PROGETTO SETTIMANA 10

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?

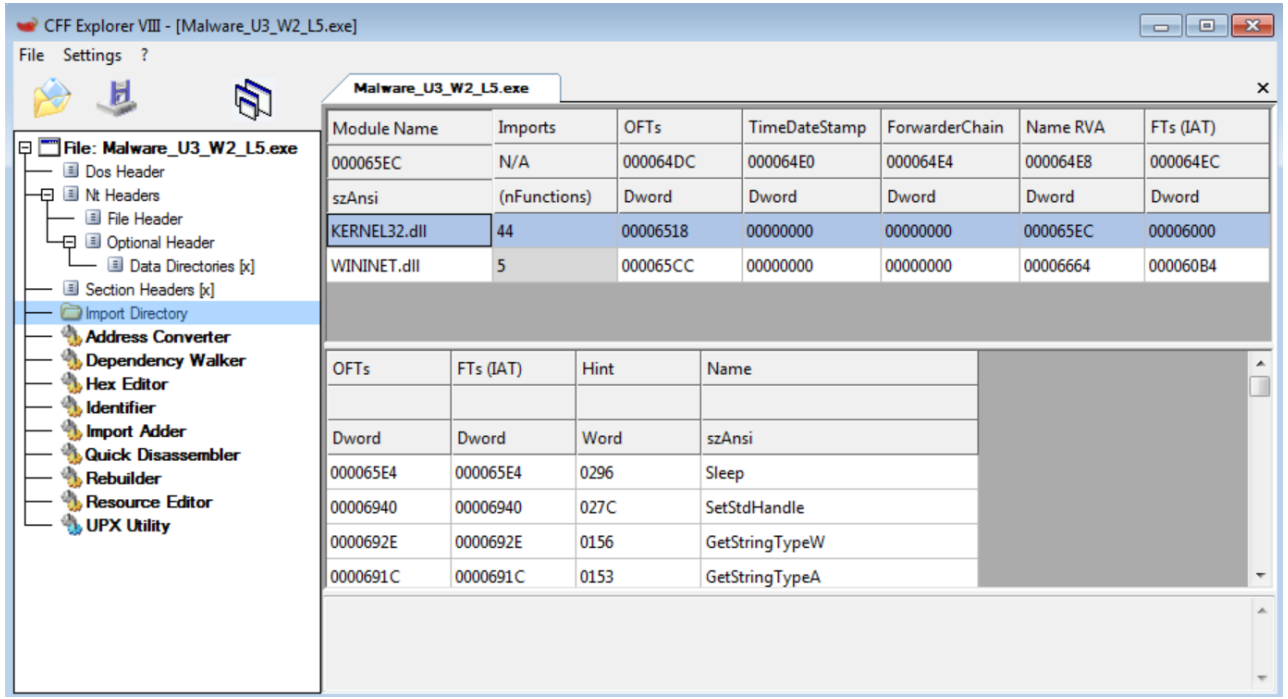
Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

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

1° RICERCA E ANALISI LIBRERIE

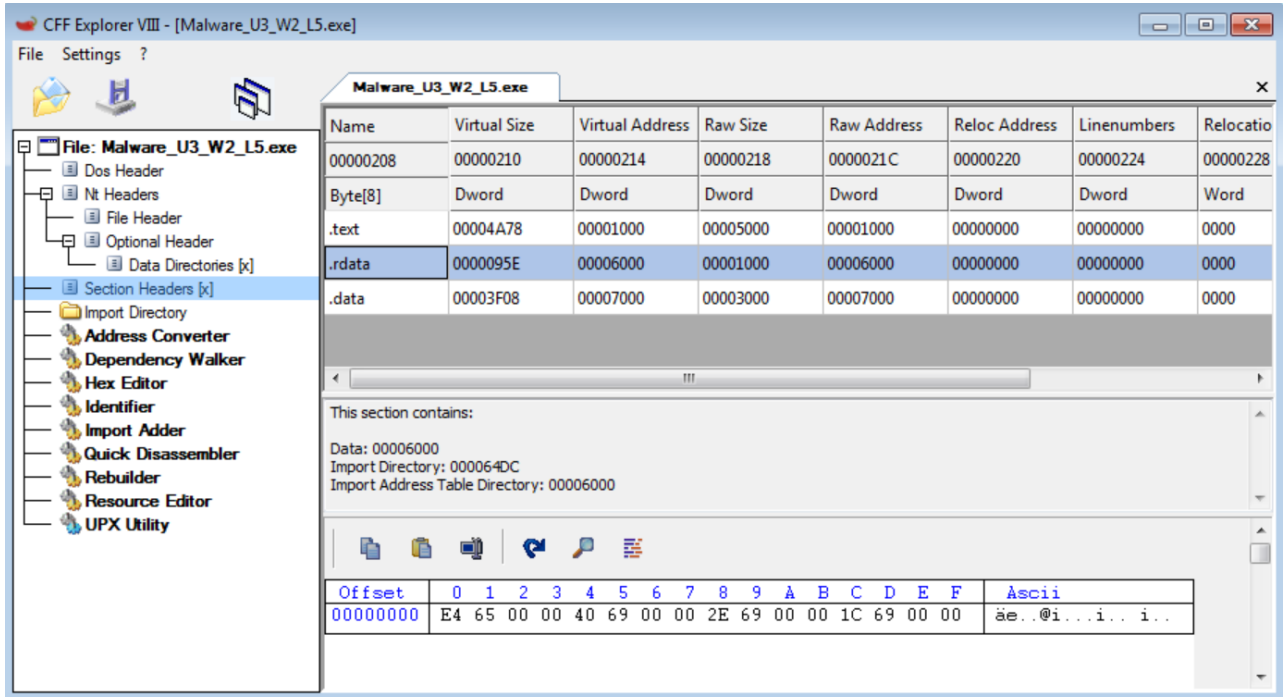
- Breve introduzione alle librerie.
- Le librerie contengono un insieme di funzioni. Quando un programma ha bisogno di una funzione «chiama» una libreria al cui interno è definita la funzione necessaria.
- Le librerie possono essere importate in tre modi diversi: staticamente, a tempo di esecuzione (runtime), dinamicamente.

1° RICERCA E ANALISI LIBRERIE



- In questo specifico caso tramite il programma CFF Explorer siamo riusci ad individuare le due librerie che il malware richiama.
- KARNELL32: : contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria.
- WININET: contiene le funzioni per l'implementazione di alcuni protocolli di rete come HTTP, FTP, NTP.

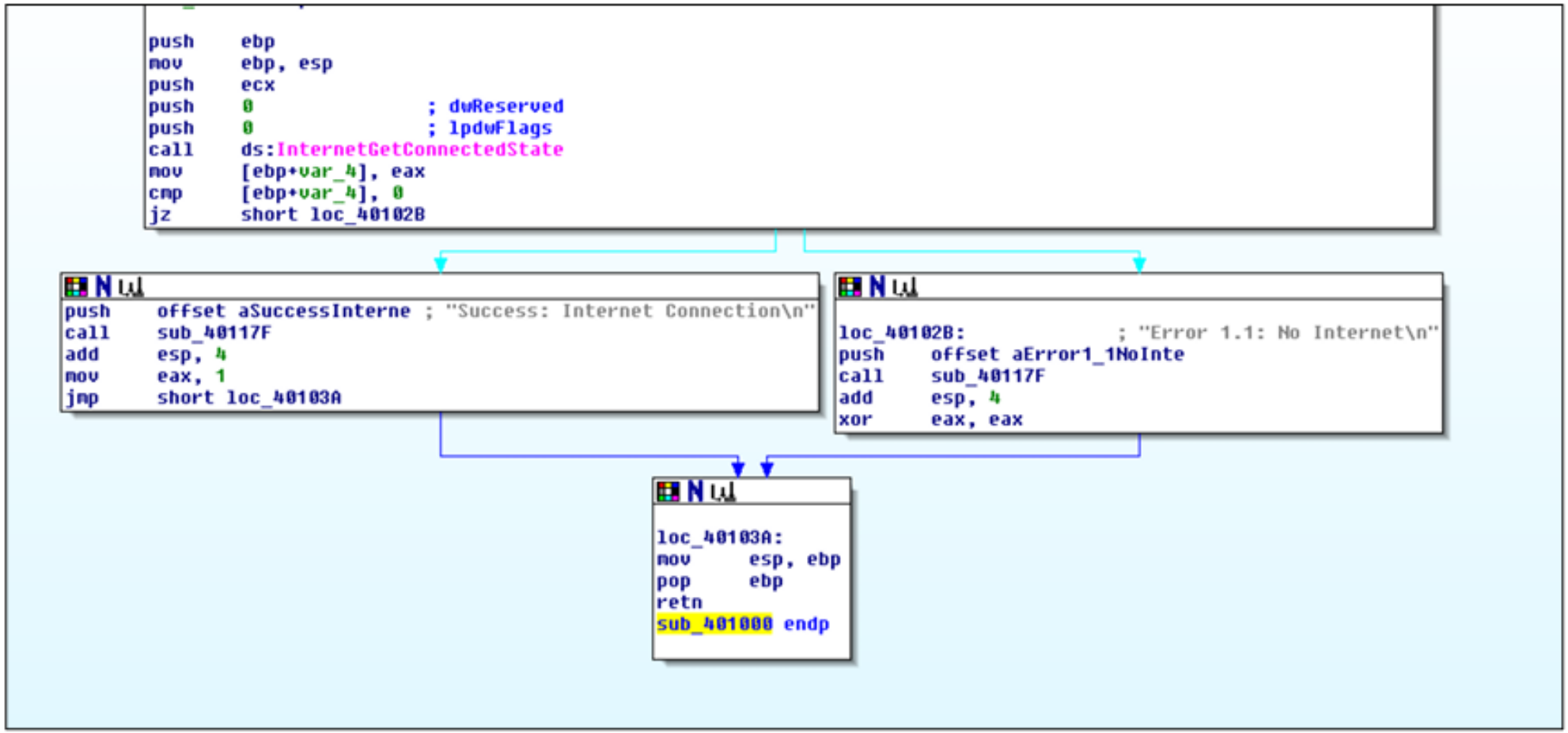
2° RICERCA E ANALISI SEZIONI



- Sempre tramite il programma CFF Explorer siamo riusci ad individuare tutte le sezioni del malware.
- .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 dalla CPU.
- .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.

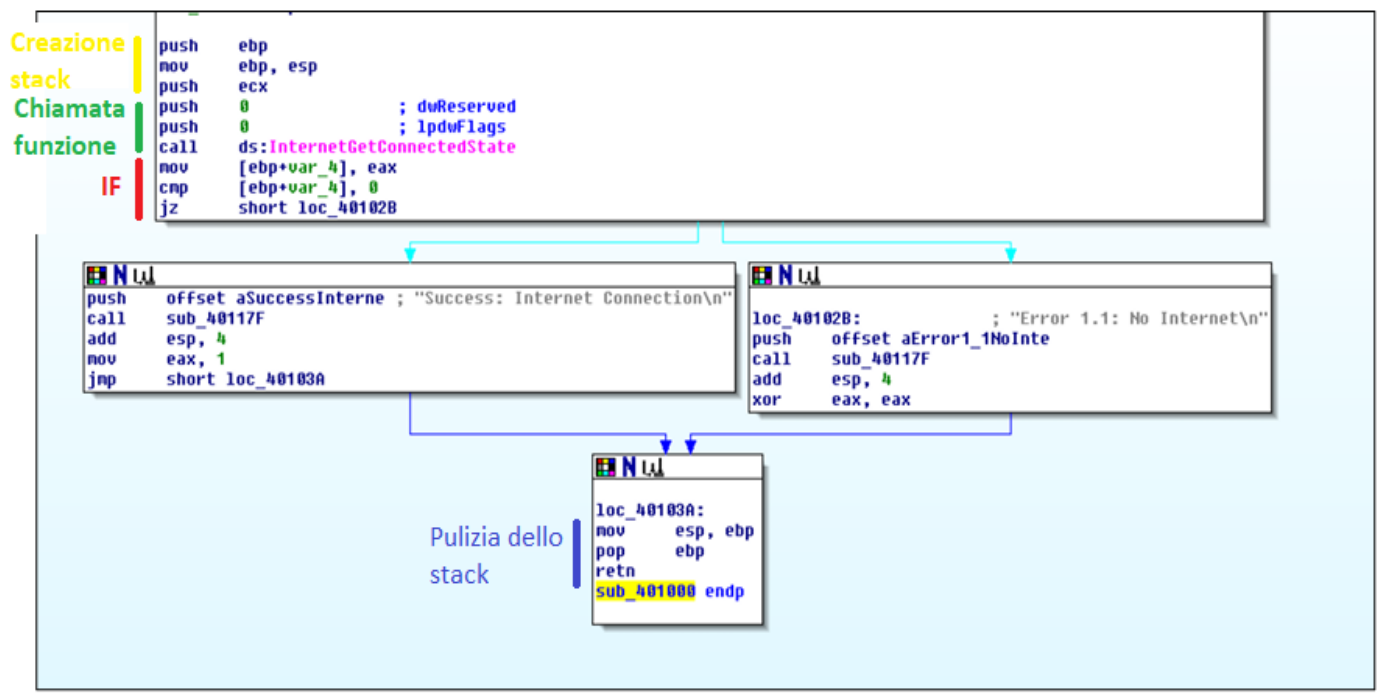
3° IDENTIFICARE I COSTRUTTI NOTI

Figura 1



3° IDENTIFICARE I COSTRUTTI NUOVI

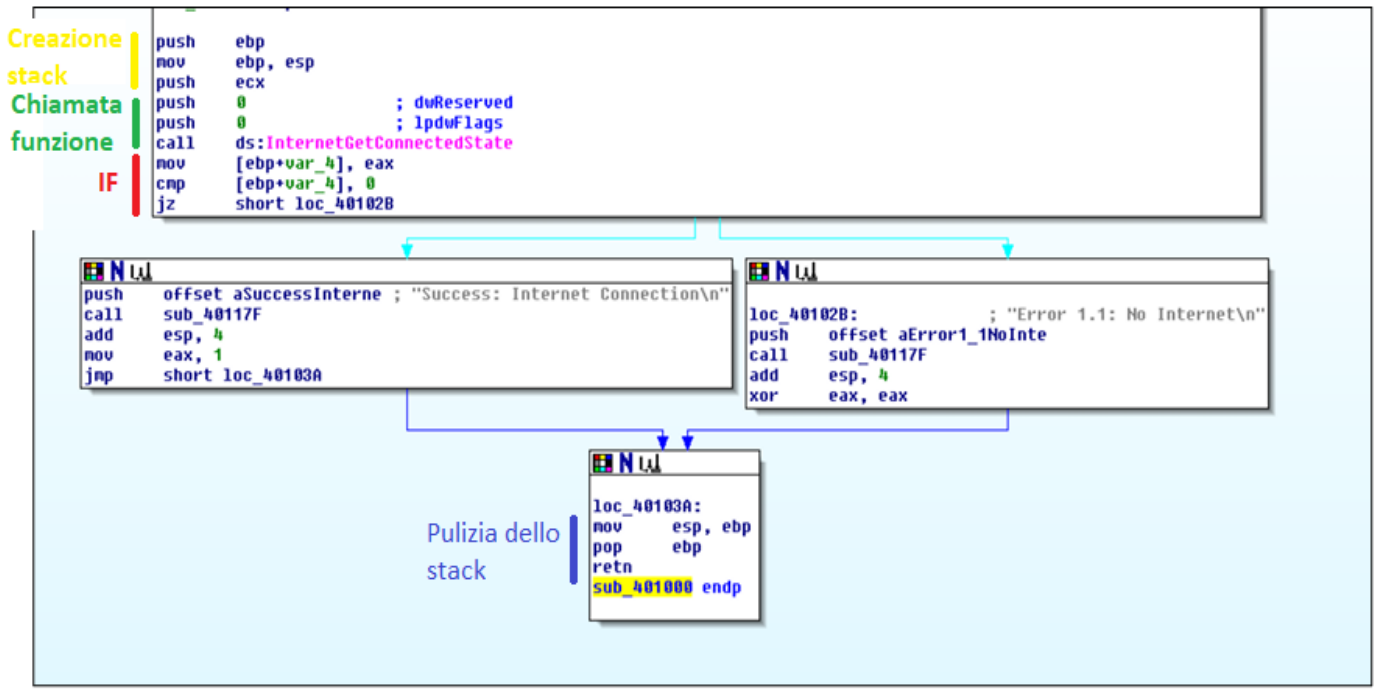
Figura 1



- Andiamo adesso a identificare e analizzare I costrutti trovati.
- Creazione dello stack: Crea uno spazio per una variabile locale senza preoccuparsi del valore di ecx.
- Chiamata della funzione: i parametri vengono «pushati» sullo 'stack' prima della chiamata alla funzione

3° IDENTIFICARE I COSTRUTTI NOTI

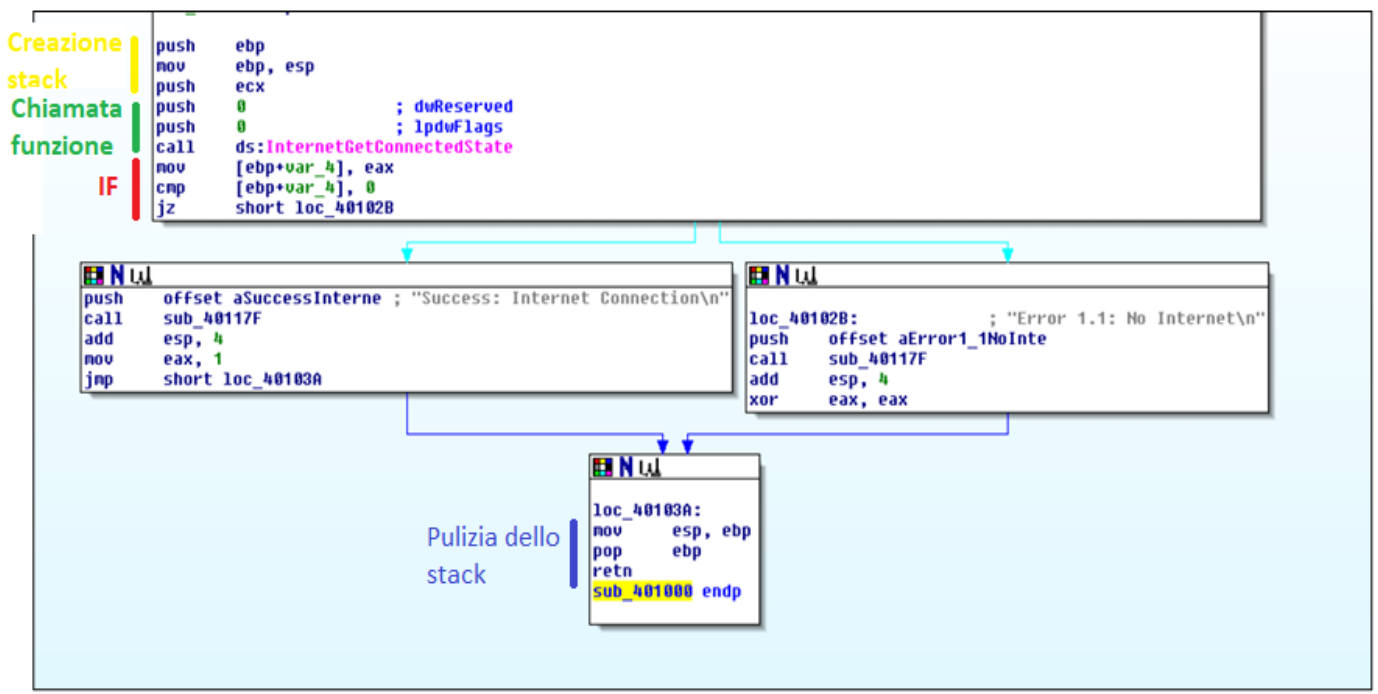
Figura 1



- IF: i costrutti if modificano l'esecuzione del programma sulla base di determinate condizioni.
  Dopo aver inizialmente assegnato due valori alle variabili, l'istruzione «cmp» unita all'istruzione jz controllano l'uguaglianza tra le due variabili. Jz poi salta alla locazione di memoria specificata se gli operandi sono diversi tra di loro, diversamente il programma scrive a schermo tramite chiamata di funzione alla funzione printf l'avvenuta connessione a internet.
- Pulizia dello stack: quando la funzione chiamata finisce il suo compito, bisogna eliminare il suo stack e le sue variabili locali non più necessarie per riportare l'esecuzione alla funzione chiamante.

4° IPOTIZZARE IL COMPORTAMENTO

Figura 1



- Da quello che riesco a intuire questa funzione testa la connessione a internet, da l'esito di questo test possono scaturire due eventualità. Una stampa l'eventuale positività della connessione, l'altra il fallimento. Entrambe terminano con la pulizia dello stack.