PROVA PRATICA

MALWARE ANALYSIS

03-31-2023

Analizzando il file eseguibile del malware con CFF Explorer possiamo ricavare varie informazioni su di esso, in questo caso eseguendo un'analisi statica mirata a scoprire le librerie importate e le sezioni che lo compongono non ci esponiamo ad alcuna eventuale azione dannosa dell'applicazione, ma analizzando la sua struttura possiamo farci un idea del suo comportamento una volta eseguito ed intercettare parti di codice volte a provocare danni alla nostra macchina.

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
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
000067A4	000067A4	0150	GetStartupInfoA
000067B6	000067B6	0126	GetModuleHandleA
000067CA	000067CA	0109	GetEnvironmentVariableA
000067E4	000067E4	0175	GetVersionExA
000067F4	000067F4	019D	HeapDestroy
00006802	00006802	019B	HeapCreate
00006810	00006810	02BF	VirtualFree
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

Come mostrato in figura l'eseguibile importa ed utilizza principalmente due librerie e varie funzioni in esse contenute:

- Kernel32.dll che contiene funzioni atte ad interagire con il sistema operativo ad esempio la manipolazione di file, processi e zone di memoria.

- Wininet.dll che contiene funzioni a supporto dei protocolli di rete e della connessione.

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

Per quanto riguarda le sezioni troviamo le classiche contenute nella maggior parte degli eseguibili rispettivamente:

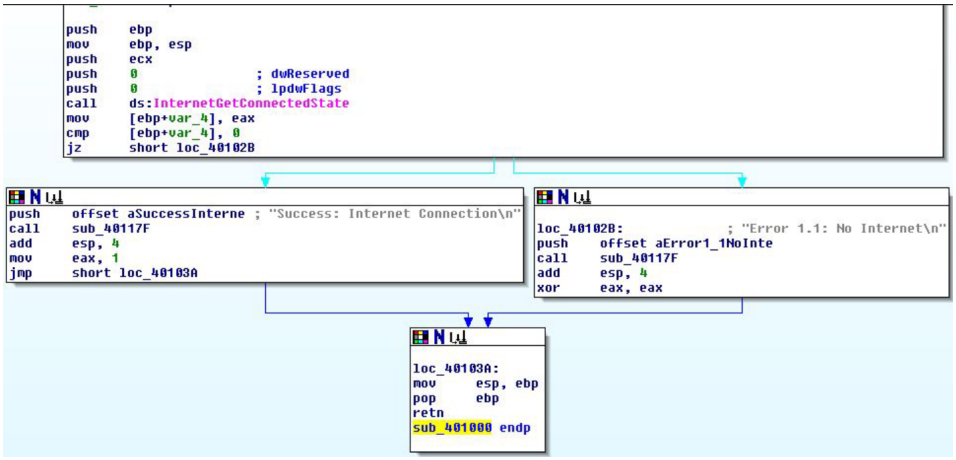
.text -> che comunica direttamente con la CPU e contiene le righe di codice che dovrà eseguire durante l'esecuzione del programma

.rdata -> contiene informazioni sulle librerie (immagini sopra)

.data -> contiene le variabili globali del programma

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	00005000	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	00001000	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

ANALISI CODICE



Il codice preso in esame ha la funzionalità di controllare se la macchina è connessa ad una rete internet dandoci un feedback tramite riga di comando, nello specifico:

push ebp
mov ebp, esp
push ecx
push 0 ; dwReserved
push 0 ; lpdwFlags

Nelle prime righe viene creato ed inizializzato lo stack

call ds:InternetGetConnectedState
mov [ebp+var_4], eax
cmp [ebprvar_u], 0
jz short loc_40102B

In questo blocco viene usata la funzione per capire se la macchina è connessa, se la risposta è negativa fa un salto condizionale se no continua (IF statement)

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

Questa parte di codice non verrà eseguita se la condizione di connessione non si verifica, se eseguita riporta una stringa di conferma e salta all'ultimo blocco di comando

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

In questo blocco si gestisce l'eccezione di connessione mancata mostrando un messaggio d'errore

loc_40103A:
mov esp, ebp
pop ebp
retn
sub_401000 endp

In fine lo stack viene eliminato ed il controllo torna alla macchina, "in ogni caso questo blocco di codice viene eseguito"

EXTRA

A seguito di un'analisi dinamica e comparando i risultati con quelli dell'analisi statica possiamo ipotizzare che si tratta di un malware che oltre a modificare le chiavi di registro effettua richieste web probabilmente per scaricare librerie mancanti oppure come Downloader di altri malware.