

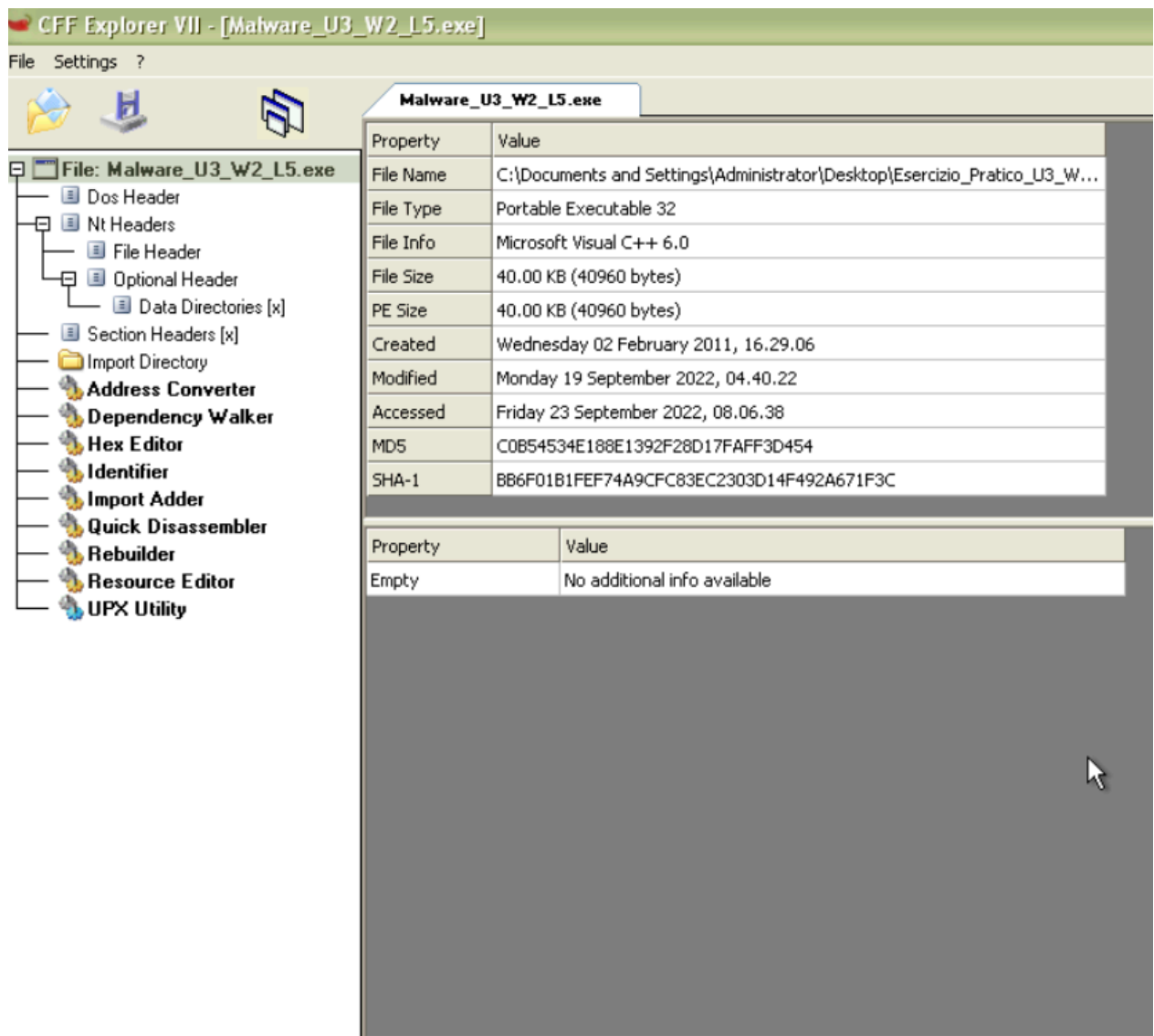
ANALISI MALWARE

L'esercizio di oggi ci richiede di fare una analisi statica basica del malware presente sulla nostra macchina windows xp.

I punti che ci chiede il progetto di oggi sono di andare a controllare quali librerie vengono importate dal file eseguibile e quali sono le sezioni di cui si compone il file eseguibile del malware.

Per analizzare staticamente il malware ci avvaleremo del tool già presente nella nostra macchina windows xp e il tool in questione si chiama **CFF EXPLORER versione VII**, che ci permette di analizzare l'header del **PE** (Portable Executable)

Il primo passo da fare è avviare il tool e inserire il malware al suo interno cosi da ricavare le informazioni che ci servono



SEZIONI DI CUI SI COMPONE IL FILE ESEGUIBILE DEL MALWARE

Spostandoci nella **SECTION HEADERS** possiamo vedere le sezioni di cui è composto il file eseguibile.

In questo caso le sezioni sono:

- .TEXT:** contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dal 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
- .DATA:** contiene tipicamente i dati/le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma

CFF Explorer VII - [Malware_U3_W2_L5.exe]

File Settings ?

Malware_U3_W2_L5.exe

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
- Rebuilder
- Resource Editor
- UPX Utility

| Name | Virtual Size | Virtual Address | Raw Size | Raw Address | Reloc Address | Linenumbers | Relocations ... | Linenumber... | Characteristics |
|----------|--------------|-----------------|----------|-------------|---------------|-------------|-----------------|---------------|-----------------|
| 00000230 | 00000238 | 0000023C | 00000240 | 00000244 | 00000248 | 0000024C | 00000250 | 00000252 | 00000254 |
| 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 |

This section contains:

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | Ascii | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------------------|
| 00000000 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 09 | 1C | 40 | 00 |!@. |
| 00000010 | 64 | 35 | 40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | AE | 1C | 40 | 00 | d5@.....!@. |
| 00000020 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000030 | 45 | 72 | 72 | 6F | 72 | 20 | 31 | 2E | 31 | 3A | 20 | 4E | 6F | 20 | 49 | 6E | 00 | Error.1.1: No. In |
| 00000040 | 74 | 65 | 72 | 6E | 65 | 74 | 0A | 00 | 53 | 75 | 63 | 63 | 65 | 73 | 73 | 3A | 00 | ternet..Success: |
| 00000050 | 20 | 49 | 6E | 74 | 65 | 72 | 6E | 65 | 74 | 20 | 43 | 6F | 6E | 6E | 65 | 63 | 00 | .Internet.Connec |
| 00000060 | 74 | 69 | 6F | 6E | 0A | 00 | 00 | 00 | 45 | 72 | 72 | 6F | 72 | 20 | 32 | 2E | 00 | tion....Error.2. |

LIBRERIE IMPORTATE DAL FILE ESEGUIBILE

Nella foto sotto possiamo vedere le librerie importate dal file eseguibile.

Le librerie importate sono:

KERNEL32.dll: libreria piuttosto comune che contiene le funzioni principali per interagire con il sistema operativo, ad esempio : manipolazione dei file, la gestione della memoria

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

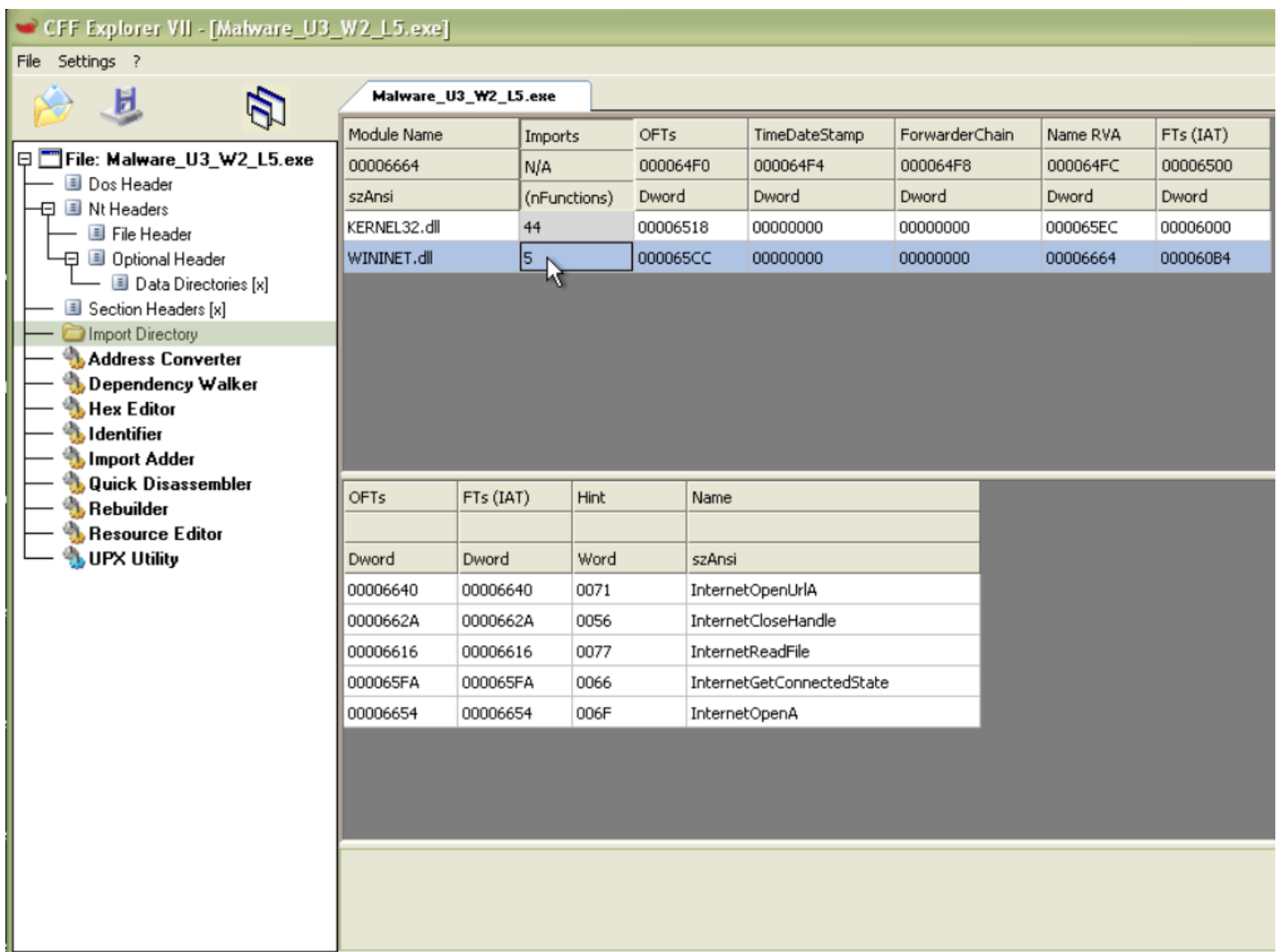
CFF Explorer VII - [Malware_U3_W2_L5.exe]

File Settings ?

Malware_U3_W2_L5.exe

| Module Name | Imports | OFTs | TimeDateStamp | ForwarderChain | Name RVA | FTs (IAT) |
|--------------|--------------|----------|---------------|----------------|----------|-----------|
| 000065EC | N/A | 000064DC | 000064E0 | 000064E4 | 000064E8 | 000064EC |
| szAnsi | (nFunctions) | Dword | Dword | Dword | Dword | Dword |
| KERNEL32.dll | 44 | 00006518 | 00000000 | 00000000 | 000065EC | 00006000 |
| WININET.dll | 5 | 000065CC | 00000000 | 00000000 | 00006664 | 000060B4 |

| OFTs | FTs (IAT) | Hint | Name |
|----------|-----------|------|---------------------|
| Dword | Dword | Word | szAnsi |
| 000065E4 | 000065E4 | 0296 | Sleep |
| 00006940 | 00006940 | 027C | SetStdHandle |
| 0000692E | 0000692E | 0156 | GetStringTypeW |
| 0000691C | 0000691C | 0153 | GetStringTypeA |
| 0000690C | 0000690C | 01C0 | LCMapStringW |
| 000068FC | 000068FC | 01BF | LCMapStringA |
| 000068E6 | 000068E6 | 01E4 | MultiByteToWideChar |
| 00006670 | 00006670 | 00CA | GetCommandLineA |
| 00006682 | 00006682 | 0174 | GetVersion |



LINGUAGGIO ASSEMBLEY

Dato il codice riportato in basso dobbiamo identificare i costrutti noti e ipotizzare il comportamento della funzionalità implementata

Il linguaggio assembly è univoco per una data architettura di un pc, ma cambia da architettura ad architettura

Dunque il linguaggio assembly servirà per leggere le istruzioni eseguite dalla CPU in formato leggibile dall'uomo

```

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

```

```

loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp

```

Creazione dello Stack

Chiamata di funzione. I parametri sono passati sullo Stack tramite le istruzioni push

Ciclo IF

```

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

```

Il blocco di codice qui di seguito si riferisce al ciclo if perché secondo una determinata condizione il programma eseguirà uno dei blocchi di codice riportati qui sotto

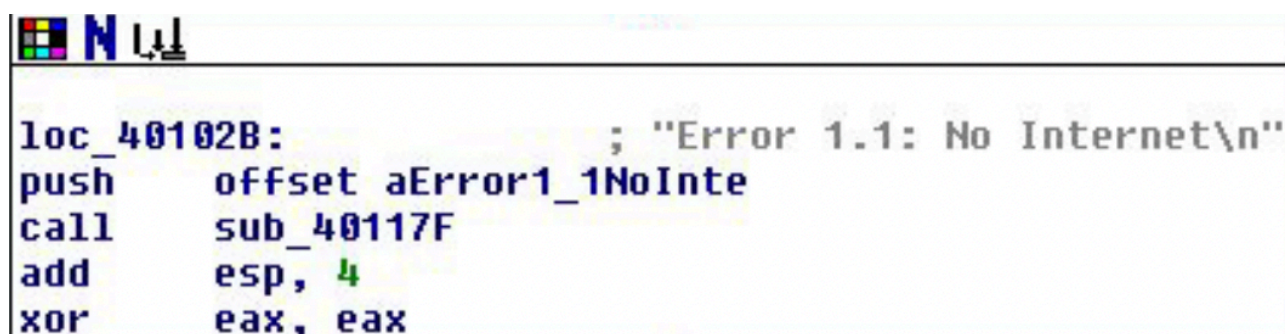
Blocco di codice connessione attiva

```

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

```

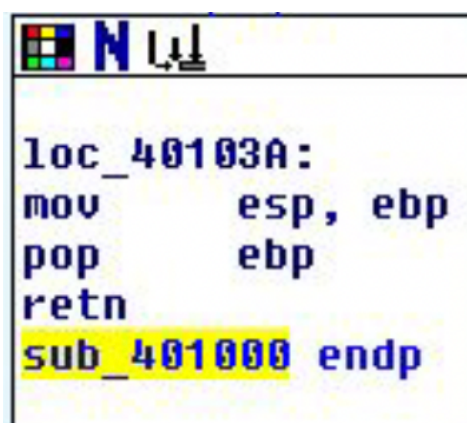
Blocco di codice connessione non attiva



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

L'ultima parte del codice invece fa la pulizia dello Stack dopo aver verificato se la connessione è attiva

Infatti come possiamo vedere nel codice dove ci dice che la connessione ha avuto successo fa un salto(jump) proprio alla locazione riportata qui di seguito



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

Avendo eseguito le analisi statiche del malware possiamo ipotizzare che la funzionalità implementata è che il malware chiama la funzione **internetgetconnectedstate** e ne controlla un IF .Se il valore di ritorno della funzione è diverso da zero allora vuol dire che c'è una connessione attiva

Il malware potrebbe sfruttare la connessione per avere la possibilità di scaricare altri malware oppure un'alta ipotesi potrebbe essere che l'attaccante usi la macchina vittima (zombie) per effettuare attacchi Ddos, queste sono solo delle ipotesi per quanto riguarda lo scopo del malware per avere più conferme si dovrebbe fare anche un'analisi dinamica del malware così da poter capire grazie all'uso dei tool specifici come ad esempio: Process Monitor, Process Explorer il comportamento che ha il malware nella macchina infetta

