



EPICODE

S10 L5

Indice

1.Traccia

2.Preparazione della macchina virtuale

3.Analisi Librerie

4.Analisi Sezioni

5.Traccia N.2

6.Identificazione Costrutti

7.Ipotesi comportamento della funzione implementata

Dario Santigliano

1 

Traccia: Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5» sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

Quali librerie vengono importate dal file eseguibile?

Quali sono le sezioni di cui si compone il file eseguibile del malware?

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

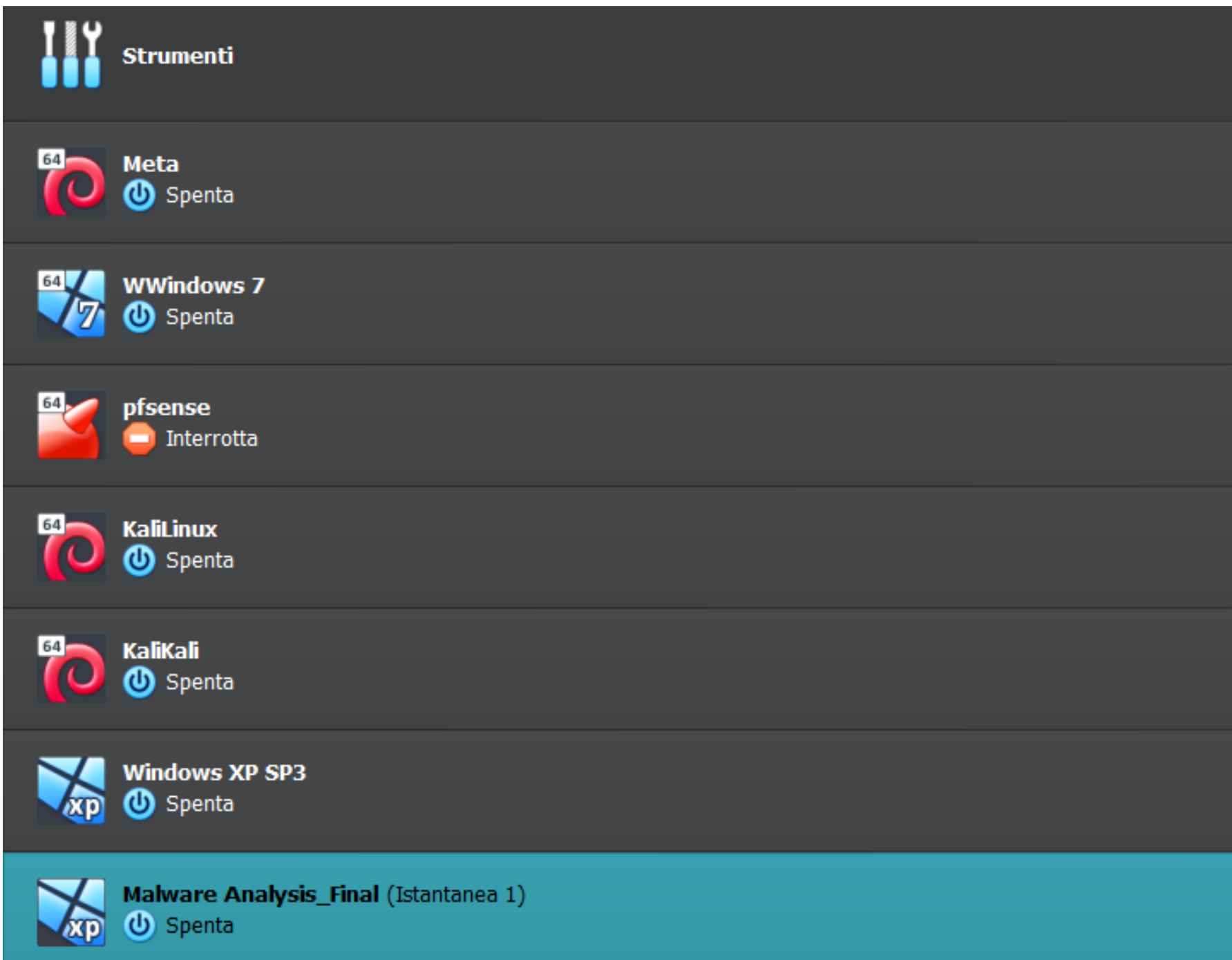
Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)

Ipotizzare il comportamento della funzionalità implementata



In questa prima fase andremo a vedere tutto l'occorrente per lo svolgimento dell'esercizio. Prima di tutto procediamo ad avviare la macchina windows XP.

Entrati all'interno della macchina procederemo ad utilizzare il tool CFF Explorer.

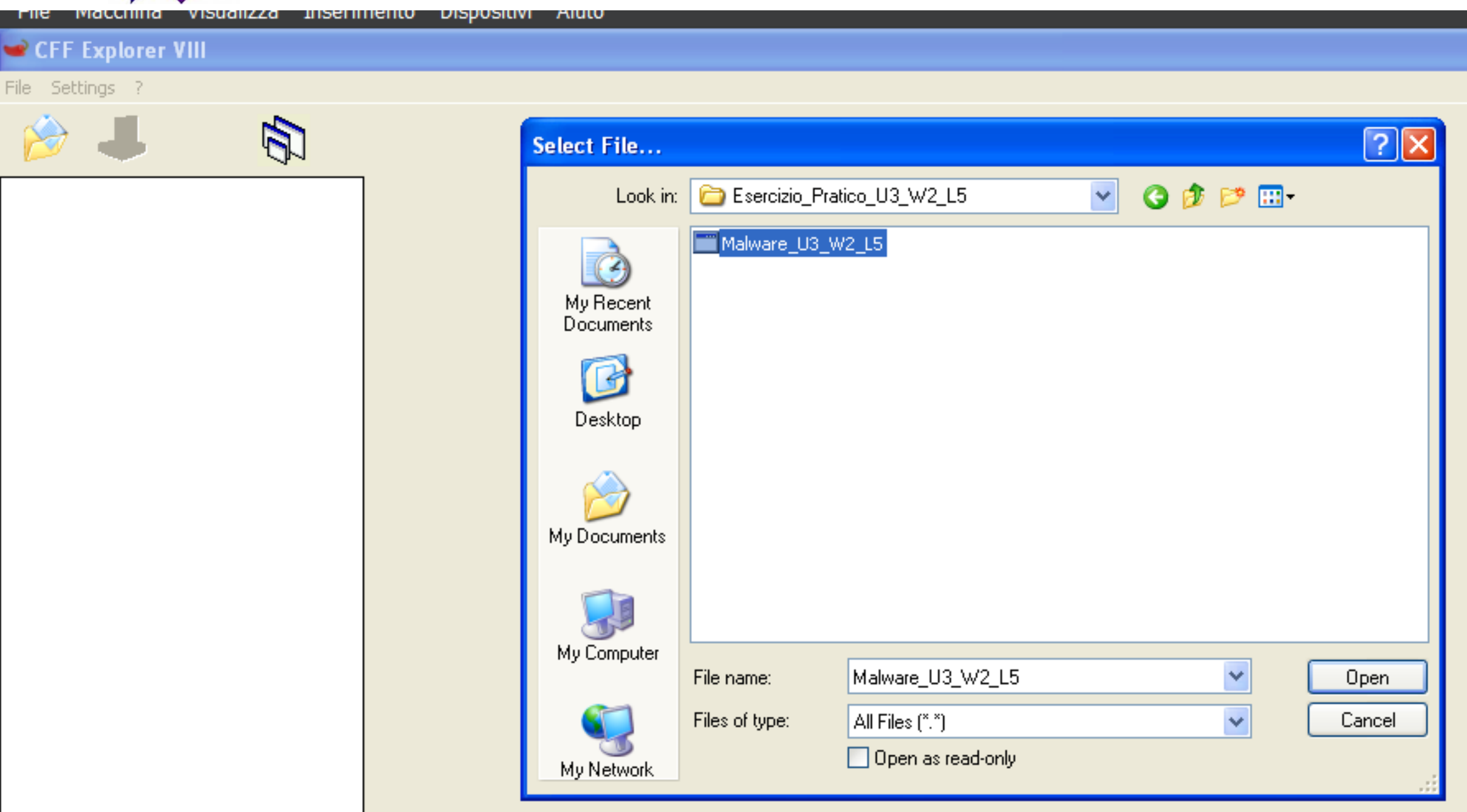


2.1

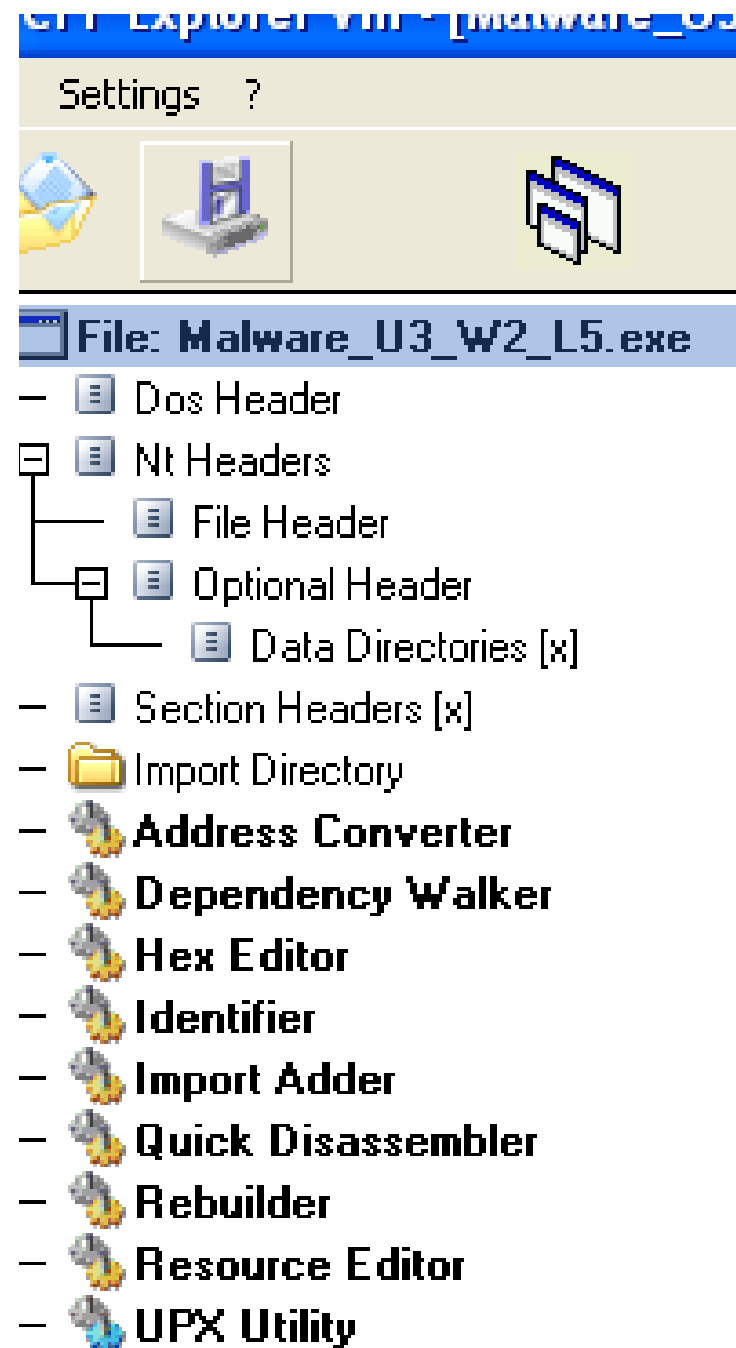
CFF EXPLORER

Cos'è CFF Explorer?

CFF Explorer è un'applicazione software utilizzata per esaminare e modificare file eseguibili di Windows, come file .exe e .dll. È uno strumento ampiamente utilizzato dagli sviluppatori e dagli appassionati di reverse engineering per analizzare la struttura interna dei file binari e per eseguire varie operazioni come l'estrazione di risorse, l'analisi delle sezioni del file, la modifica degli header e molto altro ancora. Inoltre, CFF Explorer offre anche funzionalità avanzate come la firma digitale e il supporto per l'editing del file di configurazione del processo (PE).



**Quando apriamo il tool
troveremo questa
schermata dove andremo
poi ad inserire il file
malware da analizzare. In
questo caso
“Malware_U3_W2_L5”**



Dopo aver importato il file ci verranno mostrate varie sezioni per l'analisi del malware. Noi per l'esercizio di oggi andremo a vedere "Section Headers" e "Import Directory"

In questo caso sono state importate due librerie, andiamo a vederle nello specifico.

Kernel32.dll

Nei sistemi operativi Microsoft™ Windows™, kernel32.dll è il modulo centrale che contiene i processi principali o il cuore del sistema operativo. All'avvio, il kernel32.dll viene caricato in memoria, regolando le operazioni mentre l'utente esegue varie attività e programmi. Il file kernel32.dll è così chiamato perché, come un kernel organico, contiene i processi fondamentali del sistema operativo. Il numero 32 indica un sistema operativo a 32 bit e l'estensione del file .dll sta per libreria di collegamento dinamico.

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

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

Your computer



Wininet.dll

Wininet.dll è un Windows file DLL (Dynamic Link Library) del sistema operativo. Sta per “Windows Servizi Internet” e fornisce funzioni e risorse per l'accesso e l'interazione con Internet in Windows applicazioni.

Wininet.dll fa parte dell'API WinINet (Application Programming Interface) e gestisce varie operazioni relative a Internet. Fornisce funzioni per stabilire connessioni Internet, inviare e ricevere richieste HTTP, gestire i cookie, gestire la memorizzazione nella cache, le operazioni FTP (File Transfer Protocol) e altri protocolli Internet.

explorer VIII - [Malware_U3_W2_L5.exe]

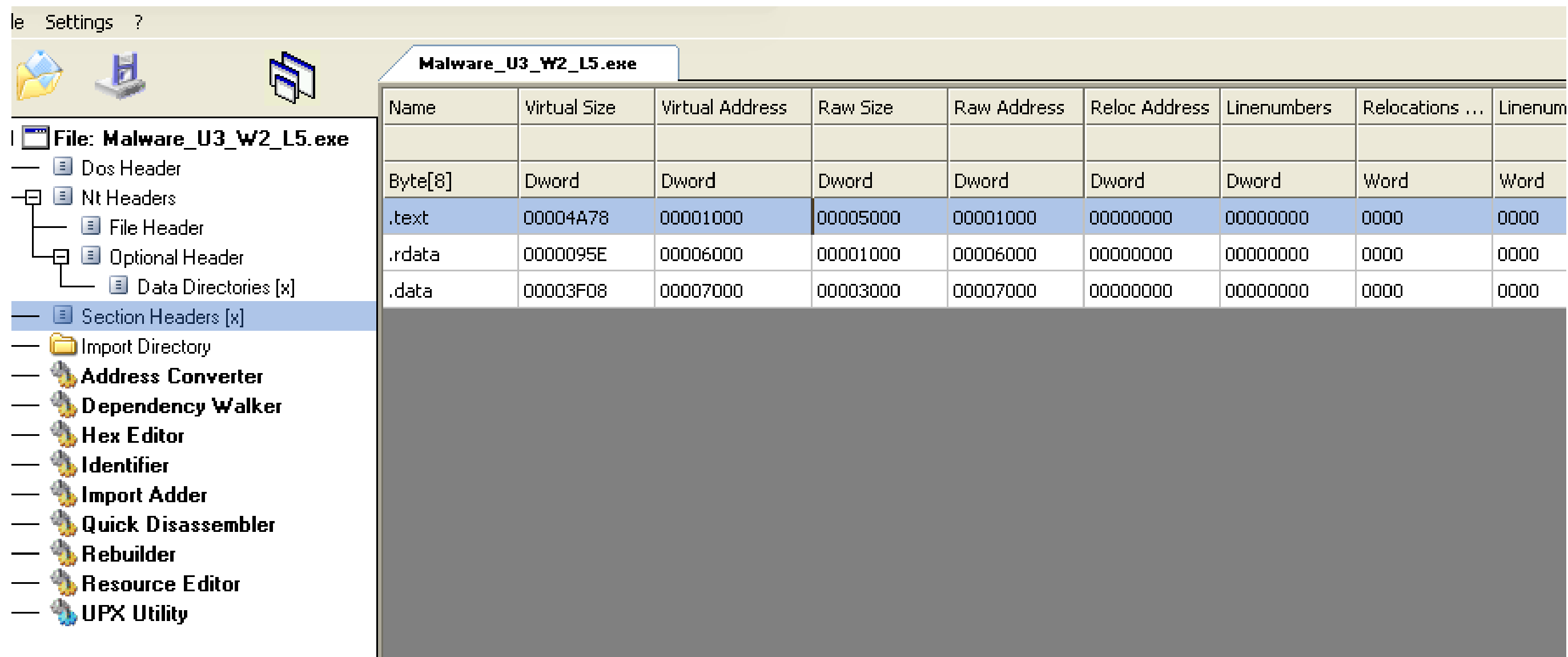
Malware_U3_W2_L5.exe

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

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
JPX Utility

Nella sezione "Section Headers" possiamo vedere come il file sia composto da ".text", ".rdata" e ".data". Andiamo a vedere brevemente il loro scopo nella prossima slide.



The screenshot shows the Immunity Debugger interface. On the left, the 'File: Malware_U3_W2_L5.exe' is loaded, and the 'Section Headers [x]' tab is selected in the left-hand pane. The main pane displays a table of section headers for the file.

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

4

.Text: la sezione ".text" è una parte critica di un file eseguibile, contenente il codice eseguibile del programma. È fondamentale per il funzionamento del software e deve essere protetta per garantire l'integrità e la sicurezza del sistema.

Nei malware, la sezione ".text" può contenere il codice malevolo effettivo del malware, come istruzioni per rubare informazioni personali, compromettere il sistema o eseguire altre azioni dannose.

.Rdata: la sezione ".rdata" è utilizzata per contenere dati costanti di sola lettura all'interno di un file eseguibile. Questi dati sono accessibili durante l'esecuzione del programma ma non possono essere modificati direttamente, il che li rende utili per memorizzare informazioni costanti utilizzate dal programma. nei malware può essere utilizzata per memorizzare dati critici e costanti utilizzati durante l'esecuzione del malware, compresi dati criptati, indirizzi di risorse esterne, parti del payload del malware e strutture di controllo.

.Data: la sezione ".data" è utilizzata per memorizzare dati inizializzati che possono essere modificati durante l'esecuzione del programma. Questa sezione è essenziale per mantenere lo stato e le informazioni dinamiche del programma durante l'esecuzione.

5



Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)

Ipotizzare il comportamento della funzionalità implementata



1) Il primo costrutto riconoscibile è la creazione dello stack, che sarà l'area di memoria RAM all'interno della quale verrà eseguita la funzione successiva. E' identificato dalle seguenti istruzioni:

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

Alla fine del codice troviamo la sua chiusura, identificata dalle istruzioni:

```
mov esp, ebp  
pop ebp
```

2) Il secondo costrutto riconoscibile è la chiamata della funzione InternetGetConnectedState. Prima della chiamata vengono aggiunte in cima allo stack i tre parametri di cui la funzione ha bisogno. Di seguito le istruzioni:

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



3) Il terzo costrutto riconoscibile è il salto condizionale jz in riferimento all'istruzione cmp, che è l'equivalente di un costrutto if/else in linguaggio C:

```
cmp [ebp+var_4], 0  
jz short loc_40102B
```

Viene confrontato il valore 0 con la variabile var_4 tramite l'istruzione cmp. Se i due operandi risultano uguali, l'istruzione setterà lo ZF a 1. Successivamente il salto condizionale jz controllerà se lo ZF risulti settato (quindi uguale a 1) e in caso affermativo salterà all'indirizzo di memoria 40102B, ignorando le istruzioni comprese tra tale indirizzo e l'indirizzo corrente.

A livello logico possiamo dire che tale costrutto serve a capire se l'output della funzione InternetGetConnectedState sia 0. In tal caso verranno eseguite le seguenti istruzioni:

loc_40102B:

```
push offset aError1_1NoInte ; "Error1.1: No Internet\n"  
call sub_40117F  
add esp, 4  
xor eax, eax
```

Altrimenti, quindi in caso di connessione presente, verranno eseguite le seguenti istruzioni:

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



Ipotesi comportamento della funzione implementata

Dall'analisi statica di base del codice si può ipotizzare che il malware in questione cerchi di connettersi ad internet dall'import della libreria wininet.dll.

Dall'analisi dell'assembly si conferma questa ipotesi, dato che nel segmento di codice evidenziato è

presente una funzione che va a controllare se la macchina target è connessa o meno ad internet.

Non

è possibile dal solo tratto di codice evidenziato capire quali altre azioni possa eseguire il malware una volta verificato che la connessione sia presente, ma potenzialmente potrebbe servirsene per scaricare altri malware sulla macchina, creare una backdoor per dare il controllo ad un eventuale attaccante o semplicemente inviargli le informazioni raccolte.