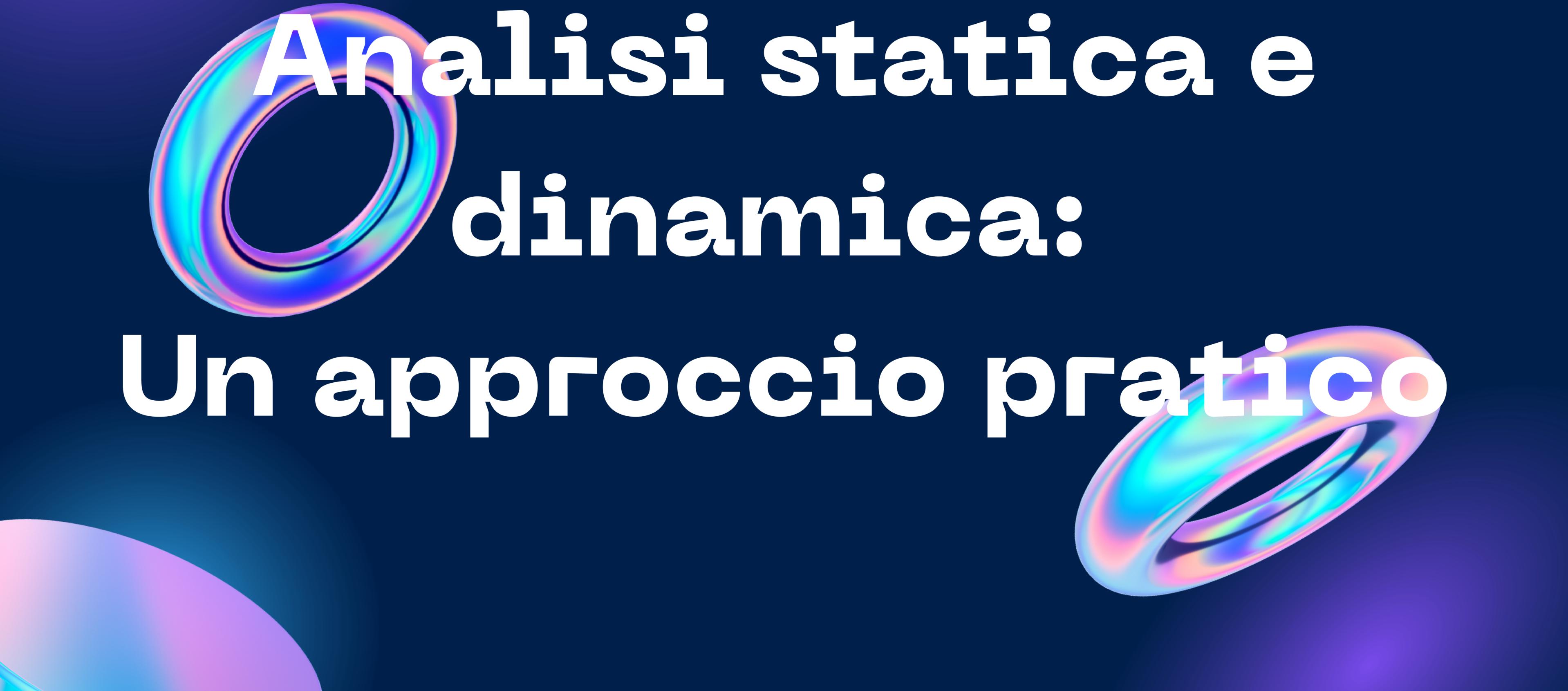




Progetto S10/L5  
Alessio Forli

# Analisi statica e dinamica: Un approccio pratico



# 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:



# Quesiti

1. Quali librerie vengono importate dal file esegibile?
  2. Quali sono le sezioni di cui si compone il file esegibile del malware?
- Con riferimento alla figura fornita, risponde ai seguenti quesiti:
3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti).
  4. Ipotizzare il comportamento della funzionalità implementata.
  5. Fare una tabella per spiegare il significato delle singole righe di codice.



Andremo ad analizzare il presunto malware  
“Malware\_U3\_W2\_L5” presente all’interno  
della cartella «Esercizio\_Pratico\_U3\_W2\_L5 »  
sul nostro desktop.

La macchina Windows 7 dove analizzeremo  
il file è una macchina virtuale configurata  
per fare una analisi in tutta sicurezza. Quindi  
procediamo con rispondere alle domande.



# Quesito 1

Per determinare quali librerie vengono importate dal file eseguibile utilizziamo CFF Explorer.

The image shows the CFF Explorer interface. On the left, there's a toolbar with icons for opening files, settings, and help. A red box highlights the 'Open File' icon (a folder). In the center, a 'Select File...' dialog box is open, showing a file named 'Malware\_U3\_W2\_L5'. A red box highlights the file name, and another red arrow points from the 'Import Directory' icon in the main window to this file name. A red arrow also points from the 'Open' button in the dialog box to the 'Import Directory' icon. On the right, the main CFF Explorer window displays the file structure of 'Malware\_U3\_W2\_L5.exe'. A red box highlights the 'Import Directory' node under the file header. Below the tree view, there's a list of tools: Address Converter, Dependency Walker, Hex Editor, Identifier, Import Adder, Quick Disassembler, and Rebuilder. A red arrow points from the 'Import Directory' icon in the main window to the 'Import Directory' node in the tree view. At the bottom, a table titled 'Malware\_U3\_W2\_L5.exe' lists imported modules. The 'Imports' column shows the number of functions imported. A red box highlights the 'Imports' column header and the row for 'KERNEL32.dll'.

Module Name	Imports	OFTs	TimeDateStamp	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

## Quesito 1

Come possiamo vedere in figura le librerie importate da “Malware\_U3\_W2\_L5” sono:

- KERNEL32.dll
- WININET.dll

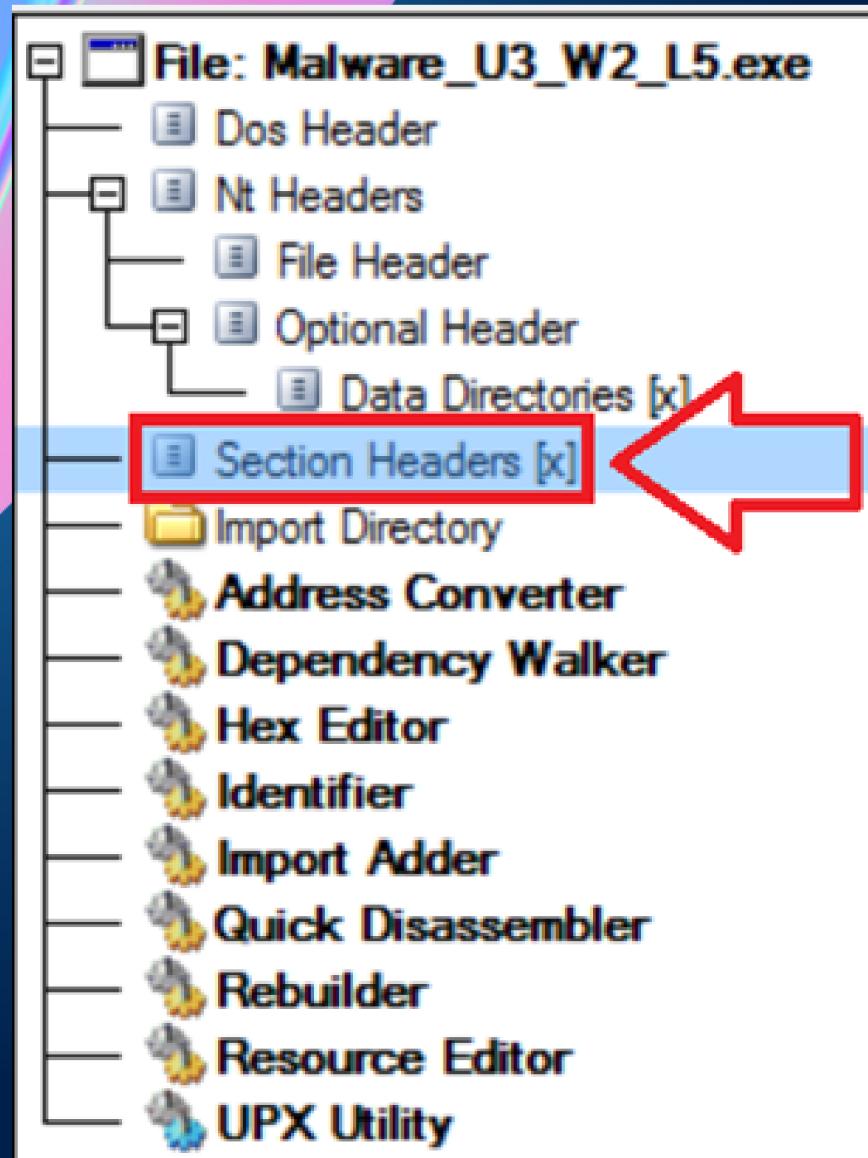
Sono 2 delle librerie più utilizzate.

KERNEL32.dll è una libreria che gestisce operazioni di base come aprire e salvare file, gestire la memoria e avviare programmi. È essenziale per il funzionamento del sistema operativo.

Mentre WININET.dll gestisce le connessioni di rete, i cookie e la cache, permette di scaricare e caricare file da e verso server web e garantisce comunicazioni crittografate sicure tramite funzioni per l'uso di SSL (Secure Sockets Layer) e TLS (Transport Layer Security).

## Quesito 2

Per determinare quali sono le sezioni di cui si compone il presunto malware utilizziamo ancora CFF Explorer.



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	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

## Quesito 2

Come possiamo vedere in figura le sezioni di cui si compone il presunto malware sono:

- .text
- .rdata
- .data

La sezione .text contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.

La sezione .rdata include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile.

La sezione .data contiene tipicamente le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.

# Codice Assembly fornito per l'esercizio:

```
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
```

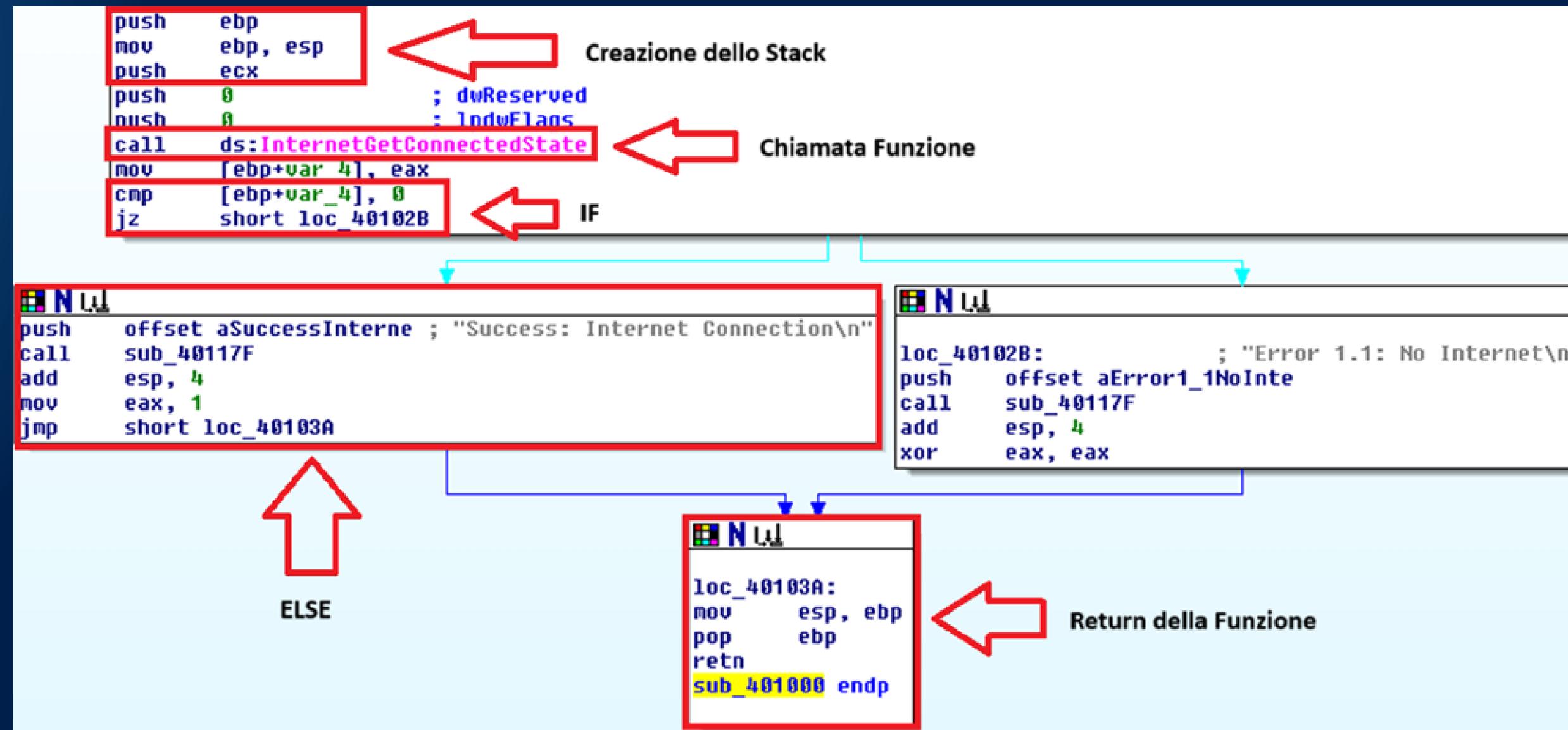
```
NUL
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add    esp, 4
mov    eax, 1
jmp    short loc_40103A
```

```
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
```

# Quesito 3

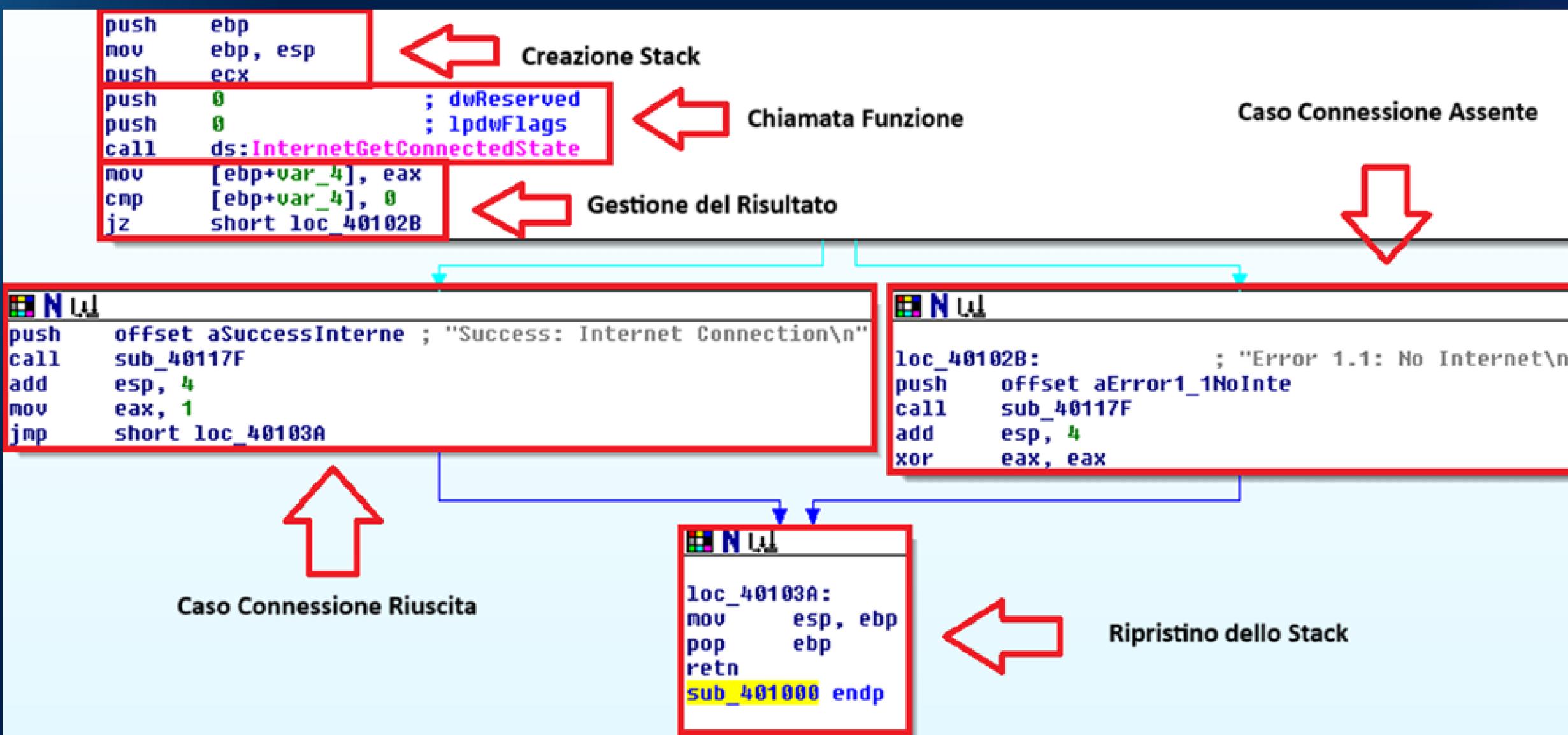
Identificazione dei costrutti noti.



Oltre alla creazione dello stack, possiamo notare un costrutto "tipo if-else".

## Quesito 4

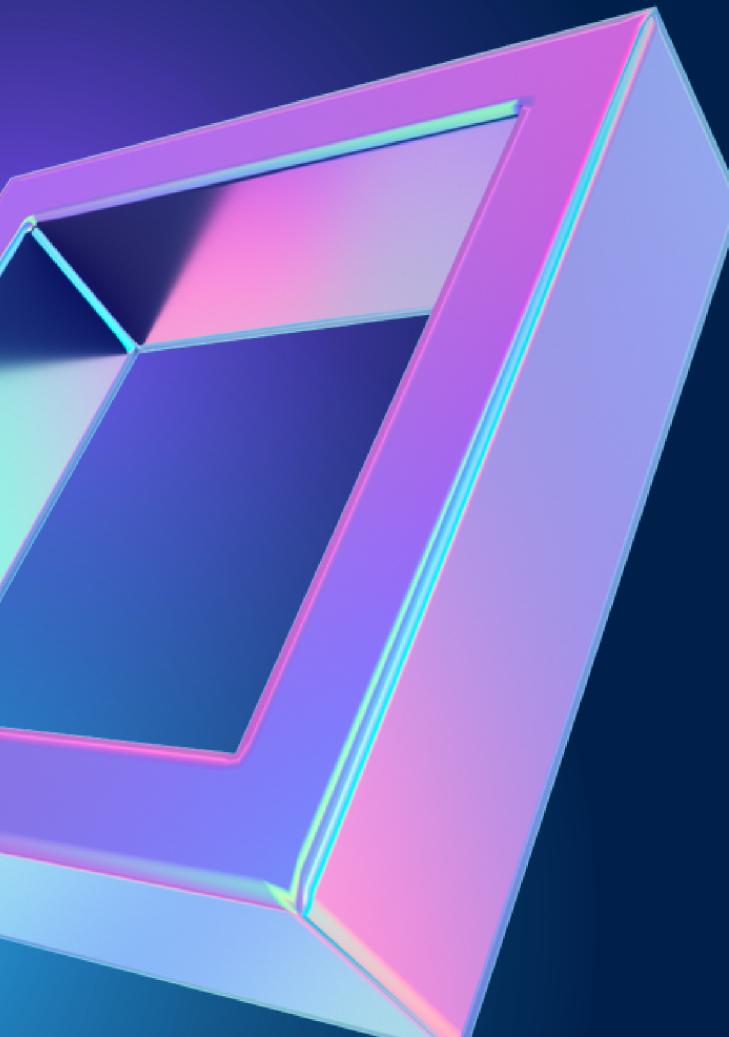
Dal codice fornito possiamo evincere l'implementazione della funzionalità di controllare lo stato della connessione e restituire un messaggio a schermo in base allo stato (connesso o non connesso).



## Quesito 5

BONUS: Tabella per spiegare le singole righe di codice:

RIGHE DI CODICE ASSEMBLY	SIGNIFICATO
push ebp	Aggiunge "ebp" allo stack
mov ebp, esp	Copia il contenuto del registro "esp" nel registro "ebp"
push ecx	Aggiunge "ecx" allo stack
push 0 ; dwReserved	Aggiunge "0" allo stack, usato come parametro "dwReserved"
push 0 ; lpdwFlags	Aggiunge "0" allo stack, usato come parametro "lpdwFlags"
call ds:InternetGetConnectedState	Chiama la funzione "InternetGetConnectedState"
mov [ebp+var_4], eax	Copia il contenuto del registro "eax" in [ebp+var_4]
cmp [ebp+var_4], 0	Compara il contenuto di [ebp+var_4] con "0"
jz short loc_40102B	Se ZF=1 salta alla posizione "loc_40102B"
push offset aSuccessInterne ; "Success: Internet Connection\n"	Aggiunge l'indirizzo di messaggio "Success: Internet Connection\n" sullo stack
call sub_40117F	Chiama la funzione "sub_40117F"
add esp, 4	Somma "4" a "esp" e salva il risultato in "esp"
mov eax, 1	Copia il valore "1" in "eax"
jmp short loc:40103A	Salta alla posizione "loc:40103A"
loc_40102B: ; "Error 1.1: No Internet\n"	Indica la posizione
push offset aError1_1Nolnte	Aggiunge l'indirizzo di messaggio "Error 1.1:No Internet\n" sullo stack
call sub_40117F	Chiama la funzione "sub_40117F"
add esp, 4	Somma "4" a "esp" e salva il risultato in "esp"
xor eax, eax	Utilizza un "or" esclusivo per settare "eax" a "0"
loc_40103A:	Indica la posizione
mov esp, ebp	Copia il contenuto del registro "ebp" nel registro "esp"
pop ebp	Elimina "ebp" dallo stack
retn	Effettua un return per pulire lo stack
sub_401000 endp	Fine della funzione



# Thank You