# ESERCITAZIONE WEEK 10 – UNIT 3

Con Riferimento al file Malware_U3_W2_L5:

- **Quali librerie vengono importate dal file eseguibile?**
- **Quali sono le sezioni di cui si compone il file eseguibile del malware?**

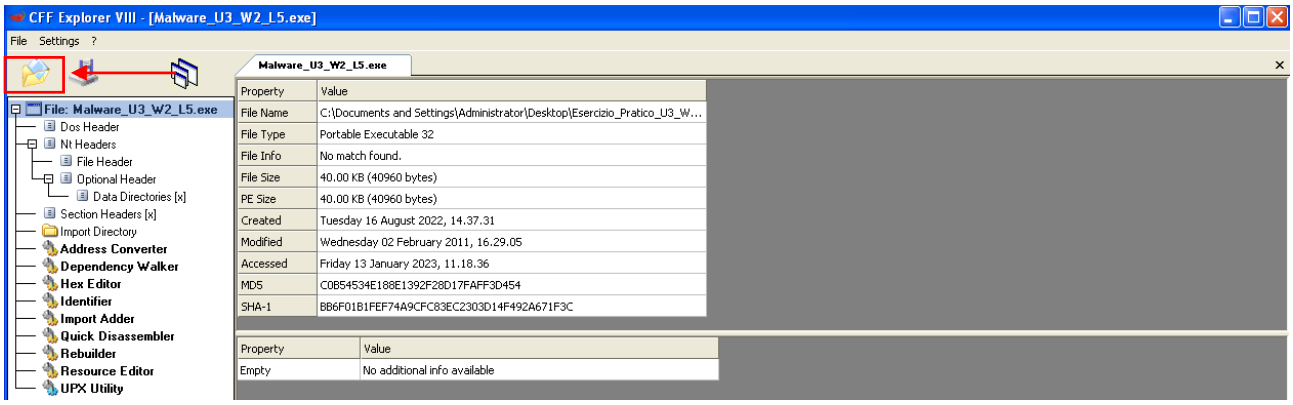
## FASE 1

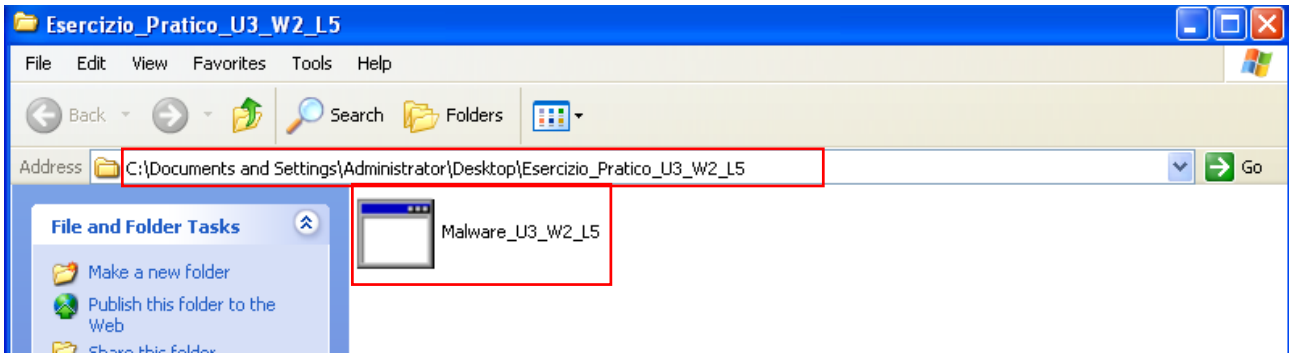
Per prima cosa apriamo il nostro **CFF Explorer** per poter analizzare in maniera statica il file eseguibile (analisi del file malevolo senza necessità di eseguirlo). Per prima cosa controlliamo che il nostro "laboratorio" sia stagno, senza connessione ad internet e senza connessione di alcun tipo con altre macchine (porte usb/cartelle condivise ecc.).


Apriamo CFF Explorer per analizzare le librerie importate e sezioni di cui si compone il file eseguibile

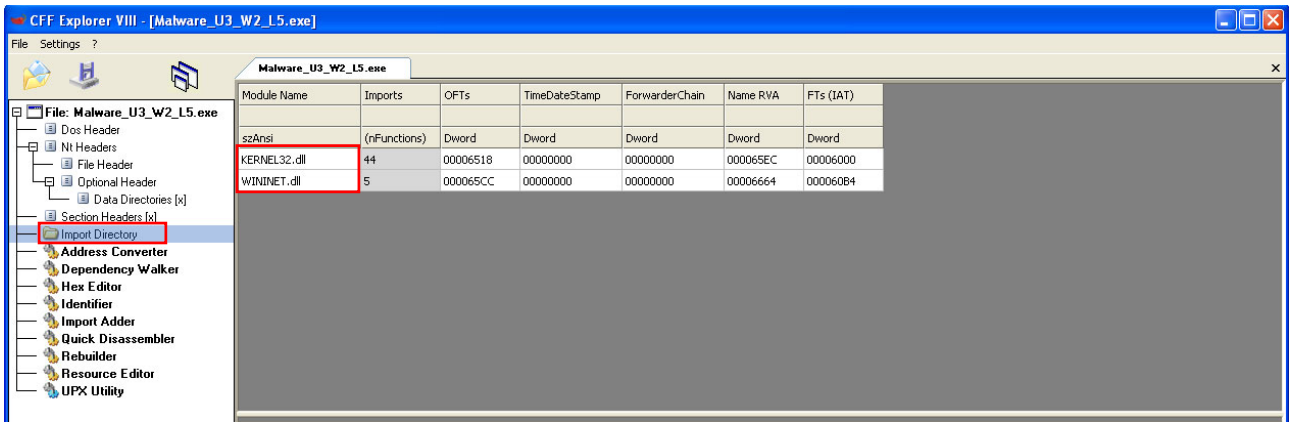
## FASE 2

Nella seconda fase iniziamo ad analizzare il file malevolo. Per prima cosa carichiamo il file su CFF Explorer



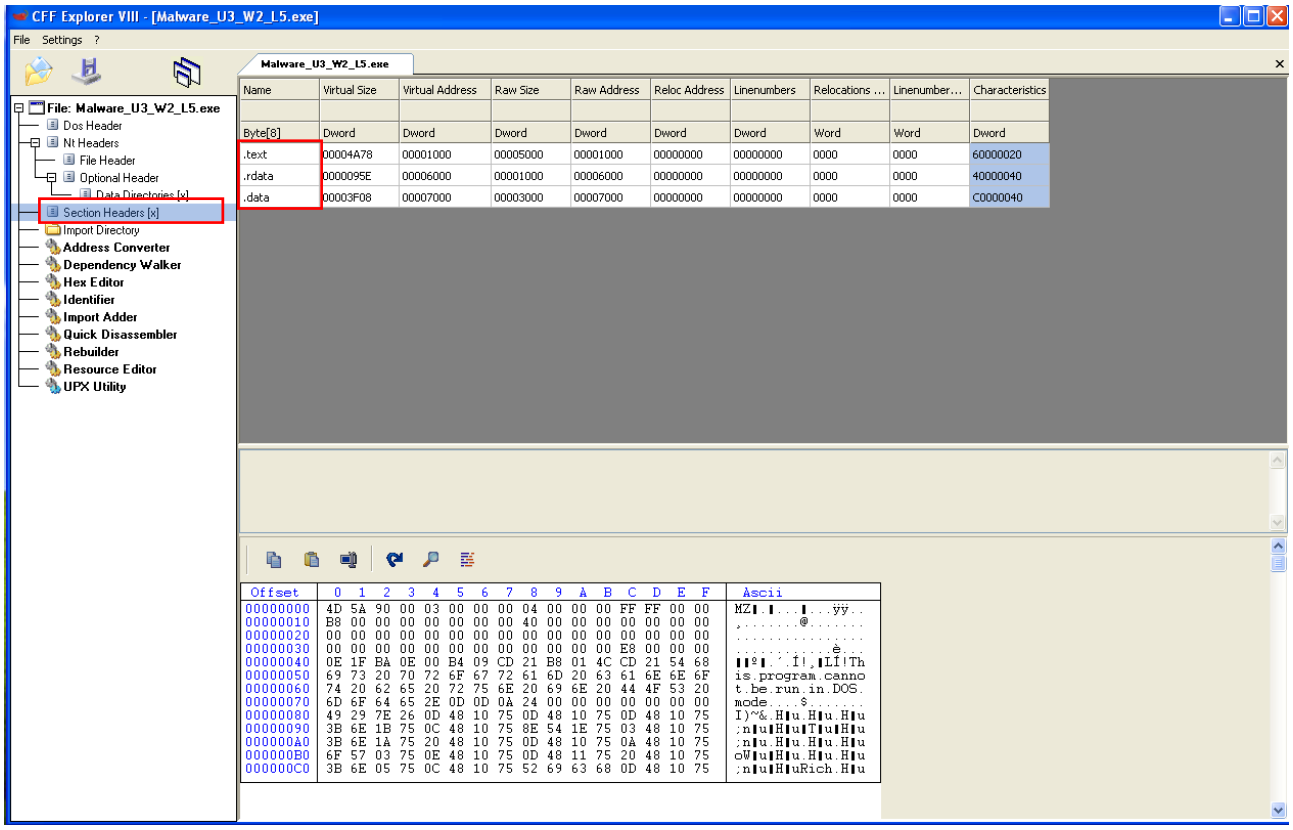




Nella sezione "**Import Directory**" possiamo vedere le Librerie importate dal nostro eseguibile




- **KERNEL32.dll** – librerie delle funzioni principale per interagire con il sistema operativo (gestione memoria / manipolazione file)
- **WININET.dll** – libreria delle funzioni per protocolli di rete (http/FTP)

Spostandoci invece nella sezione “**Section Headers**” possiamo vedere di quali sezioni si compone il nostro eseguibile




**.text** – sezione che contiene le righe di codice che la CPU eseguirà

**.rdata** – sezione che contiene le variabili importate

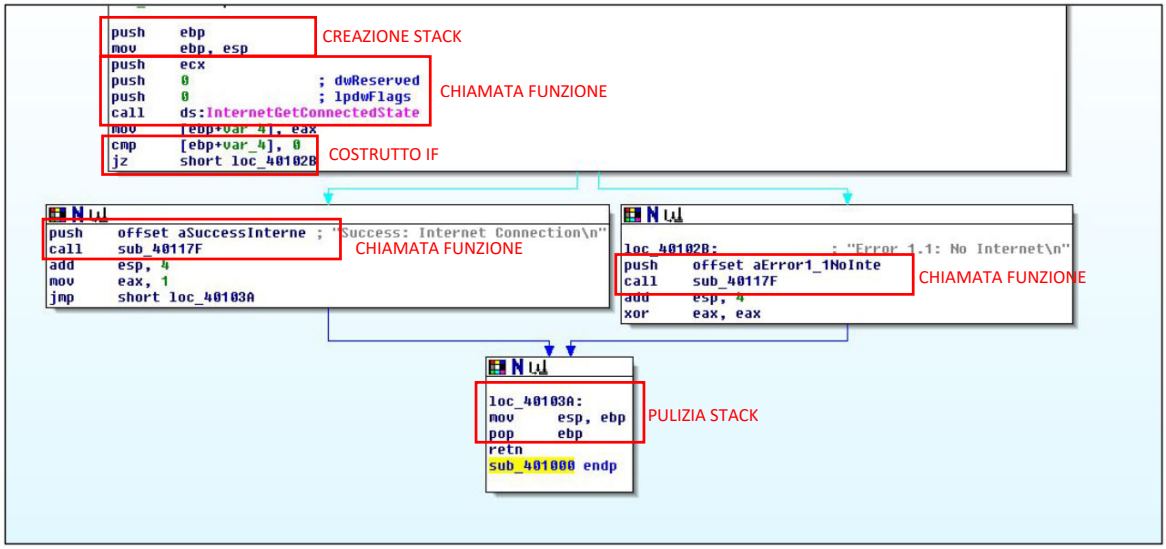
**.data** – sezione che contiene variabili globali

## PUNTO 2

Con riferimento alla figura 1

- **Identificare i costrutti noti**
- **Ipotizzare il comportamento della funzionalità implementata**

**Figura 1**



# COSRTUTTI IDENTIFICATI

- ## CREAZIONE STACK

```
# Push ebp
# Mov ebp, esp
```

- ## IF

```
# Cmp [ebp+var_4],  0
# Jz short loc_40102B
```

- ## CHIAMATA DI FUNZIONE

```
# Push exc
# Push 0 ; dwReserved
# Push 0; lpdwFlags
# Call  ds:InternetGetConnectedState
************************
# Push offset aSuccesInterne ; “Succes: Internet Connection\n”
# Call sub_40117F
************************
# Push offset aError1_1noInte
# Call sub_40117F
```

- ## PULIRE STACK

```
# Loc_40103A:
# Mov esp, ebp
# Pop ebp
```

## IPOTESI

In conclusione possiamo dire quasi con certezza che il nostro file malevolo controlla se la macchina su cui viene eseguito abbia una connessione ad internet o meno, mandando in stampa l’esito del controllo.

## BONUS

**Push ebp** - passare i paramentri ebp in cima allo stack
**Mov ebp, esp** - copia il valore della sorgente (esp), nel destinatrio (ebp)
**Push exc** - passare i paramentri exc in cima allo stack
**Push 0** ; **dwReserved** - passare il valore dei paramentri (valore0) di una variabile
**Push 0**; **lpdwFlags** - passare il valore dei paramentri (valore0) di una variabile - ; commento
**Call ds:InternetGetConnectedState** – chiamata di funzione tramite **ds**: che crea un nuovo EIP per la funzione chiamata (https://stackoverflow.com/questions/3819699/what-does-ds40207a-mean-in-assembly)
**Mov [ebp+var_4], - eax** - copia il valore della sorgente (eax), nel destinatrio (ebp+var_4)
**Cmp [ebp+var_4], 0** - mette a confronto destinatario e sorgente per restituire un valore binario
**Jz short loc_40102B** - (salta alla locazione di memoria specificata se ZF=1) - salta alla locazione di memoria se ZF = 1 riferito al cmp precedente

**Push offset aSuccesInterne ; "Succes: Internet Connection\n"** – passa l'alloggio di memoria aSuccesInterne in cima allo stack
**Call sub_40105F -** chiamata di funzione
**Add esp, 4** - somma il valore della sorgente (4) con quello del destinatario (esp)
**Mov eax, 1** - copia il valore della sorgente (1), nel destinatrio (eax)
**Jmp shot loc_40103A** - fa un salto all'istruzione nella locazione data (40103A)
**40102B ----------** - istruzione dove il jz "salta"

**Push offset_aError1_1NoInter** - passa l'alloggio di memoria **offset_aError1_1NoInter** in cima allo stack
**Call sub_40117F** - chiamata di funzione
**add esp, 4** - somma il valore della sorgente (4) con quello del destinatario (esp)
**xor eax,eax** - inizializza il valore di eax a zero tramite operartore logico **XOR** (se i valori sono uguali restituisce 0)

**Mov esp,ebp** - copia il valore della sorgente (ebp), nel destinatrio (esp)
**pop ebp** – rimozione dello stack
**retn** – pone fine ad una procedura
(https://stackoverflow.com/questions/1396909/ret-retn-retf-how-to-use-them)
**sub_401000 endp** - Segna la fine procedura precedentemente nominata (**sub_401000**)
https://learn.microsoft.com/en-us/cpp/assembler/masm/endp?view=msvc-170