

Analisi Statica e Dinamica: un approccio pratico

Lo scopo della prima parte del progetto è di analizzare il malware denominato **Malware_U3_W2_L5** che si trova all'interno della nostra macchina virtuale per l'analisi dei malware.

Per prima cosa, dopo aver inserito il Malware sul programma **CFF Explorer** ho identificato le seguenti librerie:

Malware_U3_W2_L5.exe						
Module Name	Imports	OFTs	TimeStamp	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

KERNEL32.dll: libreria che si occupa di interagire con il sistema operativo, da accesso alla gestione della memoria e alla manipolazione dei file.

WININET.dll: contiene le implementazioni per alcuni protocolli di rete come **HTTP**, **NTP** e **FTP**.

Ciascuna di queste due librerie che ho individuato presenta una serie di **funzioni**, ovvero dei blocchi di codice che eseguono compiti specifici.

Funzioni Libreria Kernel32.dll

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
000066C6	000066C6	02AD	UnhandledExceptionFilter
000066E2	000066E2	0124	GetModuleFileNameA
000066F8	000066F8	00B2	FreeEnvironmentStringsA
00006712	00006712	00B3	FreeEnvironmentStringsW

Funzioni Libreria Wininet.dll

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

Questo Malware si compone di **tre sezioni** che sono **.text**, **.rdata**, **.data**

.text: contiene le istruzioni che verranno mandate alla CPU per poter essere eseguite all' avvio del software.

.rdata: contiene le informazioni sulle librerie e sulle funzioni che vengono importate ed esportate dall' eseguibile.

.data: contiene le variabili globali del programma eseguibile.

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

Analisi Codice Assembly

La seconda parte del progetto ci richiedeva di analizzare il seguente codice assembly, identificando i costrutti noti ed ipotizzando il comportamento della funzionalità del codice.

```
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
```

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

```
loc_40102B:
; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

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

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

Queste prime due righe di codice identificano la **creazione dello stack**.

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

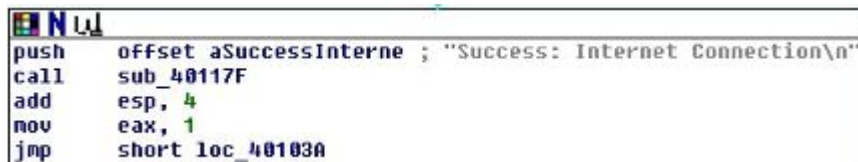
Questa è una chiamata alla funzione **InternetGetConnectedState**, in cui il

programma prende due valori e li passa come parametri, andando poi a verificare tramite un ciclo if, con le righe successive, se il sistema ha o meno la connessione ad internet.

```
cmp     [ebp+var_4], 0
jz      short loc_401028
```

Ciclo if, ovvero verifica se il valore di ritorno della funzione è **diversa** da **0** tramite il cmp,

in quel caso significa che c'è una connessione attiva e si avrà in output il seguente messaggio:



```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

Chiusura dello stack:

```
mov esp, ebp  
pop ebp  
retn
```

Il programma in sostanza cerca di determinare se sulla macchina bersaglio ci sia o meno una connessione internet attiva attraverso la funzione **InternetGetConnectedState**. Tramite il costrutto `if` esegue un controllo sulla funzione e, a seconda del parametro restituito -uguale a **0** o diverso da **0** indica la presenza o meno della connessione internet stampando a schermo, in output, un messaggio di errore (**Error 1.1: no internet**) o di successo (**Success: internet connection**).