

Progetto

Settimana 10

Lezione 5

Francesco Alfonsi

Indice

Traccia.....	3
CFF Explorer	
Introduzione.....	4
Librerie Importate dal file eseguibile oggetto dell'analisi.....	5 - 6 - 7
Sezioni di cui si compone l'eseguibile oggetto dell'analisi.....	8 - 9
Costrutti noti del codice.....	10
Analisi del codice assembly.....	11 - 12

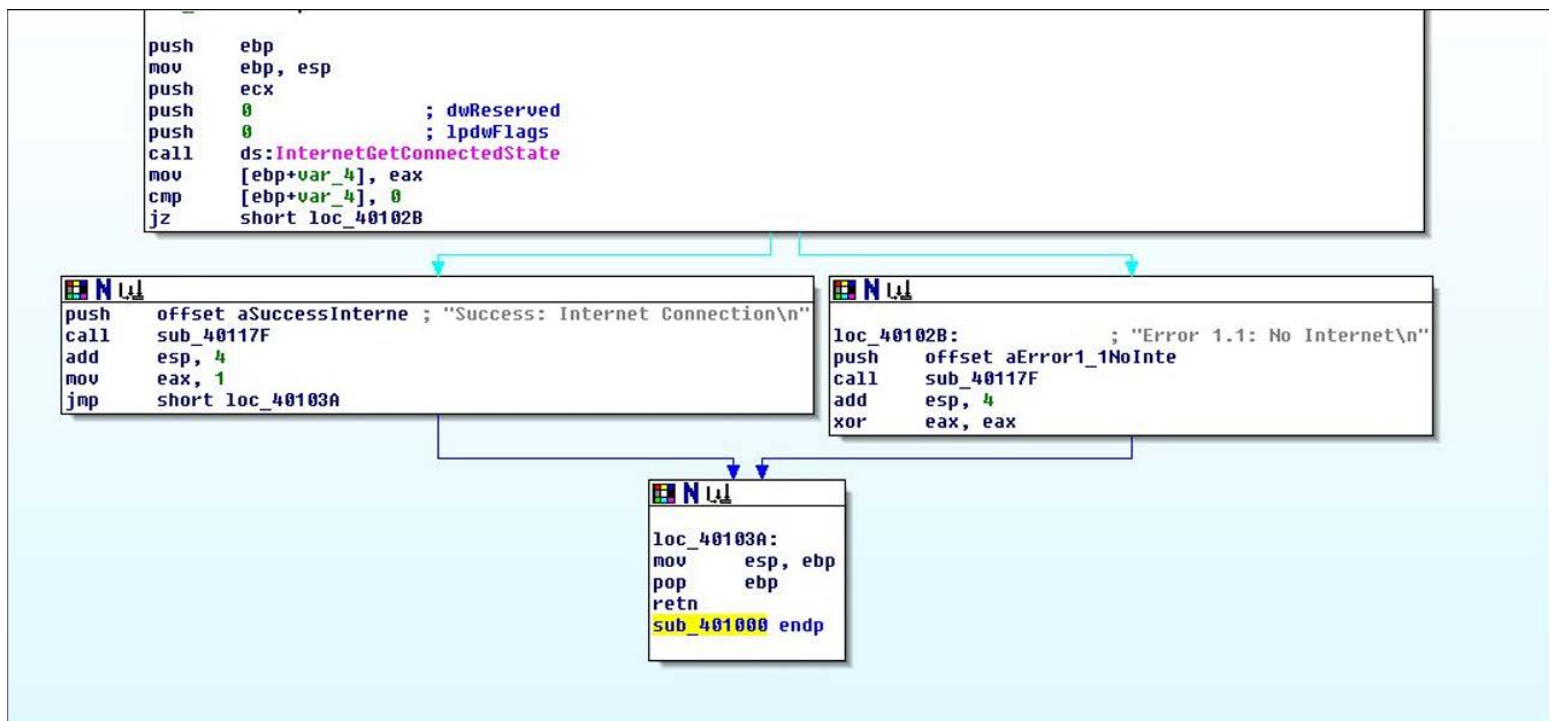
Traccia:

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 al codice in figura, rispondere ai seguenti quesiti:

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




Introduzione

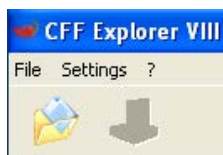
Per questo e per il successivo task, ci siamo avvalsi di CFF Explorer.

CFF Explorer è un pacchetto gratuito di strumenti specializzati nell'analisi di file Portable Executable (PE), principalmente sui sistemi Windows. Trova la sua principale applicazione nell'analisi avanzata del malware. Permette all'utente di avere un accesso approfondito agli interni PE, alla modifica del codice esadecimale e al linguaggio di scripting. Grazie alle importanti e complesse funzionalità offerte, il software consente:

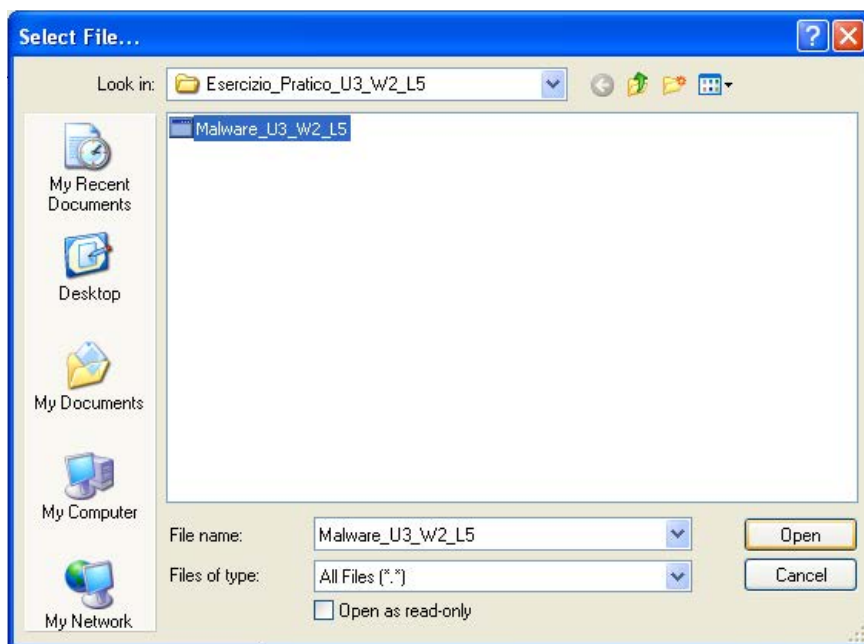
- Identificazione del codice offuscato o funzionalità nascoste.
- Tracciare e comprendere il flusso di esecuzione del malware.
- Dissezione di packer personalizzati e tecniche di anti-analisi.
- Disassemblaggio e riassemblaggio di sezioni di codice.
- Inserimento o modifica manuale del codice per ulteriori indagini.
- Visualizzazione di informazioni dettagliate sui file PE, incluse risorse, importazioni, esportazioni e sezioni.
- Identificazione di potenziali vulnerabilità o codice dannoso all'interno dei file PE.
- Indagare su crash o comportamenti imprevisti per identificarne la causa principale.

Primi passi: apertura di CFF Explorer e scelta del file da analizzare.

Avviamo CFF Explorer dal Desktop cliccando semplicemente due volte sulla sua icona .



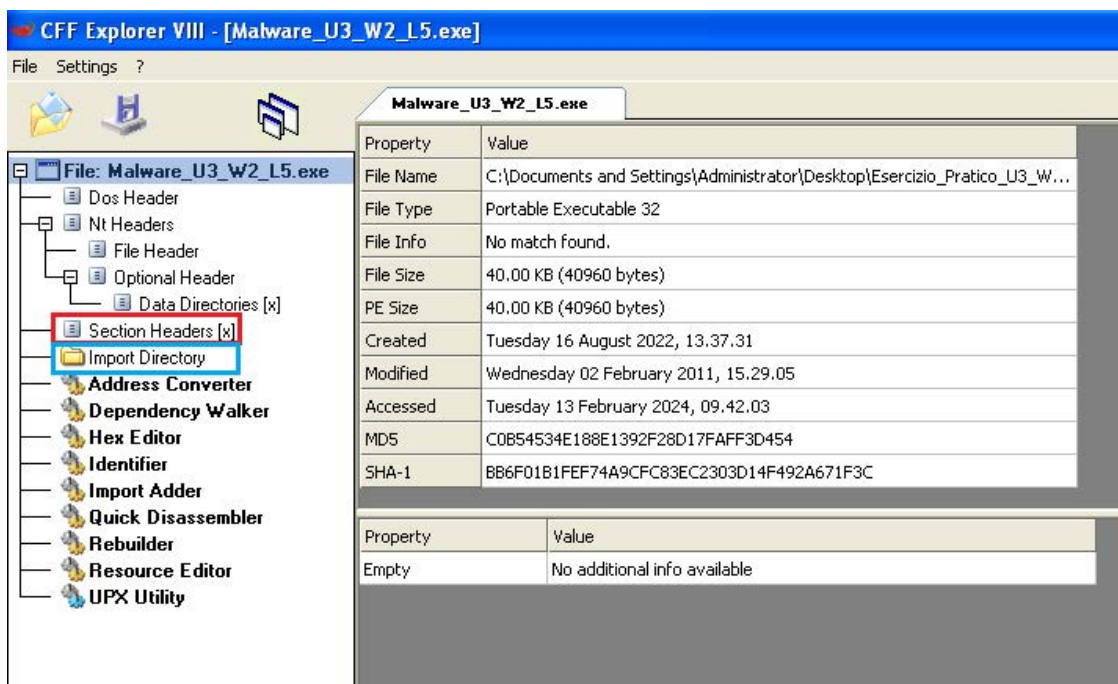
All'avvio dell'applicazione, cliccando sull'icona a forma di cartella posta in alto sulla sinistra (o, analogamente, scegliendo **file** -> **open**), si aprirà la finestra di dialogo dedicata alla ricerca del file che si andrà ad esaminare.



Navighiamo quindi all'interno delle cartelle del sistema fino ad arrivare a quella contenente il programma di cui si vuole approfondire il funzionamento - nel nostro caso, il malware richiesto nell'esercizio. Una volta trovato, clicchiamo su **open** per completare l'importazione dell'eseguibile nel software di analisi ed avviarne l'indagine.

Librerie Importate dal file eseguibile oggetto dell'analisi

Una volta acquisito il file da controllare, CFF Explorer mostrerà, sulla parte sinistra dello schermo, un menù di operazioni eseguibili. Per la nostra analisi, ci avvarremo delle funzionalità **Import Directory** (per visualizzare le librerie importate) e **Section Headers** (per scoprire le sezioni di cui il malware si compone).



Import Directory

Una libreria, o modulo, è un insieme di funzioni predefinite. Quando un applicativo necessita di una funzione, la *chiede in prestito* alla libreria nella quale è definito il suo funzionamento. Questo processo è detto di *importazione della libreria*. Per il controllo delle librerie, scegliamo Import Directory dal pannello principale a sinistra. Si aprirà, nella parte centrale dello schermo, il dettaglio delle librerie importate. Nel nostro caso, possiamo notare che il malware importa le librerie **Kernel32.dll** e **WININET.dll**. Vediamo, nelle pagine successive, quali sono le loro principali funzioni.

Librerie Importate dal file eseguibile oggetto dell'analisi

Kernel32.dll è una libreria fondamentale nei sistemi operativi Windows. Fornisce funzioni essenziali alle applicazioni per interagire con il sistema operativo. Le applicazioni possono utilizzare le funzioni al suo interno per accedere alle risorse di sistema, gestire la memoria ed eseguire varie attività. Di particolare interesse sono le funzionalità per:

Memoria: Funzioni come malloc e free consentono alle applicazioni di allocare e rilasciare memoria.

File e I/O: Funzioni come CreateFile e ReadFile consentono alle applicazioni di lavorare con file e cartelle.

Gestione di processi e thread: Funzioni come CreateProcess e ExitProcess aiutano le applicazioni a gestire processi e thread.

Informazioni di sistema: Funzioni come GetSystemTime e GetLastError forniscono accesso alle informazioni di sistema.

Sicurezza: Funzioni come CreateMutex e OpenProcess offrono meccanismi di sicurezza di base per le applicazioni.

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

Malware_U3_W2_L5.exe

File: Malware_U3_W2_L5.exe

- Dos Header
- Nt Headers
 - File Header
 - Optional Header
 - Data Directories [x]
- Section Headers [x]
- Import Directory
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler
- Rebuilder
- Resource Editor
- UPX Utility

Module Name	Imports	OFTs	TimeDateStamp	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
00006670	00006670	00CA	GetCommandLineA
00006682	00006682	0174	GetVersion
00006690	00006690	007D	ExitProcess
0000669E	0000669E	029E	TerminateProcess
000066B2	000066B2	00F7	GetCurrentProcess

Librerie Importate dal file eseguibile oggetto dell'analisi

WININET.dll è un'altra libreria fondamentale nei sistemi Windows, focalizzata in particolare sull'accesso e sulla funzionalità di Internet. È un componente chiave dell'API Win32 Internet Extensions (WinInet), che fornisce funzioni per le applicazioni per:

Effettua richieste HTTP e FTP: consente alle applicazioni di recuperare dati da server Web e trasferire file utilizzando protocolli Internet comuni.

Gestisci cookie e cache: WININET.dll aiuta le applicazioni a gestire i cookie (piccoli file di dati archiviati sul dispositivo dell'utente) e i contenuti Web memorizzati nella cache per una navigazione efficiente.

Esegui l'autenticazione: le funzioni all'interno della libreria consentono alle applicazioni di autenticarsi con i server Web utilizzando vari metodi come l'autenticazione di base o i certificati client.

Risolvi nomi host e indirizzi IP: questa funzionalità traduce gli indirizzi dei siti Web (URL) in indirizzi IP numerici comprensibili ai computer.

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

Malware_U3_W2_L5.exe

File: Malware_U3_W2_L5.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [x]
- Section Headers [x]
- Import Directory
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler
- Rebuilder
- Resource Editor
- UPX Utility

Module Name	Imports	OFTs	TimeDateStamp	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

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

Sezioni di cui si compone l'eseguibile oggetto dell'analisi

Section Headers

In un file PE (Portable Executable), le sezioni sono elementi fondamentali che memorizzano specifici tipi di dati e codice. Svolgono un ruolo cruciale nell'organizzazione e gestione delle funzionalità del programma. Sebbene esistano numerose sezioni, comprenderne alcune chiave è fondamentale per l'analisi malware di base.

Di seguito le principali sezioni su cui porre particolare attenzione:

- **.text:** Questa sezione contiene il codice effettivo che il programma esegue. È qui che vengono archiviate le istruzioni che indicano al computer cosa fare. Analizzare questa sezione nell'analisi malware aiuta a identificare chiamate di funzione sospette, riferimenti a dati e potenziali attività dannose.
- **.rdata:** Questa sezione contiene dati di sola lettura come stringhe, costanti e risorse utilizzate dal programma. Il malware potrebbe incorporare stringhe offuscate o sfruttare vulnerabilità all'interno di questa sezione. Esaminarla può rivelare messaggi nascosti, dati di configurazione o indizi sulla sua origine.
- **.data:** Questa sezione memorizza variabili di dati inizializzate utilizzate dal programma durante l'esecuzione. Il malware potrebbe utilizzare questa sezione per archiviare dati temporanei o impostazioni di configurazione relative alle sue attività dannose. Analizzarla può aiutare a scoprire variabili nascoste, canali di comunicazione o codice iniettato.
- **.rsrc:** Questa sezione contiene le risorse del programma, come icone, immagini e menù. Sebbene apparentemente innocua, gli aggressori potrebbero abusare di questa sezione per nascondere codice dannoso o sfruttare vulnerabilità nelle librerie di analisi delle risorse. Esaminarla può rivelare payload nascosti o funzionalità mascherate.
- **.idata:** Questa sezione contiene informazioni sulle funzioni importate da altre librerie (DLL). Analizzare questa sezione può identificare importazioni sospette o inaspettate che potrebbero suggerire le capacità e le dipendenze del malware.

Altre sezioni da considerare

- **.edata:** Contiene variabili di dati non inizializzate, a volte utilizzate per memorizzare dati di configurazione o valori temporanei nel malware.
- **.reloc:** Contiene informazioni di rilocazione per regolare gli indirizzi dopo il caricamento del programma in memoria. Il malware potrebbe utilizzare questa sezione per l'iniezione di codice o tecniche anti-analisi.

Sezioni di cui si compone l'eseguibile oggetto dell'analisi

In particolare, il malware che stiamo studiando, è composto dalle seguenti sezioni, che, per i motivi visti nella precedente pagina, dovranno essere oggetto di analisi più approfondita:

- .text
- .rdata
- .data

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

File: Malware_U3_W2_L5.exe

- Dos Header
- Nt Headers
 - File Header
 - Optional Header
 - Data Directories [x]
- Section Headers [x]
- Import Directory
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler
- Rebuilder
- Resource Editor
- UPX Utility

Malware_U3_W2_L5.exe

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

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZÿ...
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	,.....@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	E8	00	00	00è.....
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	!!? ...I! L!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS.
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode...\$.....
00000080	49	29	7E	26	0D	48	10	75	0D	48	10	75	0D	48	10	75	I)~&.H!u.H!u.H!u
00000090	3B	6E	1B	75	0C	48	10	75	8E	54	1E	75	03	48	10	75	!n!u!H!u!T!u!H!u
000000A0	3B	6E	1A	75	20	48	10	75	0D	48	10	75	0A	48	10	75	!n!u.H!u.H!u.H!u
000000B0	6F	57	03	75	0E	48	10	75	0D	48	11	75	20	48	10	75	oW!u!H!u.H!u.H!u
000000C0	3B	6E	05	75	0C	48	10	75	52	69	63	68	0D	48	10	75	!n!u!H!uRich.H!u
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000E0	00	00	00	00	00	00	00	00	50	45	00	00	4C	01	03	00PE...L!
000000F0	A1	CC	49	4D	00	00	00	00	00	00	00	00	00	E0	00	0F	!i!M...P...à..
00000100	0E	01	06	00	00	50	00	00	00	50	00	00	00	00	00	00	! ...P...P...
00000110	B0	11	00	00	00	10	00	00	00	60	00	00	00	00	40	00	*@..
00000120	00	10	00	00	00	10	00	00	04	00	00	00	00	00	00	00

Costrutti noti del codice

Passiamo ora ad esaminare il codice in linguaggio assembly fornito.

Tra i costrutti che sappiamo riconoscere, di particolare interesse per l'analisi sono:

Riquadro giallo -> 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
```

Riquadro blu -> Costrutto condizionale IF



Riquadro verde -> Rimozione dello stack

```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

Nella pagina a seguire, vedremo un'analisi più dettagliata del codice, lavorando sui singoli costrutti, e cercheremo di comprendere il funzionamento del malware.

Analisi del codice assembly

Prologo della funzione

- **push ebp:** Salva il puntatore di base (EBP) sullo stack, preservando l'attuale frame dello stack.
- **mov ebp, esp:** Imposta il puntatore di base sul puntatore dello stack corrente (ESP), creando un nuovo frame dello stack per la funzione.
- **push ecx:** Salva il registro ECX sullo stack, potenzialmente conservando il suo valore per un uso successivo.

Verifica della connessione Internet

- **push 0:** Inserisce 0 sullo stack come argomento dwReserved per la funzione InternetGetConnectedState.
- **push 0:** Inserisce un altro 0 sullo stack come argomento lpdwFlags.
- **call ds:InternetGetConnectedState:** Chiama la funzione InternetGetConnectedState per verificare la presenza di una connessione Internet attiva.
- **mov [ebp+var_4], eax:** Memorizza il valore di ritorno della funzione (0 per nessuna connessione, 1 per connesso) in una variabile locale a ebp-4.

Branch condizionale

- **cmp [ebp+var_4], 0:** Confronta il valore di ritorno memorizzato con 0 per vedere se esiste una connessione Internet.
- **jz short loc_40102B:** Salta all'etichetta loc_40102B se la connessione non è disponibile (flag di zero impostato).

Messaggio di successo

- **push offset aSuccessInternet:** Inserisce l'indirizzo di una stringa "Successo: Connessione Internet\n" sullo stack.
- **call sub_40117F:** Chiama un'altra funzione, probabilmente per stampare la stringa sulla console.
- **add esp, 4:** Pulisce lo stack rimuovendo l'argomento stringa inserito.
- **mov eax, 1:** Imposta il valore di ritorno della funzione a 1, indicando il successo.
- **jmp short loc_40103A:** Salta all'epilogo della funzione per uscire.

Messaggio di errore

- **loc_40102B:** Etichetta per il caso senza internet.
- **push offset aError1_1NoInte:** Inserisce l'indirizzo di una stringa "Errore 1.1: Nessun Internet\n" sullo stack.
- **call sub_40117F:** Chiama la stessa funzione per stampare il messaggio di errore.
- **add esp, 4:** Pulisce nuovamente lo stack.
- **xor eax, eax:** Imposta il valore di ritorno della funzione a 0, indicando un errore.

Analisi del codice assembly

Epilogo della funzione

- **loc_40103A:** Etichetta per il punto di uscita della funzione.
- **mov esp, ebp:** Ripristina il puntatore dello stack al suo valore originale prima della chiamata alla funzione.
- **pop ebp:** Ripristina il puntatore di base al suo valore precedente.
- **retn:** Restituisce dalla funzione, utilizzando il valore in EAX come valore di ritorno.

Considerazioni finali

Questo frammento di codice definisce una funzione che controlla la connessione Internet e stampa un messaggio di successo o di errore di conseguenza. Restituisce 1 per il successo e 0 per l'errore. Si basa su una funzione esterna `sub_40117F` per l'output dei messaggi e sulla funzione API di Windows `InternetGetConnectedState` per i controlli di connessione Internet.