

# Settimana 10 - Esercizio 5

Alessio Golfetto, 16/02/2024

## **Traccia:**

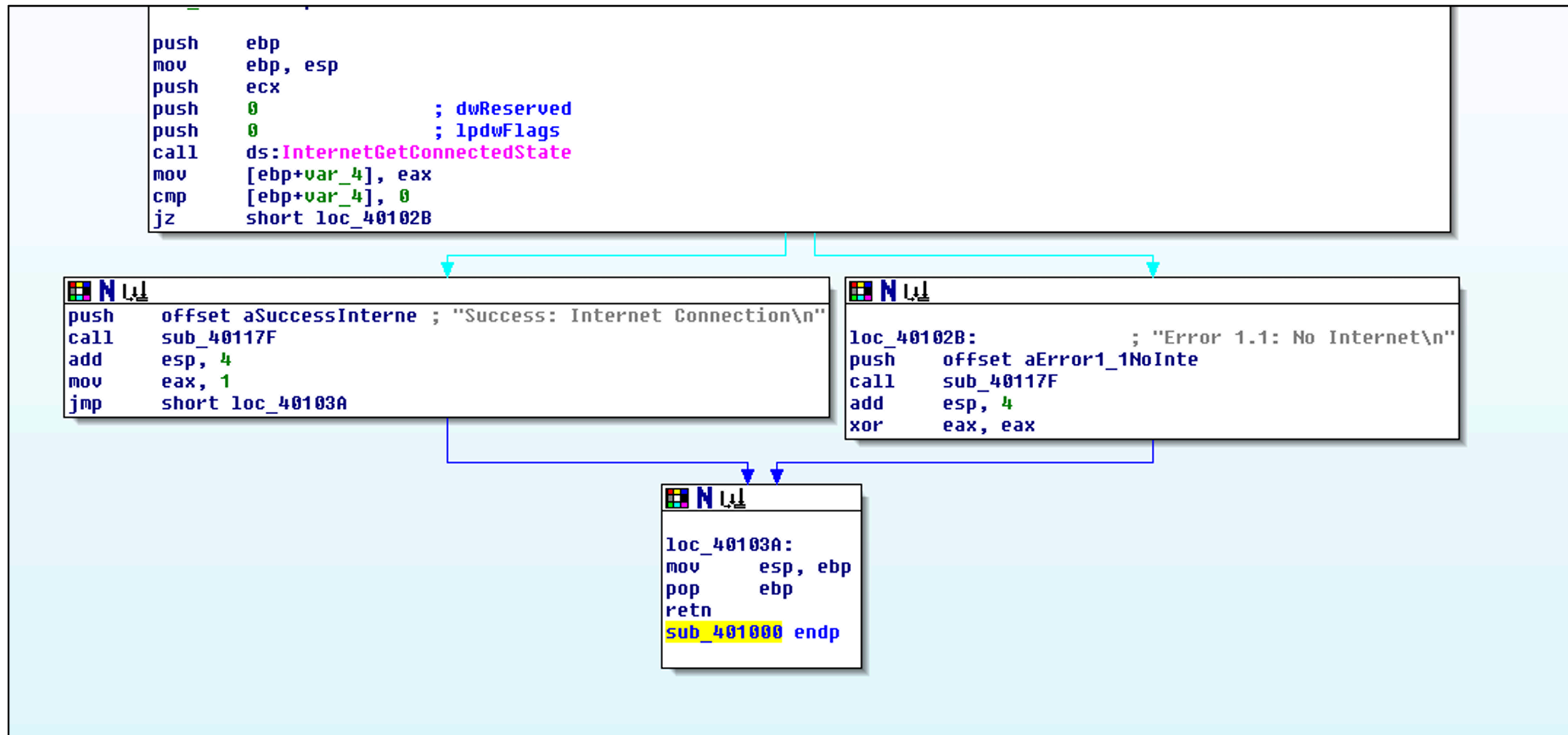
Con riferimento al file Malware\_U3\_W2\_L5 presente all'interno della cartella «Esercizio\_Pratico\_U3\_W2\_L5 » sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

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 1 in slide 3, risponde ai seguenti quesiti:

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

Figura 1



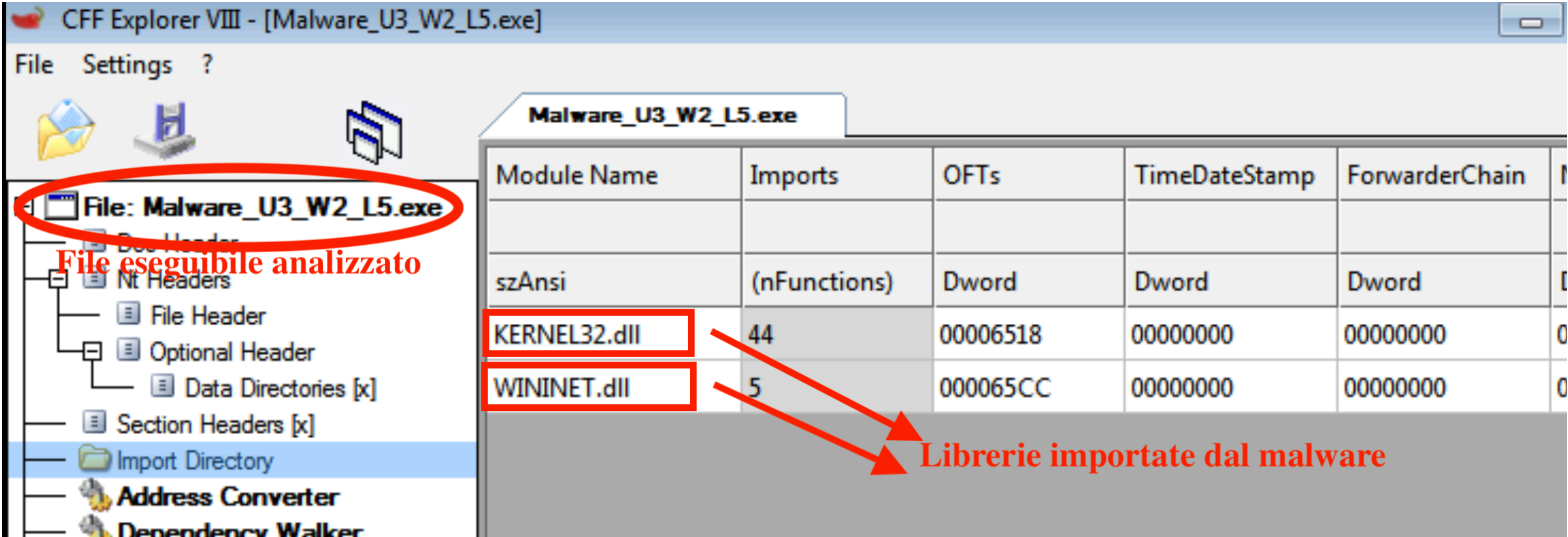
Con riferimento al file **Malware\_U3\_W2\_L5** presente all'interno della cartella «Esercizio\_Pratico\_U3\_W2\_L5 » sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

1. Quali librerie vengono importate dal file eseguibile?

Con il tool CFF Explorer possiamo controllare le funzioni importate ed esportate. Lo utilizziamo quindi per esaminare l'header del formato PE del nostro malware (file eseguibile). Possiamo evincere dalla risultato (figura sotto) che il malware importa le librerie **kernel32.dll** e **wininet.dll**.

**kernel32.dll:** è una libreria di collegamento dinamico nei sistemi operativi Microsoft Windows e svolge un ruolo fondamentale nell'esecuzione di diverse funzioni di basso livello necessarie per il corretto funzionamento del sistema operativo (ad esempio: manipolazione dei file, la gestione della memoria). È una componente fondamentale per molte applicazioni Windows e fornisce funzionalità chiave per il funzionamento stabile del sistema operativo.

**Wininet.dll:** Questa DLL fornisce una serie di funzionalità relative alle operazioni di rete e alla connettività Internet. Wininet.dll è spesso utilizzata da applicazioni Windows per effettuare operazioni di rete, come il download di file da Internet, l'invio di richieste HTTP, e la gestione delle connessioni di rete.





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:

**2. Quali sono le sezioni di cui si compone il file eseguibile del malware?**

Sempre con l'utilizzo del tool CFF Explorer e cliccando sulla voce "Section Headers" possiamo risalire alle sezioni di cui si compone il file eseguibile del malware. In questo caso ne troviamo tre: **.text**, **.rdata**, **.data**.

**.text:** questa sezione contiene le istruzioni eseguibili in linguaggio macchina che saranno eseguite dal processore

**.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




File Eseguibile analizzato


Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linen
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dwor
.text	00004A78	00001000	00005000	00001000	00000000	00000
.rdata	0000095E	00006000	00001000	00006000	00000000	00000
.data	00003F08	00007000	00003000	00007000	00000000	00000

Sezioni da cui è composto il malware

Con riferimento alla figura 1 in slide 3, risponde ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti )

<pre>push    ebp mov     ebp, esp</pre>		Creazione dello stack
<pre>push    ecx push    0           ; dwReserved push    0           ; lpdwFlags call    ds:InternetGetConnectedState</pre>		Chiamata della funzione
<pre>mov     [ebp+var_4], eax cmp     [ebp+var_4], 0 jz      short loc_40102B</pre>		Ciclo IF

<pre>loc_40103A: mov     esp, ebp pop     ebp retn sub_401000 endp</pre>		Chiusura della funzione
--	--	-------------------------

***Con riferimento alla figura 1 in slide 3, risponde ai seguenti quesiti:***

***4. Ipotizzare il comportamento della funzionalità implementata***

Il codice serve a verificare lo stato della connessione Internet utilizzando InternetGetConnectedState. Successivamente ci restituisce un messaggio che darà esito negativo se il confronto con zero del risultato ottenuto dalla chiamata a “InternetGetConnectedState” sarà zero, in caso contrario invece ci darà esito positivo.

Con riferimento alla figura 1 in slide 3, risponde ai seguenti quesiti:

5. BONUS fare tabella con significato delle singole righe di codice assembly

Spiegazione codice riga per riga	
<i>push ebp</i>	Serve a salvare il valore corrente del registro di base (ebp) nello stack
<i>mov ebp, esp</i>	Imposta il nuovo Ebp
<i>push ecx</i>	Salva il valore ecx
<i>push 0 dwReserved</i>	Setta il valore 0 come argomento di dwReserved
<i>push 0 lpdwFlags</i>	Setta il valore 0 come argomento di lpdwFlags
<i>call ds:InternetGetConnectedState</i>	Chiama la funzione InternetGetConnectedState
<i>mov [ebp+var_4], eax</i>	Salva il risultato in ebp+var_4
<i>cmp [ebp+var_4], 0</i>	Compara il risultato con 0
<i>jz short loc_40102B</i>	Ci rimanda a loc_40102B se 0
<i>push offset asuccessInterne</i>	Rimanda a una funzione che darà come risultato a schermo “success internet connection”
<i>call sub_40105F</i>	Se diverso da 0 chiama la subfunzione sub_40105F
<i>add esp, 4</i>	Regola lo stack
<i>mov eax, 1</i>	Sposta il valore 1 in Eax impostandone il valore
<i>jmp short loc_40103A</i>	Rimanda alla loc_40103A
<i>push offset aError1_1NoInte</i>	Se risultato è uguale a 0 restituisce messaggio “Error 1.1 no internet connection”
<i>call sub_40117F</i>	Chiama la subfunzione sub_40117F
<i>add esp, 4</i>	Regola lo stack
<i>mov esp, ebp</i>	Ripristina lo stack
<i>pop ebp</i>	Ripristina Ebp
<i>retn</i>	Rimanda alla funzione chiamante
<i>sub_401000 endp</i>	Chiude la funzione



Fine