# Progetto Settimana 10

# Analisi Malware

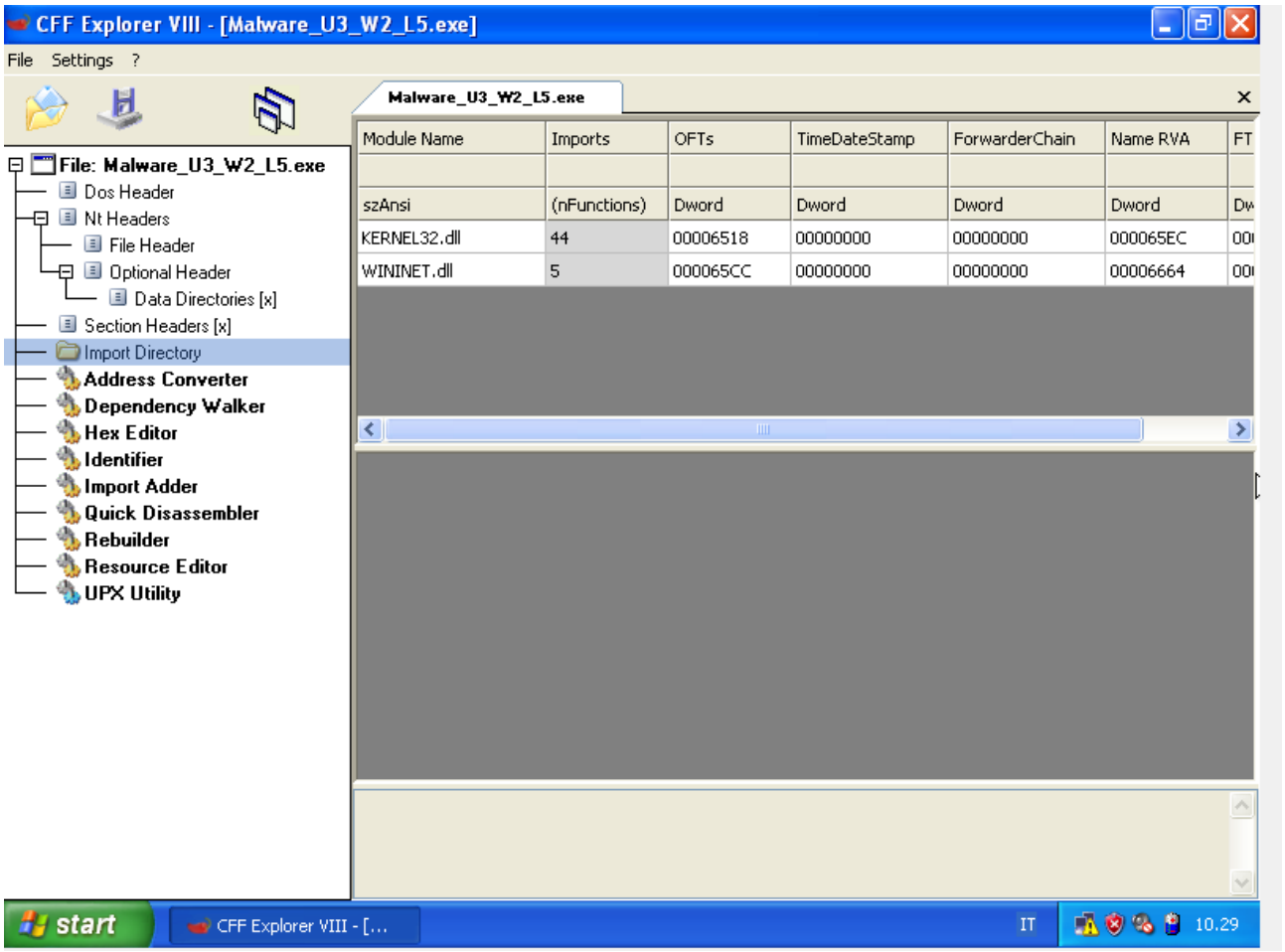
Oggi andremo ad effettuare l'analisi statica basica di un malware.

L'analisi statica prevede lo studio del malware senza che esso venga eseguito, con lo scopo di ottenere informazioni generiche sulle sue funzionalità.

Ci sono diversi tool che ci aiutano in questo compito. Possiamo usare m5deep, che ci consente di ottenere l'hash del file. Una volta ottenuto l'hash potremmo andare su siti come virustotal per capire (se il malware è presente nei registri del sito) con che tipo di malware abbiamo a che fare.

Un altro tool da poter usare è sicuramente CFF Explorer, che ci permette (una volta caricato il file del malware sul programma) di andare a vedere quali siano gli header del malware stesso e che librerie è andato ad importare.

L'esercizio di oggi, infatti, ci chiede proprio di andare ad analizzare un malware e di capire quali librerie lo stesso vada ad importare.

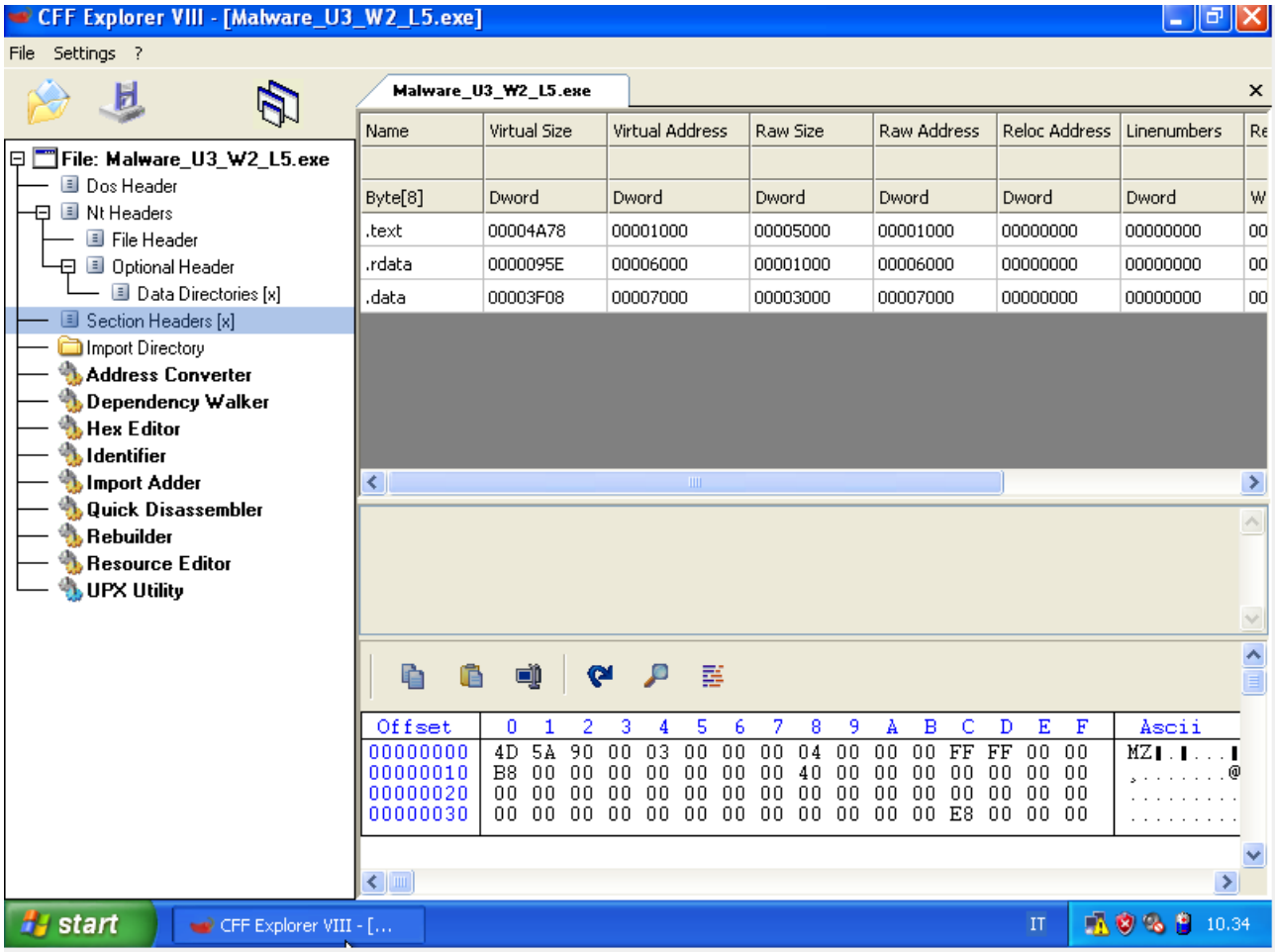


Dalla schermata, possiamo notare che il malware importa 2 librerie:

- Kernel32.dll: Questa libreria contiene le funzioni principali per interagire con il sistema operativo, come la gestione della memoria o la manipolazione dei file;
- Wininet.dll: Questa libreria, invece, contiene funzioni per l'implementazione di alcuni protocolli di rete come HTTP, FTP e NTP.

L'altra cosa che dovremo andare a capire è quali sono le sezioni di cui si compone il file eseguibile di questo malware.

Per trovare la soluzione, ci basterà andare alla sezione "Section Headers" che conterrà le sezioni di cui è composto l'eseguibile.



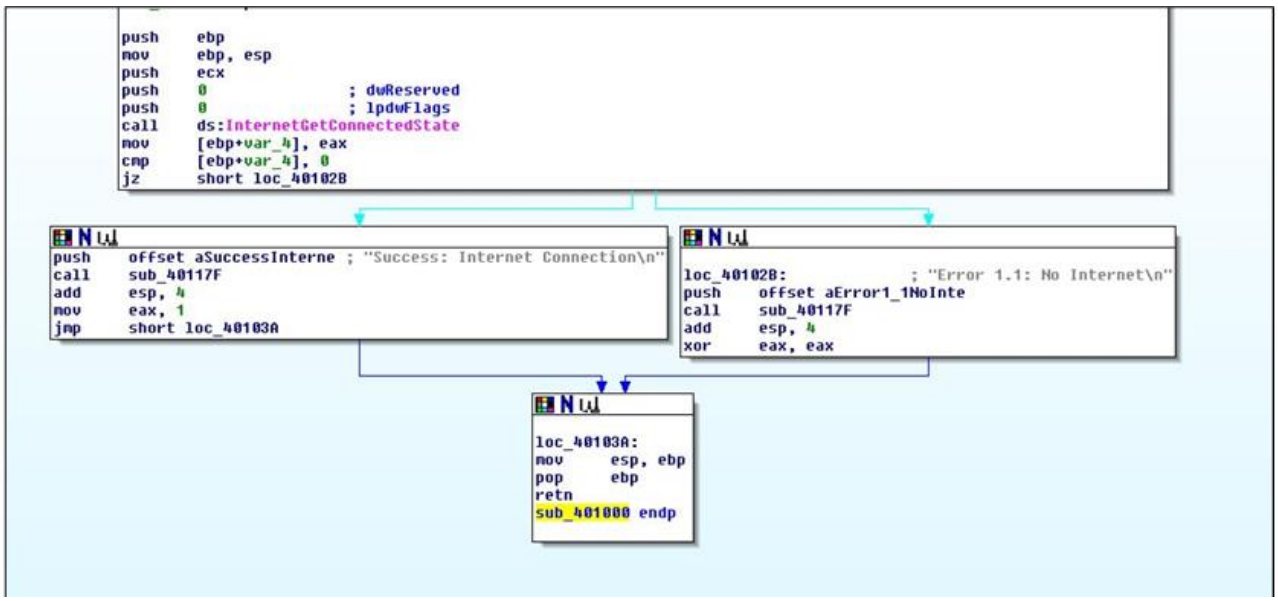

Come possiamo notare, questo eseguibile è composto da 3 sezioni:

- .text: contiene le righe di codice che la CPU andrà ad eseguire una volta che il software sarà avviato. Generalmente, questa è l'unica sezione che viene eseguita direttamente dalla CPU, in quanto le altre contengono informazioni a supporto;
- .rdata: include le informazioni riguardanti le librerie importate ed esportate dall'eseguibile;
- .data: contiene generalmente i dati e/o le variabili globali dell'eseguibile, che dovranno essere disponibili per qualsiasi parte del programma.

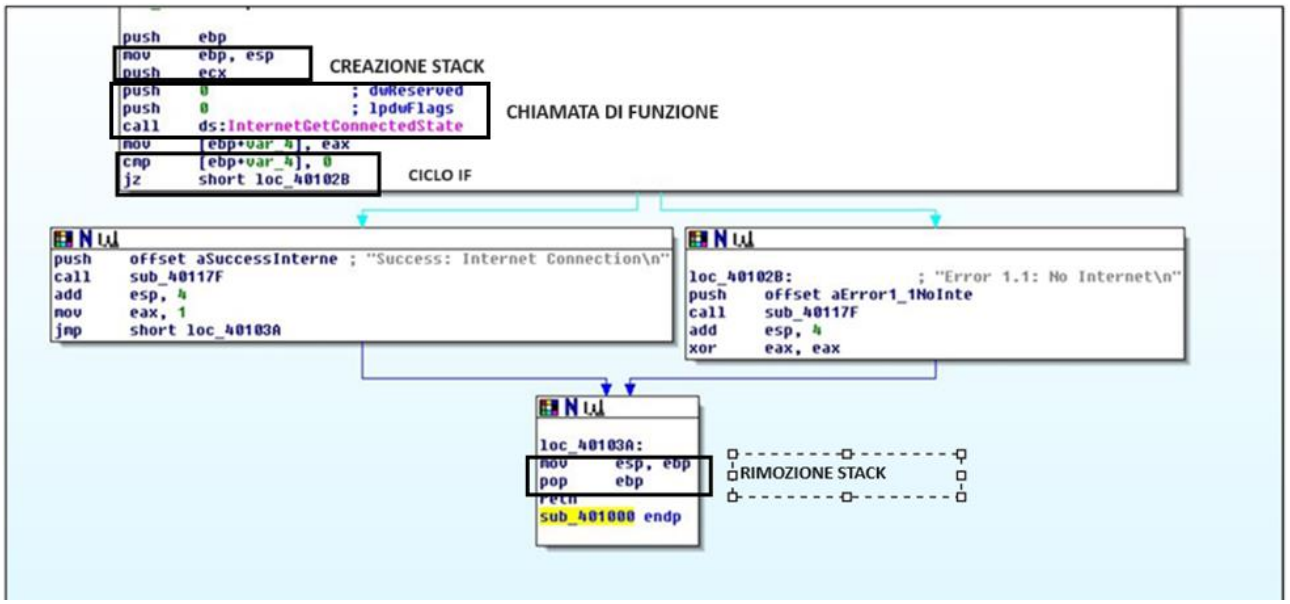
# Parte seconda – Analisi codice assembly

La seconda parte dell'esercizio ci chiede, data la seguente figura, di andare a:

- Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti);
- Ipotizzare il comportamento della funzionalità implementata.




Analizzando il disegno, possiamo identificare i seguenti costrutti:




Troveremo quindi i seguenti costrutti:

- Il primo, riguardante la creazione dello stack;
- la chiamata di funzione. Il programma eseguirà un blocco specifico di istruzioni ed eventualmente restituirà un risultato;
- Il ciclo IF. Viene identificato quando sono presenti le istruzioni cmp e jz. L'istruzione cmp, unita al jump, verificano l'ugaglianza tra le variabili. Nel caso in cui l'uguaglianza non venga

confermata, effettueremo un jump alla locazione di memoria specificata. In caso invece le variabili risultassero uguali, il codice proseguirà senza effettuare il jump.

- La rimozione dello stack: Una volta che il registro EBP avrà terminato il suo compito, verrà rimosso dallo stack per riportare l'esecuzione alla funzione chiamante.

# Ipotesi comportamento della funzionalità implementata

Dovendo fare un'ipotesi su quale sia il funzionamento di tale codice, possiamo dire che questo frammento di codice va a determinare se è presente o meno una connessione ad internet attraverso la funzione "InternetGetConnectedState".

Attraverso il costrutto IF avviene un controllo sulla funzione e, a seconda del risultato (uguale o diverso da 0), ci verrà fornita la risposta. Essa sarà "Success: Internet Connection" nel caso in cui ci sia effettivamente connessione o "Error 1.1: No Internet" nel caso in cui invece non sia presente connessione.