

FUNDAMENTALS OF MALWARE ANALYSIS AND REVERSE ENGINEERING



S10L5

TABLE OF CONTENT

• Cover	pag. 1
• Table Of Content	pag. 2
• Traccia	pag. 3
• Introduction: Analisi statica e Analisi Dinamica	pag. 4
• Project - Malware Analysis	pag. 5-8
• Project - Assembly	pag. 9
• Bonus: tabella con significato righe di codice	pag. 10

TRACCIA

S10L5

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 sotto, risponde ai seguenti quesiti:

- Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti)
- Ipotizzare il comportamento della funzionalità implementata
- BONUS fare tabella con significato delle singole righe di codice assembly

```
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
```

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

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

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

Introduction

La malware analysis è un campo cruciale per comprendere e difendersi dalle minacce informatiche rappresentate dal malware. Esistono due principali approcci per analizzarlo:

- **Analisi statica:** gli analisti esaminano il codice del malware senza eseguirlo direttamente. Questo approccio può essere di base, dove si identificano firme e stringhe di testo nel codice per determinare la natura del malware. Strumenti come disassemblatori come IDA Pro e Ghidra sono essenziali in questo contesto, consentendo di esplorare il codice assembly e comprendere le funzioni del malware senza il rischio di infezioni. Tuttavia, l'analisi statica può essere complicata da tecniche avanzate di offuscamento e crittografia utilizzate dai malware per nascondere il loro comportamento.
- **Analisi dinamica:** coinvolge l'esecuzione controllata del malware in un ambiente sicuro come una sandbox. Questo metodo consente di osservare il comportamento reale del malware in azione. Nell'approccio di base, gli analisti monitorano i comportamenti fondamentali come le connessioni di rete e le modifiche ai file. Strumenti come Process Monitor, Wireshark e RegShot sono utilizzati per registrare e analizzare queste attività. Tuttavia, l'analisi dinamica richiede tempo e risorse per configurare e monitorare l'ambiente di test, e può essere elusa da tecniche sofisticate di evasione utilizzate dai malware per rilevare ambienti virtualizzati o di sandbox.
- **Analisi statica avanzata:** l'analisi statica avanzata va oltre, utilizzando tecniche come il disassembling per tradurre il codice binario in linguaggio assembly e il decompiling per ottenere una visione più chiara del codice sorgente ad alto livello. Questo permette agli analisti di esaminare dettagliatamente le funzionalità del malware e le tecniche utilizzate, comprendendo come il malware interagisce con il sistema operativo e manipola le risorse.
- **Analisi dinamica avanzata:** analogamente, l'analisi dinamica avanzata si concentra sul monitoraggio dettagliato del comportamento del malware durante l'esecuzione. Gli analisti utilizzano debugger per esaminare il codice passo dopo passo, monitorando registri e memoria per comprendere le chiamate alle API e le interazioni con il sistema operativo. Questo livello di analisi rivela tecniche sofisticate di evasione e persistenza utilizzate dai malware per sfuggire alla rilevazione e mantenere il controllo sul sistema infetto.

L'analisi statica e dinamica sono complementari ed offrono una panoramica completa delle minacce malware e fornendo le basi per sviluppare contromisure efficaci per proteggere i sistemi informatici.

Project - Malware Analysis : Analisi Statica

L'analisi statica del malware utilizzando

CFF Explorer ci permette di dedurre diverse informazioni importanti riguardo al comportamento del programma maligno.

Come visto a lezione, andremo su Import Directory e potremo vedere quali librerie sono state importate.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA
KERNEL32.dll	44	00006518	00000000	00000000	000065EC
WININET.dll	5	000065CC	00000000	00000000	00006664

Il malware importa funzioni da due DLL principali: KERNEL32.dll e WININET.dll.

KERNEL32.dll è una libreria di sistema critica che fornisce accesso a molte funzioni fondamentali del sistema operativo Windows, come la gestione dei processi e della memoria.

WININET.dll è utilizzata per funzioni relative a Internet, suggerendo che il malware potrebbe avere funzionalità di comunicazione tramite rete.

Andremo ora ad analizzare le funzioni importate dalle due librerie e cercheremo di ipotizzare quale scopo abbia questo malware.

Le funzioni WriteFile, FlushFileBuffers, SetFilePointer, e CloseHandle potrebbero essere utilizzate per creare, modificare, o cancellare file nel sistema, il che potrebbe includere la modifica di file critici di sistema o il salvataggio di dati raccolti.

Funzioni come HeapCreate, HeapDestroy, HeapAlloc, HeapFree, HeapReAlloc,

VirtualAlloc, e VirtualFree potrebbero indicare che il malware è capace di allocare e gestire memoria dinamicamente, potenzialmente per caricare codice malevolo aggiuntivo o per memorizzare temporaneamente dati rubati.

Le funzioni GetEnvironmentVariableA, GetVersionExA, GetVersion, GetCPIInfo, GetACP,

e GetOEMCP suggeriscono che il malware raccoglie informazioni sul sistema e sull'ambiente di esecuzione, adattando il proprio comportamento in base alle specifiche del sistema.

Funzioni come Sleep, RtlUnwind, e GetLastError indicano che il malware potrebbe implementare tecniche di evasione, gestione degli errori e recupero da eccezioni, rendendosi più difficile da rilevare e analizzare.

Project - Malware Analysis : Analisi Statica

Inoltre, GetProcAddress, LoadLibraryA sono utilizzate per caricare dinamicamente altre librerie e ottenere indirizzi di funzioni, suggerendo che il malware può estendere le proprie capacità caricando ulteriori moduli in runtime.

Sotto possiamo vedere gli screenshot dell'analisi statica descritta sopra.

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain
000065EC	N/A	000064DC	000064E0	000064E4
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000
WININET.dll	5	000065CC	00000000	00000000

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain
000065EC	N/A	000064DC	000064E0	000064E4
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000
WININET.dll	5	000065CC	00000000	00000000

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain
000065EC	N/A	000064DC	000064E0	000064E4
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000
WININET.dll	5	000065CC	00000000	00000000

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain
000065EC	N/A	000064DC	000064E0	000064E4
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000
WININET.dll	5	000065CC	00000000	00000000

Project - Malware Analysis : Analisi Statica

Le funzioni importate da **WININET.dll** (`InternetOpenUrlA`, `InternetCloseHandle`, `InternetReadFile`, `InternetGetConnectedState`, `InternetOpenA`) indicano che il malware è capace di comunicare con server remoti, aprire connessioni URL, leggere dati da Internet e verificare lo stato della connessione di rete.

Questo suggerisce che il malware potrebbe:

- Scaricare ulteriori componenti malevoli o aggiornamenti dal server di comando e controllo.
- Inviare dati raccolti (come informazioni di sistema, file, credenziali) a server remoti.
- Ricevere comandi e istruzioni da server remoti per ulteriori azioni.

The screenshot shows the CFF Explorer interface for analyzing the malware executable `Malware_U3_W2_L5.exe`. The left pane displays the file structure, including sections like Dos Header, Nt Headers, File Header, Optional Header, Data Directories, and Section Headers. The right pane shows the 'Imports' table for the module `szAnsi`. The table includes columns for Module Name, Imports, OFTs, TimeStamp, ForwarderChain, Name RVA, and FTs (IAT). The `WININET.dll` row is highlighted, and an arrow points to the corresponding row in the detailed view of the Import Directory table below, which lists the specific functions `InternetOpenUrlA`, `InternetCloseHandle`, `InternetReadFile`, `InternetGetConnectedState`, and `InternetOpenA`.

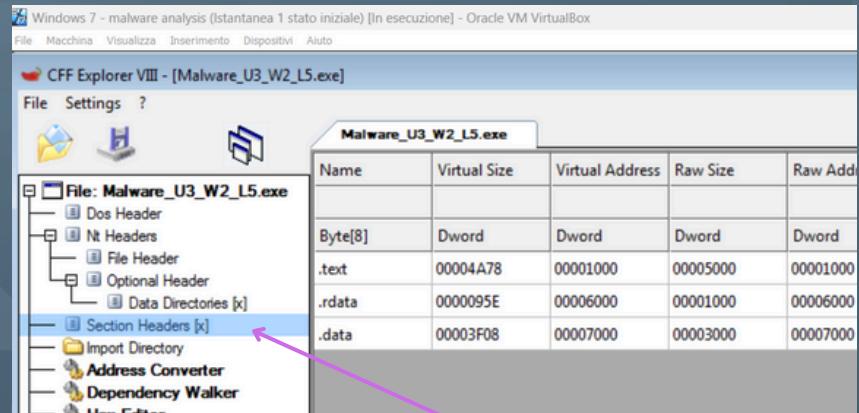
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
00006664	N/A	000064F0	000064F4	000064F8	000064FC	00006500
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

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

In sintesi: il malware analizzato sembra essere progettato per essere altamente versatile e capace di compiere diverse azioni malevoli, dal furto di dati e comunicazione con server remoti alla manipolazione e danneggiamento del sistema infetto, tutto mentre cerca di evadere il rilevamento e mantenere la propria presenza nel sistema compromesso.

Project - Malware Analysis : Analisi Statica

Ora, come richiesto dall' esercizio, andando su **Section Headers [x]**, potremo vedere quali sono le sezioni di cui si compone il file eseguibile.

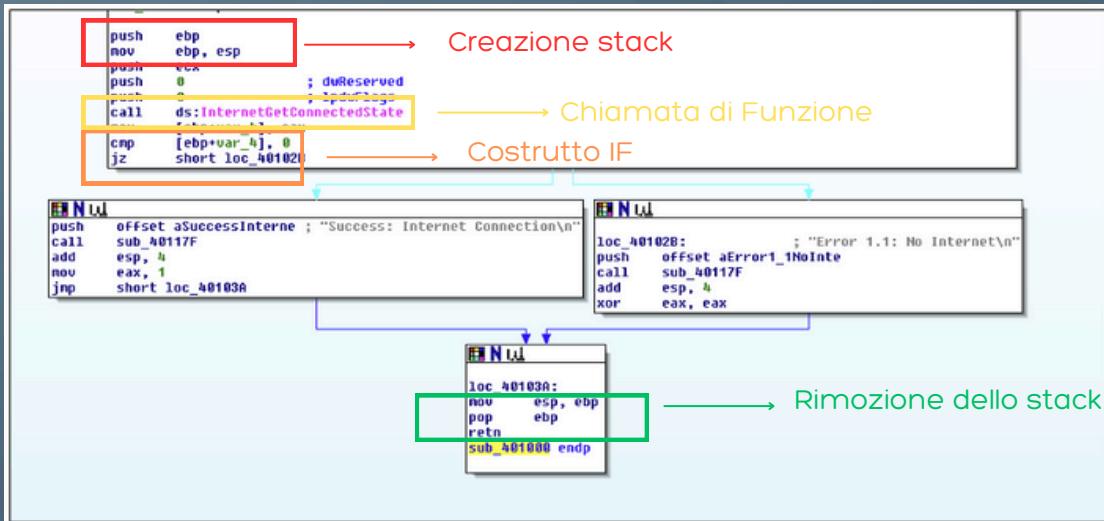


Le sezioni che troviamo riportate nello screenshot sono le seguenti:

- **.text** : contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.
- **.rdata** : include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer.
- **.data** : contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile.

Project - Assembly

Riprendiamo ora l'immagine nella traccia e andiamo ad analizzare il codice Assembly



Identificazione dei Costrutti Noti

Creazione dello Stack: Viene eseguita con le istruzioni `push ebp` e `mov ebp, esp` all'inizio della funzione.

Chiamata di Funzioni: Viene usata l'istruzione `call` per chiamare funzioni esterne.

Confronto e Salto Condizionato (costrutto IF): L'istruzione `cmp` confronta due valori e `jz` (jump if zero) esegue un salto condizionato in base al risultato del confronto.

Pulizia dello Stack e Ritorno: Viene eseguita con `mov esp, ebp`, `pop ebp` e `ret`.

Ipotesi comportamento della Funzionalità Implementata

La funzione implementata verifica se c'è una connessione Internet attiva utilizzando `InternetGetConnectedState`.

Se c'è una connessione, stampa "Success: Internet Connection" e ritorna 1.

Se non c'è connessione, stampa "Error 1.1: No Internet" e ritorna 0.

Project - Assembly

BONUS: Tabella con significato delle righe di codice

Riga di codice	Significato
push ebp	Salva il valore di ebp nello stack
mov ebp, esp	Imposta il registro ebp per puntare alla base dello stack
sub esp, 4	Riserva spazio nello stack per una variabile locale
Push 0	Passa 0 come argomento (dwReserved) alla funzione chiamata
Push 0	Passa 0 come argomento (dwFlags) alla funzione chiamata
Call ds:InternetGetConnectedState	Chiama la funzione InternetGetConnectedState per verificare lo stato della connessione
mov [ebp+var_4], eax	Memorizza il risultato della funzione nella variabile locale.
cmp [ebp+var_4], 0	Confronta il risultato con 0
jz short loc_40102B	Salta all' etichetta loc_40102B se il risultato è zero (nessuna connessione)
push offset aSuccessInterne	Passa il messaggio Success: Internet Connection come argomento
call sub_40117F	Chiama una funzione per gestire l'output del messaggio
add esp, 4	Ripulisce lo stack dopo ka chiamata alla funzione
mov eax, 1	Imposta eax a 1 per indicare successo
jmp short loc_401030	Salta alla fine della funzione
loc_40102B	Etichetta per la gestione dell'errore
push offset aError1_1NoInte	Passa il messaggio Error1.1: No Internet come argomento
call sub_40117F	Chiama una funzione per gestire l'output del messaggio di errore
add esp, 4	Ripulisce lo stack dopo ka chiamata alla funzione
xor eax eax	Imposta eax a 0 per indicare errore
loc_401030:	Etichetta per la fine della funzione
mov esp, ebp	Ripristina il puntatore dello stack
pop ebp	Ripristina il valore di ebp dello stack
ret	Ritorna alla funzione chiamante