

In questo esercizio andremo a fare un'analisi statica basica di un malware su windows XP.

L'analisi statica basica consente di analizzare un eventuale malware senza la necessità di eseguirlo, dandoci comunque delle informazioni importanti a riguardo. E' sicuramente la tecnica di analisi più rapida ed intuitiva, anche se la meno efficace per quanto riguarda i malware più avanzati.

Iniziamo verificando le librerie importate dal malware, possiamo notare subito **KERNEL32.dll** e **WININET.dll**.

Kernel32.dll è una libreria che contiene le funzioni principali per interagire col sistema operativo o per gestire la memoria.

In questo caso possiamo notare due funzioni in particolare; GetProcAddress e LoadLibraryA, queste due ci fanno capire che la libreria in questione viene caricata dinamicamente, ovvero vengono chiamate soltanto quando il programma (o malware) in questione ha bisogno di utilizzarle. Caricando le librerie dinamicamente permette ad un eventuale malware di essere meno invasivo e meno rilevabile.

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

File: Malware_U3_W2_L5.exe

Dos Header

Nt Headers

File Header

Optional Header

Data Directories [x]

Section Headers [x]

Import Directory

Address Converter

Dependency Walker

Hex Editor

Identifier

Import Adder

Quick Disassembler

Rebuilder

Resource Editor

UPX Utility

Malware_U3_W2_L5.exe

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
000065B0	00006098	0000688E	00006890
Dword	Dword	Word	szAnsi
0000681E	0000681E	019F	HeapFree
0000682A	0000682A	022F	RtlUnwind
00006836	00006836	02DF	WriteFile
00006842	00006842	0199	HeapAlloc
0000684E	0000684E	00BF	GetCPInfo
0000685A	0000685A	00B9	GetACP
00006864	00006864	0131	GetOEMCP
00006870	00006870	02BB	VirtualAlloc
00006880	00006880	01A2	HeapReAlloc
0000688E	0000688E	013E	GetProcAddress
000068A0	000068A0	01C2	LoadLibraryA
000068B0	000068B0	011A	GetLastError
000068C0	000068C0	00AA	FlushFileBuffers
000068D4	000068D4	026A	SetFilePointer
00006950	00006950	001B	CloseHandle

start

CFF Explorer VIII - [...]

L'altra libreria in questione è **Wininet.dll**, una libreria che permette al malware di importare funzioni per l'utilizzo di protocolli di rete (ad esempio HTTP ed FTP).

Questo ci dà parecchie info su ciò che potrebbe fare il malware, molto probabilmente potrebbe connettersi ad internet, per scaricare altro materiale malevolo oppure per condividere informazioni sensibili dal dispositivo ad un server remoto o anche una sorta di backdoor.

Analizzeremo nel dettaglio il programma in assembly più avanti, durante l'esercizio, per ottenere qualche informazione in più sul malware in questione.

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

File: Malware_U3_W2_L5.exe

Dos Header

Nt Headers

File Header

Optional Header

Data Directories [x]

Section Headers [x]

Import Directory

Address Converter

Dependency Walker

Hex Editor

Identifier

Import Adder

Quick Disassembler

Rebuilder

Resource Editor

UPX Utility

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

start

CFF Explorer VIII - [...]

Il secondo passaggio che andremo a fare è verificare di che "sezioni" è composto il file. Le sezioni ci danno un'ulteriore idea su come funziona il malware, andiamo ad analizzarle.

.text è la sezione di file che contiene tutte le informazioni e le istruzioni che la CPU dovrà eseguire a malware avviato.

.rdata include tutte le informazioni sulle librerie importate ed esportate dal programma (cosa che possiamo appunto verificare con CFF Explorer, come negli screenshot precedenti).

.data invece include le variabili globali dell'eseguibile, sono globali perché devono essere disponibili a qualsiasi funzione all'interno dell'eseguibile.

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

File: Malware_U3_W2_L5.exe

Dos Header

Nt Headers

File Header

Optional Header

Data Directories [x]

Section Headers [x]

Import Directory

Address Converter

Dependency Walker

Hex Editor

Identifier

Import Addr

Quick Disassembler

Rebuilder

Resource Editor

UPX Utility

Malware_U3_W2_L5.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
000001E0	000001E8	000001EC	000001F0	000001F4	000001F8	000001FC	00000200	00000202	00000204
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:
Code Entry Point: 000011B0

Offset

0123456789A B C D E F

Ascii

00000000	55	8B	EC	51	6A	00	6A	00	FF	15	C0	60	40	00	89	45	U!Qj.j.y!A`@.IE
00000010	FC	83	7D	FC	00	74	14	68	48	70	40	00	E8	5E	01	00	ü}ü.tihHp@.è^l.
00000020	00	83	C4	04	B8	01	00	00	00	00	00	EB	0F	68	30	70	!A!...èih0p@.
00000030	E8	4A	01	00	00	83	C4	04	33	C0	8B	E5	5D	C3	CC	CC	èJ!...!A!3A!â!A!i
00000040	55	8B	EC	81	EC	10	02	00	00	6A	00	6A	00	6A	00	6A	U!i!i!...j.j.j.j
00000050	00	68	F4	70	40	00	FF	15	C4	60	40	00	89	45	F4	6A	.h0p@.y!A`@.IE0j
00000060	00	6A	00	6A	00	6A	00	68	C4	70	40	00	8B	45	F4	50	.j.j.j.hâp@.IE0P
00000070	FF	15	B4	60	40	00	89	45	F0	83	7D	F0	00	75	1E	68	y!`@.IEâ!}â.u!h
00000080	A8	70	40	00	E8	F6	00	00	00	83	C4	04	8B	4D	F4	51	p@.è0...!A!M0Q
00000090	FF	15	B8	60	40	00	32	C0	E9	8F	00	00	00	8D	55	F8	y!`@.2Aè!...!Uè
000000A0	52	68	00	02	00	00	8D	85	F0	FD	FF	FF	50	8B	4D	F0	Rh!...!âÿÿÿP!Mâ
000000B0	51	FF	15	BC	60	40	00	89	45	FC	83	7D	FC	00	75	25	Qÿ!â`@.IEü!}ü.uâ
000000C0	68	88	70	40	00	E8	B5	00	00	83	C4	04	8B	55	F4		h!p@.èp...!A!U0
000000D0	52	FF	15	B8	60	40	00	8B	45	F0	50	FF	15	B8	60	40	Rÿ!`@.IEâPÿ!`@
000000E0	00	32	C0	EB	47	0F	BE	8D	F0	FD	FF	FF	83	F9	3C	75	.2AèG!âÿÿÿ!ûcu
000000F0	2C	0F	BE	95	F1	FD	FF	FF	83	FA	21	75	20	0F	BE	85	!â!âÿÿÿ!û!u.â!
00000100	F2	FD	FF	FF	83	F8	2D	75	14	0F	BE	8D	F3	FD	FF	FF	ôÿÿÿ!è-u!â!ôÿÿÿ
00000110	83	F9	2D	75	08	8A	85	F4	FD	FF	FF	EB	0F	68	68	70	!û-u!â!ôÿÿÿ!hbp
00000120	40	00	E8	58	00	00	00	83	C4	04	32	C0	8B	E5	5D	C3	@.èX...!A!2A!â!A

start

CFF Explorer VIII - [...]

Il programma qui a fianco è scritto in assembly x86.

Il suo scopo è verificare un'eventuale connessione ad internet utilizzando la funzione "InternetGetConnectedState". Utilizza un costrutto IF (lo analizzeremo nella prossima slide), nel caso la funzione venga verificata correttamente, il programma userà la parte "Success: Internet Connection", in caso contrario, la parte "Error 1.1: No internet".

Possiamo correlare questo codice in assembly alla libreria "wininet.dll" vista in precedenza, confermando il fatto che il malware vada ad utilizzare una componente che si connette ad internet.

Analizziamo nel dettaglio i blocchi di 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
```

"Success: Internet Connection"

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

"Error 1.1: No internet"

```
Loc_40102B:
push offset aError1_NoInte
call sub_40117F
add esp, 4
xor eax, eax
```

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

Il programma crea lo stack	push ebp mov ebp, esp push ecx
Chiama la funzione "InternetGetConnectedState"	push 0 ; dwReserved push 0 ; lpdwFlags call ds:InternetGetConnectedState
Copia il valore in [ebp+var_4]	mov [ebp+var_4], eax
Confronta il risultato con 0.	cmp [ebp+var_4], 0
Se uguale a 0, fa un jump all'indirizzo loc_40102B (costrutto IF, connessione fallita)	jz short loc_40102B
Connessione riuscita	push offset aSuccessInterne ; "Success: Internet Connection\n" call sub_40117F add esp, 4 mov eax, 1 jmp short loc_40103A
Connessione fallita	loc_40102B: push offset aError1_NoInte ; "Error 1.1: No Internet\n" call sub_40117F add esp, 4 xor eax, eax
Resetta lo stack e ritorna (retn) alla funzione	loc_40103A: mov esp, ebp pop ebp retn