

Malware_U3_W2_L5

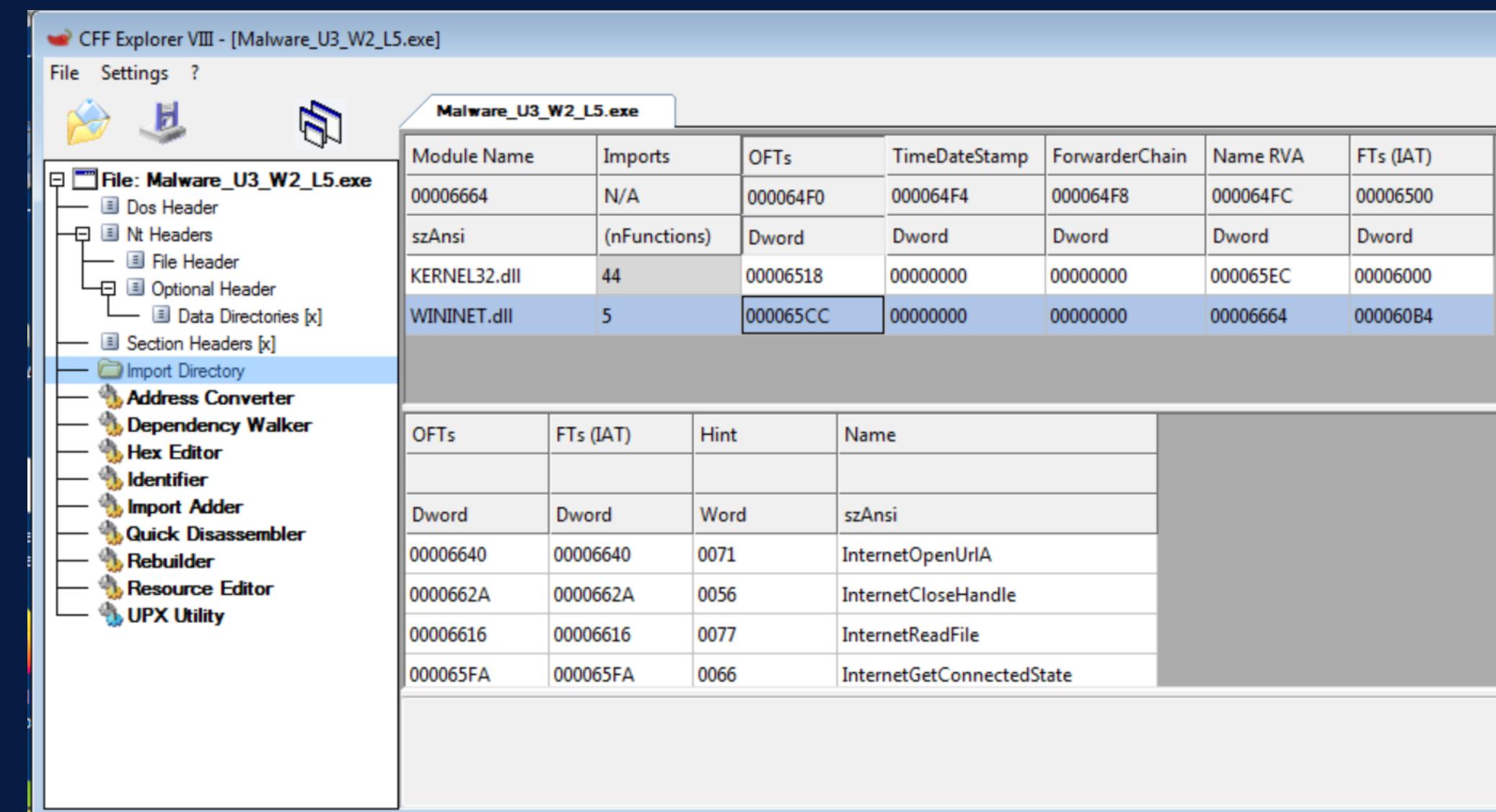
Traccia:

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:

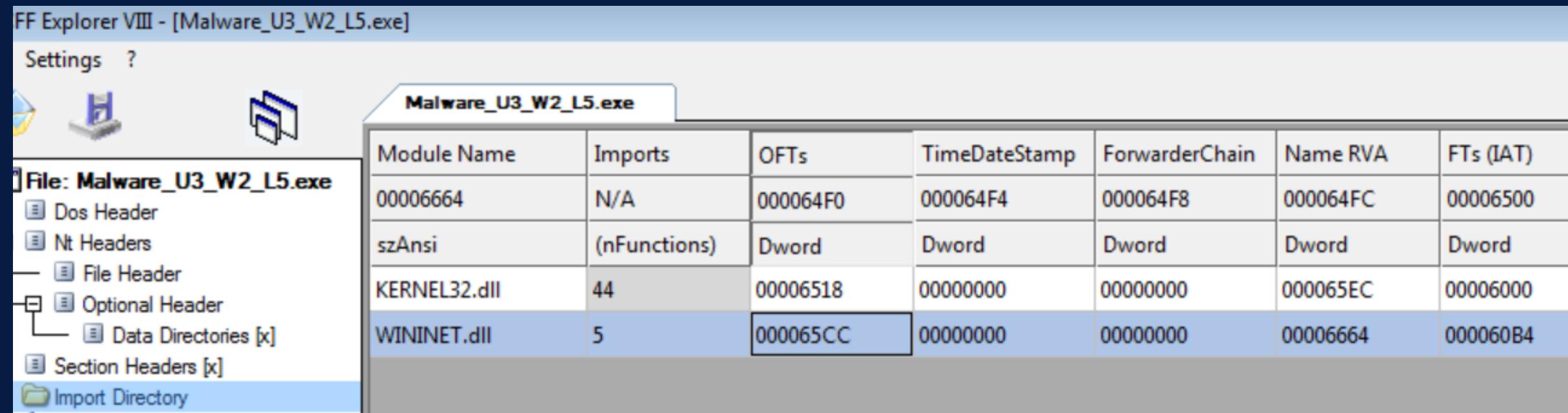
1. Quali librerie vengono importate dal file eseguibile?
2. Quali sono le sezioni di cui si compone il file eseguibile del malware? Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:
3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti).
4. Ipotizzare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

TOOL

CFF Explorer è un software che funge da visualizzatore e editor di file eseguibili in formato Portable Executable (PE). I file PE sono il formato standard per i file eseguibili e i file di librerie dinamiche su sistemi operativi Windows



LIBRERIE IMPORTATE DA FILE ESEGUIBILE

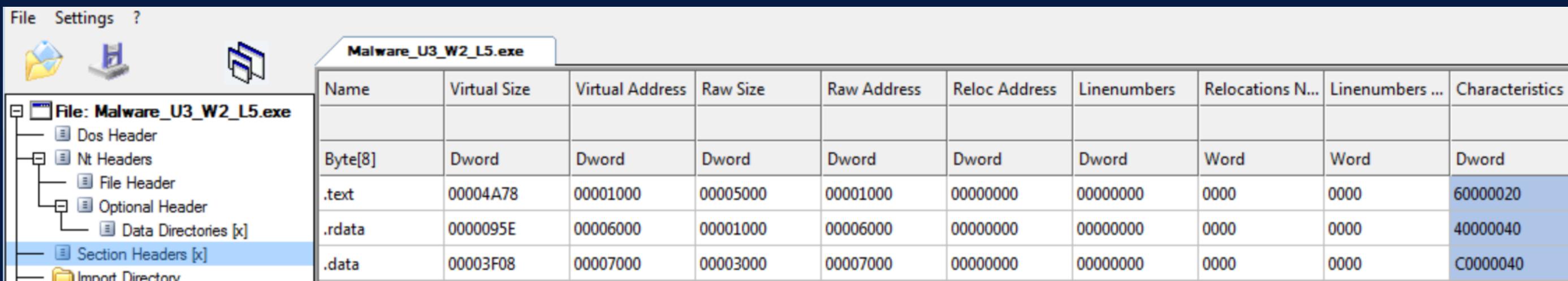


Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
00006664	N/A	000064F0	000064F4	000064F8	000064FC	00006500
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

- Kernel32.dll: contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria.
- Advapi32.dll: contiene le funzioni per interagire con i servizi ed i registri del sistema operativo.
- Wininet.dll: contiene le funzioni per l'implementazione di alcuni protocolli di rete come HTTP, FTP, NTP.

SEZIONI

- **.text**: contiene le istruzioni (le righe di codice) 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.
- **.rdata**: include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer.
- **.data**: contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile.



The screenshot shows the CFF Explorer interface. On the left, a tree view displays the file structure of 'Malware_U3_W2_L5.exe', including the Dos Header, Nt Headers, File Header, Optional Header, Data Directories, Section Headers, and Import Directory. The 'Section Headers' node is currently selected. On the right, a table provides detailed information for each section:

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

CODICE ASSEMBLY

```
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 aSuccessInternet ; "Success: Internet Connection\n"
call    sub_40117F
add    esp, 4
mov    eax, 1
jnp    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
retm
sub_401000 endp
```

COSTRUTTI: CREAZIONE DELLO STACK

```
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
```

Viene effettuato il salvataggio del puntatore alla base dello stack (ebp) e l'allocazione dello spazio per le variabili locali.

Viene pushato il registro ecx nello stack.

COSTRUTTI: CHIAMATA ALLE FUNZIONI

```
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
```

Chiamata a
InternetGetConnectedState:

Vengono pushati i parametri per
la chiamata alla funzione
InternetGetConnectedState.

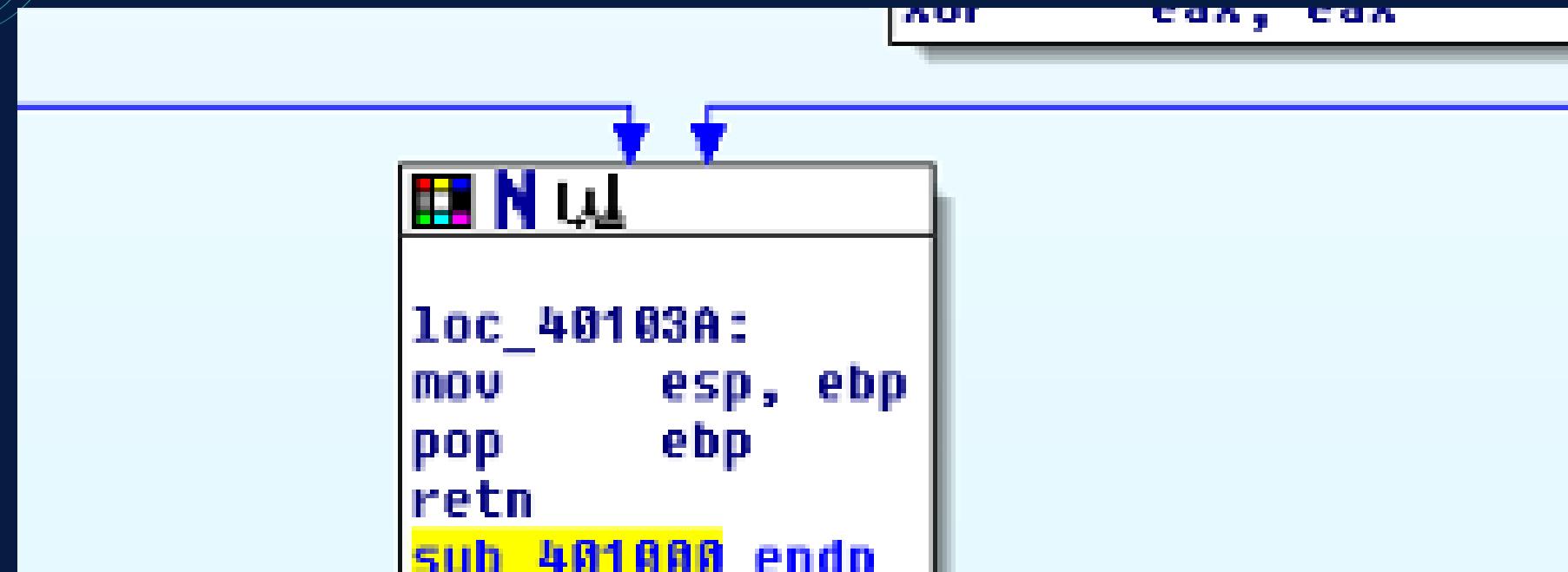
Viene eseguita la chiamata alla
funzione attraverso call
ds:InternetGetConnectedState.
Il risultato della chiamata viene
salvato nella variabile locale
[ebp+var_4].

CICLO: IF

```
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
```

Verifica dello stato della connessione:

Viene effettuato un confronto del valore salvato nella variabile locale con zero (cmp [ebp+var_4], 0). Se il risultato del confronto è uguale a zero (nessuna connessione), salta a loc_40102B. Altrimenti, continua a eseguire il codice successivo.



```
loc_40103A:
    mov    esp, ebp
    pop    ebp
    retn
    sub    401000 endo
```

Fine della funzione:
Viene effettuato il
ripristino del
puntatore alla base
dello stack (pop ebp)
e il ritorno dalla
funzione (retn).

COMPORTAMENTO FUNZIONE IMPLEMENTATA

La funzione ha lo scopo di verificare lo stato della connessione Internet e fornire una risposta in base a tale stato.

In sintesi, la funzione sembra essere progettata per informare l'utente sullo stato della connessione Internet, restituendo un messaggio di successo o errore in base a tale stato.

B O N U S

push ebp: Salva il valore corrente del registro base dello stack (ebp) nello stack. Questa istruzione è comune all'inizio di molte funzioni per creare uno stack frame.

mov ebp, esp: Imposta il registro base dello stack (ebp) con il valore corrente dello stack pointer (esp).

push ecx: Salva il valore corrente del registro ecx nello stack. ecx verrà utilizzato successivamente.

push 0 e push 0: Vengono pushati due zeri nello stack come parametri per la funzione successiva.

call ds:InternetGetConnectedState: Chiama la funzione InternetGetConnectedState. Questa funzione verifica lo stato della connessione Internet e restituisce il risultato.

mov [ebp+var_4], eax: Salva il risultato della chiamata alla funzione in una variabile locale [ebp+var_4].

cmp [ebp+var_4], 0: Compara il valore salvato nella variabile locale [ebp+var_4] con zero.

jz short loc_40102B: Salta a loc_40102B (linea 13) se il risultato della comparazione è zero (cioè, nessuna connessione).

push offset asuccessInterne: Pusha l'indirizzo di una stringa "Succes Internet Connection\n" nello stack come argomento per la funzione successiva.

call sub_40105F: Chiama la funzione sub_40105F, che probabilmente si occupa di stampare il messaggio.

add esp, 4: Ripristina lo stack dopo la chiamata della funzione, eliminando il parametro precedentemente pushato.

mov eax, 1: Imposta il registro eax a 1, indicando un successo.

jmp short loc_40103A: Salta a loc_40103A (fine della funzione).

push offset aError1_1Nolnte: Pusha l'indirizzo di una stringa "Error 1.1: No Internet\n" nello stack come argomento per la funzione successiva.

call sub_40117F: Chiama la funzione sub_40117F, che probabilmente si occupa di stampare il messaggio di errore.

add esp, 4: Ripristina lo stack dopo la chiamata della funzione, eliminando il parametro precedentemente pushato.

xor eax, eax: Imposta il registro eax a zero, indicando un fallimento.

mov esp, ebp: Ripristina il valore originale dello stack pointer (esp).

pop ebp: Ripristina il valore originale del registro base dello stack (ebp).

retn: Restituisce il controllo al chiamante. La funzione termina.