

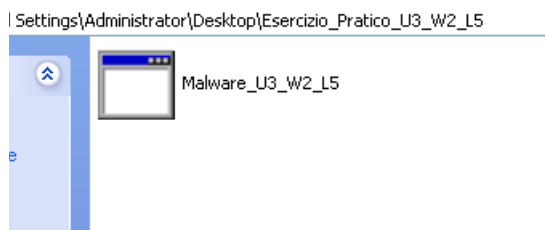
REPORT ANALISI MALWARE

Per studiare, indagare e capire il comportamento di un malware verranno utilizzate un insieme di tecniche e tool al fine di averne un quadro preciso, tramite analisi statica e dinamica, ossia senza e con esecuzione.

In particolare si analizzeranno tramite analisi basica statica :

1. Quali librerie verranno importate dal file eseguibile.
2. Quali sono le sezioni di cui si compone
3. Identificazione costrutti
4. Ipotesi

Il file per cui è richiesta l'analisi, si trova in Virtual lab su macchina appositamente settata per non avere alcun tipo di connessione con l'esterno, potendo essere sicuri di eseguirlo in tutta sicurezza.



Importante poiché i tipi di librerie importate, con funzioni annesse, restituisce un'ipotesi generica sulle azioni che il potenziale file dannoso potrebbe compiere sul sistema.

1. Librerie importate e relative funzioni

Si utilizzerà quindi il tool presente sulla macchina CFF EXPLORER, per analizzare l'Header del file PE (Portable Executable) in cui sono contenute le info relative al sistema operativo e alla sua gestione file relativa al codice.

Malware_U3_W2_L5.exe

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

La sezione dedicata alle librerie è appunto la “Import directory” , dove per l'appunto ci verrà restituita una chiara tabella delle librerie e delle sue funzioni nel riquadro sottostante.

Nello specifico caso di questo PE vengono importate due librerie:

Winlnet.dll: Libreria che contiene funzioni per implementazioni protocolli di rete come http, FTP, NTP, accedere ad internet, download e tutto ciò concernente connessioni a rete.

Comprendente 5 funzioni:

InternetOpenUrlA: Chiamata che avviene solo dopo che una richiesta http è andata a buon fine .

InternetCloseHandle : Termina tutte le operazioni in sospeso sull’handle e ne elimina i dati

InternetReadFile:handle precedentemente restituito da InternetOpenUrlA, recupera e ne legge i dati scaricati

InternetGetConnectedState: Recupera lo stato connesso del sistema globale

InternetOpenA : Inizializza l’uso di un’applicazione delle funzioni Winlnet

Seconda libreria importata è invece la **KERNEL32.dll** comprendente ben 44 funzioni.

KERNEL32.dll : Libreria comune che contiene le funzioni principali per interagire con il sistema operativo, e ne gestisce allocazioni di memoria. Libreria quasi sempre importata dai Malware per le sue notevoli possibilità di manipolazione file e sistema

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

Particolarmente rilevanti e sospette risultano le funzioni:

Sleep = Delay esecutivo per evitare la rilevazione

MultiBytetoWideChar= Funzione di mappatura che permette multiple rappresentazioni di una stessa stringa, lasciando l'applicazione potenzialmente esposta ad attacchi

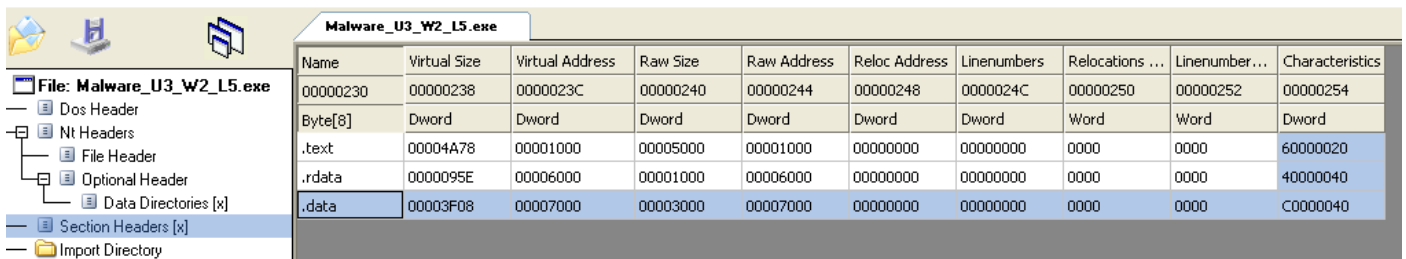
GetModuleFileNameA = Funzione usata per recuperare il filename di un modulo del processo. I malware possono utilizzare questa funzione per modificare o copiare i file in processi in corso.

5. Sezioni di cui si compone il Malware.

Quali sono le sezioni di cui si compone quindi il file eseguibile in questione? E perché è importante analizzarle?

I malware possono utilizzare packer per nascondere codici malevoli all'interno di una delle sezioni del PE padre, estraendolo e cacicandolo in memoria in runtime, quindi per individuare un packer la "import section" mostrerà pochi importi, ed un alto livello di entropia, dovuto ad una compressione per risultare quanto più nascosto possibile.

Anche per il recupero di questa informazione sarà possibile servirci di CFF Explorer, questa volta spostandosi sulla sezione "**Section Headers**"



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
00000230	00000238	0000023C	00000240	00000244	00000248	0000024C	00000250	00000252	00000254
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

.Text = Sezione comprendente le righe di codice che verranno eseguite dalla CPU una volta avviato

.rdata = Sezione comprendente dati inizializzati "Read-only", informazioni relative all'importazione ed esportazione librerie e conseguenti funzioni

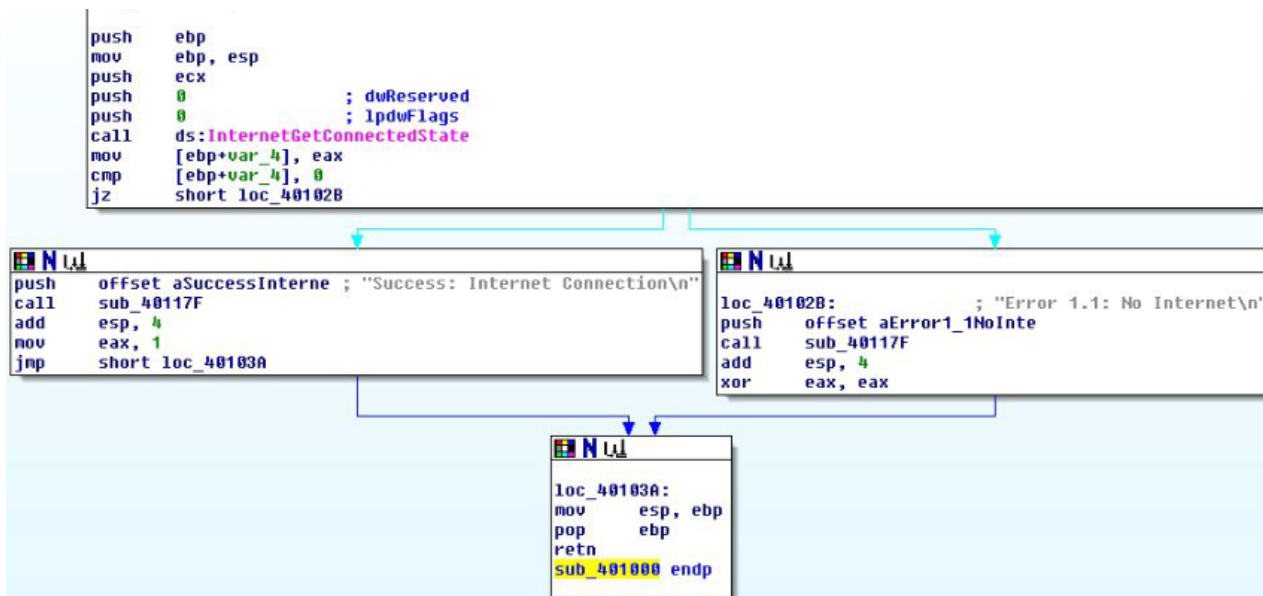
.data = Sezione contenente dati inizializzati e variabili globali (Ossia globalmente dichiarate e quindi accessibili da tutte le procedure e dal programma principale, definite al di fuori delle procedure)

6.

Analisi Costrutti noti

Un fondamentale step nella Malware analysis risiede nell'analisi statica avanzata che presuppone fondamenti di "reverse engineering", ricostruzione ad alto livello delle funzionalità di un malware tramite analisi del suo codice in assembly.

Di seguito riportato la porzione di codice fornitoci, che verrà in seguito sezionata.



1.

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

Creazione dello stack

2.

```

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

```



Passaggio parametri con istruzione Push e chiamata a funzione "InternetGetConnectedState"

3.

```

mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B

```

Spostamento valori da registro EBP in EAX. Ciclo If , comparazione tra 0 (False) ed un jz (Jump se è zero). Il salto a locazione (short loc_40102B) avverrà quindi con successo se InternetGetconnectedState ritorna valore 0 (Falso) (Connessione assente)

7.

```

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

```

Risultato dell'If statement nel caso InternetGetConnectedState ritorni valore Vero, con salto a locazione "short loc_40102B" non avvenuto e restituzione stringa a schermo "Success: Internet Connection" tramite la subroutine _40117F, infine conclude con salto a locazione _40103A (ritorno di processo)

5.

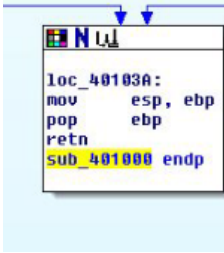
```

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

```

Risultato dell'If statemente nel caso InternetGetConnectedState ritorni valore Falso, con salto a locazione "short loc_40102B" avvenuto e restituzione stringa "Error 1.1 : No Internet" tramite sub routine _40117F

6.



Ritorno a processo con locazione _40103A, chiusura e pulizia stack su subroutine _401000 (Indirizzo Virtuale).

4. Ipotesi comportamento

Dal risultato delle analisi si può affermare che per prima cosa il malware cerchi di stabilire una connessione su target vittima, e una volta trovata, stampi a schermo il risultato di avvenuta connessione, in seguito cerca di connettersi tramite internet ad un indirizzo specifico.

Potrebbe trattarsi di un downloader o di un Ransomware (Ipotesi avanzata per l'uso di funzioni quali HeapReAlloc, FlushFileBuffer, VirtualFree e molte altre riconducibili a manipolazione, sovrascrittura e criptazione)