



INDICE:

Traccia	1.
Preparazione VM e Tool	2.
Analisi Librerie	3.
Analisi Sezioni	4.
Traccia 2 Figura 3	5.
Identificazione costrutti noti	6.
Ipotesi comportamento	7.

1. Traccia

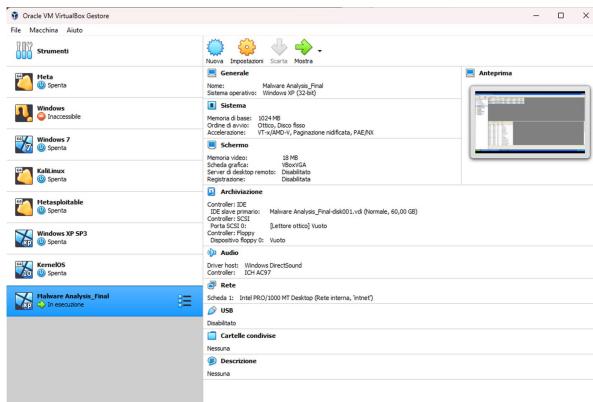
Con riferimento al file **Malware_U3_W2_L5** presente all'interno della cartella «**Esercizio_Pratico_U3_W2_L5**» sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

- 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:

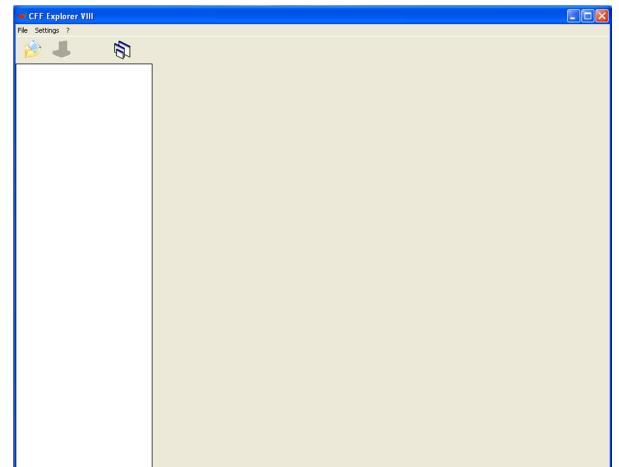
- Identificare i costrutti noti (creazione dello stack, eventuali cicli, costrutti)
- Ipotizzare il comportamento della funzionalità implementata

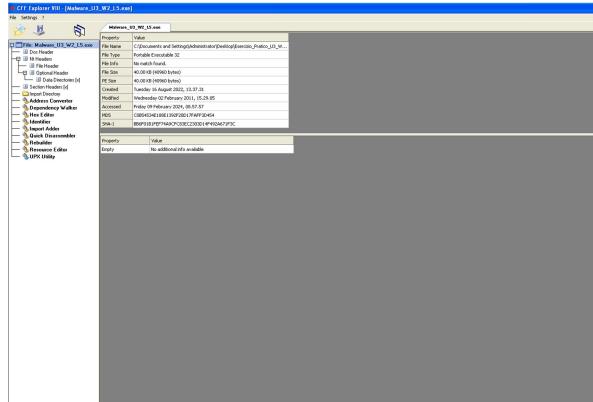
2. Preparazione VM e Tool



Per prima cosa avviare la macchina virtuale di Malware Analysis basata su Windows XP, data in precedenza dalle slide. Una volta avviata, accedere con la password ed avviare il Tool “CFF Explorer VIII”.

Una volta aperto, ci troveremo davanti una schermata vuota dove andremo ad importare il file malware per analizzarlo. Clicchiamo sul primo pulsante in alto a sinistra (icona cartella gialla) e scegliamo il file “Malware_U3_W2_L5” all'interno della cartella “Esercizio_Pratico_U3_W2_L5”.





Una volta importato possiamo vedere che ci sono varie sezioni per l'analisi del malware. Noi ci concentreremo su "import Directory" per l'analisi della librerie importate e su "Section Headers" per vedere le sezioni di cui si compone il file. Andiamo a vederle più nel dettaglio con un analisi più approfondita...

3. Analisi Librerie

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	00006084

Nella sezione "Import Directory" possiamo vedere che sono state importate le librerie "KERNEL32.dll" e "WININET.dll"

VEDIAMOLE NELLO SPECIFICO...



1. kernel32.dll:

- *Funzione principale:* Kernel32.dll è una libreria di sistema di Windows che contiene numerose funzioni fondamentali per la gestione delle risorse di sistema e per l'interazione con il kernel del sistema operativo.
- *Funzioni comuni:* Alcune delle funzioni più comuni fornite da Kernel32.dll includono la gestione della memoria, la creazione e la gestione di processi e thread, l'accesso ai file, la gestione degli errori e la comunicazione interprocessuale.
- *Utilizzo nel malware:* Il malware potrebbe utilizzare le funzioni di Kernel32.dll per manipolare i processi e i thread, accedere ai file di sistema, gestire la memoria o comunicare con altri processi o componenti del sistema.

2. wininet.dll:

- *Funzione principale:* Wininet.dll è una libreria di sistema di Windows che fornisce funzionalità per l'accesso e la gestione delle risorse di rete tramite protocolli Internet come HTTP, FTP e altri.
- *Funzioni comuni:* Le funzioni fornite da Wininet.dll includono la creazione e la gestione delle connessioni di rete, il download e l'upload di file, la gestione dei cookie e delle cache, e l'accesso alle risorse web.
- *Utilizzo nel malware:* Il malware potrebbe utilizzare Wininet.dll per comunicare con server remoti, scaricare o caricare file, inviare dati raccolti dalla macchina infetta o ricevere istruzioni dal server di comando e controllo.

In breve, kernel32.dll fornisce funzioni fondamentali per la gestione delle risorse di sistema, mentre wininet.dll offre funzionalità per l'accesso e la gestione delle risorse di rete. Entrambe queste librerie sono spesso utilizzate da malware per eseguire operazioni di sistema e comunicare con risorse esterne, come server di comando e controllo.

4. Analisi Sezioni

CFF Explorer VIII - [Malware_U3_W2_L5.exe]

File Settings ?

Malware_U3_W2_L5.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linumber...	Characteristics
Byte[8]	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

File: Malware_U3_W2_L5.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [x]
- Section Headers [x] (selected)
- Import Directory
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler
- Rebuilder
- Resource Editor

Nella sezione "Section Headers" possiamo vedere che il file eseguibile del malware si compone di ".text" ".rdata" ".data"

VEDIAMOLE NELLO SPECIFICO...

.text

- Questa sezione contiene il codice eseguibile del programma, cioè le istruzioni macchina che vengono eseguite dal processore. È dove si trova il codice vero e proprio del programma, che definisce le operazioni da eseguire.
- Il codice in questa sezione può includere istruzioni per eseguire calcoli, condizioni di controllo, chiamate a funzioni di sistema e altro ancora.
- Nei malware, questa sezione conterrà il codice che svolge le funzionalità dannose o dannose del programma, come la propagazione, il danneggiamento dei file o la raccolta di informazioni.

.rdata

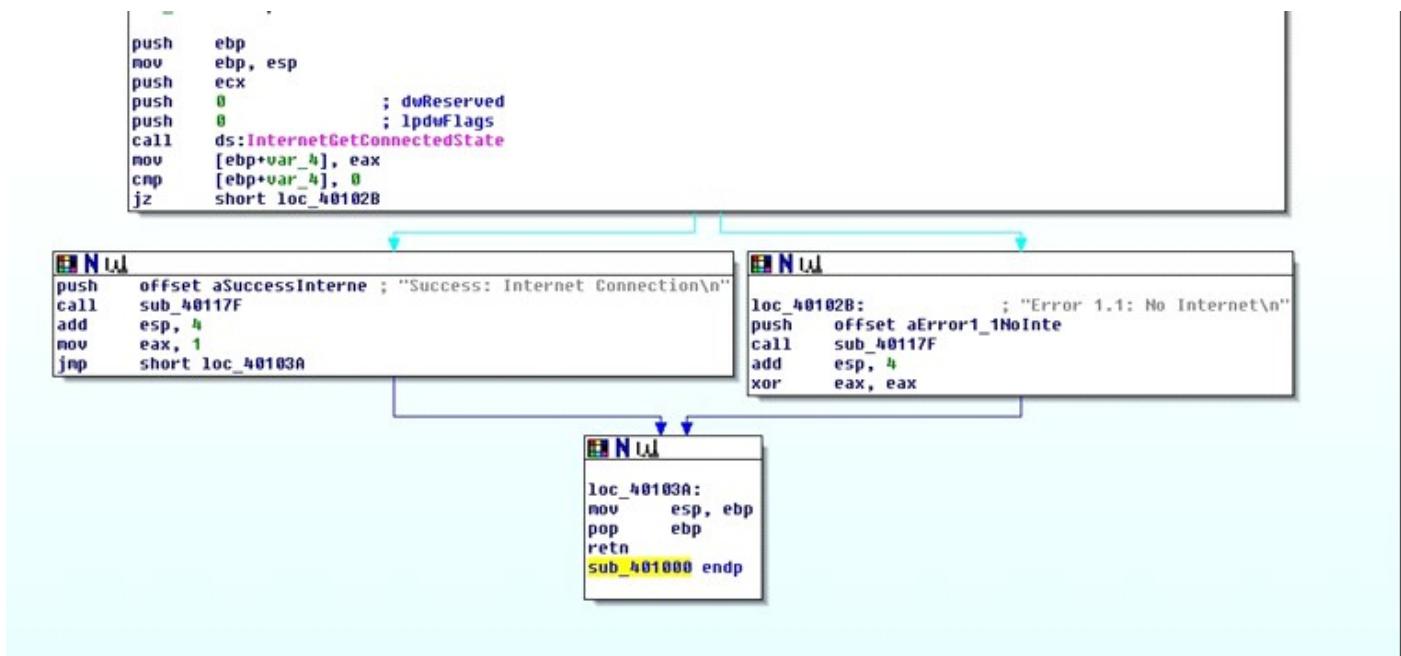
- Questa sezione contiene dati di sola lettura (read-only data) necessari al programma durante l'esecuzione.
- I dati presenti in questa sezione non possono essere modificati durante l'esecuzione del programma.
- Esempi comuni di dati presenti in questa sezione sono stringhe di testo, tabelle di costanti e altre informazioni che il programma necessita durante l'esecuzione.
- Nei malware, questa sezione potrebbe contenere stringhe di testo codificate, costanti o altri dati utilizzati dal codice maligno.

.data

- Questa sezione contiene dati modificabili durante l'esecuzione del programma, cioè dati che possono essere letti e scritti dal programma stesso.
- I dati presenti in questa sezione possono essere utilizzati per memorizzare informazioni dinamiche durante l'esecuzione del programma, come variabili, strutture dati, buffer e altro ancora.
- Nei malware, questa sezione può essere utilizzata per memorizzare dati temporanei, configurazioni, stato del programma e altre informazioni dinamiche necessarie per il funzionamento del malware.

In sintesi, le sezioni ".text", ".rdata" e ".data" suddividono il file eseguibile in diverse parti in base al tipo di contenuto. La sezione ".text" contiene il codice eseguibile, ".rdata" contiene dati di sola lettura e ".data" contiene dati modificabili durante l'esecuzione del programma.

5. Traccia 2 Figura 3



6. Identificazione costrutti noti

Istruzioni Assembly:

- ❖ **push**: Utilizzato per spingere dati sullo stack.
- ❖ **mov**: Utilizzato per muovere (copiare) dati da un'origine a una destinazione.
- ❖ **call**: Utilizzato per chiamare una procedura o una funzione.
- ❖ **cmp**: Utilizzato per confrontare due valori.
- ❖ **jz**: Jump se zero. Se l'ultimo confronto ha dato un risultato uguale a zero, salta all'indirizzo specificato.
- ❖ **jmp**: Jump incondizionato. Salta incondizionatamente all'indirizzo specificato.
- ❖ **xor**: Operazione XOR bit a bit.
- ❖ **pop**: Utilizzato per estrarre dati dallo stack.
- ❖ **retn**: Utilizzato per ritornare da una procedura o una funzione.

Etichette:

Le etichette come "loc_401028" e "loc_401030" sono punti di riferimento nel codice sorgente. Possono essere utilizzate come destinazioni per i salti condizionati o incondizionati.

Stringhe:

- ❖ Le stringhe come "aSuccessInterne" e "aError1_1Nolnte" sono semplici sequenze di caratteri utilizzate per la visualizzazione di messaggi.
- ❖ "aSuccessInterne" potrebbe essere utilizzato per indicare un messaggio di successo, ad esempio, quando il malware è riuscito a completare una determinata azione con successo.
- ❖ "aError1_1Nolnte" potrebbe essere utilizzato per indicare un messaggio di errore, ad esempio, quando il malware non è in grado di stabilire una connessione Internet.



7. Ipotesi comportamento:

1. Il primo blocco in alto contiene istruzioni per la creazione dello stack e una chiamata a una funzione per verificare lo stato della connessione Internet.
2. C'è un salto condizionale che porta a due blocchi di codice diversi a seconda del risultato della verifica della connessione Internet.
3. Il secondo blocco sulla sinistra visualizza un messaggio "Success: Internet Connection" se la connessione Internet è attiva.
4. Il terzo blocco sulla destra visualizza un messaggio "Error 1.1: No Internet" se non c'è connessione Internet.
5. Ogni blocco ha delle frecce colorate che indicano il flusso del codice.

Questo frammento di codice assembly esegue una verifica sulla connessione Internet e stampa un messaggio appropriato a seconda del risultato. È interessante notare come vengano utilizzati i costrutti di controllo per gestire i diversi scenari.