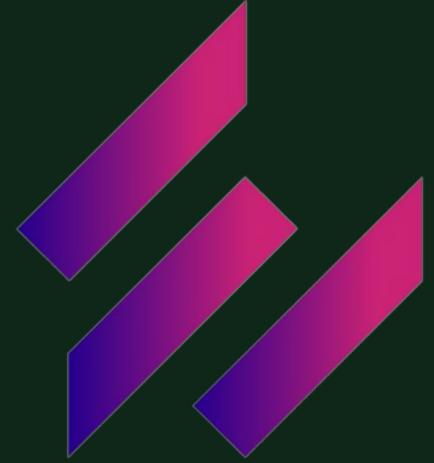


---

Malware analysis

S10

Team



# GIORNO 5

L5

Progetto S10\_L5

# Indice generale

---

- 
- 01 ANALISI LIBRERIE DEL MALWARE \_U3\_W2\_L5
  - 02 ANALISI SEZIONI DEL MALWARE \_U3\_W2\_L5
  - 03 IDENTIFICAZIONE COSTRUTTI
  - 04 IPOTESI COMPORTAMENTO FUNZIONALITÀ
  - 05 BONUS TABELLA RIGHE DI CODICE ASSEMBLY

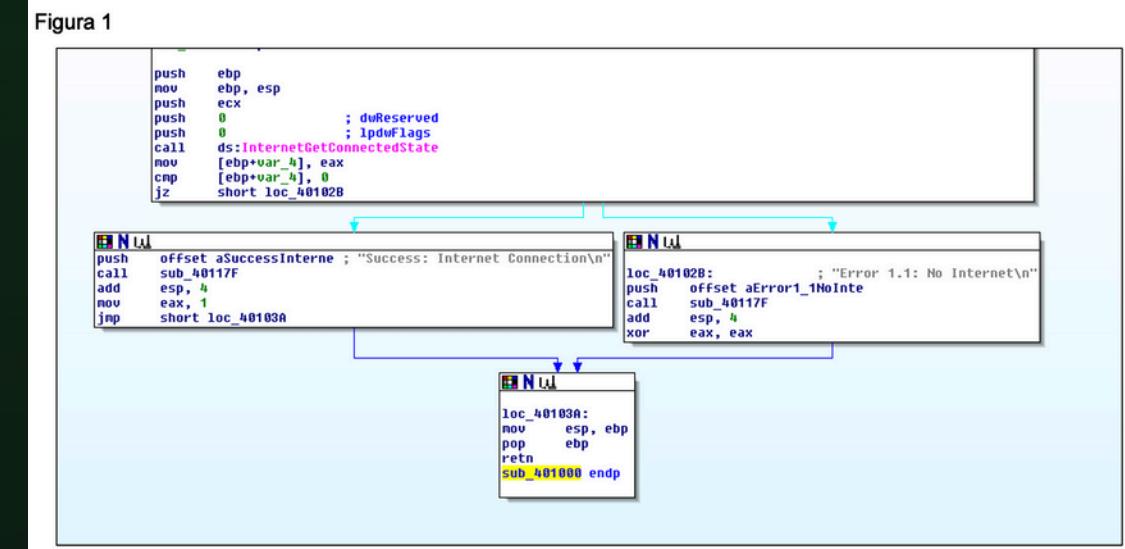
# Traccia giorno 5

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:

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

Con riferimento alla figura 1, rispondere ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli)
4. Ipotizzare il comportamento della funzionalità
5. BONUS fare tabella con significato delle singole righe di codice Esercizio Traccia e requisiti altri costrutti ) implementata assembly.



# 1 - Malware\_U3\_W2\_L5

## Librerie

Per analizzare le librerie di malware Malware\_U3\_W2\_L5, utilizziamo il tool CFF Explorer. Abbiamo individuato le seguenti librerie chiave:

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

### KERNEL32.dll

Questa libreria contiene numerose funzioni essenziali per il funzionamento del sistema operativo e delle applicazioni Windows. Le principali aree di gestione includono:

- **Memoria:** Funzioni come VirtualAlloc, VirtualFree, HeapCreate, e HeapAlloc sono utilizzate per l'allocazione e la liberazione della memoria.
- **Processi e Thread:** Funzioni come CreateProcess, CreateThread, TerminateProcess, e TerminateThread permettono la creazione e gestione di processi e thread.
- **Input/Output:** Funzioni come ReadFile, WriteFile, CreateFile, e CloseHandle gestiscono operazioni di lettura, scrittura e gestione di file e dispositivi.
- **File:** Funzioni come CopyFile, DeleteFile, e MoveFile gestiscono la manipolazione dei file nel file system.
- **Tempi e Date:** Funzioni come GetSystemTime, SetSystemTime, e GetTickCount forniscono e manipolano informazioni temporali.
- **Risorse:** Funzioni come LoadLibrary, GetProcAddress, e FreeLibrary permettono di caricare, ottenere indirizzi di funzioni e liberare librerie dinamiche.

Nel contesto del malware, queste funzioni possono essere utilizzate per vari scopi malevoli, come:

- **Persistenza:** Il malware può utilizzare funzioni di gestione dei processi e dei thread per rimanere attivo nel sistema anche dopo il riavvio.
- **Evasione:** Funzioni di gestione della memoria possono essere sfruttate per nascondere il malware o offuscare il suo codice.
- **Comunicazione:** Operazioni di input/output e gestione dei file possono essere utilizzate per esfiltrare dati o comunicare con server di comando e controllo.

# 1 - Malware\_U3\_W2\_L5

## Librerie

Per analizzare le librerie di malware Malware\_U3\_W2\_L5, utilizziamo il tool Explorer. Abbiamo individuato le seguenti funzioni chiave:

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

### WININET.dll

Questa libreria contiene funzioni per l'implementazione di vari protocolli di rete, come HTTP, FTP e NTP. Fornisce un'API che consente alle applicazioni Windows di:

- Effettuare richieste di rete: Funzioni come InternetOpen, InternetConnect, HttpOpenRequest, e HttpSendRequest permettono di inviare richieste HTTP ai server web.
- Gestire operazioni di rete: Funzioni come FtpGetFile, FtpPutFile, e InternetReadFile facilitano il download e l'upload di file tramite FTP e la lettura di dati da internet.

In un contesto di malware, le funzioni di WININET.dll possono essere sfruttate per:

- Download e Upload di Payload: Il malware può utilizzare funzioni di rete per scaricare componenti aggiuntivi o caricare dati rubati.
- Comunicazione con Server C2: Le richieste HTTP possono essere usate per comunicare con server di comando e controllo per ricevere istruzioni o inviare dati.

# 1 - Malware\_U3\_W2\_L5

## KERNEL32

Malware_U3_W2_L5.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	000064EC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

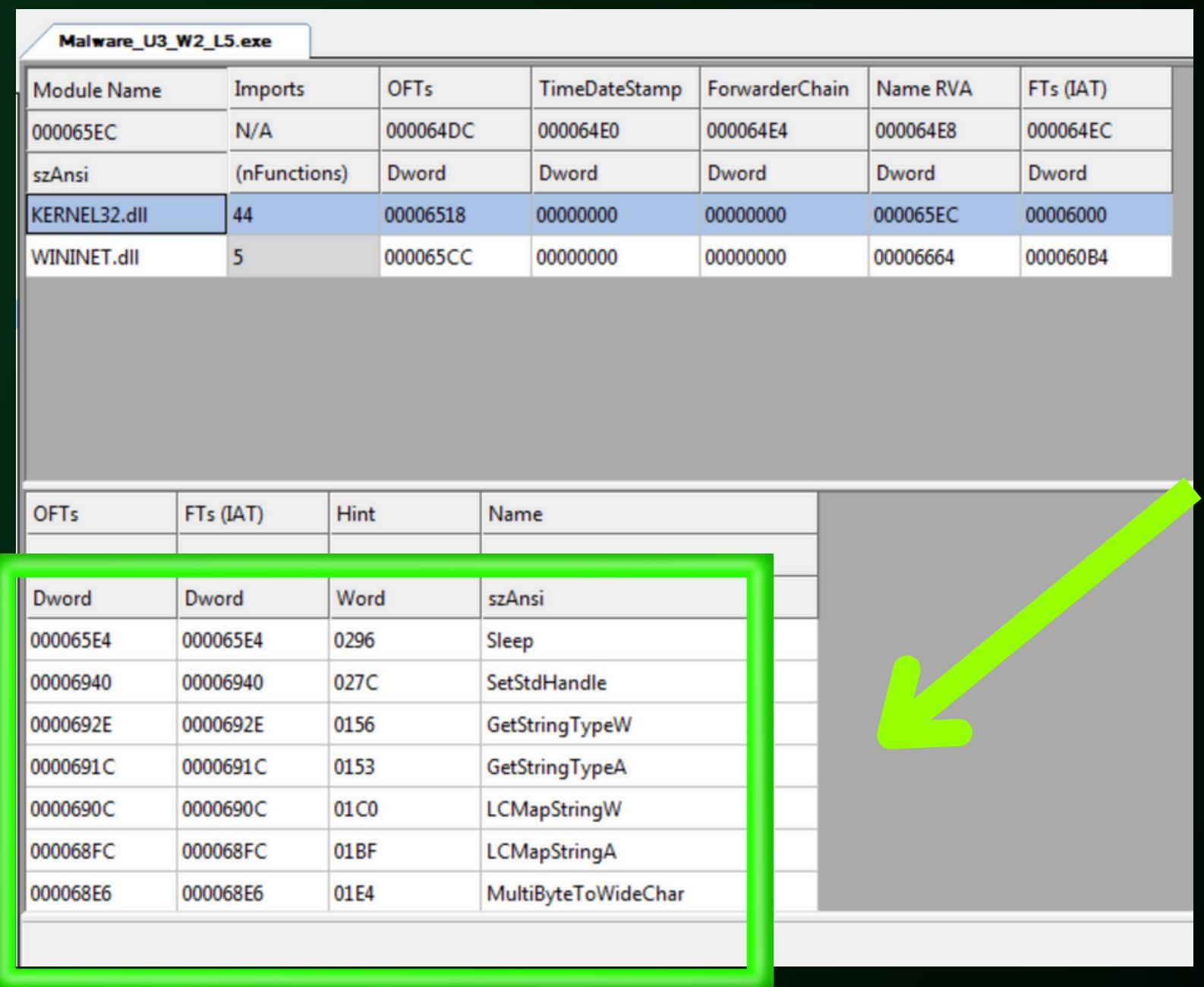
  

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMMapStringW
000068FC	000068FC	01BF	LCMMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar

La libreria **KERNEL32**, parte integrante del sistema operativo Windows, fornisce una vasta gamma di funzionalità essenziali per la gestione delle risorse di sistema e l'esecuzione dei programmi. Tra le 44 funzioni che importa, alcune delle più significative includono:

- **Sleep**: Questa funzione di sistema sospende temporaneamente l'esecuzione di un thread per un periodo specificato. Durante questo tempo, altri thread possono essere eseguiti, permettendo una gestione più efficiente delle risorse di calcolo. Ad esempio, un thread che non ha bisogno di essere eseguito continuamente può essere messo in pausa, liberando tempo di CPU per altri processi.
- **CloseHandle**: È un'API utilizzata per chiudere un handle, ossia un riferimento o puntatore a un oggetto kernel. Questa funzione è cruciale per la gestione delle risorse, poiché rilascia le risorse allocate a un handle una volta che non sono più necessarie, prevenendo perdite di memoria e migliorando la stabilità del sistema.

- **GetProcAddress**: Questa funzione permette ai programmatori di ottenere un puntatore a una funzione specifica all'interno di una DLL (Dynamic Link Library) caricata in memoria. Viene utilizzata per chiamare funzioni esportate da una DLL in modo dinamico durante l'esecuzione del programma, anziché durante la fase di compilazione. Questo è particolarmente utile per plugin, estensioni o per caricare moduli opzionali senza ricompilare l'intera applicazione.
- **VirtualAlloc**: Utilizzata principalmente per riservare o allocare memoria virtuale per un processo. Questa funzione consente di riservare una porzione di memoria virtuale senza dover allocare immediatamente la memoria fisica corrispondente. È fondamentale per la gestione efficiente della memoria, poiché permette di controllare l'allocazione delle risorse in modo più granulare e flessibile.
- **VirtualFree**: È complementare a VirtualAlloc e viene utilizzata per liberare la memoria precedentemente allocata. Questa funzione è essenziale per la gestione della memoria dinamica, poiché garantisce che la memoria non più necessaria venga restituita al sistema, riducendo così il rischio di frammentazione e migliorando l'efficienza complessiva del sistema.



The screenshot shows the 'Imports' section of a debugger. The table has columns for Module Name, Imports, OFTs, TimeStamp, ForwarderChain, Name RVA, and FTs (IAT). It lists imports from KERNEL32.dll (44 functions) and WININET.dll (5 functions). The 'szAnsi' function from KERNEL32.dll is highlighted with a green box and a green arrow pointing to it from the text above.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	000064EC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW
000068FC	000068FC	01BF	LCMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar

Queste funzioni, tra le altre importate da **KERNEL32**, sono fondamentali per il funzionamento efficace e efficiente delle applicazioni Windows, fornendo gli strumenti necessari per la gestione delle risorse, la comunicazione inter-processo e l'allocazione dinamica della memoria.

# Malware\_U3\_W2\_L5

## Sezione del malware

The screenshot shows a software interface with two main panes. The left pane is a tree view of the file structure for 'File: Malware\_U3\_W2\_L5.exe', listing sections like Dos Header, Nt Headers, File Header, Optional Header, Data Directories, Section Headers, and Import Directory. The right pane is a table titled 'Malware\_U3\_W2\_L5.exe' with columns for Name, Virtual Size, Virtual Address, Raw Size, Raw Address, Reloc Address, Linenumbers, Relocations N..., Linenumbers ..., and Characteristics. The table contains four rows corresponding to the sections listed in the tree view.

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

### .text

La sezione .text contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta avviato il software. Generalmente, questa è l'unica sezione di un file eseguibile che viene eseguita direttamente dalla CPU, mentre tutte le altre sezioni contengono dati o informazioni di supporto.

### .rdata

La sezione .rdata include generalmente le informazioni sulle librerie e le funzioni importate ed esportate dall'eseguibile. Queste informazioni possono essere ricavate utilizzando strumenti come CFF Explorer.

### .data

La sezione .data contiene tipicamente i dati e le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile è considerata globale quando non è definita all'interno del contesto di una funzione, ma è dichiarata globalmente ed è quindi accessibile da qualsiasi funzione all'interno dell'eseguibile.

Identificazione del Malware

# Ricerca su VirusTotal

Effettuando una ricerca su VirusTotal tramite l'hash ricavato da CFF Explorer, è stato riscontrato che il malware in questione potrebbe essere un Trojan.

The screenshot shows the VirusTotal analysis interface for the file hash `b71777edbf21167c96d20ff803cbcb25d24b94b3652db2f286dc6efd3d8416a`. The main summary indicates that 38 out of 72 security vendors flagged the file as malicious. Below this, detailed file information is provided: Name (Lab06-02.exe), Size (40.00 KB), Last Modification Date (6 days ago), and Type (EXE). The file is associated with several threat labels: peexe, checks-network-adapters, runtime-modules, direct-cpu-clock-access, and armadillo. At the bottom of the analysis page, there is a summary bar with three sections: 'Popular threat labels' (which includes 'trojan.r002c0pdm21'), 'Threat categories' (which includes 'trojan'), and 'Family labels' (which includes 'r002c0pdm21'). A red oval has been drawn around the 'Popular threat labels' section to highlight the specific threat label found.

## 4 - Ipotesi comportamento della funzionalità

Il codice in Assembly x86 implementa una funzione che verifica la connessione a Internet utilizzando la funzione “InternetGetConnectedState”. Prima di chiamare la funzione, viene creato lo stack e vengono inseriti i tre parametri che saranno passati alla funzione: “ecx”, “dwReserved”, e “lpdwFlags”. La funzione verifica lo stato della connessione e restituisce il valore nel registro “eax”, che viene poi salvato nella variabile locale “var\_4”.

A questo punto viene effettuato un confronto tramite l'istruzione “cmp”, che esegue una sottrazione tra i due operandi. In questo caso confronta “var\_4” e “0”. Se sono uguali, l'istruzione “jz” salta alla locazione “loc\_40102B” e stampa a video un errore, indicando che non c'è connessione. Altrimenti, se “var\_4” e “0” non sono uguali, il salto non verrà eseguito e il codice continuerà, stampando a video "Success: Internet Connection".

# InternetGetConnectedState

Questa funzione verifica lo stato della connessione Internet su un sistema operativo Windows. Quando la funzione restituisce TRUE, la connessione Internet è attiva; se restituisce FALSE, non è presente alcuna connessione.

Dettagli della Funzione InternetGetConnectedState:

**dwReserved**

Nella funzione InternetGetConnectedState, il parametro dwReserved è riservato per usi futuri e attualmente non viene utilizzato.

**lpdwFlags**

Il prefisso "lp" indica un "long pointer", ovvero un puntatore a un valore DWORD. Nella funzione InternetGetConnectedState, il parametro lpdwFlags viene utilizzato per restituire lo stato della connessione Internet. La funzione scrive i dati relativi allo stato della connessione all'indirizzo di memoria puntato da lpdwFlags, consentendo al chiamante di ottenere tali informazioni dopo l'esecuzione della funzione.

# 5 - Costrutti noti e significato di tutte le righe del codice

push ebp	viene salvato il valore attuale di ebp, puntatore alla base dello stack, nel nuovo stack che stiamo creando. In questo modo il valore di ebp può essere ripristinato alla fine della funzione.
mov ebp, esp	Viene copiato il valore del registro esp, che è lo stack pointer, nel registro ebp. In questo modo il registro ebp punta alla funzione corrente e questo rende più facile l'accesso a variabili locali tramite offset rispetto a ebp.
push ecx	salva il valore attuale del registro ecx. Serve sempre per preservare il registro e ripristinarlo alla fine della chiamata.
push 0 ;dwReserved	Carica il valore 0 nello stack. Questo 0 rappresenta il parametro dwReserved che deve essere 0 perché è riservato e quindi non utilizzabile.
push 0 ;lpdwFlags	Carica un altro valore 0 nello stack. Questa volta lo 0 è associato al parametro lpdwFlags perché non si vogliono ulteriori informazioni riguardo il tipo di connessione.
call ds:InternetGetConnectedState	chiama la funzione tramite il registro segmentazione ds. La funzione "InternetGetConnectedState" è una funzione API di Windows che riceve i 3 parametri precedentemente caricati nello stack, e verifica se il sistema è connesso o meno ad internet.
mov [ebp+var_4], eax	Copia il valore del registro eax(utilizzato per contenere il valore di ritorno della funzione chiamata)nella variabile locale var_4 che si trova probabilmente ad un offset di -4 dal valore del registro ebp.
cmp [ebp+var_4], 0	confronta il valore appena salvato nella variabile var_4 (cioè eax) con lo 0
jz short loc_40102B	Jump if zero. Questa istruzione effettua un salto alla locazione loc_40102B se la sottrazione tra var_4 e 0 è uguale a 0, e quindi se i due valori sono uguali e di conseguenza il flag ZF = 1.
push offset aSuccessInterne ;"Success: Internet Connection\n"	carica nello stack l'indirizzo della stringa aSuccessInterne. Offset aSuccessInternet serve per ottenere l'indirizzo di memoria della stringa "Success: Internet Connection\n"
call sub_40117F	chiama la funzione sub_40117F che potrebbe essere progettata per stampare una stringa. In questo caso vien da sé che riceve come parametro l'indirizzo della stringa appena descritta sopra
add esp, 4	Aggiunge 4 al valore di esp. Questa istruzione serve per ripulire lo stack pointer rimuovendo, in questo caso, un argomento di 4 byte, cioè la variabile locale var_4
mov eax, 1	copia il valore 1 all'interno del registro eax
jmp short loc_40103A	salto incondizionato di breve distanza, "short"(-128 +127 byte), alla locazione loc_40103A

=costruzione dello stack

=costrutto if

loc_40102B:	
push offset aError1_1NoInte ; "Error 1.1: No Internet\n"	carica nello stack l'indirizzo della stringa aSuccessInternet. Offset aSuccessInternet serve per ottenere l'indirizzo di memoria della stringa "Success: Internet Connection\n"
call sub_40117F	chiama la funzione sub_40117F che potrebbe essere progettata per stampare una stringa. In questo caso vien da sé che riceve come parametro l'indirizzo della stringa appena descritta sopra
add esp, 4	Aggiunge 4 al valore di esp. Questa istruzione serve per ripulire lo stack pointer rimuovendo, in questo caso, un argomento di 4 byte, cioè la variabile locale var_4
xor eax, eax	inizializza a 0 il registro EAX. Infatti l'operatore logico tra due bit identici restituisce sempre 0.

loc_40103A:	
mov esp, ebp	copia il valore del registro ebp nel registro esp. Ripristina il valore dello stack pointer esp che aveva all'inizio della funzione eliminando tutte le variabili locali allocate sullo stack
pop ebp	rimuove dallo stack il registro ebp che era stato aggiunto all'inizio con push
retn	Restituisce il controllo alla funzione chiamante
sub_401000 ends	indica la fine della funzione. non è un'istruzione eseguibile.

=rimozione dello stack



# TEAM ALBA



**Zhongshi Liu**



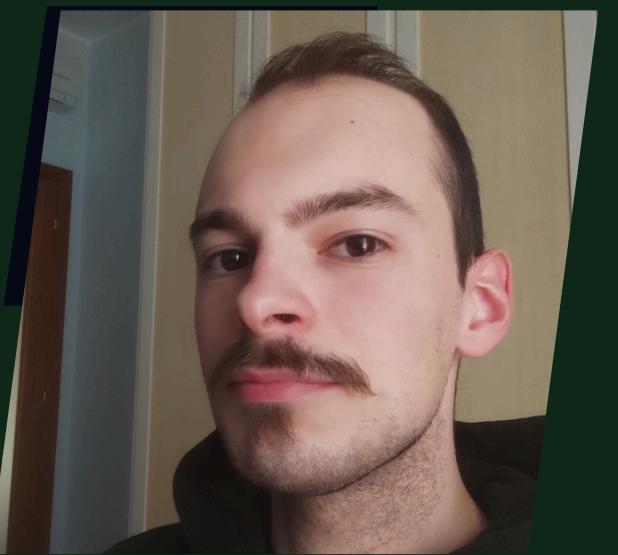
**Mara Dello Russo**



**Mario Marsicano**



**Luca Lenzi**



**Giovanni Sannino**



**Andre Vinicius**

---

# THANK YOU

---