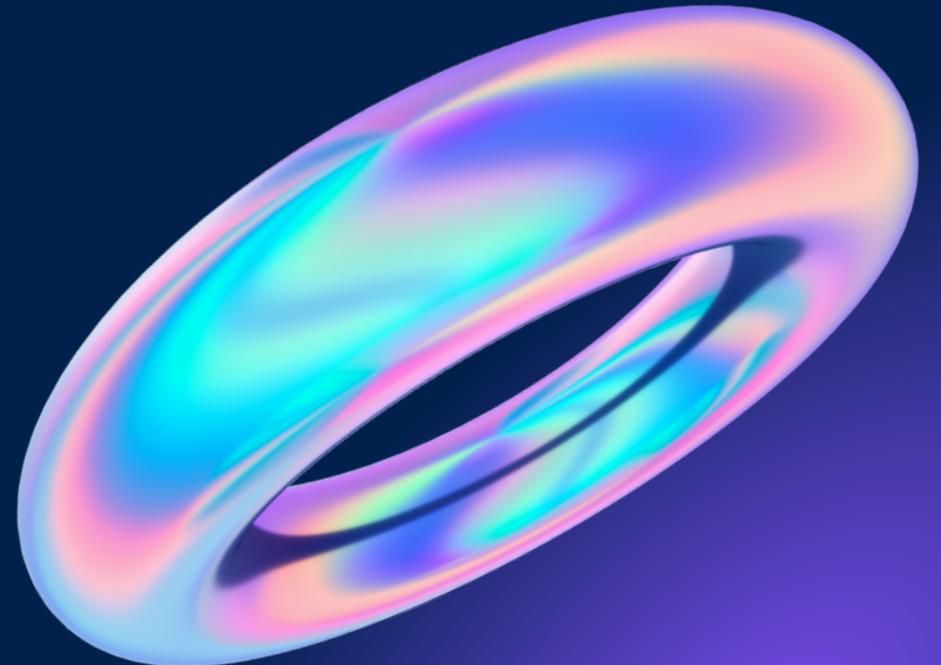




# Progetto S10-L5

malware analysis



# INDICE

- **traccia + codice malware (pag. 3)**
- **introduzione (pag. 4->7)**
- **svolgimento fase 1 (pag. 8-> 10)**
- **svolgimento fase 2 (pag. 11->13)**



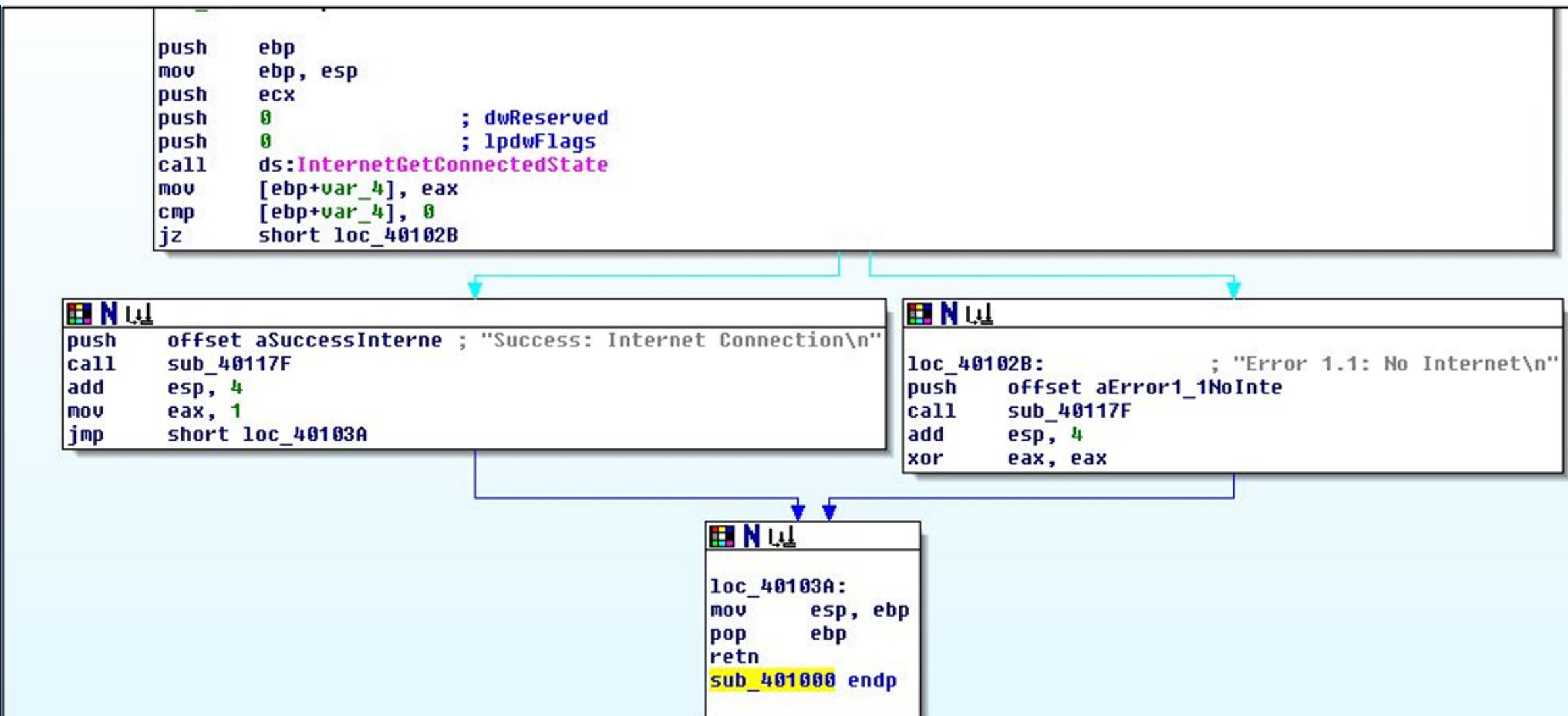
### Traccia:

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 in slide 3, risponde ai seguenti quesiti:

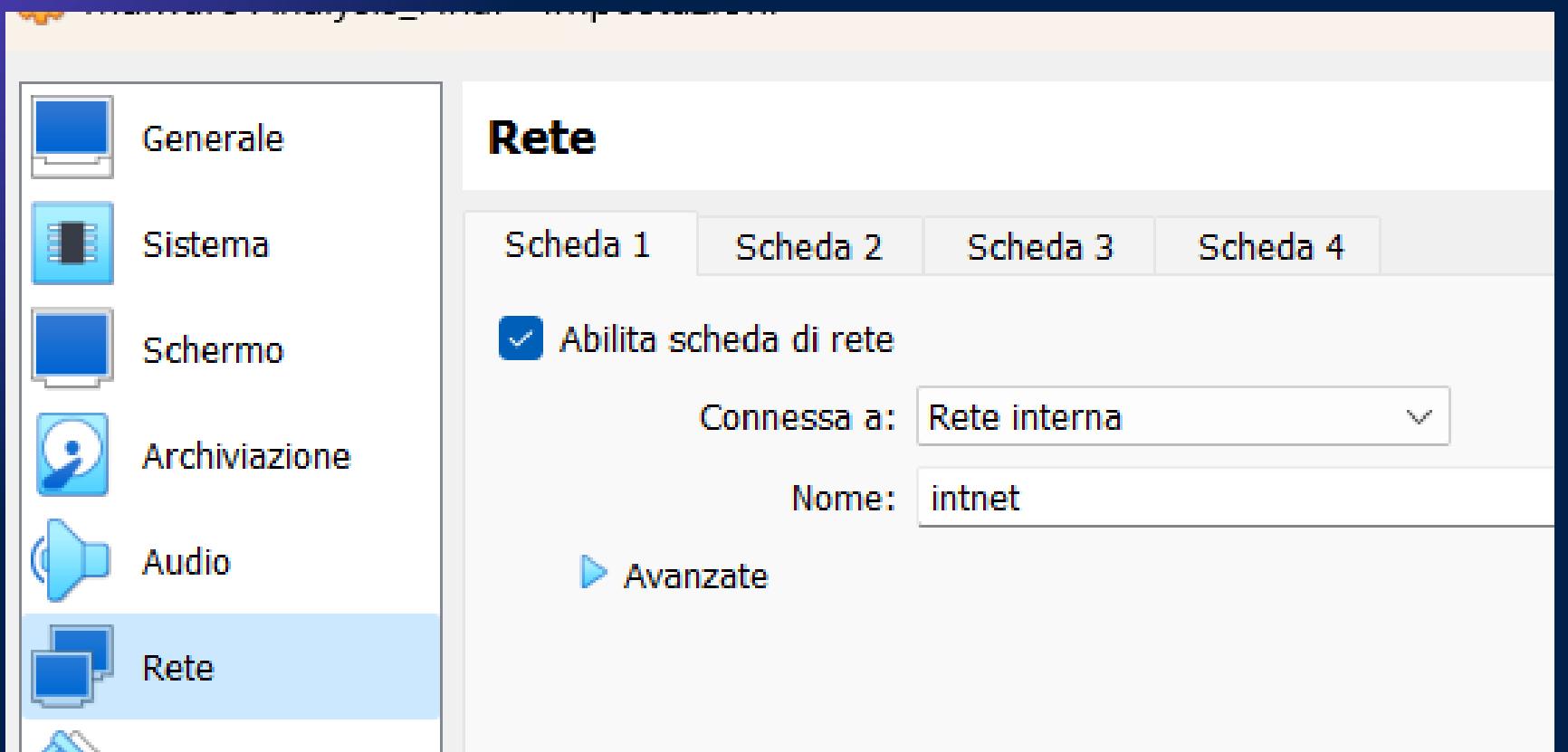
- Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)
- Ipotizzare il comportamento della funzionalità implementata



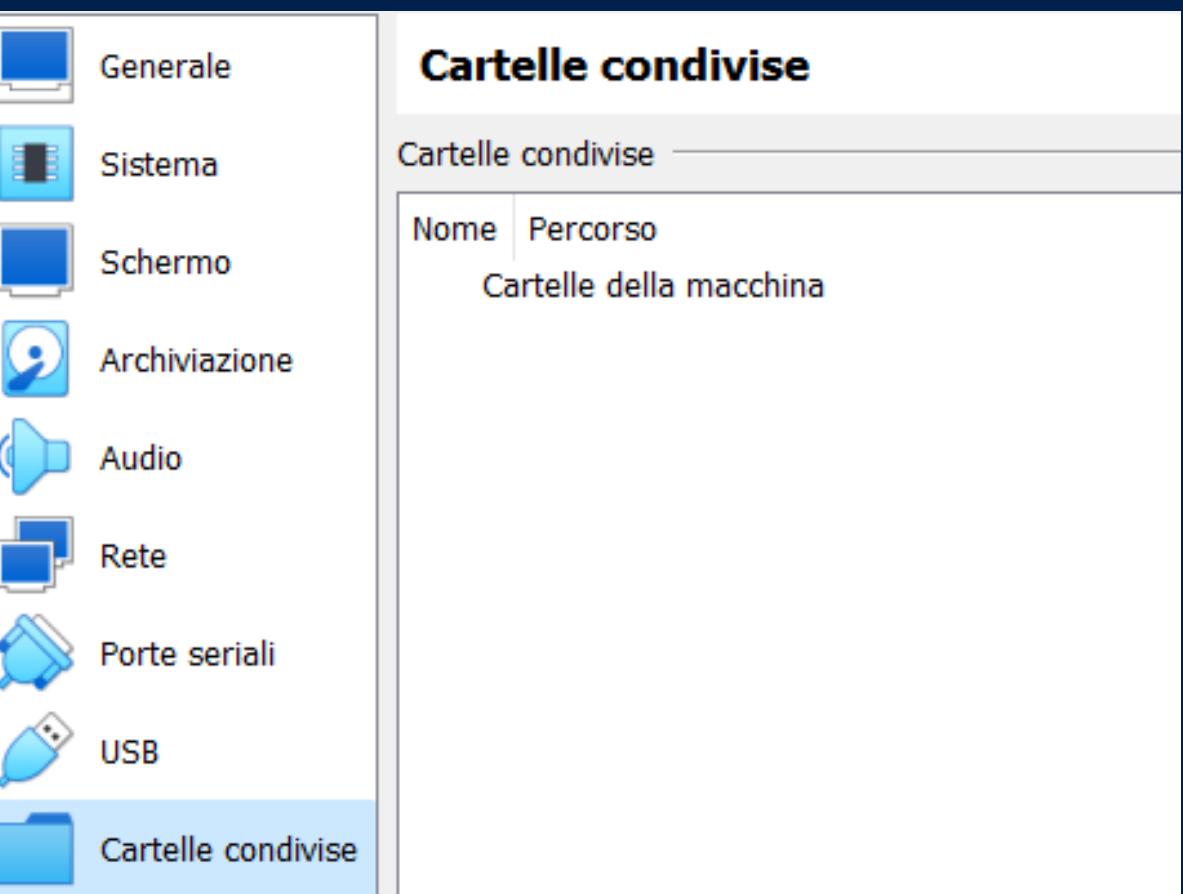
# introduzione

**Prima di iniziare assicuriamoci di avere la macchina in sicurezza così da evitare che il malware si propaghi sulla nostra macchina fisica o che comunque crei danni, ciò va fatto con i seguenti step:**

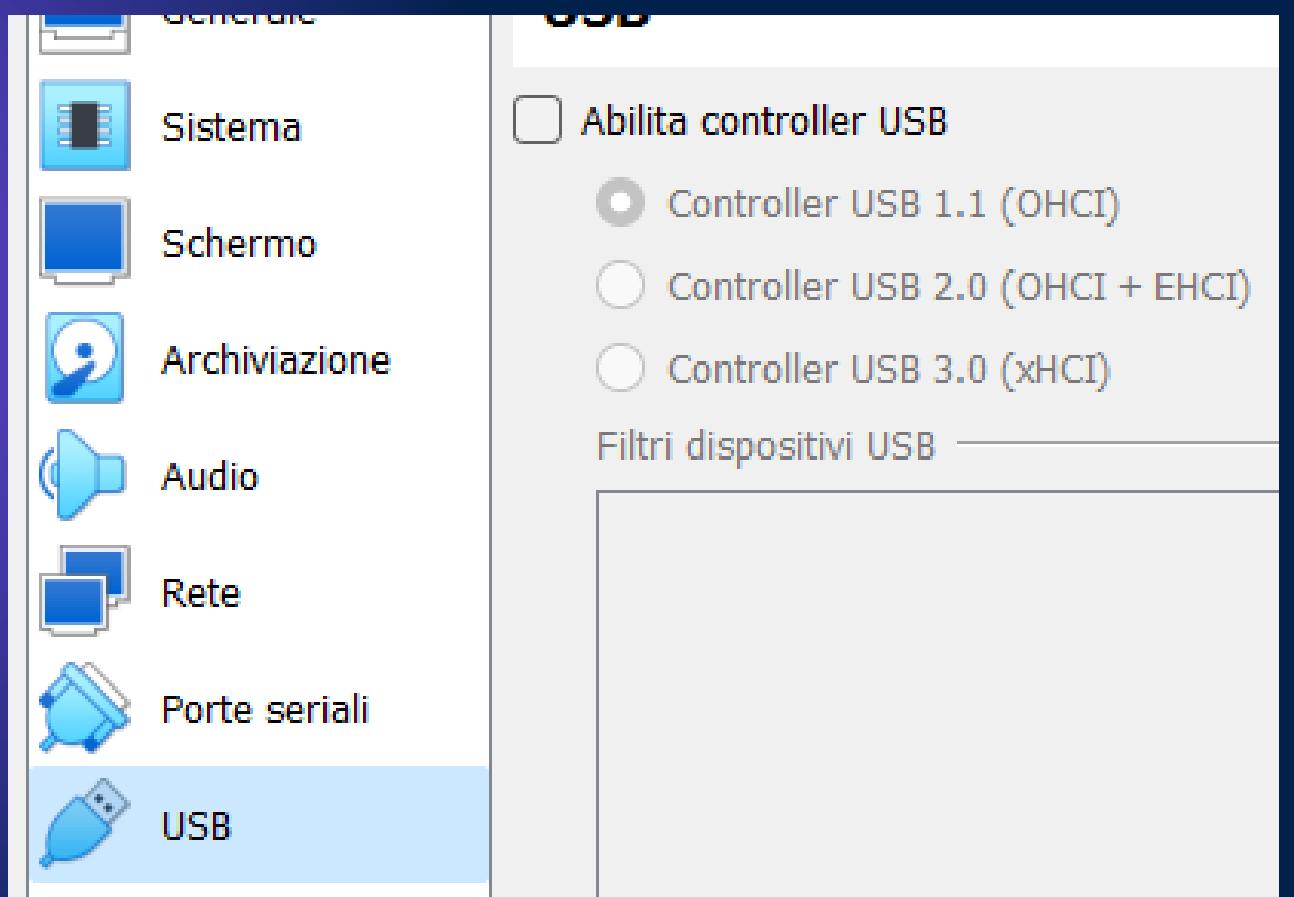
**1. assicuriamoci che la macchina sia impostata su rete interna**



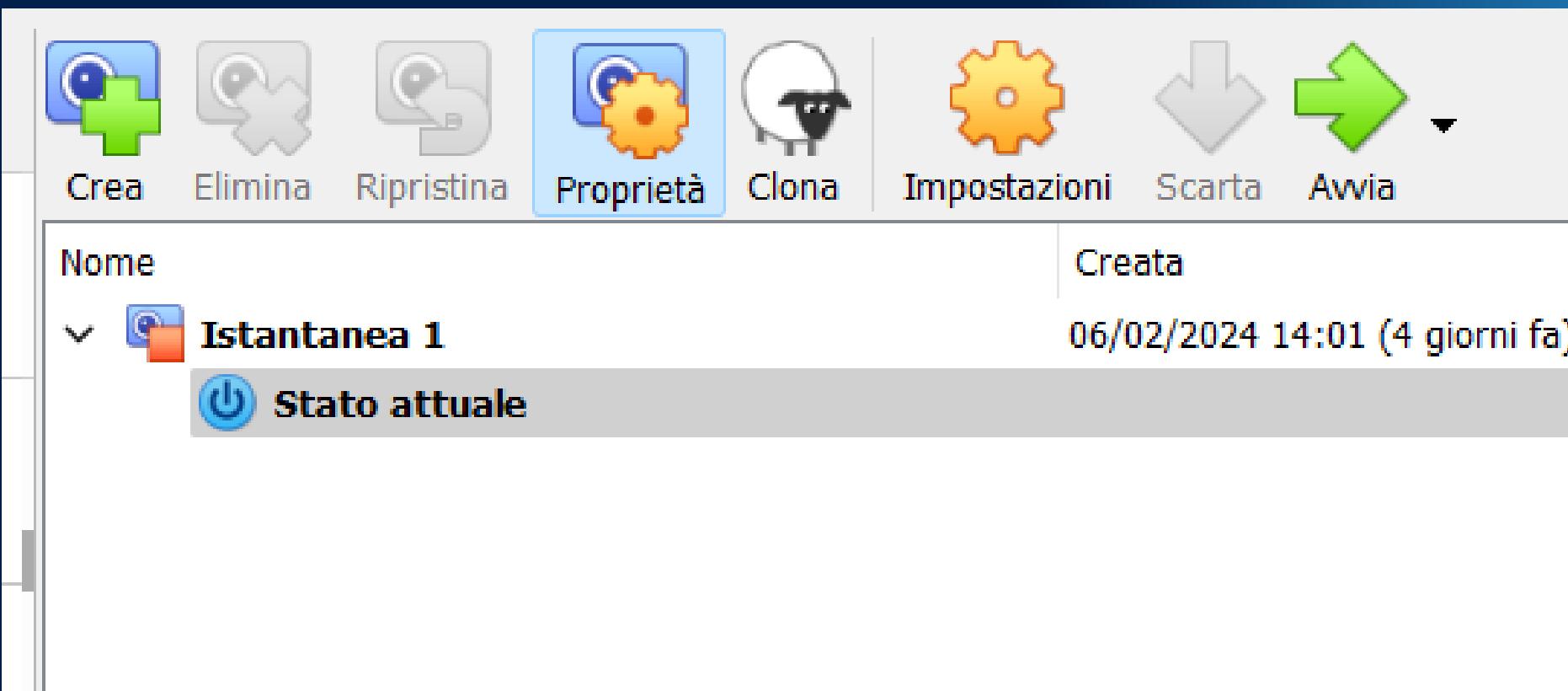
**2. controlliamo che non ci siano cartelle condivise tra la macchina virtuale e la nostra**



### 3. verifichiamo che i dispositivi **usb** siano disabilitati



4. creiamo un **istantanea** della macchina così da avere una sorta di backup nel caso le cose si mettano male. Questo ci permette di tornare alla macchina virtuale prima dell'esecuzione del malware



**Una volta avviata la macchina  
andremo ad avviare il tool che  
useremo per analizzare il  
malware CFF Explorer**



**CFF explorer:**

**E' un software che useremo per  
eseguire un' analisi statica del  
malware così da poter verificare quali  
librerie sono state importate o quali  
sono le sezioni dello stesso. Cio' ci  
permette, in minima parte, di capire  
come funziona.**

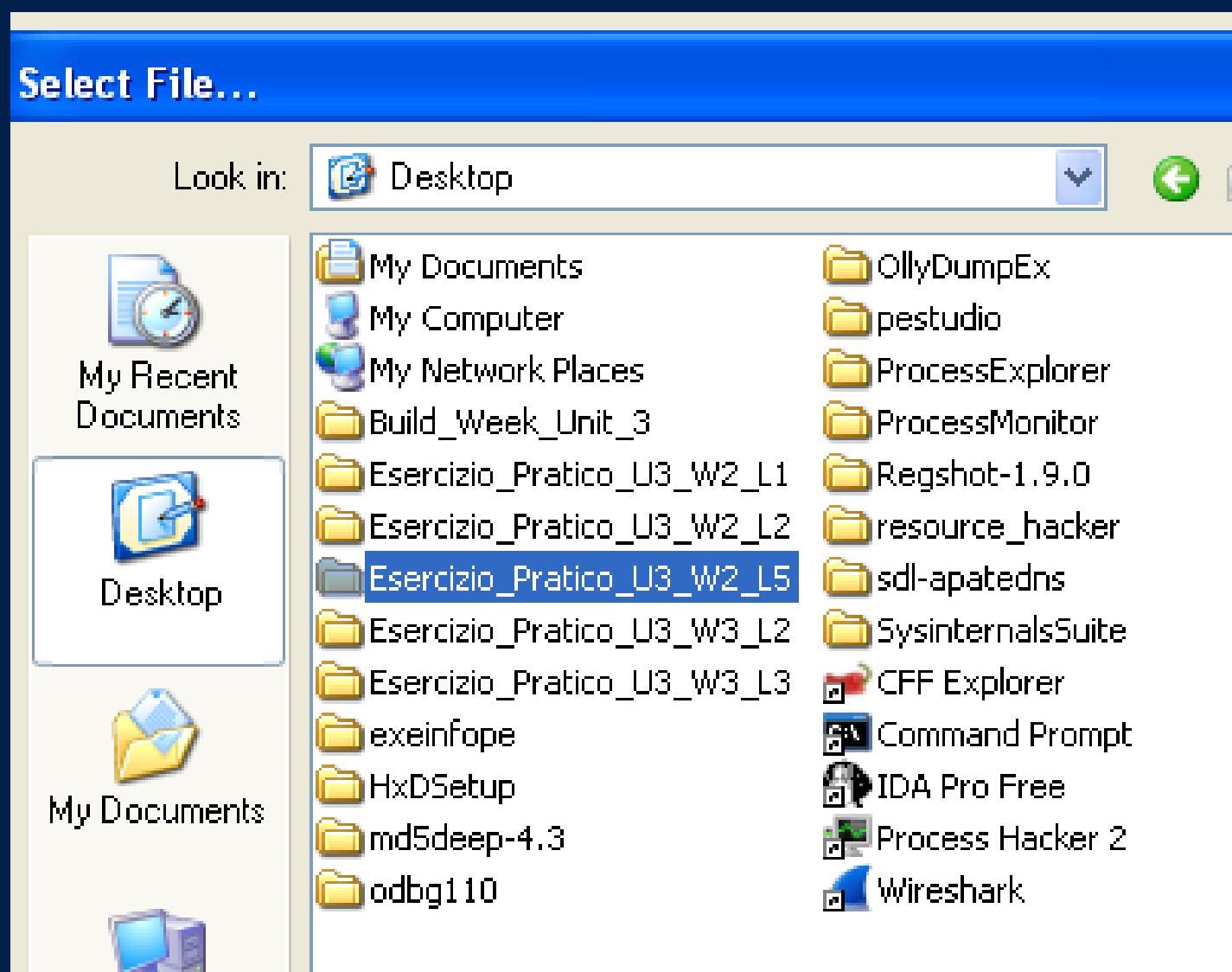
**Per avviarlo basta un doppio clic  
sull'icona**

# Svolgimento Fase 1

Per avviare l'analisi statica dovremo selezionare la cartella in cui e' presente l'eseguibile del malware cio' viene fatto coi seguenti step:



1. selezionare l'icona della cartella in alto a sinistra



2. selezionare l'icona desktop e poi selezionare la cartella a noi interessata, in questo caso **Esercizio\_Pratico\_U3\_W2\_L5**

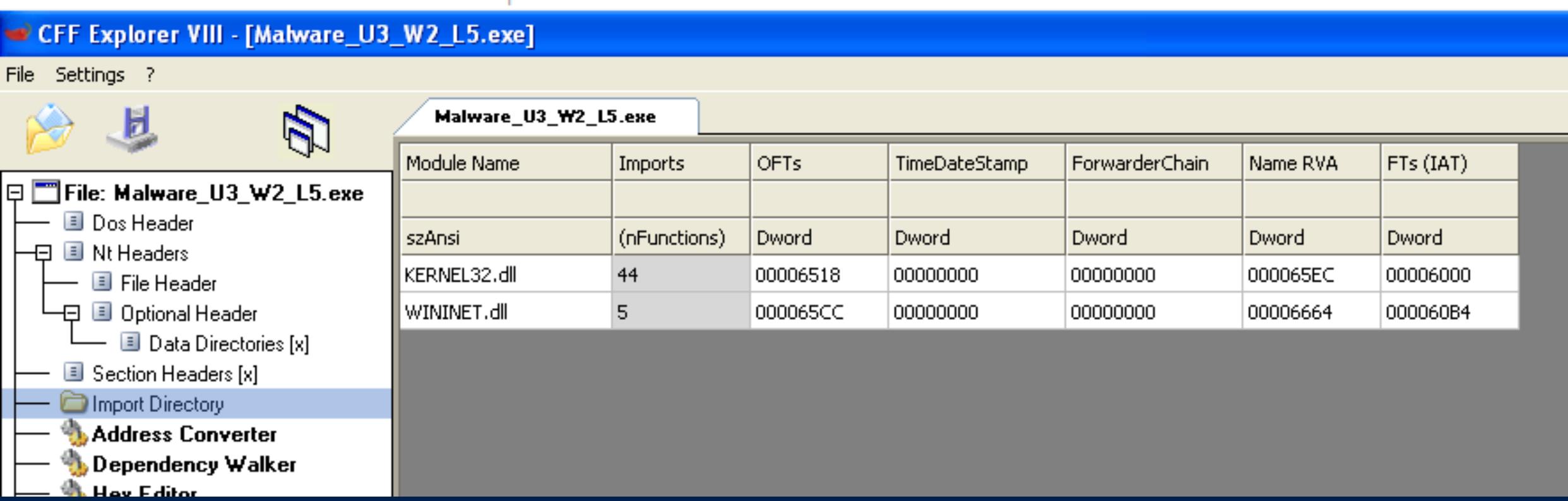
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword	
.text	00004478	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

una volta avviato su **CFF Explorer** selezioniamo la voce **Section Headers[x]** questo ci farà vedere le sezioni in cui è diviso il malware. Possiamo notare le seguenti:

**.txt:** Questa sezione contiene le righe di istruzione del malware che verranno eseguite dalla CPU una volta avviato

**.rdata:** di solito questa sezione contiene dati del malware come le librerie importate o le funzioni del codice. Tutto ciò può essere rilevato con **CFF Explorer**

**.data:** Questa sezione invece contiene le variabili globali del programma che devono essere disponibili



**selezionando invece la cartella, import Directory ci troveremo davanti le librerie importate dal programma nel nostro caso sono le seguenti:**

**KERNEL32.dll:** Una libreria in cui sono contenute le funzioni principali di un sistema operativo , cio' puo' dunque permettere la manipolazion di spazi di memoria o dei file se usata in modo improrio

**WININET.dll:** Questa libreria contiene le funzioni per i protocolli di rete HTTP, NPT e FTP

# Svolgimento fase 2 (costrutti noti)

il linguaggio usato per questo malware e'  
**Assembly**, un linguaggio a basso livello  
considerato tale per via del fatto che le  
**istruzioni sono correlate alla CPU e alle  
istruzioni della macchina**

```
push    ebp  
mov     ebp, esp  
push    ecx  
push    0          ; dwReserved  
push    0          ; lpdwFlags  
call    ds:InternetGetConnectedState  
mov     [ebp+var_4], eax  
cmp     [ebp+var_4], 0  
jz      short loc_40102B
```

```
[NUL]  
push    offset aSuccessInterne ; "Success: Internet Connection\n"  
call    sub_40117F  
add     esp, 4  
mov     eax, 1  
jmp    short loc_40103A
```

```
[NUL]  
loc_40102B:           ; "Error 1.1: No Internet\n"  
push    offset aError1_1NoInte  
call    sub_40117F  
add     esp, 4  
xor    eax, eax
```

```
[NUL]  
loc_40103A:  
mov     esp, ebp  
pop    ebp  
ret  
sub_401000 endp
```

- Creazione dello stack
- Ciclo if
- Rimozione dello stack

## Descrizione delle azioni eseguite

- **call e Jump:** utilizzati per richiamare una funzione o spostare il controllo ad una subroutine specifica.
- **push e pop:** Sono istruzioni usate per inserire o rimuovere valori dallo stack.
- **mov:** Istruzione usata per copiare i dati da una zona all'altra.
- **cmp e jz:** Queste due istruzioni sono utilizzate per confrontare due valori e raggiungere un etichetta specifica nel caso fossero uguali.
- **Offset:** Viene utilizzata per ottenere l'etichetta all'interno di un programma.
- **add:** In linguaggio assembly, utilizzato per eseguire un'operazione di addizione. Somma il contenuto di due registri o di una costante a un registro.
- **xor:** (eXclusive OR) in linguaggio assembly viene utilizzata per eseguire un'operazione di xor bit a bit tra due operandi. La sintassi generale è simile a quella di 'add'.