

Analisi Statica e Assembly

Traccia

Con riferimento al file Malware_U3_W2_L5 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:

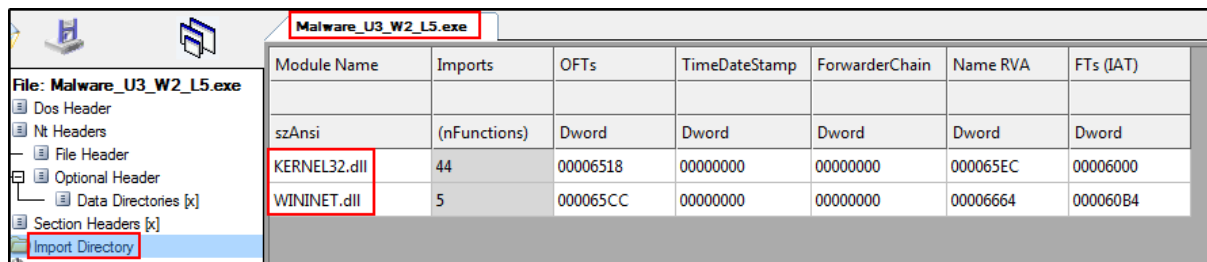
3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
4. Ipotesizzare il comportamento della funzionalità implementata
5. BONUS fare tabella con significato delle singole righe di codice assembly (Eseguito nel file Excel in allegato)

Traccia 1

Usando CFF Explorer > Import Directory andiamo a trovare le librerie importate dal file:

KERNEL32.dll: Contiene funzioni per interagire col sistema operativo, come gestione dei file, memoria, processi, thread e errori.

WININET.dll: Contiene funzioni per l'accesso a internet, gestione delle connessioni, caching, autenticazione, cookie e certificati di sicurezza.



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

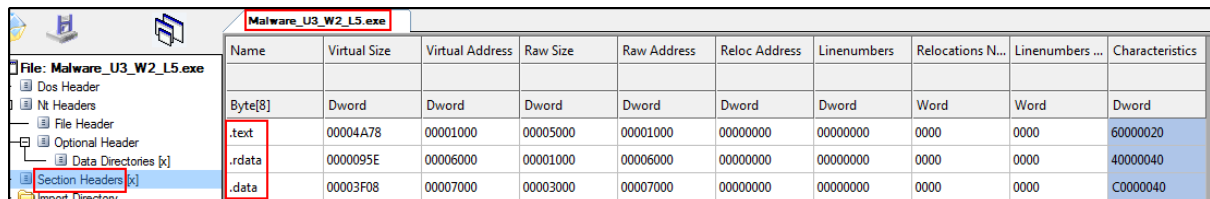
Traccia 2

Usando CFF Explorer > Section Headers andiamo a trovare le sezioni di cui si compone il file:

.text: Rappresenta la porzione di memoria che contiene il codice eseguibile del programma. Generalmente è l'unica sezione eseguita dalla CPU in quanto le altre sezioni contengono dati o informazioni a supporto

.rdata: Sta per Read-Only Data e svolge un ruolo simile a .data ma a differenza di quest'ultima contiene dati costanti o destinati alla sola lettura. Include generalmente le informazioni circa le librerie e funzioni importate ed esportate dall'eseguibile

.data: Contiene dati/variabili globali, ovvero dati che devono essere disponibili in qualsiasi parte del programma, e viene usata per capire meglio come un eseguibile gestisce le informazioni dinamiche

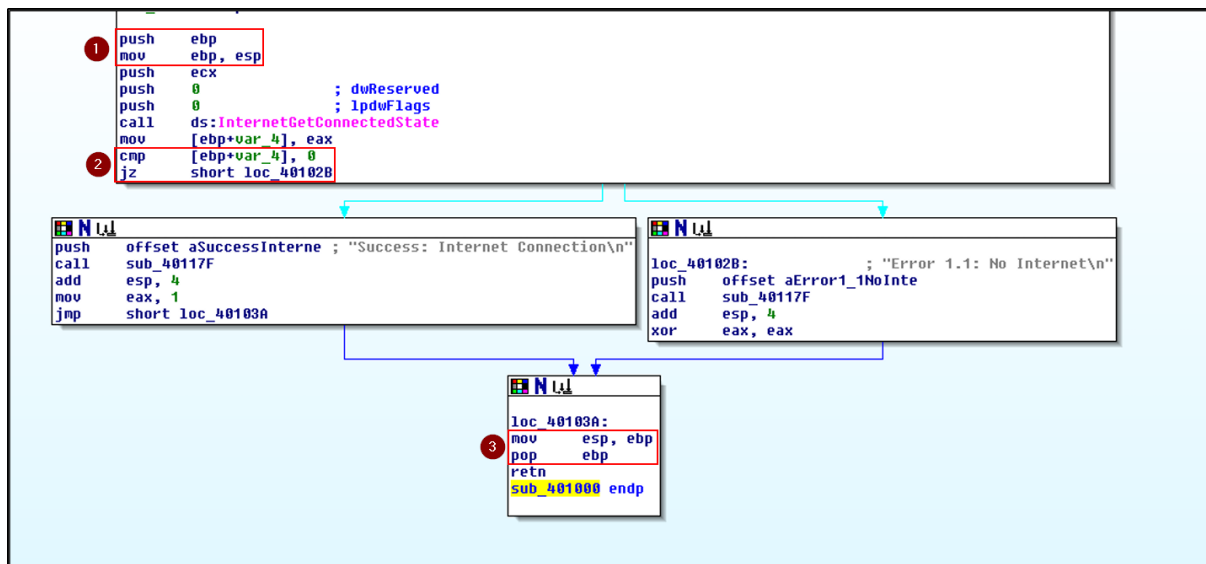


Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000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

Traccia 3

I costrutti noti presenti nel codice sono i seguenti:

1. **Creazione dello Stack:** Riguarda l'inizializzazione e la gestione di una struttura dati chiamata Stack. I passi principali includono l'inizializzazione del registro Stack Pointer (SP), che tiene traccia della cima dello stack, e l'allocazione di spazio per le variabili locali durante l'esecuzione di una funzione
2. **Condizione IF:** Condizione gestita attraverso un confronto seguito da un'istruzione di salto condizionato. Se la condizione è vera, il controllo del programma salta a un blocco di codice specifico. In sintesi, si effettua un confronto e, se la condizione è soddisfatta, si eseguono le istruzioni nel blocco IF
3. **Rimozione dello Stack:** Si riferisce al processo di deallocare lo spazio precedentemente assegnato per le variabili locali e altri dati dallo Stack



Traccia 4

La funzione InternetGetConnectedState ritorna un valore TRUE se c'è almeno una connessione a Internet disponibile, ma non garantisce che si possa stabilire una connessione a un host specifico.

Restituisce TRUE se è presente un modem attivo o una connessione Internet LAN o FALSE se non è presente alcuna connessione Internet o se tutte le connessioni Internet possibili non sono attualmente attive. Per altre informazioni, vedere la sezione Osservazioni.

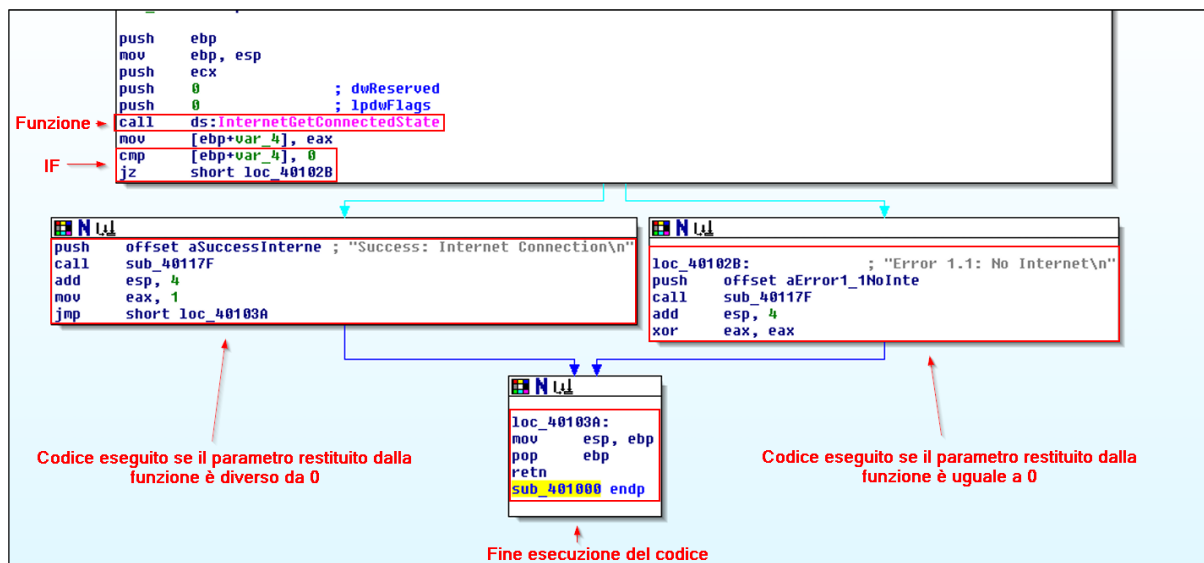
Il costrutto IF controlla il parametro restituito da quest'ultima funzione e:

Se uguale a 0 viene stampato a schermo Error 1.1: No Internet

Se diverso da 0 viene stampato a schermo Success: Internet Connection

Prendendo in considerazione quanto scritto sopra possiamo ipotizzare che la subroutine sub_40117F equivale a printf()

In sintesi viene controllato se avviene una connessione ad internet e viene restituito a schermo se quest'ultima è presente o meno. Questo fa pensare a un Malware che controlla se è disponibile una connessione a internet per eseguire azioni specifiche non definite in questo frammento di codice.



Ulteriore Analisi Statica con IDA

Aprendo il malware in questione (Malware_U3_W2_L5.exe) con IDA andiamo ad esaminare più a fondo il codice assembly per capire meglio il funzionamento del Malware

Notiamo che nella funzione Main viene chiamata la subroutine 401000 che corrisponde al frammento di codice analizzato nell'esercitazione di oggi

```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
main proc near

var_8= byte ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h

push    ebp
mov     ebp, esp
sub     esp, 8
call    sub_401000
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jnz     short loc_401148
```

```
sub_401000 proc near

var_4= dword ptr -4

push    ebp
mov     ebp, esp
push    ecx
push    0
push    0
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

Se il risultato non è zero salta a un altro blocco di codice dove chiama la subroutine 401040

```
cmp     [ebp+var_4], 0
jnz     short loc_401148
```

```
loc_401148:
call    sub_401040
mov     [ebp+var_8], al
```

Questa subroutine contiene due funzioni, InternetOpenA e InternetOpenUrlA, le quali:

InternetOpenA = Inizializza strutture di dati interni, in questo caso l'user agent "Internet Explorer 7.5/pma", e si prepara per future chiamate dall'applicazione

InternetOpen è la prima funzione WinINet chiamata da un'applicazione. Indica alla DLL Internet di inizializzare le strutture di dati interne e prepararsi per le chiamate future dall'applicazione. Al termine dell'uso delle funzioni Internet, l'applicazione deve chiamare **InternetCloseHandle** per liberare l'handle e le risorse associate.

InternetOpenUrlA = Controlla se la connessione a un URL specifico, in questo caso "http://www.practicalmalwareanalysis.com", avviene con successo e restituisce un handle valido in caso positivo, altrimenti restituisce NULL

Restituisce un handle valido all'URL se la connessione viene stabilita correttamente o **NULL** se la connessione ha esito negativo. Per recuperare un messaggio di errore specifico, chiamare **GetLastError**. Per determinare il motivo per cui l'accesso al servizio è stato negato, chiamare **InternetGetLastResponseInfo**.

```
Sub_401040 proc near
var_210= byte ptr -210h
var_20F= byte ptr -20Fh
var_20E= byte ptr -20Eh
var_20D= byte ptr -20Dh
var_20C= byte ptr -20Ch
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= byte ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 210h
push    0
push    0
push    0
push    0
push    offset aInternetExplor ; "Internet Explorer 7.5/pma"
call    ds:InternetOpenA
mov     [ebp+var_C], eax
push    0
push    0
push    0
push    0
push    offset aHttpWww_practi ; "http://www.practicalmalwareanalysis.com"...
mov     eax, [ebp+var_C]
push    eax
call    ds:InternetOpenUrlA
mov     [ebp+var_10], eax
cmp     [ebp+var_10], 0
jnz     short loc_40109D
```

In sintesi questo frammento di codice controlla se è possibile stabilire una connessione al sito http://practicalmalwareanalysis.com usando l'user agent Internet Explorer 7.5/pma

In caso di esito positivo salta al blocco di codice che contiene loc_40109D

```
loc 40109D:  
lea     edx, [ebp+var_8]  
push    edx  
push    200h  
lea     eax, [ebp+var_210]  
push    eax  
mov     ecx, [ebp+var_10]  
push    ecx  
call    ds:InternetReadFile  
mov     [ebp+var_4], eax  
cmp     [ebp+var_4], 0  
jnz     short loc_4010E5
```

Qui troviamo la funzione `InternetReadFile` che viene utilizzata per leggere dati da una risorsa su Internet, in questo caso vengono precedentemente passati 200h (512 in decimale) byte in una variabile e di conseguenza la funzione ritornerà vero se riesce a leggere i primi 512 byte dalla pagina internet chiamata in precedenza


`InternetReadFile` funziona in modo molto simile alla funzione `ReadFile` di base, con alcune eccezioni. In genere, `InternetReadFile` recupera i dati da un handle DELLA RETE COME flusso sequenziale di byte. La quantità di dati da leggere


In caso di esito positivo avverrà un salto alla location loc_4010E5


Qui avvengono diverse operazioni di comparazione di valori esadecimali che essendo di 3 caratteri non sembrano appartenere a dei numeri. Ad ogni comparazione in caso di esito positivo si procede con la successiva mentre in caso di esito negativo si salta a un frammento di codice che stampa su schermo "Error 2.3: Fail to get command\n"


Convertendo i valori esadecimali in ASCII vengono fuori i seguenti caratteri: < ! - -


In sintesi questo frammento di codice controlla se sono presenti i caratteri <!-- all'inizio dei dati letti in precedenza e in caso di esito positivo per ogni carattere salta alla location loc_40111D

N 
loc_4010E5:
movsx ecx, [ebp+var_210]
cmp ecx, 3Ch
jnz short loc_40111D

N 
movsx edx, [ebp+var_20F]
cmp edx, 21h
jnz short loc_40111D

N 
movsx eax, [ebp+var_20E]
cmp eax, 2Dh
jnz short loc_40111D

N 
movsx ecx, [ebp+var_20D]
cmp ecx, 2Dh
jnz short loc_40111D

N 
loc_40111D: ; "Error 2.3: Fail to get command\n"
push offset aError2_3FailTo
call sub_40117F
add esp, 4
xor al, al