



FLORES ALEX

# EPICODE

S10

# Indice

---

- 01 Traccia
  - 02 La macchina virtuale
  - 03 Cosa è un malware?
  - 04 Librerie e sezioni
  - 05 Identificazione dei costrutti noti
  - 06 Comportamento del codice Assembly
  - 07 Conclusioni finali
-

# Traccia

# 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:

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

Con riferimento alla figura che mostrerò successivamente, rispondere ai seguenti quesiti:

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

# La macchina virtuale

# La macchina virtuale

Una macchina virtuale basata su Windows XP è un ambiente simulato che emula il sistema operativo Windows XP su un computer ospite moderno. Questa macchina virtuale consente agli utenti di eseguire applicazioni e programmi progettati per Windows XP su hardware più recente, senza dover installare direttamente questo sistema operativo obsoleto sul proprio computer principale.

All'interno di questa macchina virtuale, è possibile eseguire le stesse attività e utilizzare le stesse applicazioni che sarebbero disponibili su un computer con Windows XP nativo. Questo permette agli utenti di sperimentare e testare diverse configurazioni senza influenzare il sistema operativo principale del computer. Nel nostro caso Epicode ci ha fornito una versione già completa di tutti i tool ed i malware che andremo ad analizzare in questo ultimo mese.



# Cosa è un malware?

di che tipologia sono?

# Cosa è un malware?

**Un malware, abbreviazione di "software dannoso" (malicious software), è un programma progettato per infiltrarsi o danneggiare un computer, un sistema informatico o una rete senza il consenso dell'utente e può causare una serie di problemi, tra cui rallentamento del computer, perdita di dati, furto di informazioni personali, danni finanziari e compromissione della sicurezza delle reti.**

**Esistono vari tipi di malware, ognuno con scopi e comportamenti diversi che vedremo nelle slide successive.**

# Malware: di che tipologia sono?

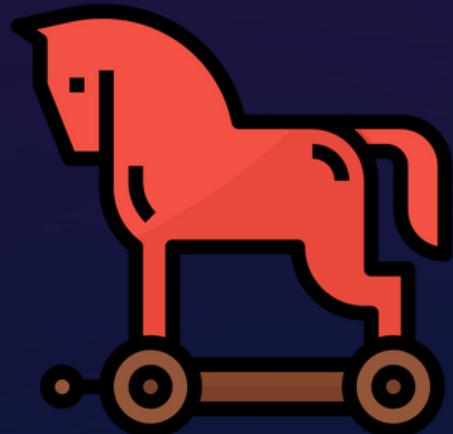
- Virus: Il virus è un tipo di malware che si attacca a file eseguibili e si replica quando il file viene eseguito. Può danneggiare o alterare i file o il sistema operativo.
- Worm: I worm sono malware progettati per replicarsi e diffondersi autonomamente attraverso reti informatiche, sfruttando vulnerabilità nel sistema operativo o nelle applicazioni.
- Trojan: Un trojan, o cavallo di Troia, è un tipo di malware che si maschera da software legittimo per ingannare gli utenti e ottenere l'accesso non autorizzato al sistema. Possono essere utilizzati per rubare dati sensibili, installare backdoor o altri malware, o danneggiare il sistema.



**Virus malware**



**Worm malware**



**Trojan malware**

# Malware: di che tipologia sono?

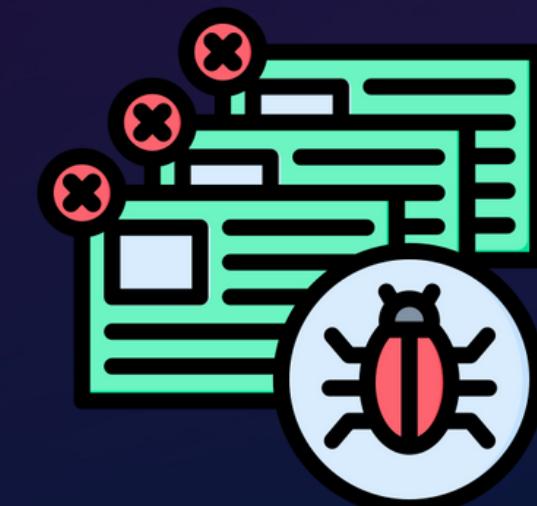
- Spyware: Lo spyware è progettato per monitorare e raccogliere informazioni sugli utenti senza il loro consenso. Può registrare attività di navigazione, tasti premuti, informazioni personali e altro ancora, per scopi di marketing, furto di identità o spionaggio.
- Ransomware: Il ransomware è un tipo di malware che blocca l'accesso ai file o al sistema dell'utente e richiede un pagamento (ransom) per ripristinare l'accesso. Può crittografare i file dell'utente o bloccare l'intero sistema, minacciando di distruggere o rendere inaccessibili i dati se il pagamento non viene effettuato.
- Adware: L'adware è progettato per visualizzare pubblicità indesiderate sul computer dell'utente, spesso generando entrate per gli autori del malware attraverso clic fraudolenti o installazioni di software sponsorizzati.



**Spyware malware**



**Ransomware malware**



**Adware malware**

# Librerie e Sezioni

CFF Explorer

## CFF Explorer

CFF Explorer è uno strumento potente e flessibile utilizzato principalmente per analizzare ed esplorare file eseguibili, librerie dinamiche (DLL) e altri file binari in ambiente Windows. Quando si utilizza CFF Explorer per analizzare un malware, lo strumento rivela tutta una serie di informazioni cruciali che consentono agli analisti di comprendere appieno il funzionamento e l'impatto del software dannoso come per esempio le librerie utilizzate dal malware e le sue sezioni



## Le librerie

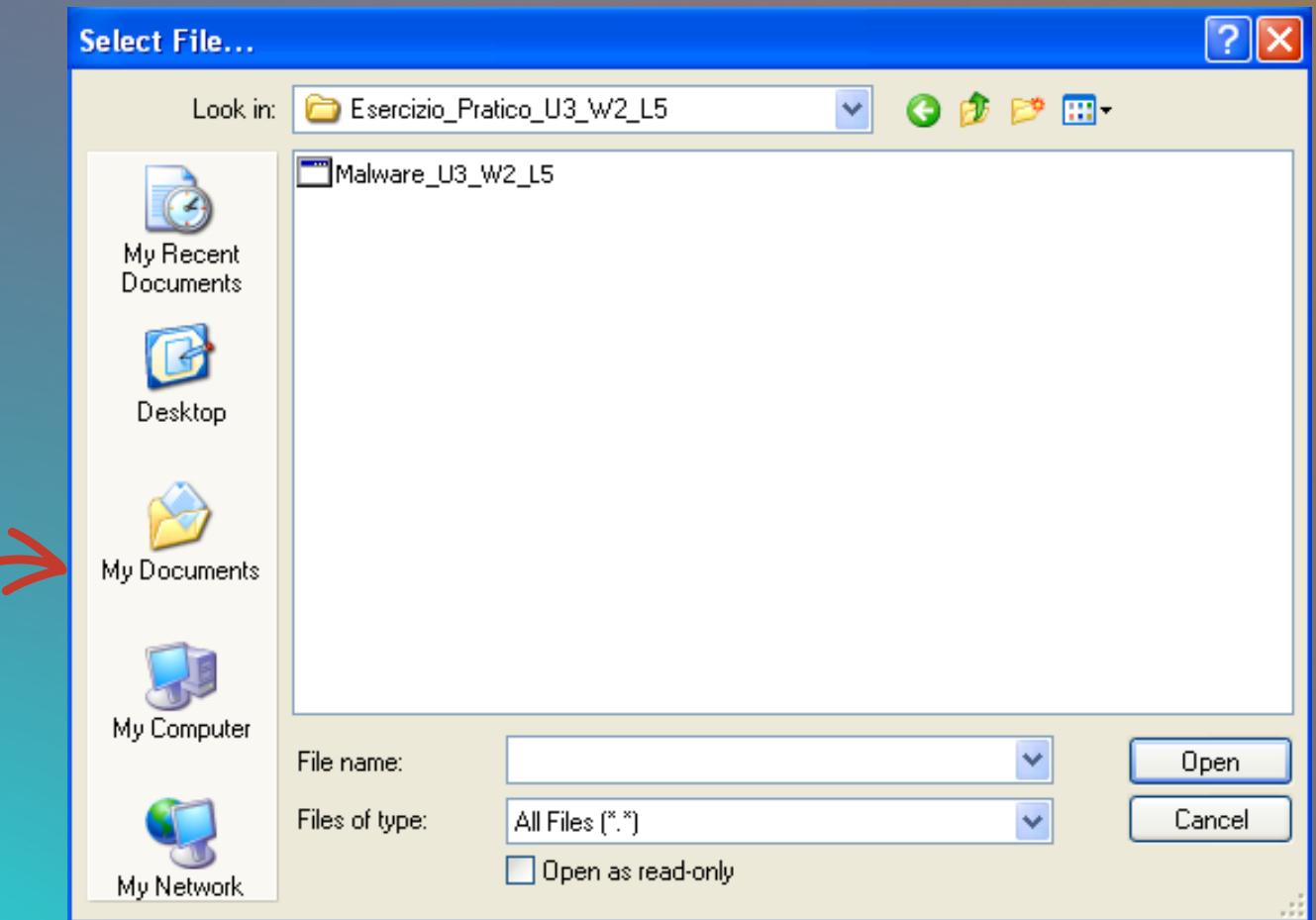
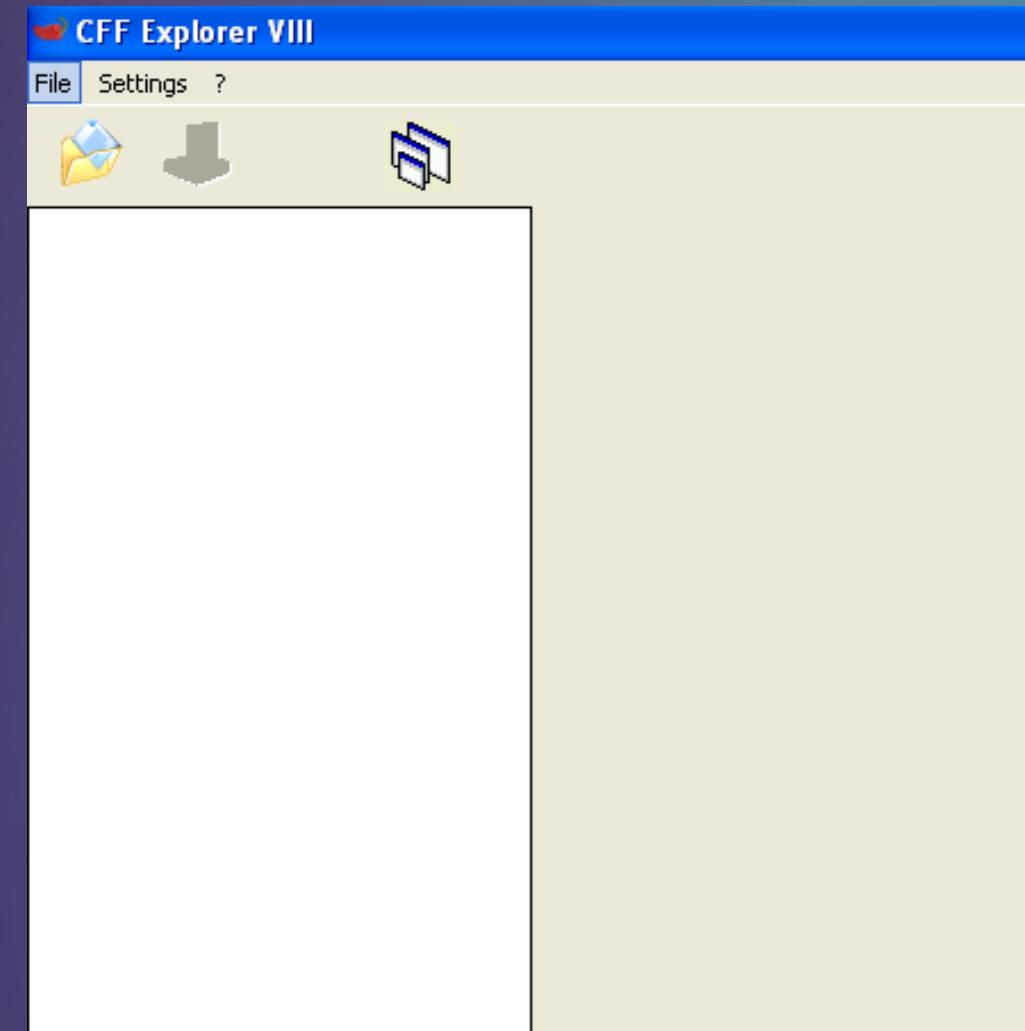
Le librerie dinamiche (DLL) sono file contenenti codice e risorse che possono essere richiamati e utilizzati da altri programmi in esecuzione sul sistema operativo Windows. I malware spesso sfruttano DLL per eseguire funzionalità specifiche senza dover incorporare tutto il codice nel file eseguibile principale. Questo li rende più furtivi e leggeri. I malware possono caricare DLL dannose o manipolare le DLL di sistema per ottenere accesso privilegiato, rubare informazioni o eseguire altre azioni dannose.

# Le librerie

Quindi per andare ad analizzare le librerie utilizzate dall'eseguibile "Malware\_U3\_W2\_L5" useremo CFF Explorer:



clicchiamo sull''icona a forma di cartella

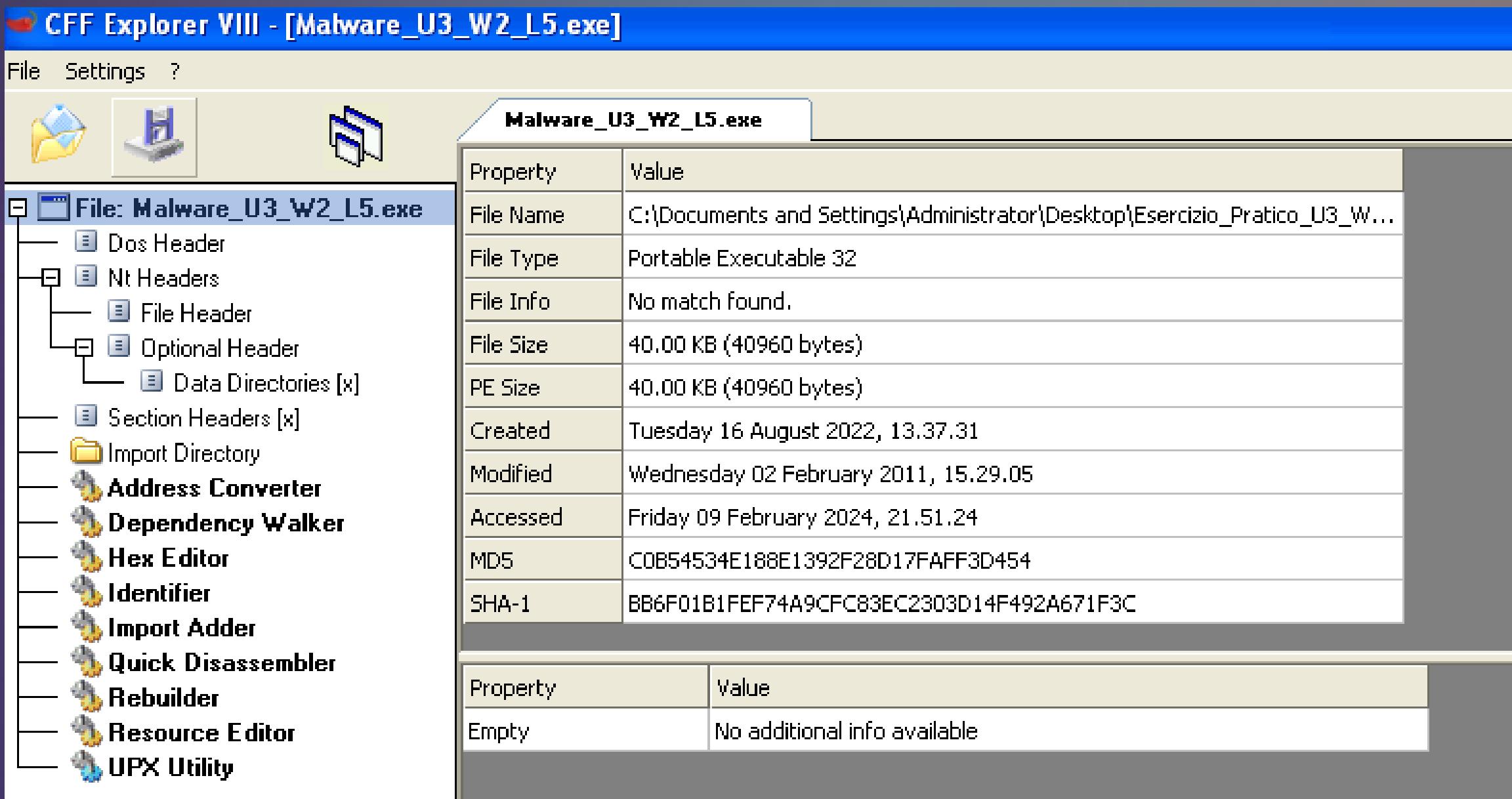


selezioniamo il file eseguibile "Malware\_U3\_W2\_L5"

# Le librerie

Quindi per andare ad analizzare le librerie utilizzate dall'eseguibile "Malware\_U3\_W2\_L5" useremo CFF Explorer:

VISUALIZZEREMO UNA PAGINA SIMILE ALLA SEGUENTE



Selezioniamo "**Import Directory**" dal pannello a sinistra per visualizzare tutte le librerie utilizzate dal malware

# Le librerie

Quindi per andare ad analizzare le librerie utilizzate dall'eseguibile "Malware\_U3\_W2\_L5" useremo CFF Explorer:

Malware_U3_W2_L5.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

**Le librerie utilizzate dal malware sono le seguenti:**

- **Kernel32.dll, che include le funzioni core del sistema operativo**
- **Wininet.dll, include le funzione per implementare i servizi di rete come ftp, ntp, http**

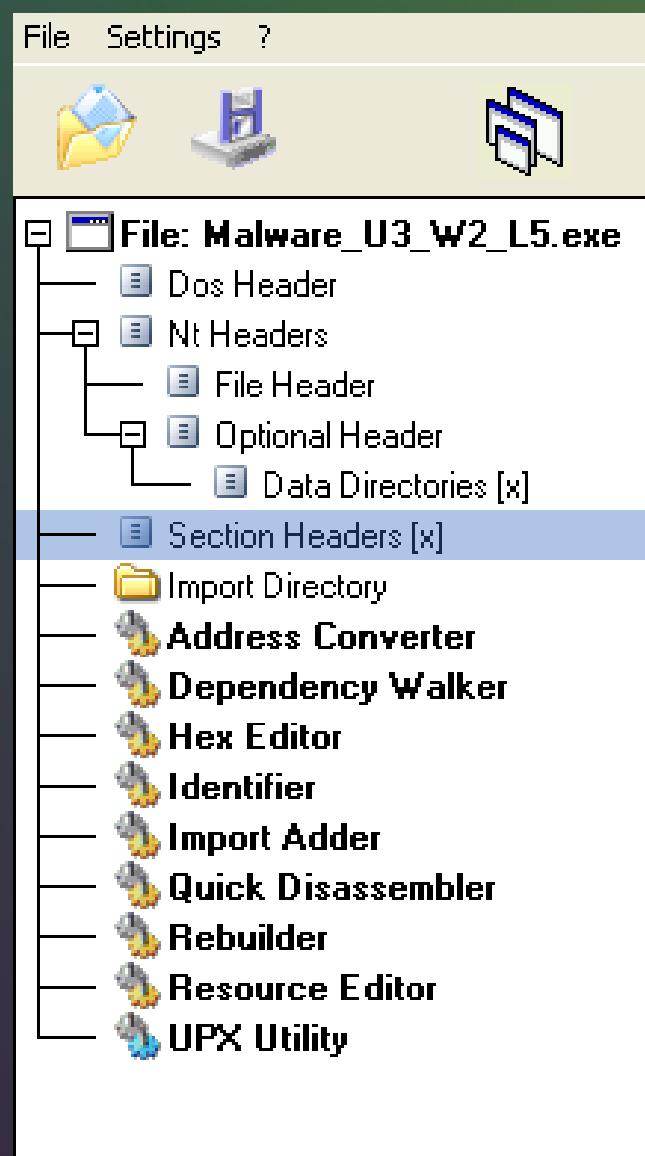
## Le sezioni

Le sezioni in un file eseguibile rappresentano parti specifiche del codice, dei dati o delle risorse. Le sezioni possono includere il codice eseguibile, i dati inizializzati e non inizializzati, le risorse come immagini, icone o stringhe, e altro ancora. I malware possono utilizzare sezioni per organizzare e nascondere il proprio codice, crittografare parti del payload o incorporare risorse aggiuntive utilizzate nel corso dell'esecuzione. Manipolando le sezioni, i malware possono evitare la rilevazione da parte degli strumenti di sicurezza e complicare l'analisi da parte degli investigatori.

# Le sezioni

Quindi per andare ad analizzare le sezioni utilizzate dall'eseguibile "Malware\_U3\_W2\_L5" useremo CFF Explorer:

Selezioniamo la voce "Section Headers [x]" nel menù alla nostra sinistra



# Le sezioni

Quindi per andare ad analizzare le sezioni utilizzate dall'eseguibile "Malware\_U3\_W2\_L5" useremo CFF Explorer:

Malware_U3_W2_L5.exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber... Characteristics	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

**Le sezioni utilizzate dal malware sono le seguenti:**

- **.text:** la sezione «text» contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato.
- **.rdata:** la sezione «rdata» include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile
- **.data:** la sezione «data» contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.

# Identificazione dei costrutti noti

## I costrutti noti

In assembly, i "costrutti noti" possono essere equiparati ai mattoncini LEGO permettendoti di fare le seguenti cose:

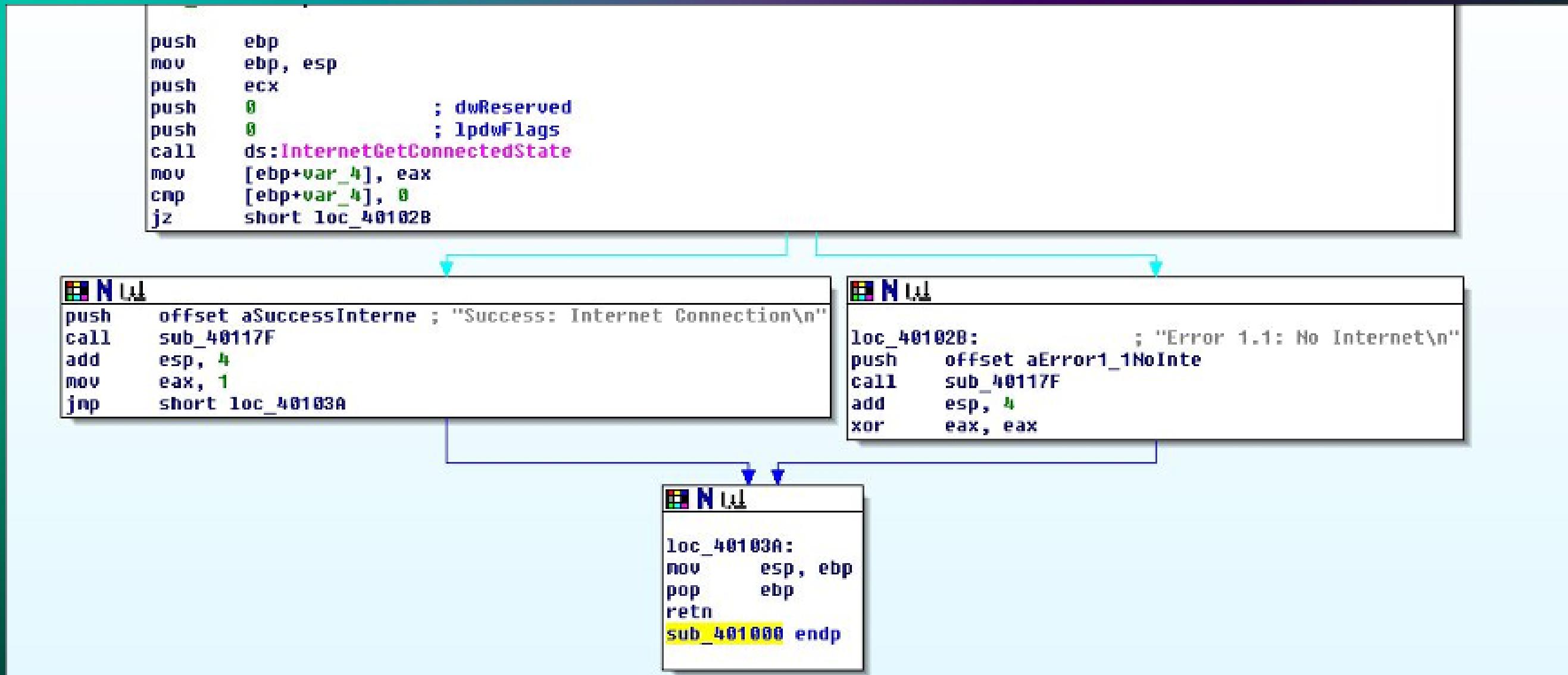
- **Carica e Salva (Load and Store):** Prendi dati dalla memoria o li metti dentro.  
Come prendere i mattoncini dallo scaffale e poi li rimetti al loro posto.
- **Muovi i Dati:** Copia i dati da un posto all'altro. È come spostare i mattoncini da un posto all'altro sul tavolo.
- **Operazioni Matematiche e Logiche:** Fai calcoli o combini i mattoncini in modi diversi (aggiungi, sottrai, moltiplica..)  
Come unire o separare i blocchi
- **Salta:** Decidi cosa fare in base a condizioni o vai in un altro posto nel tuo progetto. È come decidere se continuare a costruire una torre o passare a un'altra parte del tuo progetto.
- **Chiamate di Funzione e Stack:** Quando hai un compito che vuoi fare più volte, puoi "chiamare" una funzione. È come chiamare un amico per aiutarti. Lo "stack" è come una pila di piatti, dove metti temporaneamente le cose mentre lavori su qualcos'altro.

Questi costrutti sono le basi su cui si costruiscono i progetti di assembly, proprio come quando usi mattoncini LEGO per creare diverse strutture.

---

## I costrutti noti

La traccia del progetto richiede di identificare i costrutti noti del seguente codice in linguaggio assembly:



## I costrutti noti

All'interno del programma ho identificato 6 costrutti noti e sono i seguenti:

- il primo è per la crezione dello stack e comprende la seguente coppia di istruzioni :

```
push    ebp  
mov    ebp, esp
```

- il secondo è per l'utilizzo della funzione "internet get connected state"

```
push    ecx  
push    0          ; dwReserved  
push    0          ; lpdwFlags  
call    ds:InternetGetConnectedState
```

- il terzo è un costrutto IF che controlla il valore di ritorno di "internet get connected state" con 0, se il valore sarà uguale a 0 salterà all'indirizzo di memoria specificato da jz .

```
cmp    [ebp+var_4], 0  
jz    short loc_40102B
```

## I costrutti noti

All'interno del programma ho identificato 6 costrutti noti e sono i seguenti:

- il quarto è un costrutto printf che stampa a schermo un messaggio nel caso in cui non è stato effettuato il jump

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
```

- il quinto è un costrutto printf nel caso in cui il jump sia avvenuto stampando quindi un messaggio su schermo

```
loc_40102B:          ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
```

- il sesto svuota lo stack e termina il programma

```
mov     esp, ebp
pop    ebp
ret
sub_401000 endp
```

# Comportamento del codice assembly

# Comportamento del codice

**Il seguente codice assembly è stato progettato per verificare la presenza di una connessione Internet su una macchina. Utilizzando la funzione "getinternetconnectstate", il programma controlla lo stato della connessione.**

**Il costrutto IF esegue un controllo condizionale sul valore restituito dalla funzione "getinternetconnectstate". Se il valore è uguale a 0, il programma visualizza il messaggio "No internet" sullo schermo e termina l'esecuzione.**

**Altrimenti, se il valore è diverso da 0, il programma visualizza il messaggio "Success: Internet Connection" prima di completare la sua esecuzione.**

**Questo codice fornisce un modo semplice ed efficiente per verificare lo stato della connessione Internet su una macchina e informare l'utente di conseguenza.**

# Conclusioni finali

Durante il corso del progetto, abbiamo intrapreso un'analisi approfondita di un malware e di un codice assembly al fine di comprendere i loro comportamenti, le loro funzionalità e le loro possibili implicazioni per la sicurezza informatica.