



Progetto S10-L5

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:

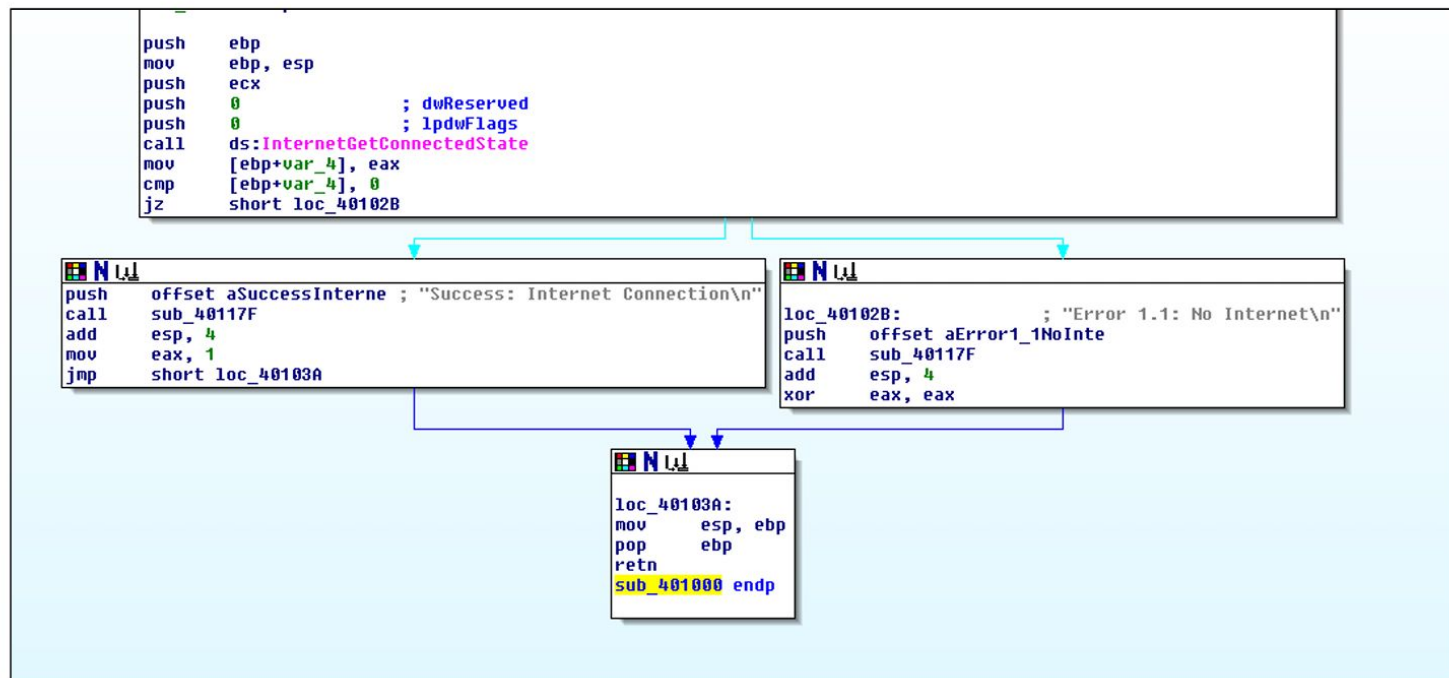
1. Quali **librerie** vengono importate dal file eseguibile?
2. Quali sono le **sezioni** di cui si compone il file eseguibile del malware?

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

3. Identificare i **costrutti** noti (creazione dello stack, eventuali cicli, altri costrutti)
4. **Ipotizzare il comportamento della funzionalità implementata**
5. **BONUS** fare tabella con significato delle singole righe di codice assembly

Traccia

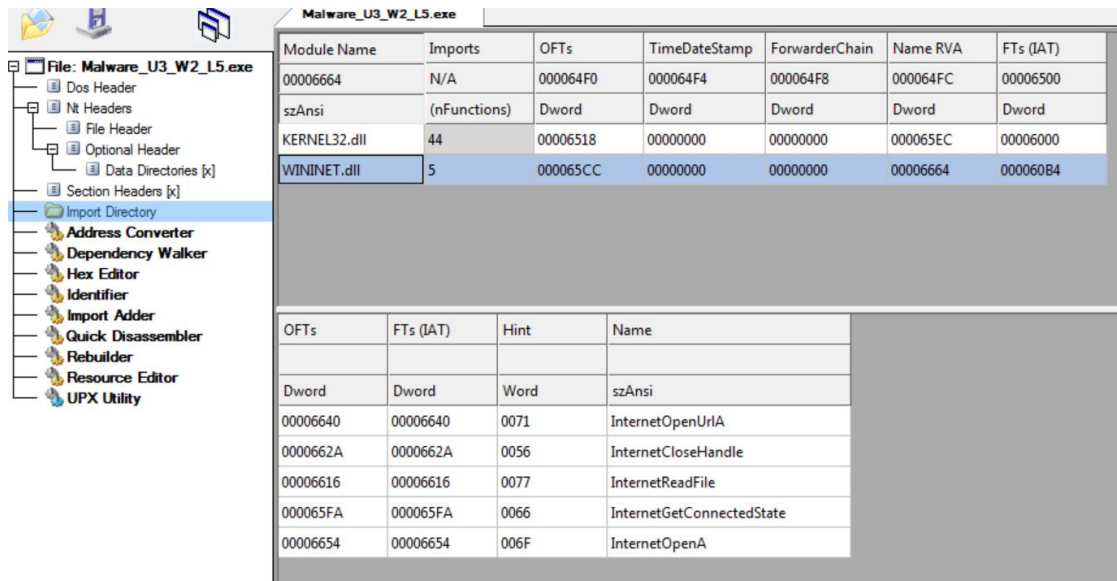
Figura 1



1. Librerie

Utilizzando CFF Explorer, vediamo dalla sezione import directory che il malware U3_W2_L5 importa 2 librerie:

1. **Kernel32.dll**: che include le funzioni core del sistema operativo
2. **Winnet.dll**: include le funzione per implementare i servizi di rete come ftp, ntp, http



The screenshot displays the CFF Explorer interface for the file **Malware_U3_W2_L5.exe**. The left-hand pane shows the file's internal structure, with the **Import Directory** section highlighted. The right-hand pane contains two tables. The top table lists imported modules, and the bottom table provides details for the **WININET.dll** module.

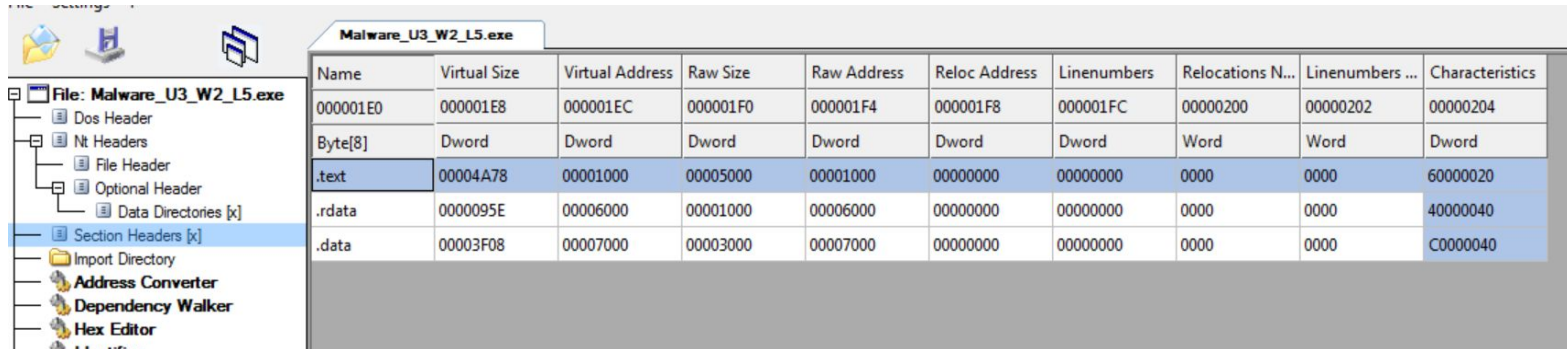
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
00006664	N/A	000064F0	000064F4	000064F8	000064FC	00006500
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

2. Sezioni

Passiamo alla sezione “section header” vediamo che l’e eseguibile si compone di 3 sezioni:

1. **.text**: contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l’unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.
2. **.rdata**: include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall’e eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer.
3. **.data**: contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile si dice globale quando non è definita all’interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all’interno dell’e eseguibile.



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
000001E0	000001E8	000001EC	000001F0	000001F4	000001F8	000001FC	00000200	00000202	00000204
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

3. Costrutti noti

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
```

Rimozione dello stack

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

4. Ipotizzare il comportamento



La funzione sembra controllare lo stato della connessione Internet chiamando `InternetGetConnectedState` e restituisce un valore diverso a seconda che la connessione sia attiva o meno. In caso di successo, stampa un messaggio di successo; in caso di errore, stampa un messaggio di errore.

5. Bonus: significato delle righe del codice

Codice Assembly	Significato
push ebp	Salva il valore corrente di EBP nello stack.
mov ebp, esp	Imposta EBP al valore corrente di ESP, creando un nuovo frame della stack.
push ecx	Salva il valore corrente di ECX nello stack.
push 0	Mette 0 nello stack (dwReserved).
push 0	Mette 0 nello stack (lpdwFlags).
call ds:InternetGetConnectedState	Chiama la funzione InternetGetConnectedState.
mov [ebp+var_4], eax	Salva il risultato di InternetGetConnectedState in [ebp+var_4].
cmp [ebp+var_4], 0	Confronta il valore salvato con 0.
jz short loc_40102B	Salta a loc_40102B se il risultato è zero (nessuna connessione Internet).
push offset asuccessInterne	Mette l'indirizzo della stringa "Success Internet Connection\n" nello stack.
call sub_40105F	Chiama la subroutine sub_40105F.
add esp, 4	Pulisce lo stack dopo la chiamata alla subroutine (aggiungendo 4 byte).
mov eax, 1	Imposta il registro EAX a 1.
jmp short loc_40103A	Salta a loc_40103A.
push offset aError1_1NoInte	Mette l'indirizzo della stringa "Error 1.1: No Internet\n" nello stack.
call sub_40117F	Chiama la subroutine sub_40117F.
add esp, 4	Pulisce lo stack dopo la chiamata alla subroutine (aggiungendo 4 byte).
mov esp, ebp	Ripristina il valore di ESP al valore di EBP, deallocando il frame della stack.
pop ebp	Ripristina il valore di EBP dallo stack.
ret	Restituisce dal sottoprogramma.