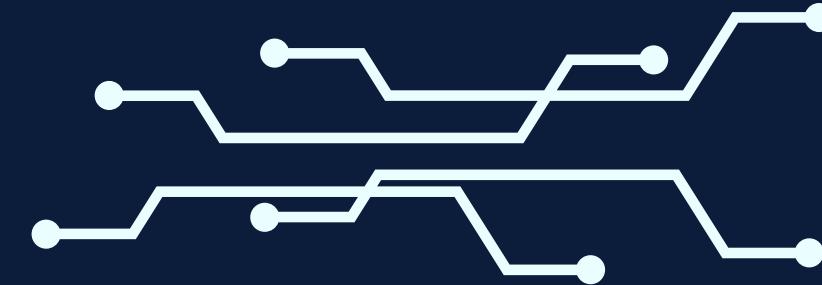
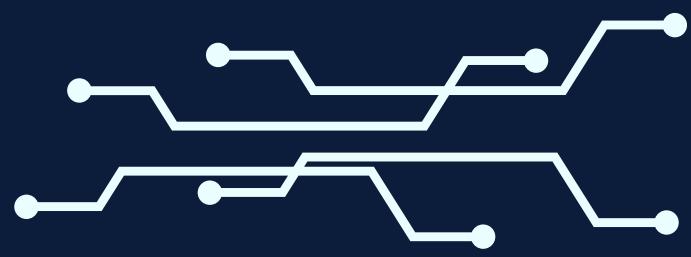
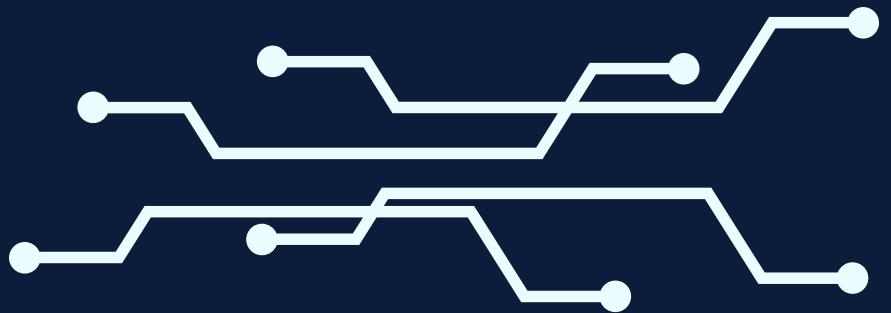


310-15





Come vedremo dalle seguenti immagini le librerie importate sono: Kernel32.dll, che include le funzioni core del sistema operativo e Wininet.dll, che include le funzione per implementare i servizi di rete come ftp, ntp, http

Windows 7 - malware analysis (Instantanea 1) [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

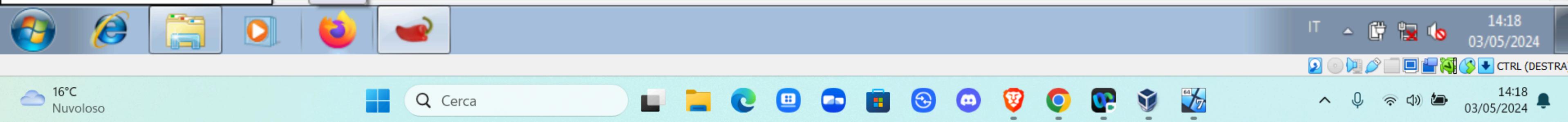
File Settings ?

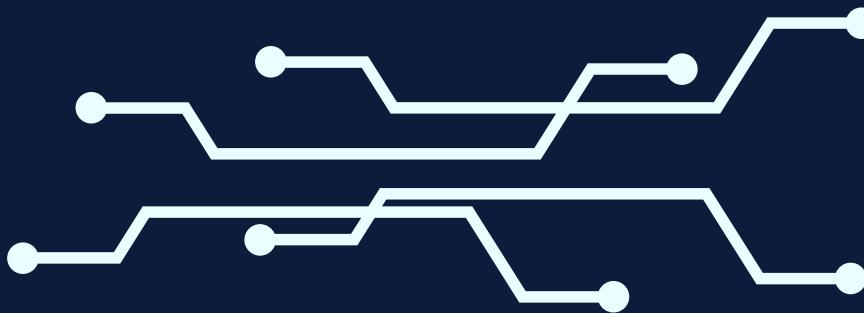
Module Name Imports OFTs TimeStamp ForwarderChain Name RVA FTs (IAT)

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

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





Le sezioni che compongono il malware sono:

.text che contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato

.rdata: che include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile.

.data: che contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.



Malware_U3_W2_L3

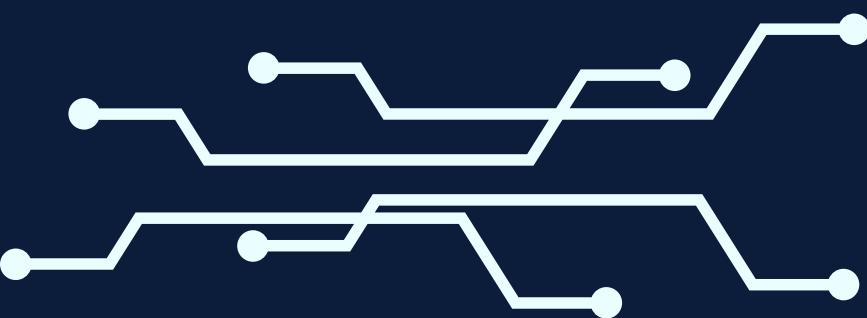
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ .L...J...yy..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	,.....@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	E8	00	00	00è..
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	þ?þ.'!í!, LÍ!TH

Fi



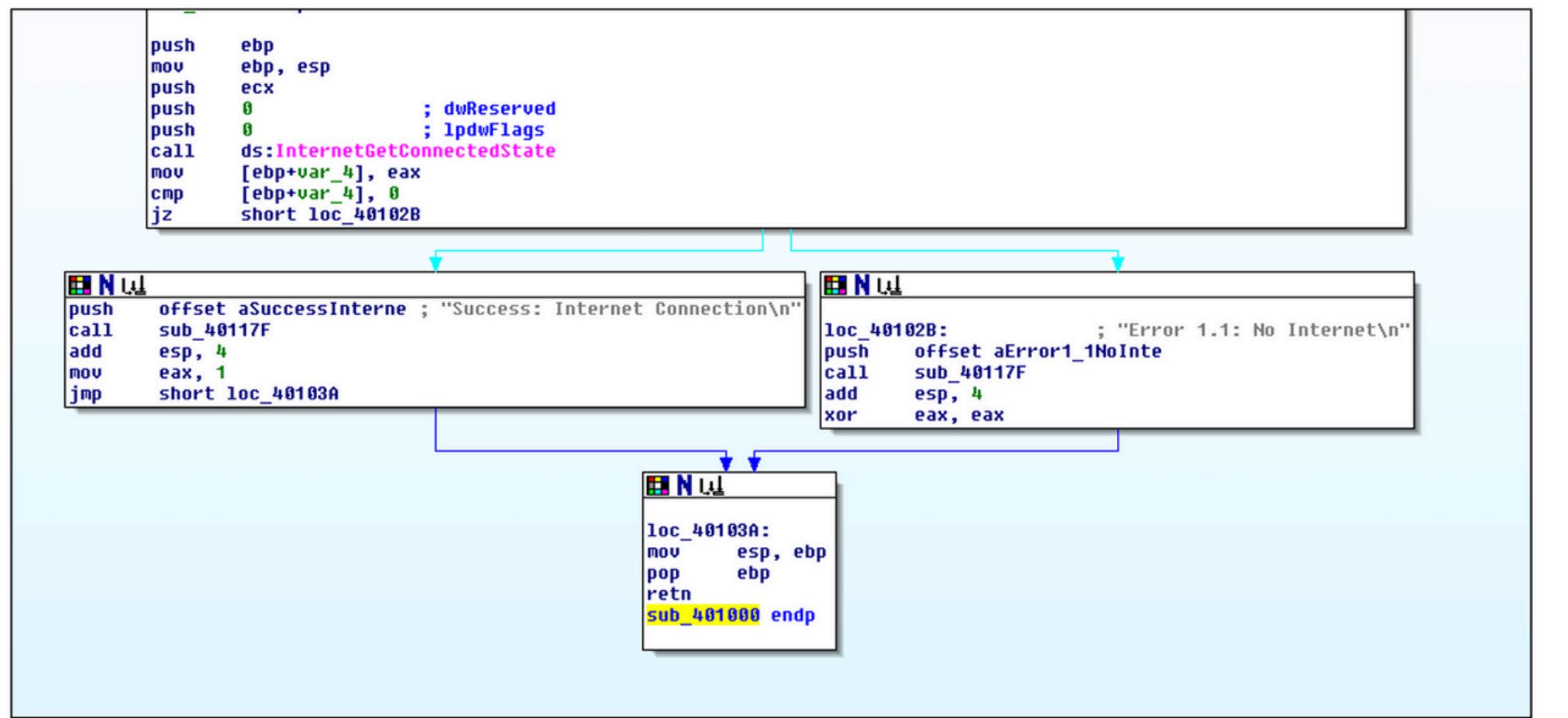


Creazione dello stack: All'inizio della funzione, viene creata una nuova cornice dello stack utilizzando l'istruzione push ebp.

Costrutto condizionale “IF” identificato da cnp e jz

Rimozione stack: Alla fine della funzione, lo stack viene ripulito utilizzando l'istruzione pop ebp. Questa istruzione ripristina il valore del registro EBP dal valore salvato nello stack

Figura 1



3

Creazione dello stack: All'inizio della funzione, viene creata una nuova cornice dello stack utilizzando l'istruzione push ebp.
Costrutto condizionale “IF” identificato da cmp e jz

Rimozione stack: Alla fine della funzione, lo stack viene ripulito utilizzando l'istruzione pop ebp. Questa istruzione ripristina il valore del registro EBP dal valore salvato nello stack