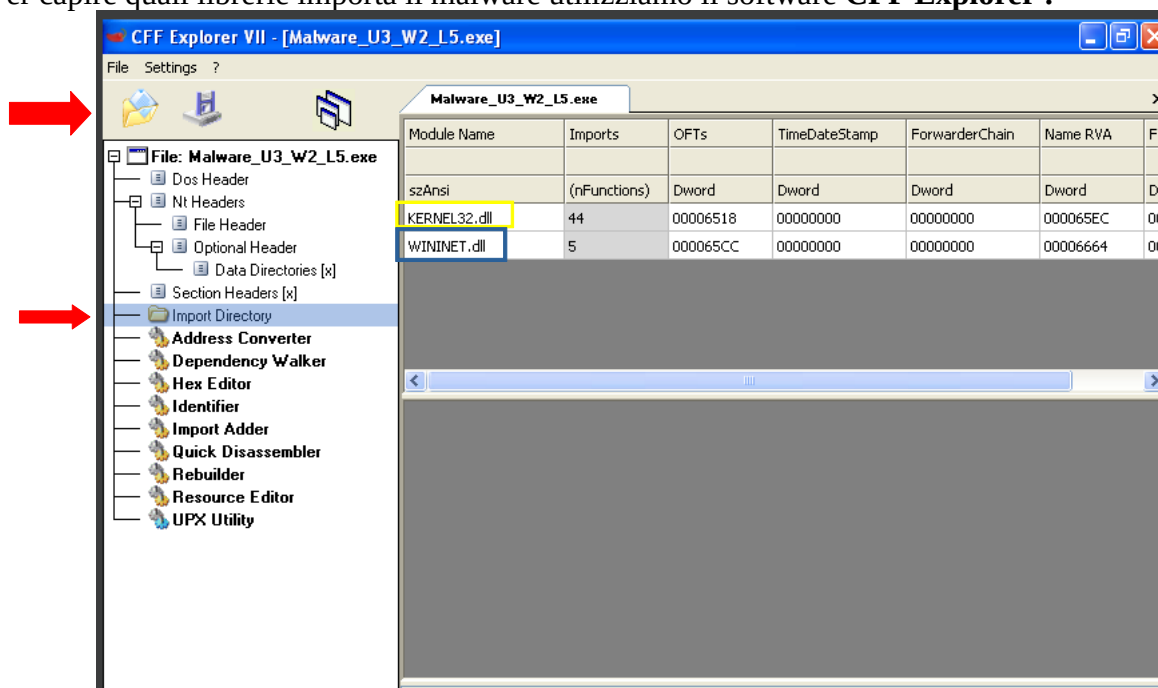


Studio malware basico completo

1) Quali librerie vengono importate dal file eseguibile :

Per capire quali librerie importa il malware utilizziamo il software **CFF Explorer** :

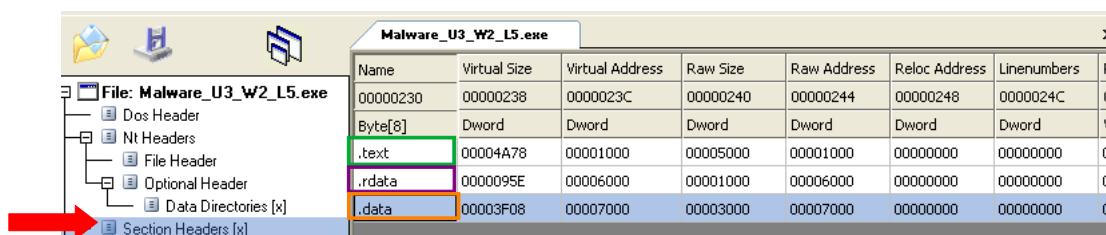


Come prima cosa andiamo inserire il file che vogliamo analizzare selezionandolo dal icona della lettera aperta. Poi per conoscere quali sono le librerie importate dal malware andiamo sulla sezione **Import Directory** :

KERNEL32.dll : Questa libreria contiene le funzioni principali per interagire con il sistema operativo: manipolazione dei file o gestione della memoria.

WININET.dll : Questa libreria contiene le funzioni per l'implementazione dei protocolli di rete come : HTTP,NTP,FTP.

2) Quali sono le sezioni di cui si compone il file eseguibile del malware :



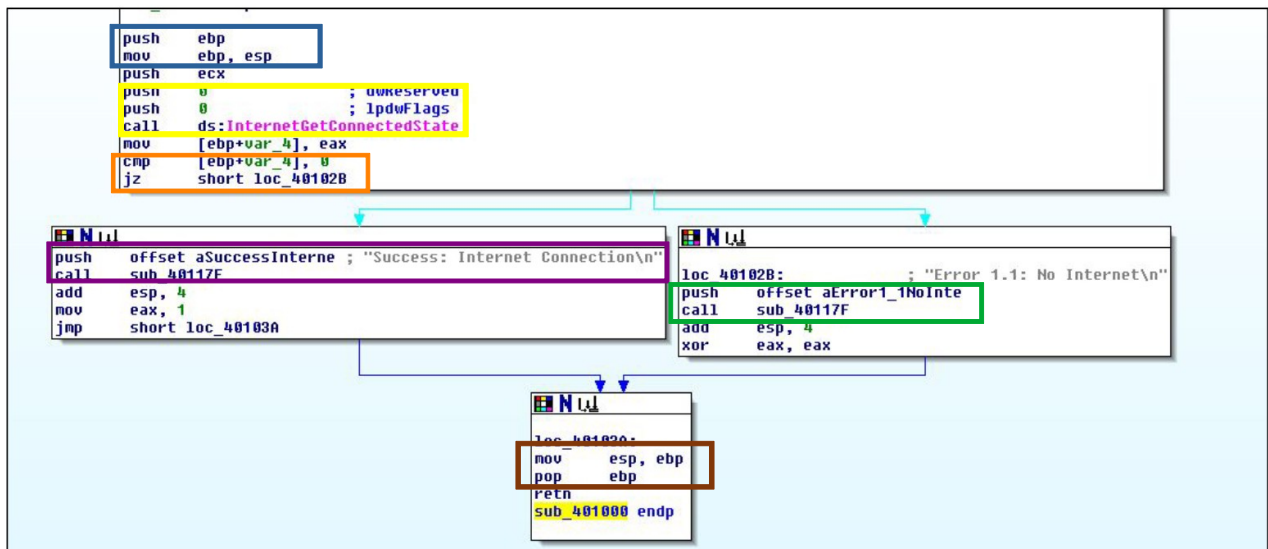
Per capire invece da quali sezioni è composto il malware utilizziamo sempre il software **CFF Explorer** questa volta nella sezione **Section Header** :

.text : Questa sezione contiene le righe di codice che la CPU eseguirà una volta che il malware sarà avviato.

.rdata : Questa sezione contiene le informazioni sulle librerie e le funzioni importate ed esportate dal malware.

.data : Questa sezione contiene i dati e le variabili globali del malware, che quindi dovranno essere disponibili a tutte le funzioni del malware.

3) Identificare i costrutti noti :



Creazione dello stack : creazione dello stack.

Richiama InternetGetConnectedState : inserisce due parametri nello stack che poi serviranno alla funzione **InternetGetConnectedState**.

Ciclo IF : le due righe sopra hanno la sintassi di un ciclo **if** infatti se il risultato del **cmp** darà 0 come risultato **jz** sotto salterà all'indirizzo di memoria **short loc_401102B** altrimenti continuerà in modo graduale.

Richiamo funzione aSuccessInterne : Queste due righe di codice richiamano la funzione **sub_40117F** che probabilmente mostrerà che la connessione è avvenuta con successo.

Richiamo funzione aError1_1noInte : Queste due righe di codice richiamano la funzione **sub_40117F** che probabilmente mostrerà che la connessione non è avvenuta con successo.

Pulizia dello stack : Queste due righe andranno a svuotare lo stack.

4) Ipotizzare il funzionamento della funzionalità implementata :

Dalla porzione di codice a nostra disposizione si può dedurre che il programma in questione verifichi se una macchina sia connessa ad internet o meno.

5) BONUS – tabella con significato delle singole righe di codice :

| Riga di codice | Descrizione |
|-----------------------------------|--|
| Push ebp | Si mette in memoria il registro ebp |
| Mov ebp, esp | Con questa riga si crea lo stack |
| Push ecx | Si mette in memoria il registro ecx |
| Push 0 | si passano i parametri per la funzione, sempre nello stack. |
| Push 0 | si passano i parametri per la funzione, sempre nello stack. |
| call ds:InternetGetConnectedState | Richiama la funzione InternetGetConnectedState |
| mov [ebp+var4], eax | sposta eax nel locazione di memoria puntata da [ebp+var4] . |
| cmp [ebp+var_4], 0 | compara il valore di [ebp+var4] con 0. |
| jz short loc_401102B | nel caso il confronto sopra produca come risultato 0 il comando eseguirà il salto. |

Tabella delle righe in caso la connessione sia avvenuta con successo

| Riga di codice | Descrizione |
|-------------------------------|---|
| push offset aSeccessInterne ; | inserisce in memoria la stringa aSeccessInterne. |
| call sub_40117F | richiama sub_40117F che potrebbe essere una procedura o simile. |
| add esp, 4 | somma al registro esp il valore decimale 4. |
| mov eax, 1 | sposta il valore decimale 1 nel registro eax. |
| jmp short loc_40103A | effettua un jump non condizionale verso short loc_40103A. |

Tabella delle righe in caso la connessione sia non avvenuta con successo

| Riga di codice | Descrizione |
|-----------------------------|--|
| Push offset aError1_1noInte | inserisce in memoria la stringa inserisce in memoria la stringa aError1_1noInte. |
| call sub_40117F | richiama sub_40117F che potrebbe essere una procedura o simile. |
| add esp, 4 | somma al registro esp il valore decimale 4. |
| Xor eax,eax | Con l'operatore xor rimettiamo a 0 il registro eax |

Tabella per l' ultima porzione di codice

| Riga di codice | Descrizione |
|-----------------|--|
| Mov esp,ebp | Copia il contenuto del registro ebp in esp |
| Pop ebp | Con questa riga a ricavare i dati dal registri ebp |
| retn | Ritorna al programma chiamante |
| Sub_401000 endp | Indica la fine della procedura Sub_401000 |