Progetto S10L5

Alessandro Moscetti

Dovevamo rispondere ai seguenti quesiti riguardanti il file ‘Malware_U3_W2_L5’:

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

Le informazioni richieste possono essere trovate tramite il tool per l’analisi dei malware **CFF Explorer.**
CFF Explorer esamina gli header dei PE (Portable Execute), file eseguibili comuni utilizzati nel sistema Windows, contenente varie informazioni riguardo il file come librerie importate, sezioni, ecc.

Punto 1

Esaminare le librerie chiamate da un eseguibile è importante in un analisi di un malware perchè permette di farsi un’idea sulle funzioni richieste dal file per eseguire il codice all’interno e per quale contesto operativo è progettato.
Per vedere le librerie importate dal file eseguibile lo andiamo ad aprire tramite CFF Explorer e clicchiamo sulla sezione ‘Import Directory’.

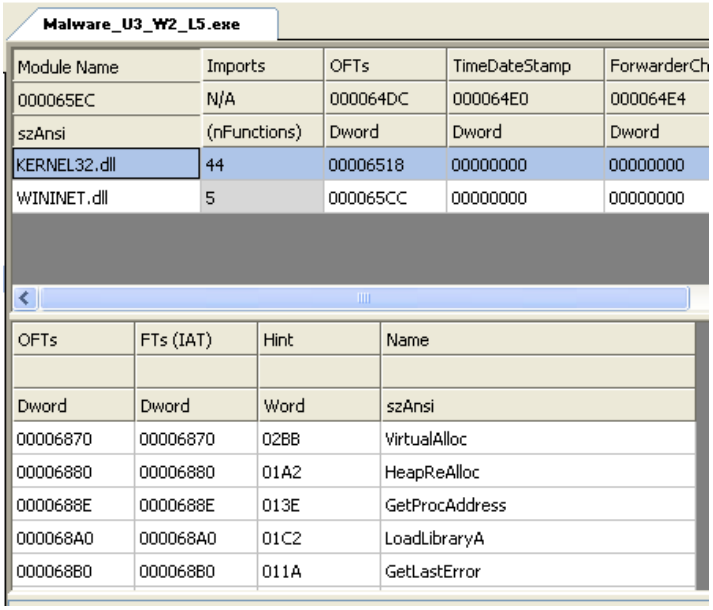
Così facendo vediamo che le librerie importate sono due:

- KERNEL32.dll: Libreria contenente le funzioni principali per interagire con il sistema operativo.
- WININET.dll: Libreria contenente le funzioni per l’implementazione di alcuni protocolli di rete.

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC
WININET.dll	5	000065CC	00000000	00000000	00006664

Possiamo anche vedere le funzioni che vengono importate tramite le librerie importate, per ipotizzare alcuni comportamenti dell’eseguibile.

La libreria KERNEL32.dll ne importava 44 andandole ad esaminare troviamo le funzioni ‘LoadLibraryA’ e ‘GetProcAddress’ che sono funzioni del sistema operativo per importare librerie.
Questo comportamento è utilizzato da molti malware che importano le librerie in runtime cioè chiamando le librerie all’occorrenza di una determinata funzione risultando meno invasivo e rilevabile.

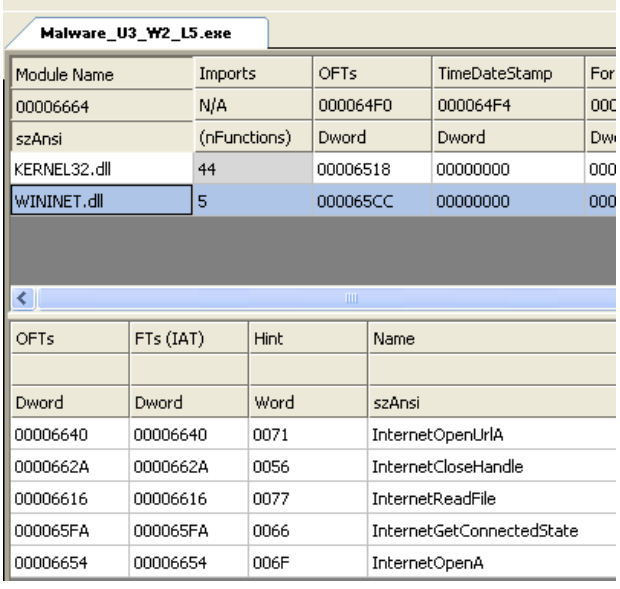


Malware_U3_W2_L5.exe

Module Name	Imports	OFTs	TimeDateStamp	ForwarderCh
000065EC	N/A	000064DC	000064E0	000064E4
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000
WININET.dll	5	000065CC	00000000	00000000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006870	00006870	02BB	VirtualAlloc
00006880	00006880	01A2	HeapReAlloc
0000688E	0000688E	013E	GetProcAddress
000068A0	000068A0	01C2	LoadLibraryA
000068B0	000068B0	011A	GetLastError

La libreria WININET.dll importava 5 funzioni, esaminandole possiamo intuire come l’eseguibile verifichi una connessione a internet sul dispositivo (InternetGetConnectedState) e cerchi un Url (InternetOpenUrlA).
Questo comportamento può portare all’ipotesi di un malware di tipo downloader cioè file malevoli che all’esecuzione scaricano un altro/altri file da internet, solitamente altri malware.
In una situazione reale non sarebbero sufficienti queste informazioni per determinare un ipotesi concreta senza proseguire nell’analisi.



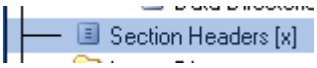
Malware_U3_W2_L5.exe

Module Name	Imports	OFTs	TimeDateStamp	For
00006664	N/A	000064F0	000064F4	000
szAnsi	(nFunctions)	Dword	Dword	Dw
KERNEL32.dll	44	00006518	00000000	000
WININET.dll	5	000065CC	00000000	000

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

Punto 2

Esaminare le sezioni di un eseguibile è importante in un analisi di un malware per permettere di identificare e comprendere meglio come il codice è organizzato. Possiamo vedere anche le sezioni importate dallo stesso tool cliccando sulla sezione ‘Section Headers’.



Così facendo vediamo che le sezioni importate sono tre:

- **.text**: Contiene le istruzioni che verranno eseguite dalla CPU (righe di codice)
- **.rdata**: Contiene generalmente le informazioni riguardanti le librerie e le sezioni importate e esportate del file.
- **.data**: Contiene solitamente i dati e le variabili globali del programma eseguibile.

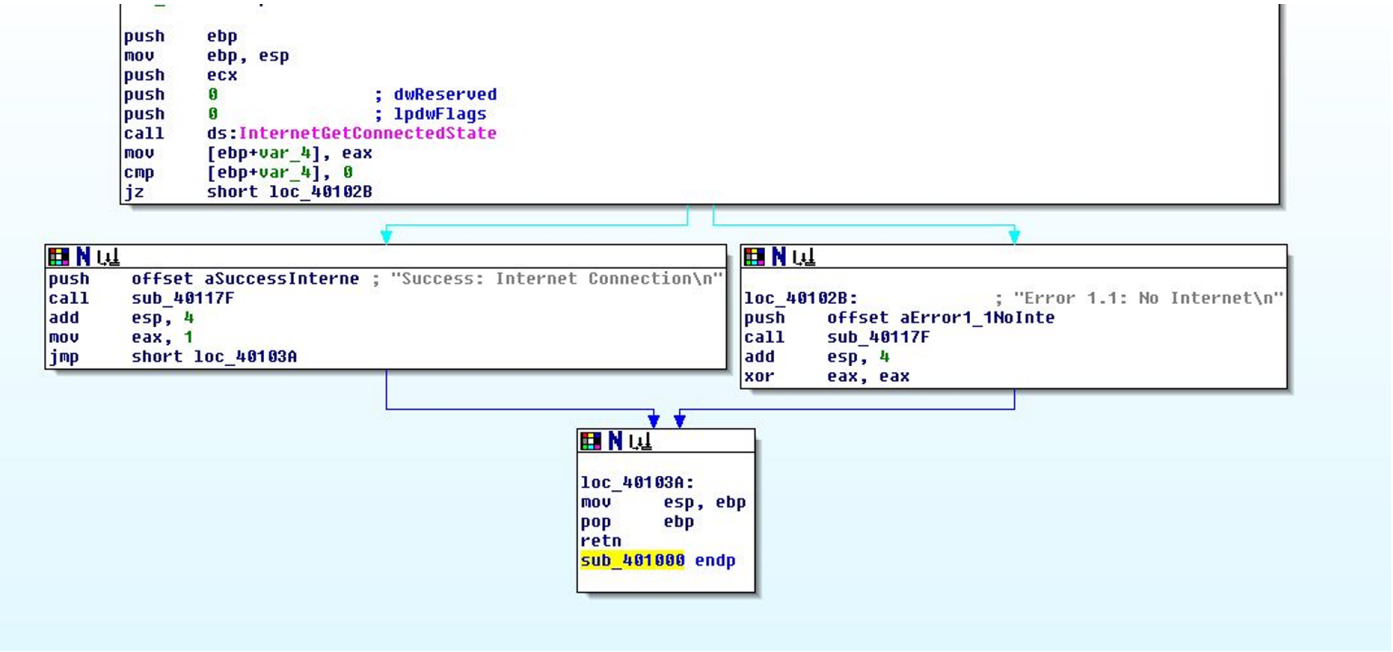
Malware_U3_W2_L5.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumb
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword
.text	00004A78	00001000	00005000	00001000	00000000	0000000C
.rdata	0000095E	00006000	00001000	00006000	00000000	0000000C
.data	00003F08	00007000	00003000	00007000	00000000	0000000C

Seconda parte progetto

Dovevamo rispondere ai seguenti quesiti riguardanti la figura sottostante:

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



Punto 3

Ho trovato i seguenti costrutti noti:

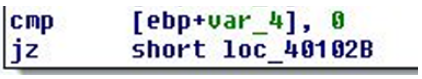
- Creazione dello stack: con l'utilizzo di queste istruzioni viene creato un nuovo frame nello stack per permettere alla funzione di utilizzare variabili locali.

```
push    ebp
mov     ebp, esp
```

- Chiamata di funzione: parametri passati allo stack tramite l'istruzione 'push' e chiamo la funzione 'InternetGetConnectedState' tramite l'istruzione 'call'.

```
push    ecx
push    0               ; dwReserved
push    0               ; lpdwFlags
call    ds:InternetGetConnectedState
```

- Ciclo If: in questo caso l'istruzione jz salterà alla locazione loc_40102B se il ZF (Zero flag) sia settato a 1 ossia la comparazione precedente (cmp) dia come risultato destinazione=sorgente ([ebp+var_4]=0).



- Chiusura stack: tramite le prime due righe libera la memoria allocata nello stack durante l'apertura del frame della funzione, l'istruzione retn effettua il salto di ritorno all'indirizzo memorizzato nello stack durante la chiamata della funzione.

Punto 4

Analizzando il codice assembly, ipotizzo che la funzionalità implementata sia la verifica dello stato di connessione a Internet del dispositivo host.
Lo si può capire dal ciclo if, basato sul valore di ritorno della chiamata a 'InternetGetConnectedState'.
Nel controllo, si salta tramite jz se il flag ZF (Zero Flag) è settato a 1, visualizzando (presumibilmente visto l'invio della stringa) il messaggio 'Error 1.1: No Internet\n'; al contrario, se ZF è 0, il flusso prosegue mostrando il messaggio 'Success: Internet Connection'.

Bonus

Spiegare il significato delle singole righe di assembly.

Codice con spiegazioni

1. **push ebp** // Viene mandato il valore di ebp nello stack
2. **mov ebp, esp** // Il valore del registro esp viene caricato nel registro ebp

3. **push ecx** // Viene mandato il valore di ecx nello stack
4. **push 0 ; dwReserved** //Viene mandato il valore 0 riguardante la variabile dwReserved nello stack.
5. **push 0 ; lpdwFlags** //Viene mandato il valore 0 riguardante la variabile lpwdFlags nello stack.
6. **call ds:InternetGetConnectedState** // Viene chiamata la funzione InternetGetConnectedState
7. **mov [ebp+var_4], eax** // Il valore del registro eax viene copiato nella variabile [ebp+var_4]
8. **cmp [ebp+var_4], 0** //Viene comparata la variabile [ebp+var_4] a 0
9. **jz short loc_40102B** // Salta alla locazione 40102B se il ZF è 1

//Success internet connection

10. **push offset aSuccessInterne ; "Success: Internet Connection\n"** // Invia la stringa nello stack
11. **call sub_40117F** // Viene chiamata la funzione alla locazione 40117F
12. **add esp, 4** // Viene aggiunto 4 al registro esp
13. **mov eax, 1** // Viene assegnato il valore 1 al registro eax
14. **jmp short loc_40103A** // Salta alla locazione 40103A

//Error internet connection

loc_40102b:

15. **push offset aError1_NoInte ; "Error 1.1: NoInternet\n"**// Invia la stringa nello stack
16. **call sub_40117F** // Viene chiamata la funzione alla locazione 40117F
17. **add esp, 4** // Viene aggiunto 4 al registro esp
18. **xor eax, eax** // Esegue un XOR tra eax ed eax, inizializzando il registro eax

//

loc_40103A:

19. **mov esp, ebp** // Il valore del registro ebp viene caricato nel registro esp
20. **pop ebp** // Viene ripristinato ebp dallo stack
21. **retn** // Effettua un salto di ritorno alla funzione chiamante
22. **sub_401000 endp** // Chiude il codice