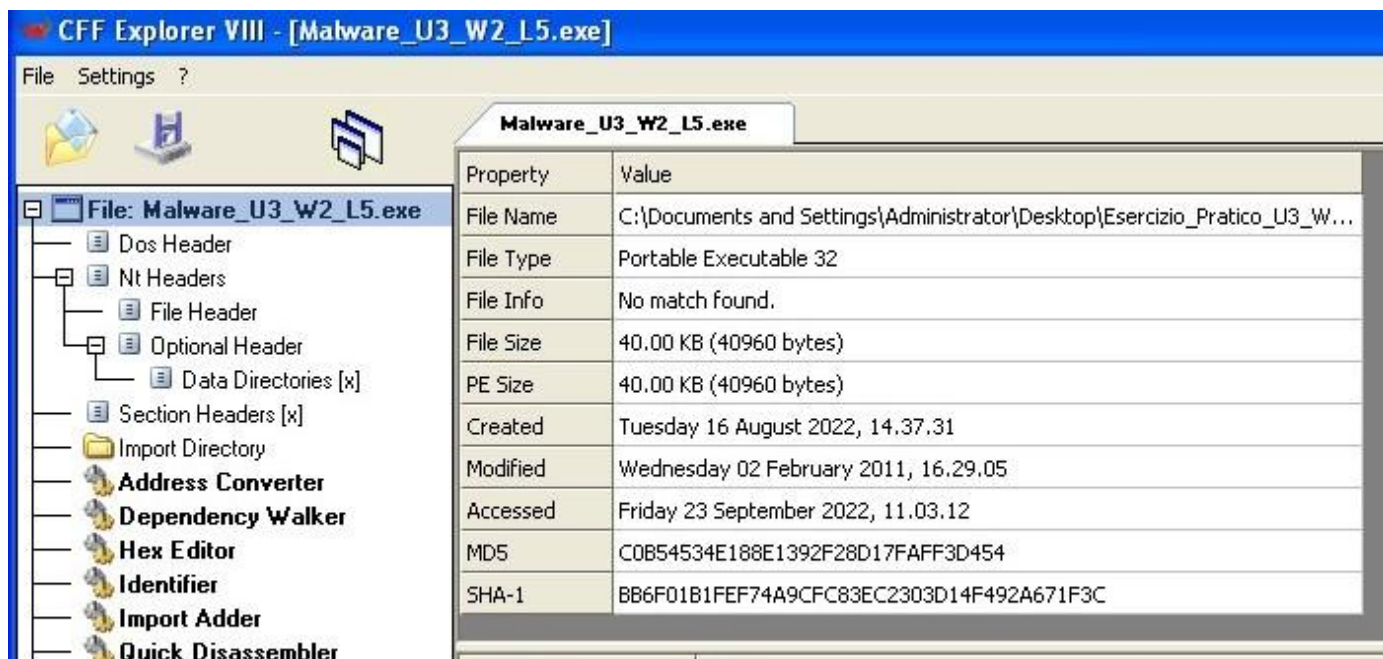


Malware Analysis

23.09.22

Oggi andremo a fare un malware analysis (*=insieme di tecniche che permettono ad un analista di indagare su un malware, così da capirne il comportamento e rimuoverlo.*) che si trova all'interno della cartella "Esercizio_Pratico_U3_W2_L5". Per prima cosa andremo ad aprire il nostro Malware (Malware_U3_W2_L5) con **CFF Explorer**.



CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

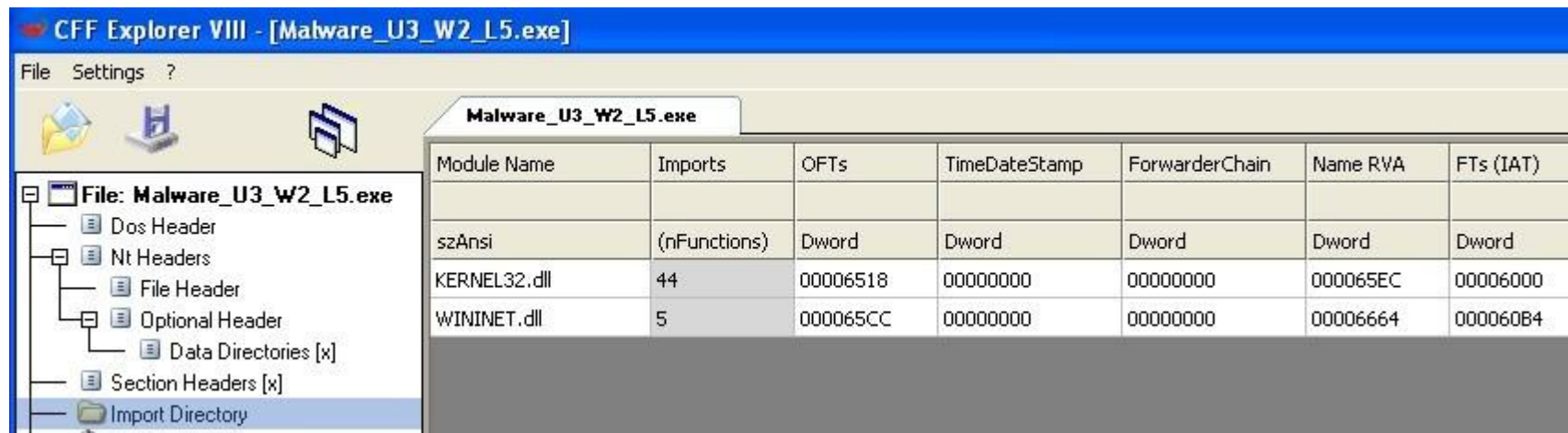
Malware_U3_W2_L5.exe

Property	Value
File Name	C:\Documents and Settings\Administrator\Desktop\Esercizio_Pratico_U3_W...
File Type	Portable Executable 32
File Info	No match found.
File Size	40.00 KB (40960 bytes)
PE Size	40.00 KB (40960 bytes)
Created	Tuesday 16 August 2022, 14.37.31
Modified	Wednesday 02 February 2011, 16.29.05
Accessed	Friday 23 September 2022, 11.03.12
MD5	C0B54534E188E1392F28D17FAFF3D454
SHA-1	BB6F01B1FEF74A9CFC83EC2303D14F492A671F3C

File: Malware_U3_W2_L5.exe

- Dos Header
- Nt Headers
 - File Header
 - Optional Header
 - Data Directories [x]
- Section Headers [x]
- Import Directory
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler

Dopo aver aperto il nostro malware, andremo su “**Import Directory**” dove potremmo vedere le librerie importate dal nostro malware.



In questo caso le librerie importate sono: “**Kernel32.dll**” e “**Wininet.dll**”

- La libreria **Kernel32.dll** contiene le funzioni principali per interagire con il sistema operativo, come la gestione della memoria e la manipolazione dei file.
- La libreria **Wininet.dll** contiene le funzioni per implementare alcuni protocolli di rete (FTP, HTTP etc.)

Dopo aver individuato le nostre librerie andremo a capire le sezioni che compongono il malware. Per vedere queste sezioni andremo su "Section Headers".

Malware_U3_W2_L5.exe

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

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZyy..
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	00E.....
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	!! Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program.canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t.be.run.in.DOS.
00000070	6D	6F	64	65	2E	0D	00	0A	24	00	00	00	00	00	00	00	mode...\$.....
00000080	49	29	7E	26	0D	48	10	75	0D	48	10	75	0D	48	10	75	I)%..Htu.Htu.Htu
00000090	3B	6E	1B	75	0C	48	10	75	8E	54	1E	75	03	48	10	75	:ntu.Htu.Htu.Htu
000000A0	3B	6E	1A	75	20	48	10	75	0D	48	10	75	0A	48	10	75	:ntu.Htu.Htu.Htu

Dove scopriremo quindi che le sezioni che compongono il nostro malware sono
“.text,.rdata,.data”

- - **.text:** Questa sezione contiene le righe di codice, e quindi le istruzioni, che la CPU andrà ad eseguire una volta che il software verrà avviato. (Unica sezione eseguita dalla CPU, le altre sezioni hanno al loro interno dati o info di supporto)
- **.rdata:** Questa sezione contiene le informazioni delle librerie e le varie funzioni esportate e importate dal malware.
- **.data:** Questa sezione contiene le variabili globali(=variabile non definita all'interno di un contesto di una funzione ma è invece dichiarata globalmente, è quindi accessibile da qualsiasi funzione del malware.) e i dati del malware che devono essere disponibili da qualsiasi parte del programma.

Parte 2 esercizio

Nella seconda parte di esercizio andremo ad identificare i costrutti e ipotizzare il comportamento della funzione implementata.


```
push    ebp
mov     ebp, esp
push    ecx
push    0          ; dwReserved
push    0          ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

```
loc_40102B:          ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

Crea lo stack




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

Chiamata di funzione.
Il push chiama
parametri messi sullo
stack



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

Ciclo if



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

Output ciclo if:

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

pusha il fatto che la connessione internet
non sia attiva

```
loc_40102B:                                ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

pusha il fatto che la connessione internet
non sia attiva

Se la funzione ha un valore di ritorno differente da 0, vuol dire che la connessione è attiva



```
loc_40103A:  
mov     esp, ebp  
pop     ebp  
retn  
sub_401000 endp
```

Chiude il ciclo e ripulisce lo stack.

Ipotesi di comportamento del malware

Il malware chiama la funzione “internetgetconnectedstate” che controlla con un if il valore di ritorno. Se la funzione ha un valore di ritorno differente da 0, vuol dire che la connessione è attiva. Da qui possiamo quindi dedurre che il malware si tratti di un downloader. Tenta di accedere alla connessione internet per scaricare altri malware.