

Report - Analisi statica e dinamica

Analizzo in maniera statica il malware “**Malware_U3_W2_L5.exe**” con lo strumento **CFF Explorer**.

Una volta aperto il file, navigo nel programma fino ad arrivare alla pagina “*Import Directory*”, la quale mi mostra le librerie e le funzioni importate dal file.

The screenshot shows the CFF Explorer VIII application window. The title bar reads "CFF Explorer VIII - [Malware_U3_W2_L5.exe]". The sidebar on the left contains a tree view with the following items: File: Malware_U3_W2_L5.exe, Dos Header, Nt Headers, File Header, Optional Header, Data Directories [x], Section Headers [x], Import Directory (selected), Address Converter, Dependency Walker, Hex Editor, Identifier, Import Adder, Quick Disassembler, Rebuilder, Resource Editor, and UPX Utility. The main pane displays the "Import Directory" for "Malware_U3_W2_L5.exe". It contains two tables.

Module Name	Imports	OFTs	TimeDateStamp	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

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
0000672C	0000672C	02D2	WideCharToMultiByte
00006742	00006742	0106	GetEnvironmentStrings
0000675A	0000675A	0108	GetEnvironmentStringsW
00006774	00006774	026D	SetHandleCount
00006786	00006786	0152	GetStdHandle
00006796	00006796	0115	GetFileType

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

Le librerie importate, in questo caso, sono due: Kernel32.dll e Wininet.dll.

Kernel32.dll, nei sistemi operativi Windows, è un modulo che contiene le funzioni principali per interagire con il sistema operativo, gestire memoria, file e processi. Ci sono in tutto 44 funzioni su questa libreria.

Questa libreria permette al malware, per esempio, di creare e modificare file sensibili, far partire altri processi malevoli, importare altre librerie con funzioni potenzialmente malevole usando “LoadLibrary” e “GetProcAddress”, come in questo caso. Può anche utilizzare funzioni per ottenere maggiori informazioni sul sistema bersaglio. Una delle funzioni in questo malware, per esempio, è “GetVersion”, che riesce infatti a ottenere la versione di Windows della macchina bersaglio.

Wininet.dll, nei sistemi Windows, dà accesso a funzionalità e operazioni legate alla connessione internet e protocolli come FTP, HTTP, NTP (Network Time Protocol) e altri. Ci sono in tutto 5 funzioni in questa libreria.

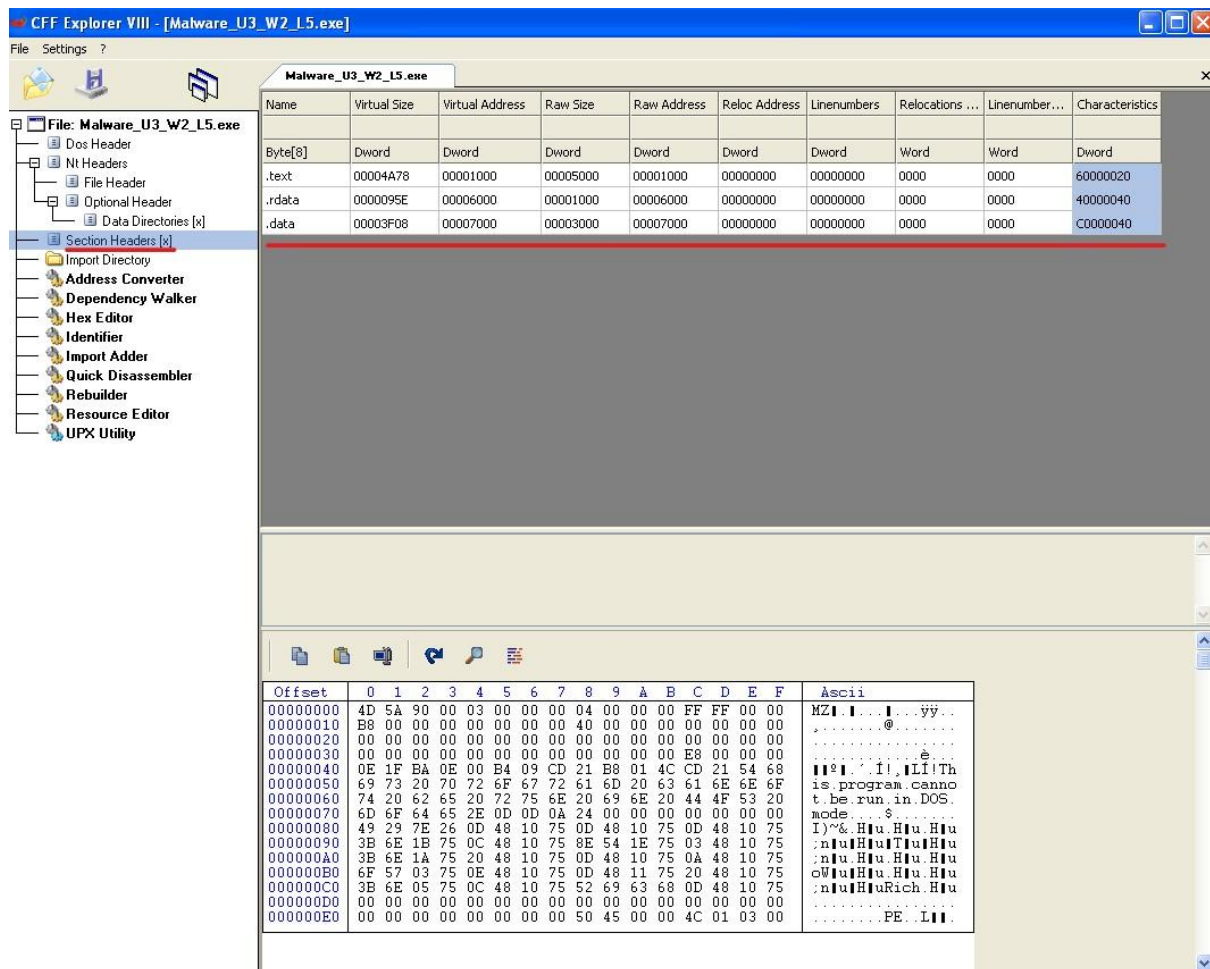
È maggiormente utilizzato per il furto e l'invio di dati sensibili su rete, per esempio su server remoti, per eseguire un'enumerazione di rete e scaricare altri componenti malevoli. Può inoltre avere l'accesso a cookie di sessione, impostazioni dei browser, manipolazione dei certificati e bypassare o modificare proxy già esistenti.

Le funzioni presenti permettono al malware di connettersi a internet, aprire url e leggere file. (InternetGetConnectedState, InternetOpenUrlA, InternetReadFile ecc.)

La funzione InternetOpenUrlA viene usata per codice scritto in ANSI ed è obsoleta, ma in questo caso può essere utile su questa versione di Windows XP.

E' altamente probabile che il malware abbia altre funzioni importate con la funzione "LoadLibrary" e "GetProcAddress".

Potrebbe inoltre usufruire di librerie importate dinamicamente, ma non è possibile scoprirlo per certo con un'analisi statica.



Il file ha tre sezioni: .text, .rdata e .data.

.text è la sezione più importante del programma che contiene le istruzioni e il codice da eseguire (tipicamente in C, C++ o assembly) dalla CPU una volta avviato l'eseguibile, con le sue funzioni e procedure. E' di solito supportato dai dati e informazioni contenuti nelle altre sezioni.

.rdata è la sezione che contiene informazioni read-only, ovvero dati che il programma legge senza modificare, come le informazioni sulle librerie e funzioni importate. Grazie a .rdata scopriamo che il malware, tramite "LoadLibrary" e "GetProcAddress", ha anche importato la libreria **MSVCRT.dll**, utile per la manipolazione di stringhe, allocazione di memoria e chiamate per input e output. Le sue funzioni saranno eseguite una volta avviato il file.

Malware_U3_W2_L5.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
00000208	00000210	00000214	00000218	0000021C	00000220	00000224	00000228	0000022A	0000022C
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

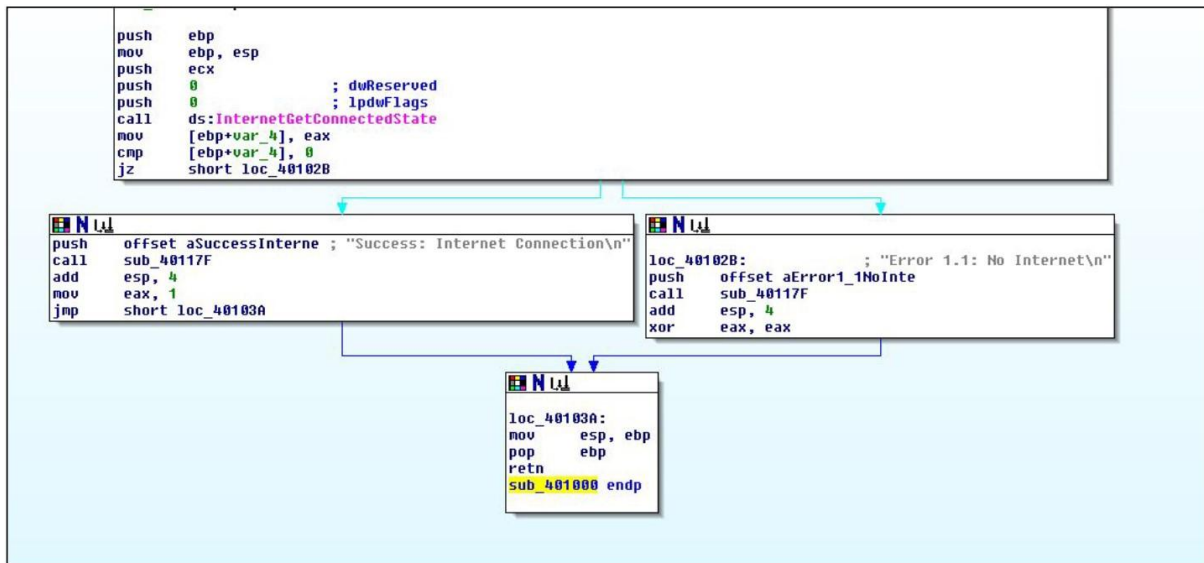
This section contains:

Data: 00006000
 Import Directory: 000064DC
 Import Address Table Directory: 00006000

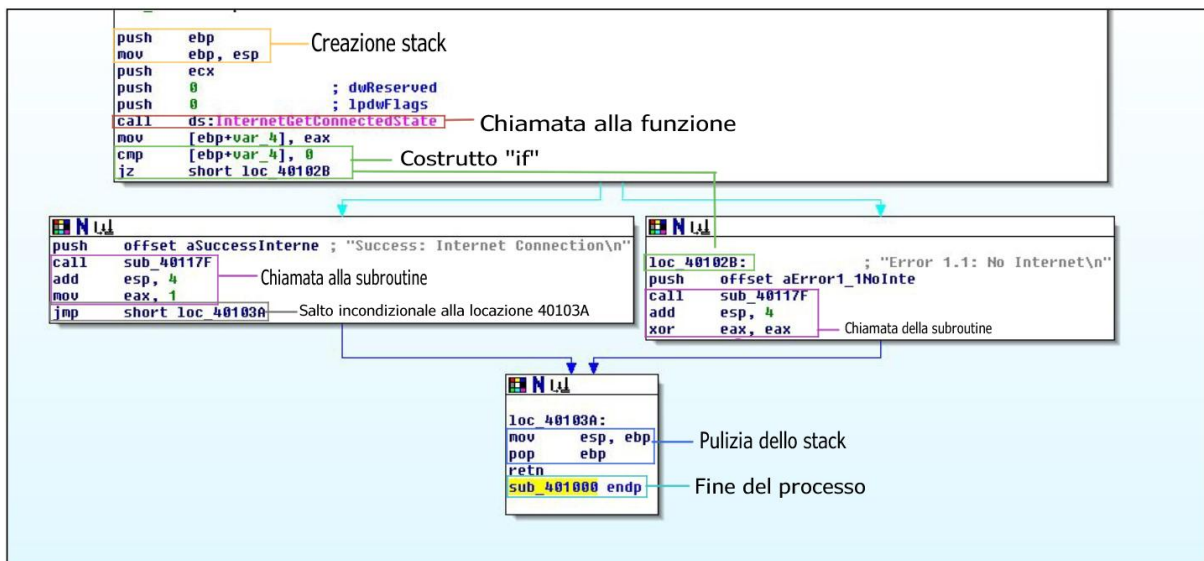
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000140	6C	00	29	00	00	00	00	00	28	6E	75	6C	6C	29	00	00	1.).....(null)...
00000150	5F	5F	47	4C	4F	42	41	4C	5F	48	45	41	50	5F	53	45	GLOBAL_HEAP_SE
00000160	4C	45	43	54	45	44	00	00	5F	5F	4D	53	56	43	52	54	LECTED...MSVCRT
00000170	5F	48	45	41	50	5F	53	45	4C	45	43	54	00	00	00	00	HEAP_SELECT....
00000180	72	75	6E	74	69	6D	65	20	65	72	72	6F	72	20	00	00	Runtime error...
00000190	0D	0A	00	00	54	4C	4F	53	53	20	65	72	72	6F	72	0DTLOSS.error..
000001A0	0A	00	00	00	53	49	4E	47	20	65	72	72	6F	72	0D	0ASING.error..
000001B0	00	00	00	00	44	4F	4D	41	49	4E	20	65	72	72	6F	72DOMAIN.error
000001C0	0D	0A	00	00	52	36	30	32	38	0D	0A	2D	20	75	6E	61R6028...una
000001D0	62	6C	65	20	74	6F	20	69	6E	69	74	69	61	6C	69	7A	ble.to.initializ
000001E0	65	20	68	65	61	70	0D	0A	00	00	00	00	52	36	30	32	e.heap.....R602
000001F0	37	0D	0A	2D	20	6E	6F	74	20	65	6E	6F	75	67	68	20	7...not.enough.
00000200	73	70	61	63	65	20	66	6F	72	20	6C	6F	77	69	6F	20	space.for.lowio.
00000210	69	6E	69	74	69	61	6C	69	7A	61	74	69	6F	6E	0D	0A	initialization..
00000220	00	00	00	00	52	36	30	32	36	0D	0A	2D	20	6E	6F	74R6026...not

.data è la sezione che contiene variabili globali e statiche accessibili da ogni parte del programma, quindi le informazioni non sono read-only come nella sezione **.rdata**.

Codice assembly



Abbiamo qui come riferimento la figura con il codice assembly in un comodo schema ad albero. Andiamo quindi a individuare i suoi costrutti.



Dentro questo codice assembly troviamo la creazione dello stack (push ebp; mov ebp, esp) dove il valore dello stack pointer viene copiato in quello del base pointer.

La chiamata alla funzione "InternetGetConnectedState" serve a identificare lo stato della connessione internet. Il programma esegue poi un costrutto if (jz short loc_40102B) per i due possibili risultati.

Se la connessione è presente, uscirà in output la stringa "Success: Internet Connection\n", verrà chiamata una subroutine (mini programma presente all'interno del programma principale accessibile in qualunque punto) e il programma farà un jump alla locazione 40103A.

Se la connessione non dovesse essere presente, quindi $cmp = 0$, la ZF sarà 1 e il programma farà un salto alla locazione 40102B, poiché il costrutto è jz, ovvero jump if zero.

Arrivato alla locazione 40103A, il programma eseguirà una pulizia dello stack e finirà il processo. (mov esp, ebp; pop ebp)

Si può ipotizzare che questa porzione di codice faccia parte di un malware più grande e complesso, e che questa parte si occupi di constatare la presenza o meno di connessione internet, dando in output errore in caso dell'assenza di connessione.