

MALWARE ANALYSIS

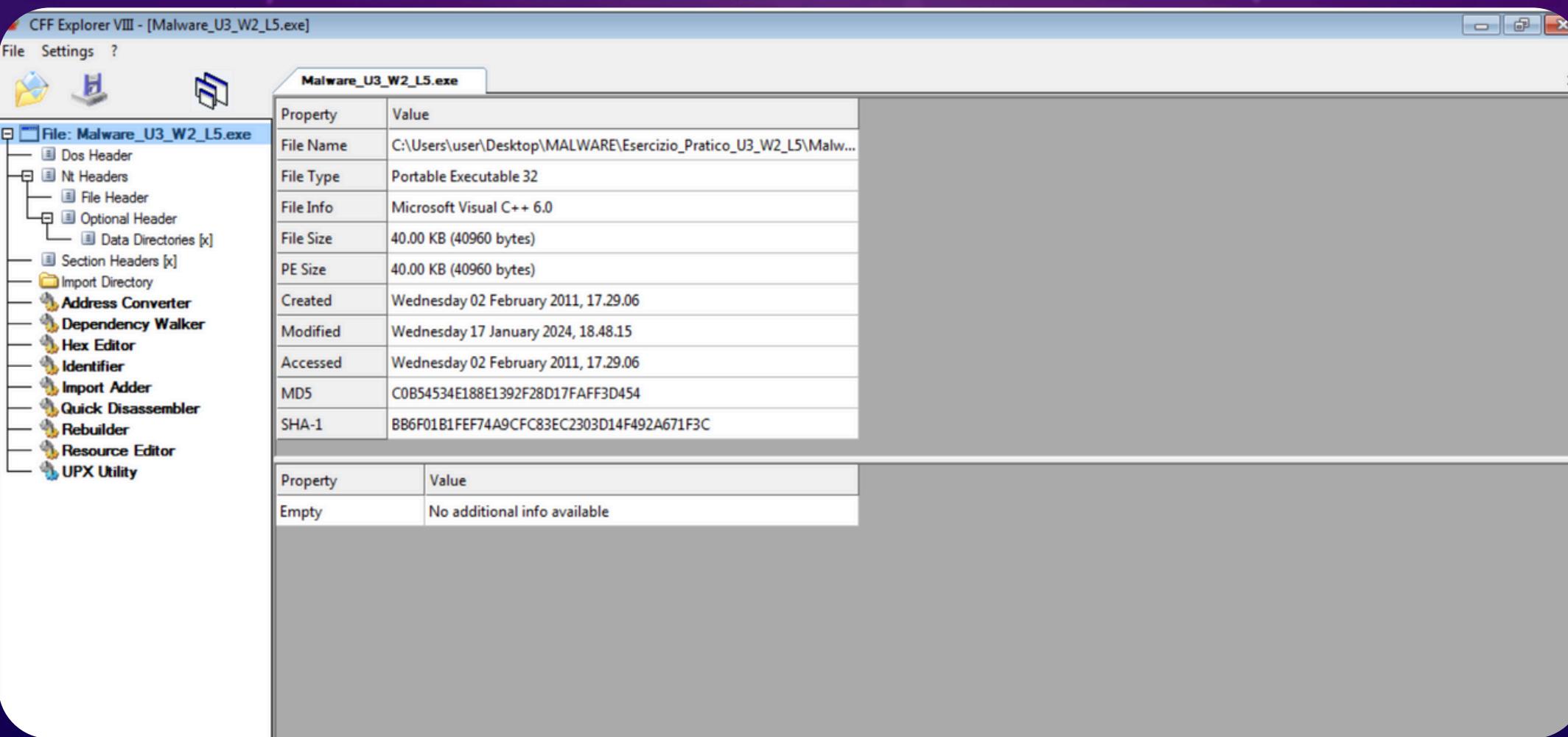
Morgan Petrelli



TRACCIA

Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5» sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

1. Quali librerie vengono importate dal file eseguibile?
2. Quali sono le sezioni di cui si compone il file eseguibile del malware?



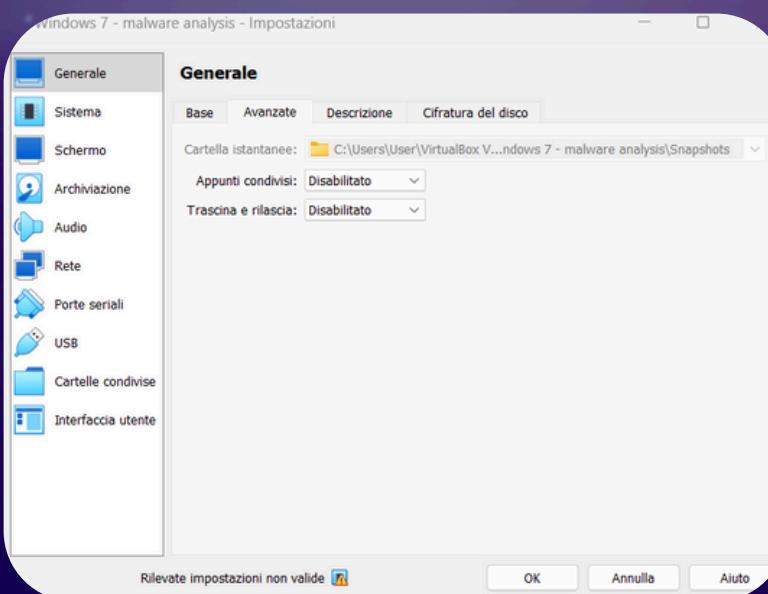
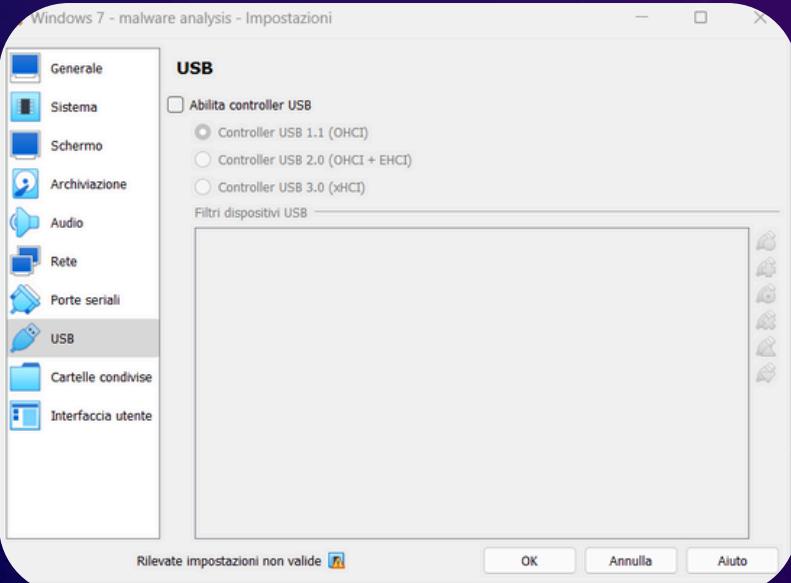
SANDBOX

prima di iniziare una malware analysis bisogna configurare un ambiente di test in modo da creare un ambiente sicuro in cui eseguire, monitorare e analizzare il malware senza rischio di diffusione o danni.

CONFIGURAZIONE DI RETE: disattivo la scheda di rete per evitare che l'ambiente non abbia accesso diretto ad internet.

DISPOSITIVI USB: per evitare che il malware si possa propagare su un possibile dispositivo usb, disattivo anche il controller usb.

APPUNTI CONDIVISI: disabilito anche gli appunti condivisi per evitare qualsiasi comunicazione fra macchina host e ambiente di test.



LIBRERIE IMPORTATE

CFF Explorer è uno strumento particolarmente utile per professionisti nel campo del reverse engineering, dell'analisi del malware e dello sviluppo software.

una volta aperto CFF Explorer carico il malware. Vado su “import directory” per controllare quali librerie siano state importate.

Kernel32.dll: è la libreria che contiene le funzioni per interagire con il sistema operativo, serve per eseguire funzioni essenziali come gestione della memoria, dei processi, dei file.

Wininet.dll: questa libreria contiene le funzioni per interagire con alcuni protocolli di rete come http, ftp, ntp.

Module Name	Imports	OFTs	TimeStamp	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



SEZIONI DEL FILE

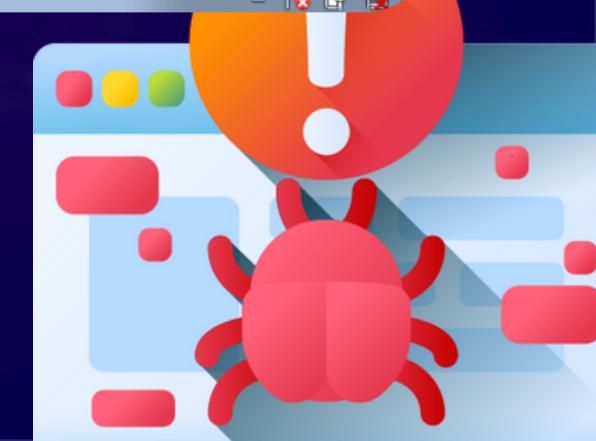
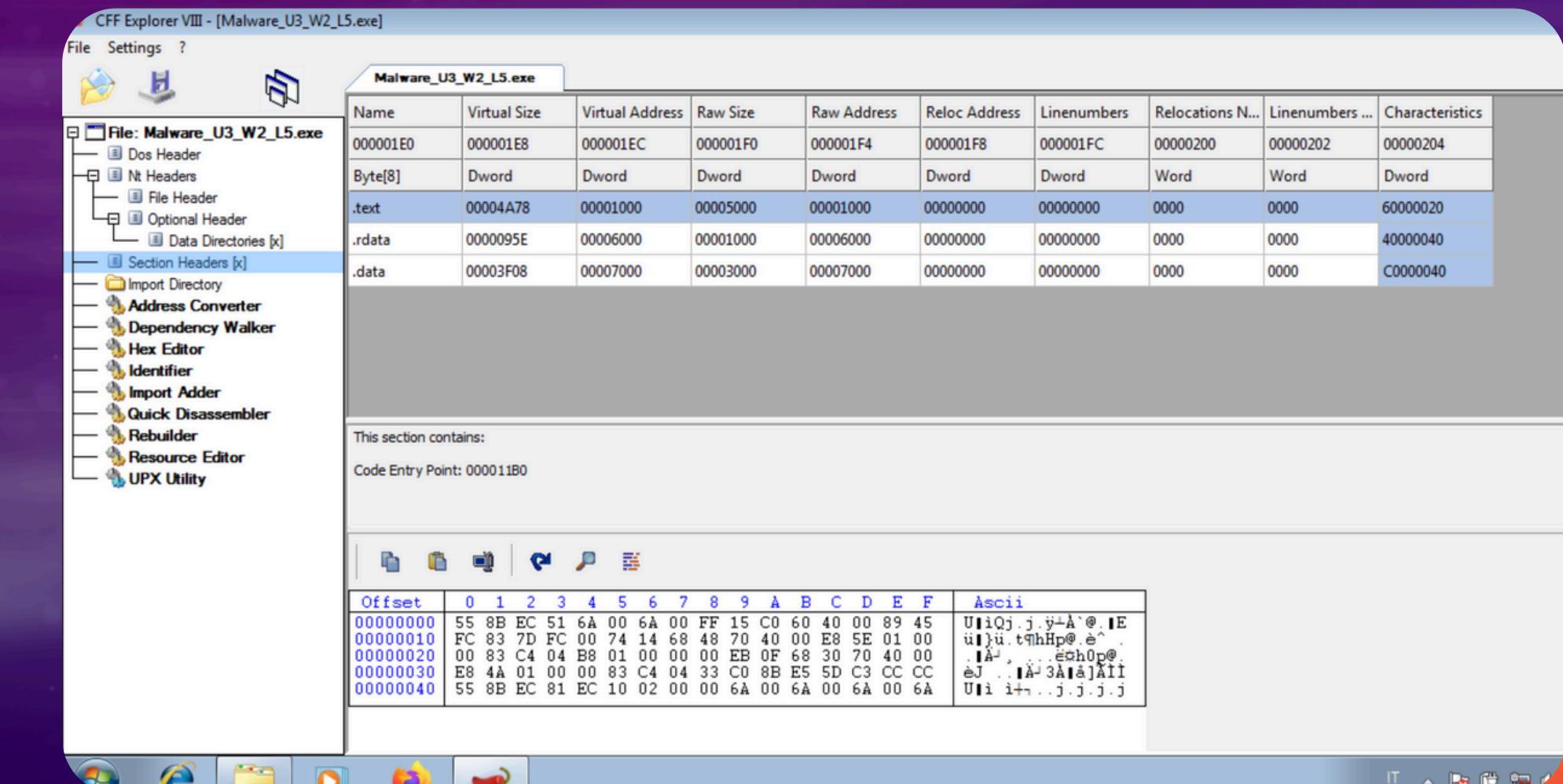
le sezioni sono parti del file eseguibile che contengono diversi tipi di dati necessari per l'esecuzione del programma. Ogni sezione ha un ruolo specifico e può contenere codice eseguibile, dati, risorse o altre informazioni necessarie per il funzionamento del programma.

il file si divide in tre diverse sezioni:

.text: contiene il codice che verrà eseguito dalla CPU una volta avviato.

.rdata: contiene le informazioni sulle librerie importate e le funzioni che servono al file.

.data: contiene dati inizializzati utilizzati dal programma durante l'esecuzione. A differenza della sezione .rdata, che contiene dati di sola lettura, la sezione .data contiene variabili e strutture che possono essere modificate dal programma.



TRACCIA

Con riferimento alla figura, risponde ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
4. Ipotizzare il comportamento della funzionalità implementata

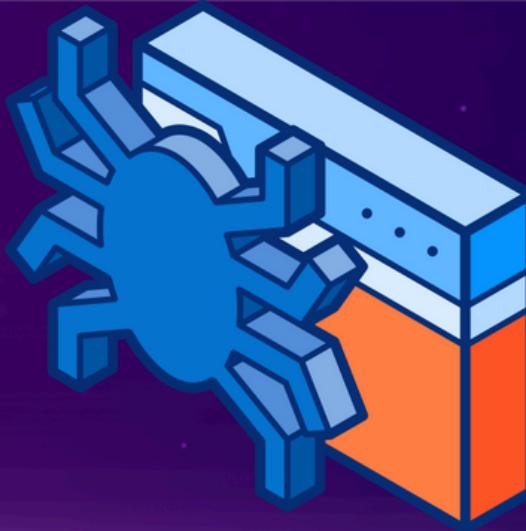
Figura 1

```
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
ret
sub_401000 endp
```



COSTRUTTI NOTI

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

questo costrutto serve per creare lo stack,
push ebp salva il valore attuale sullo stack
mov ebp, esp muove il valore di esp in ebp

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

questo è il costrutto condizionale “if”,
cmp [ebp+var_4], 0 confronta il valore di
[ebp+var_4] con 0.
jz short loc_40102B se il confronto risulta 0
salta a loc_40102B.

```
push    0
call    ds:InternetGetConnectedState
mov     [ebp+var_4], 1puwlags
call    sub_40117F
```

questi due costrutti chiamano le rispettive funzioni
InternetGetConnectedState e una funzione di
subroutine.

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

questo costrutto ripulisce lo stack e ritorna:
mov esp, ebp ripristina il valore di esp da ebp.
pop ebp ripristina il valore di ebp dallo stack.
retn ritorna dalla funzione.

IPOTESI DEL COMPORTAMENTO

Il comportamento della funzione sembra verificare lo stato della connessione e restituire un messaggio in base al risultato. Questa supposizione è supportata dal fatto che la funzione chiama `InternetGetConnectedState` per verificare lo stato della connessione Internet. Se lo stato della connessione è attivo, il valore di `cmp [ebp+var_4]`, 0 sarà diverso da zero e verrà stampato il messaggio "Success: Internet Connection". Al contrario, se il valore è 0, verrà stampato il messaggio "Error 1.1: No Internet".

