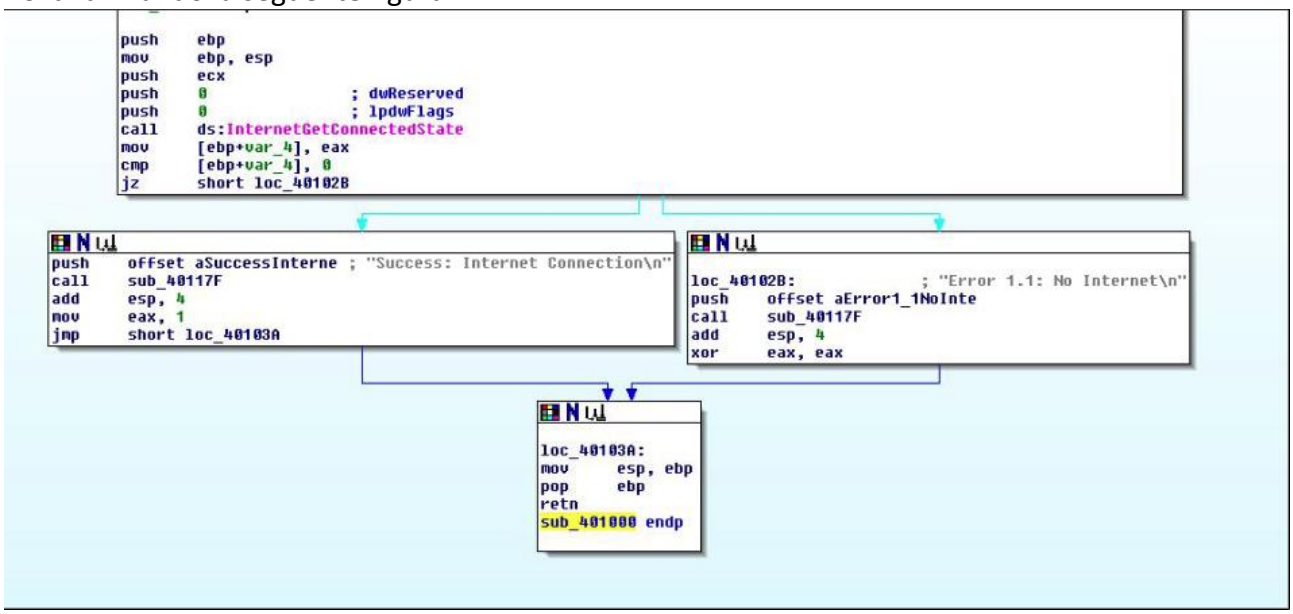
PROGETTO ANALISI MALWARE

FRANCESCO PERSICHETTI

Con riferimento al file Malware oggetto dell'esercitazione, dobbiamo analizzare il codice del Malware cercando di indentificare:

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

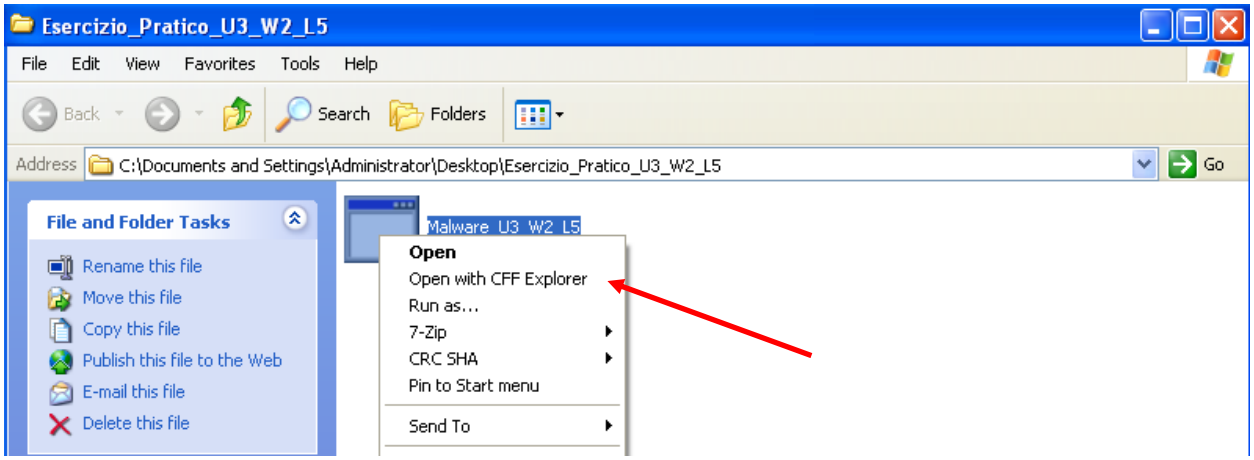
Poi analizzando la seguente figura:



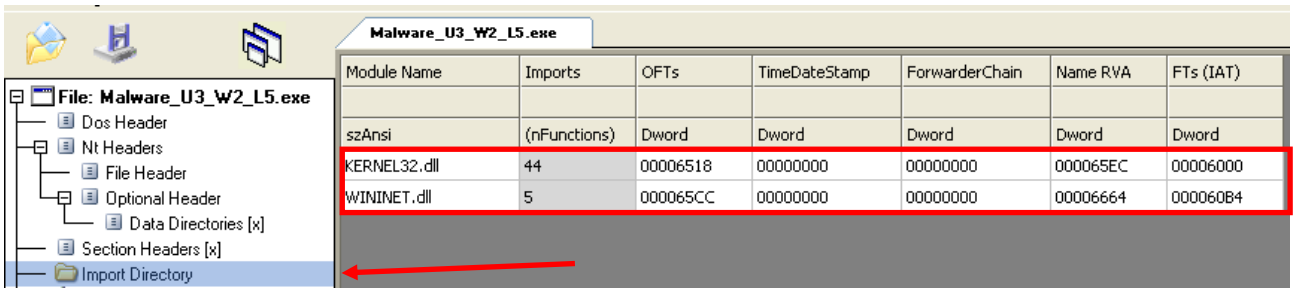
3. Identificare i costrutti noti
4. Ipotizzare il comportamento della funzionalità implementata

1.

Per iniziare con l'analisi del codice malevolo bisogna aprire quest'ultimo all'interno della nostra virtual machine creata ad hoc per non rischiare di infettare la nostra macchina principale.
All'interno della macchina virtuale apriamo il codice con l'apposito tool che ci aiuta nello studio del malware; questo tool si chiama "CFF Explorer"



Una volta aperto, all'interno dell'applicazione per conoscere le librerie importate dal malware basterà spostarsi nella categoria "import directory" nella finestra di dialogo sulla sinistra

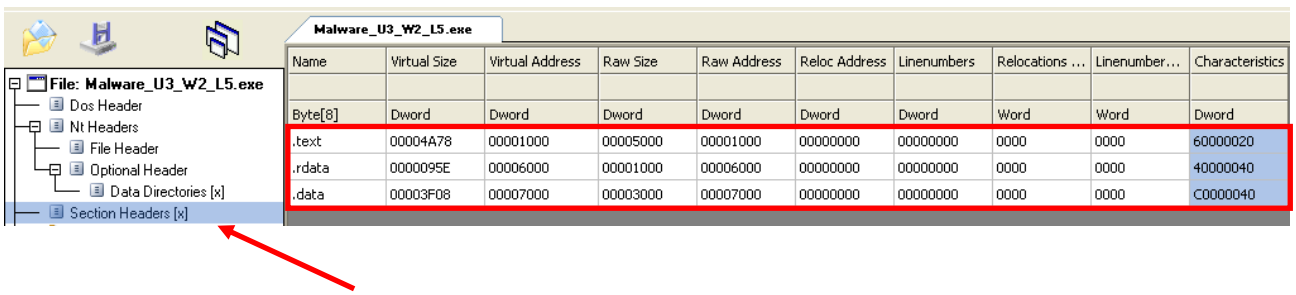


Come possiamo notare dalla figura le librerie importate sono due:

1. KERNEL32.DL → Libreria abbastanza comune che nel nostro caso importa a sua volta 44 funzioni di principale importanza per interagire con il sistema operativo
2. WININET.DL → Libreria che importa le funzioni utili per l'implementazione di alcuni protocolli di rete come HTTP,FTP,NTP. Nel nostro caso ne importa 5

2.

Sempre all'interno dell'applicazione, per continuare con la nostra analisi, andiamo a identificare le sezioni da cui è composto il nostro malware. Sempre nella finestra di dialogo alla sinistra dello schermo clicchiamo sulla categoria "Section Headers"



Dalla figura notiamo che le varie sezioni sono 3:

1. .TEXT → Contiene le righe di codice che la CPU eseguirà una volta che il software sarà avviato. Si tratta dell'unica sezione che verrà eseguita dalla CPU dato che tutte le altre contengono informazioni a supporto
2. .RDATA → Comprende le informazioni riguardo le librerie e le funzioni importate ed esportate dall'eseguibile, le stesse che abbiamo identificato e precisato nel punto precedente
3. .DATA → Contiene dati e variabili globali del programma eseguibile che devono essere disponibili da qualsiasi parte del programma

3.

I costrutti noti sono:

→ *CREAZIONE DELLO STACK*

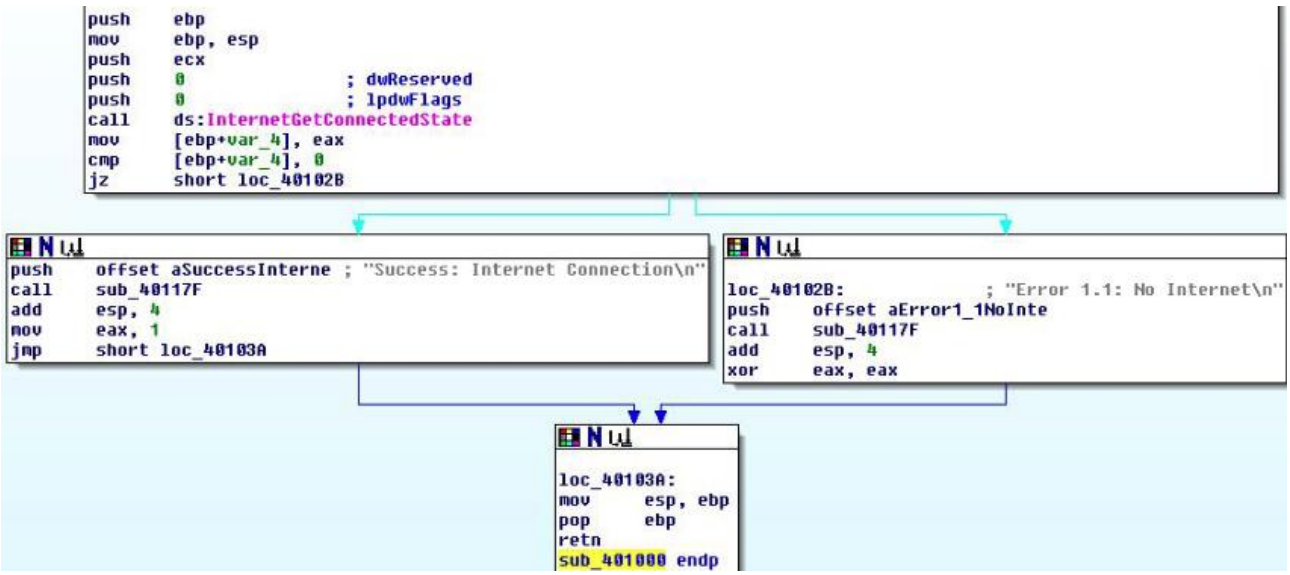
→ *CHIAMATA DI FUNZIONE CON I PARAMETRI PASSATI ALLO STACK TRAMITE LE ISTRUZIONI PUSH*

→ *CICLO IF SALTA FINO ALL'INDIRIZZO DI MEMORIA SCRITTO DOVE IL RISULTATO E' UGUALE A 0 PER CONTINUARE IL CODICE*

→ *SVUOTAMENTO DELLO STACK*

4.

Come si può evincere da questa struttura



Come già spiegato nel punto precedente, dopo aver creato lo stack arriva il momento in cui viene richiamata la funzione “ds:InternetGetConnectedState”. Dopodiché parte un ciclo IF il quale ha due possibilità:

- La connessione avviene e verrà richiamata la funzione “printf” tramite “call sub_40117F” che riporterà la stringa “Success: Internet Connection” (riquadro sinistro)
- La connessione non avviene e verrà richiamata la funzione “printf” tramite “call sub_40117F” che riporterà la stringa “Error 1.1: No Internet” (riquadro destro)

In entrambi i casi, dopo aver finito il controllo con il ciclo IF, il codice termina con lo svuotamento dello stack