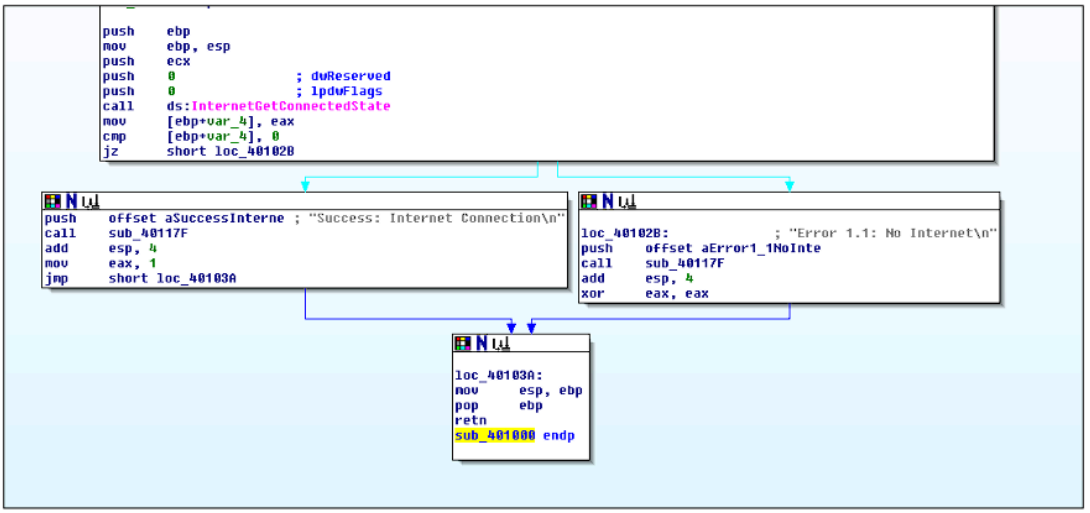
PROGETTO

Traccia:
Prima parte:
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?
Seconda parte:
Con riferimento alla figura 1 , risponde ai seguenti quesiti:
1. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
2. Ipotizzare il comportamento della funzionalità implementata
3. BONUS fare una tabella con significato delle singole righe di codice assembly

Figura 1



SOLUZIONE

PRIMA PARTE

1) Per identificare le librerie del file eseguibile bisogna eseguirlo con CFF explorer selezionando "Import directory", nella figura le due librerie importate che sono:

Malware_U3_W2_L5.exe

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

1.Kernel32.DLL: è un file DLL di Windows. DLL è l'acronimo di **D**ynamic **L**ink **L**ibrary. I file DLL sono programmi o estensioni del browser web necessari, perché contengono le risorse, i dati e il codice del programma.

2. WININET.DLL: è un modulo che contiene le funzioni Internet-relative usate dalle applicazioni di Windows.

2) Per controllare le sezioni di cui è composto il malware basta selezionare “ section headers".

Malware_U3_W2_L5.exe

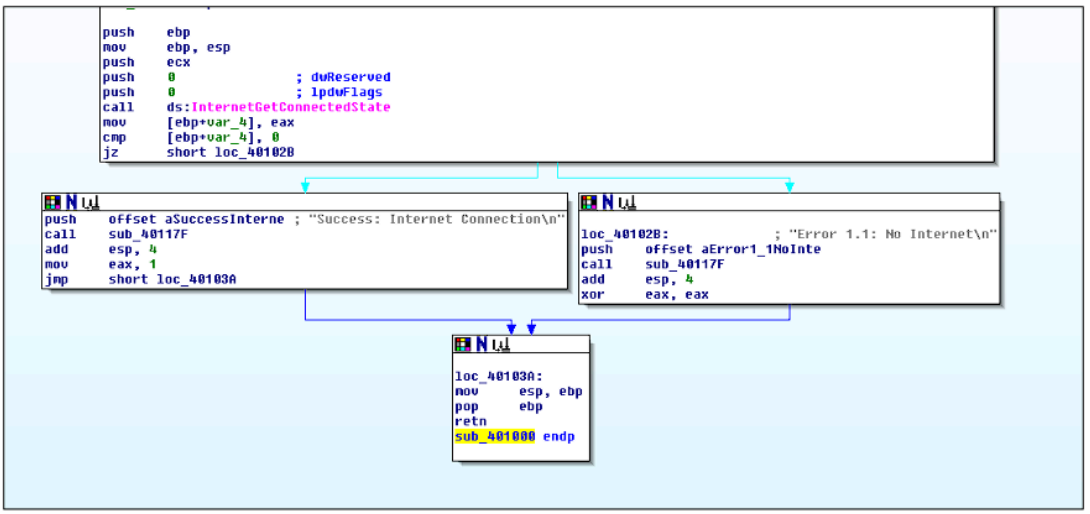
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

Questi sono le tre sezioni del malware in figura:

1. .text:contiene il codice eseguibile (si tratta in genere dell’unica sezione che contiene codice);
2. .rdata:contiene informazioni sull’import e l’export e può contenere anche altri dati read only;
3. .data:contiene dati globali accessibili da ogni punto del programma.

SECONDA PARTE:

Figura 1



1)I costrutti noti che si possono trovare nella figura sono:

1. Creazione dello stack tramite il blocco di istruzioni
```
push    ebp
mov     ebp, esp
```
2. Costrutto condizionale “IF” tramite il blocco di istruzioni

3. Rimozione dello stack tramite il blocco di istruzioni 

2) La funzionalità implementata in questo frammento di codice Assembly serve a verificare lo stato della connessione Internet e a visualizzare un messaggio di successo o di errore a seconda che la connessione sia attiva o meno.

Il comportamento della funzionalità implementata può essere descritto come segue:

1. Viene richiamata la funzione `sub_481898`.
2. La funzione verifica lo stato della connessione Internet chiamando `InternetGetConnectedState`.
3. Se la connessione Internet è attiva, la funzione visualizza un messaggio di successo ("Success: Internet Connection\n").
4. Se la connessione Internet non è attiva, la funzione visualizza un messaggio di errore ("Error 1.1: No Internet\n").
5. La funzione restituisce il controllo al chiamante.

3) In sintesi, la funzionalità implementata consente di verificare lo stato della connessione Internet e di informare l'utente sullo stato della connessione stessa.

Il frammento di codice fornito è scritto in Assembly language, specificamente per l'architettura x86. Appare essere una funzione che verifica lo stato della connessione Internet e visualizza un messaggio di successo o errore di conseguenza.

Ecco una suddivisione di ciò che il codice fa:

1. La funzione `sub_481898` inizia spingendo il puntatore base (`ebp`) e creando un nuovo frame di stack (`ebp, esp`).

2. Quindi spinge diversi valori nello stack, incluso `eax` e `ebp`, e fa una chiamata a `InternetGetConnectedState`. Questa funzione è utilizzata per determinare se esiste una connessione Internet attiva.
3. Se lo stato di connessione non è zero (ovvero esiste una connessione attiva), il codice salta a `loc_40102B`.
4. A `loc_40102B`, il codice spinge l'indirizzo di un messaggio di successo (`aSuccessInterne`) e chiama `sub_40117F`. Dopo la chiamata, il codice incrementa `eax` di 1 e salta a `loc_40103A`.
5. A `loc_40103A`, il codice pulisce lo stack e restituisce al chiamante.
6. Se non esiste una connessione Internet attiva, il codice salta il salto a `loc_40102B` e invece spinge l'indirizzo di un messaggio di errore (`aError1_1NoInte`). Quindi chiama `sub_40117F` e azzera il registro `eax` prima del ritorno.

In sintesi, questo codice verifica se esiste una connessione Internet attiva e visualizza un messaggio di successo o errore di conseguenza.