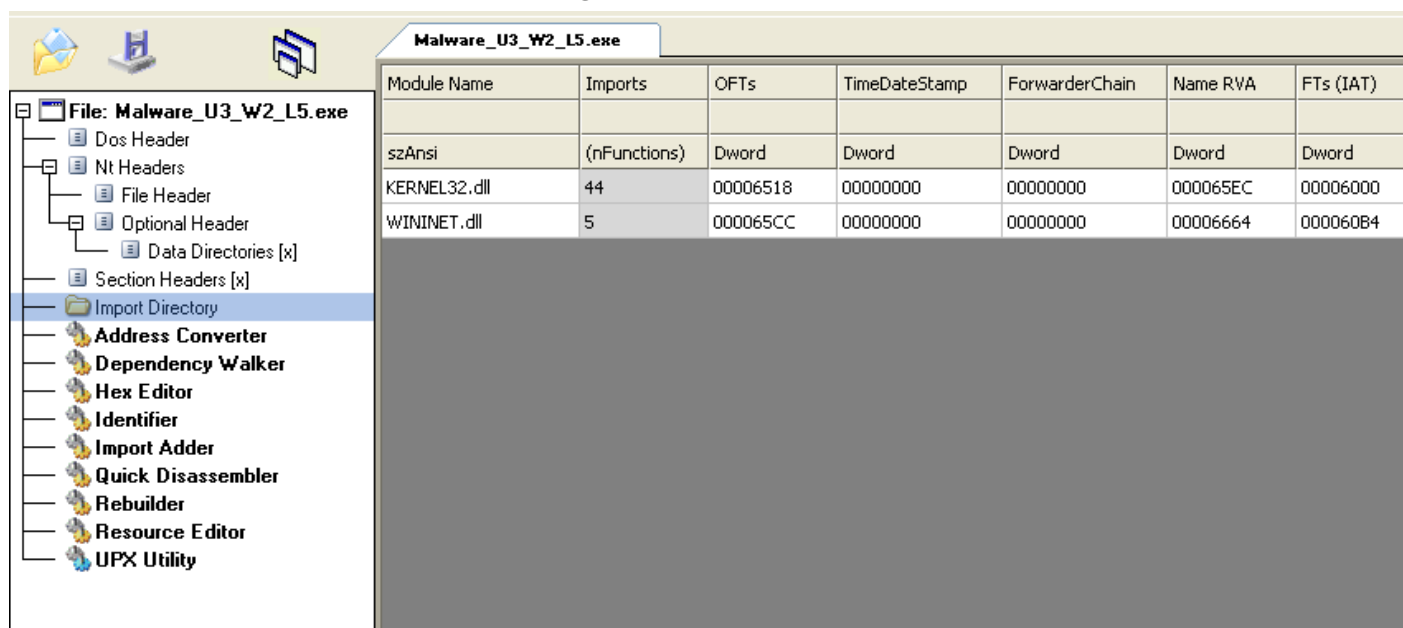
In questo esercizio andremo ad analizzare un malware con un'analisi statica basica (analizza il malware senza eseguirlo). Le domande a cui dobbiamo rispondere sono:

- 1) Quali librerie sono importate dall'eseguibile?
- 2) Quali sono le sessioni di cui si compone il file eseguibile?
- 3) Identificare i costrutti noti sulla slide 3
- 4) Ipotezzare il comportamento del malware.

Analisi statica basica

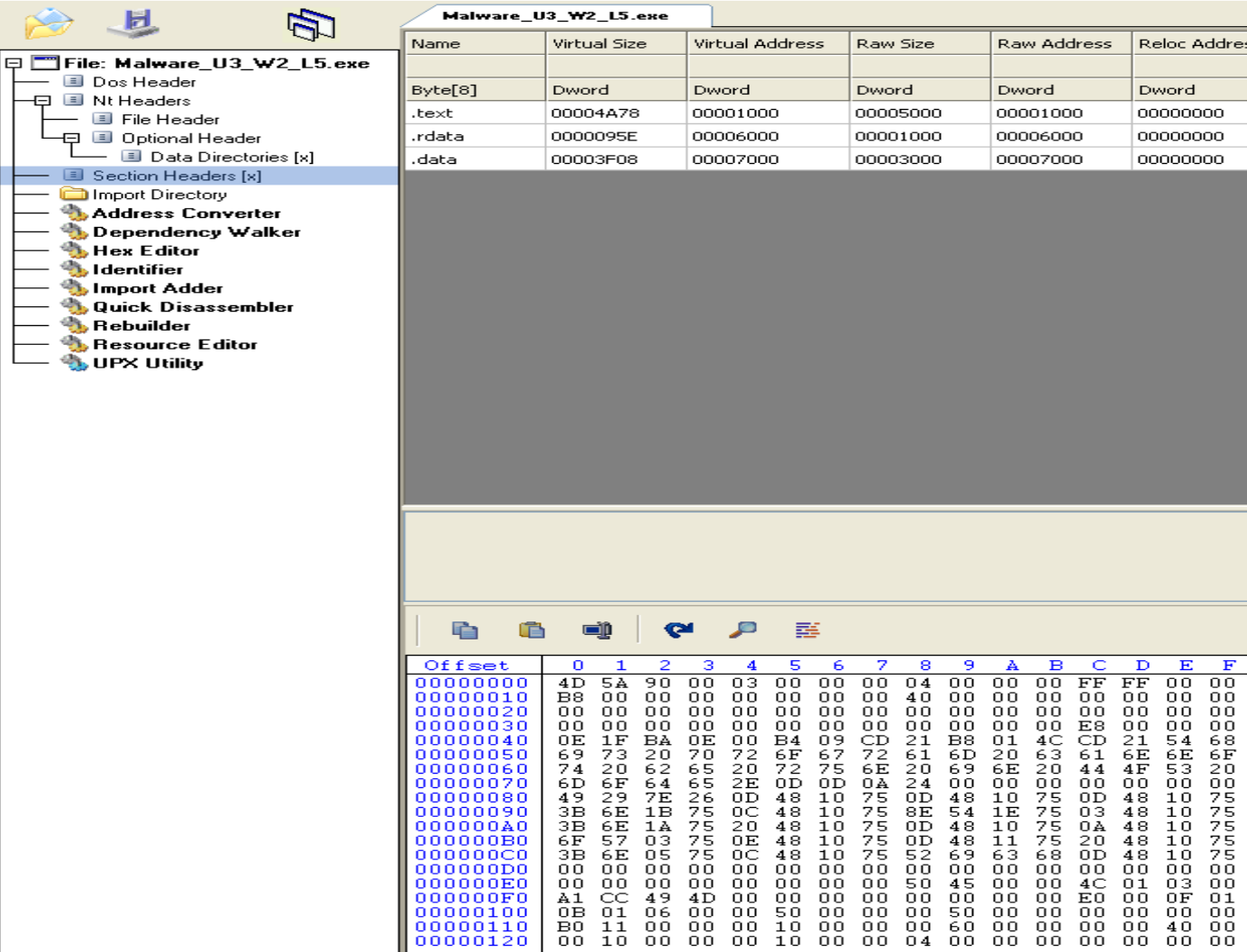
Per rispondere al 1 e 2 quesito ho aperto la nostra macchina virtuale e lanciato il malware con **CFF Explorer** (tool utilizzato per analisi dinamica per i file eseguibili).

Nello specifico ho cercato due parti fondamentali per l'analisi. Le librerie e le sezioni dell'eseguibile.



Nell'immagine vediamo che il pannello a sinistra da cliccare è import directory e le librerie sono:

- 1) **kernel32.dll**: La libreria che agisce sul sistema operativo come ad esempio manipolazione dei file e la gestione della memoria
- 2) **Wininet.dll**: Si occupa dell'implemento dei protocolli di rete HTTP, FTP, NTP



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

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	E8	00	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00
00000080	49	29	7E	26	0D	48	10	75	0D	48	10	75	0D	48	10	75
00000090	3B	6E	1B	75	0C	48	10	75	8E	54	1E	75	03	48	10	75
000000A0	3B	6E	1A	75	20	48	10	75	0D	48	10	75	0A	48	10	75
000000B0	6F	57	03	75	0E	48	10	75	0D	48	11	75	20	48	10	75
000000C0	3B	6E	05	75	0C	48	10	75	52	69	63	68	0D	48	10	75
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000E0	00	00	00	00	00	00	00	00	50	45	00	00	4C	01	03	00
000000F0	A1	CC	49	4D	00	00	00	00	00	00	00	00	E0	00	0F	01
00000100	0B	01	06	00	00	50	00	00	00	50	00	00	00	00	00	00
00000110	B0	11	00	00	00	10	00	00	00	60	00	00	00	00	40	00
00000120	00	10	00	00	00	10	00	00	04	00	00	00	00	00	00	00

Nell'immagine sopra vediamo le sezioni, il pannello a sinistra da cliccare è section headers:

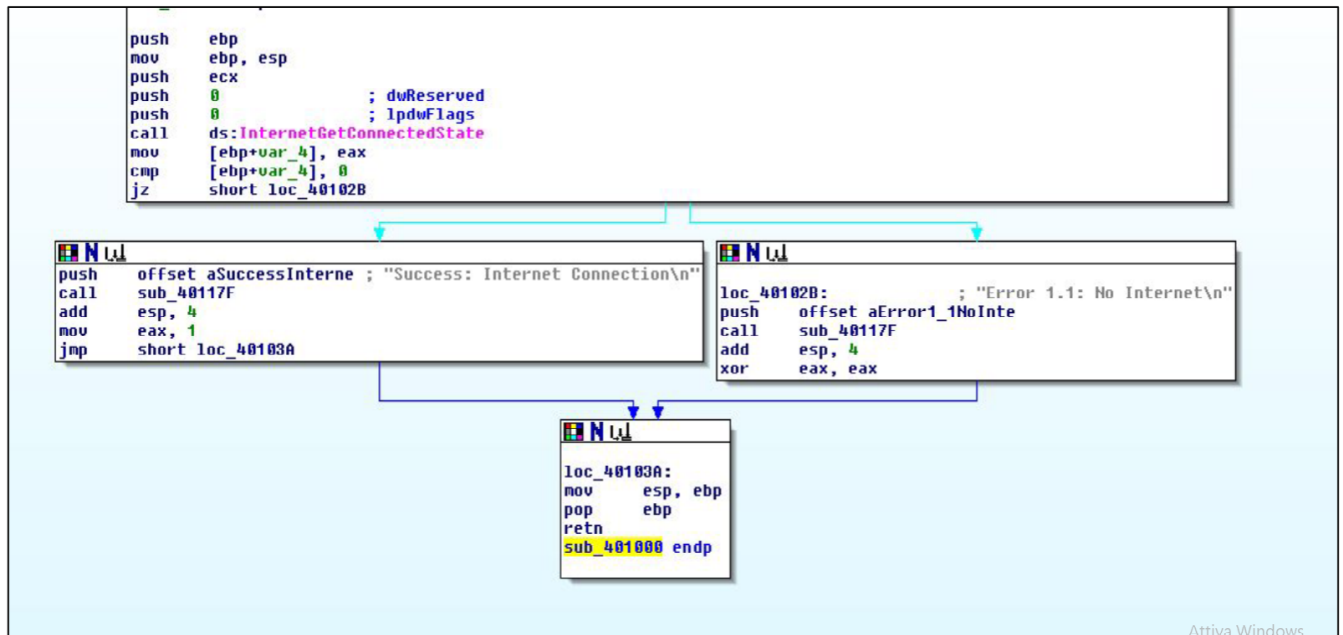
.TEXT: contiene le righe di codice che eseguirà la CPU. Generalmente è l'unica sezione del file eseguibile dalla CPU.

.RDATA: Include informazioni sulle librerie importate ed esportate dall'eseguibile.

.DATA: Contiene dati e variabili globali del file eseguibile. Le variabili globali sono accessibili da qualsiasi funzione dell'eseguibile.

Analisi costrutti noti

Una volta finita la prima parte andiamo a prendere l'immagine della slide 3 dell'esercizio per identificare i costrutti noti.



Dividiamo il codice in macroblocchi.

- 1) Da push **EBP** a mov è la creazione dello stack
- 2) Push **ecx** a call Dichiarati i parametri messi sullo stack e chiama la funzione **InternetGetConnectedState**.
- 3) Da **cmp** a **jz** è l'inizio del **CICLO IF**. Il ciclo if compara la variabile **eax** se è uguale a 0 o ad 1. Se il valore di ritorno è 0 la connessione è attiva, altrimenti il ciclo salta nella parte destra del diagramma connessione disattiva.
- 4) Parte sinistra da **Push** a **JMP**. La connessione internet è attiva.
- 5) Parte destra da **loc_40102B** a **XOR**. La connessione internet è disattiva.
- 6) Da **loc_40103A** a **Sub_401000 endp** Chiude il ciclo e ripulisce lo stack.

Ipotesi

Date le librerie utilizzate e il codice assembly possiamo dedurre che il malware prova a connettersi ad internet. Due sono le mie ipotesi o che si tratti di un **Downloader** o che si tratti di una **backdoor**.