

1) Librerie del Malware

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) |
|--------------|--------------|----------|---------------|----------------|----------|-----------|
| 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 |

File: Malware_U3_W2_L5.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [x]
- Section Headers [x]
- Import Directory

Notiamo che il programma carica 2 librerie:

KERNEL32.dll con 44 funzioni, necessaria a comunicare con il sistema operativo
WININET.dll con 5 funzioni utilizzate principalmente per accedere a servizi di rete
quindi ci aspettiamo che il programma possa comunicare.

2)Sezioni

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

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 |

File: Malware_U3_W2_L5.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [x]
- Section Headers [x]
- Import Directory

3)Costrutti

```

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

```

- in **ROSSO** possiamo vedere la creazione di un stack di memoria di grandezza indefinita.

- in **BLU** il codice ci suggerisce che il programma ha utilizzato 2 variabili per verificare con la funzione invocata in CALL la connessione ad internet della macchina.

-in **VERDE** è evidente che il programma abbia eseguito una condizione di tipo IF/ELSE data dalla coppia CMP-JZ (compare e jump)

```

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

```

-in **ROSSO** vediamo che il malware esegue la funzione PRINTF per stampare a video la frase di avvenuta connessione.

- con le altre istruzioni sposta e aggiunge valori ai registri esp e eax, infine salta a allo stack conclusivo.

```

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

```

-in **ROSSO** possiamo intuire che la connessione ai servizi di rete non sia andata a buon fine, il programma crea un messaggio di errore e lo stampa a video con la funzione printf.

-in **VERDE** ripulisce il registro eax con xor per salvare memoria

```

loc_40103A:
mov      esp, ebp
pop      ebp
retn
sub_401000 endp

```

-con questa porzione di codice il programma ripulisce lo stack.

4)Ipotesizzare il funzionamento

Se vogliamo stimare quale funzione svolge il codice analizzato possiamo presupporre che esso vada a testare la connessione ai servizi di rete della macchina attaccata. Con un ciclo if compara i dati delle variabili della funzione stabilita così da compiere l'adeguato salto di memoria seconda del risultato. Infine ripulisce lo stack.