

Aprile 2024



**InfiniTech
S.p.A.**

BUILD **WEEK III** **MALWARE** **ANALYSIS** **EPICODE**

TEAM LEADER:
Francesco Perticaroli

TEAM MEMBERS:
Manuel Buonanno
Bruno Falconi
Flaviano Sedici
Jacopo Trovato

Indice

03 **Traccia**

04 **Giorno 1**

05 **Diagramma di flusso**

06 **Funzionalità**

07 **Funzioni call**

08 **Istruzioni**

09 **Glossario**

EPICODE

Traccia

Il Malware da analizzare è nella cartella “**Build_Week_Unit_3**” presente sul desktop della macchina virtuale dedicata.

- **Giorno 1:**

Con riferimento al file eseguibile “**Malware_Build_Week_U3**”, rispondere ai seguenti quesiti utilizzando i tool e le tecniche apprese nelle lezioni teoriche:

1. Quanti parametri sono passati alla funzione Main()?
2. Quante variabili sono dichiarate all'interno della funzione Main ()?
3. Quali sezioni sono presenti all'interno del file eseguibile?
Descrivete brevemente almeno 2 di quelle identificate
4. Quali librerie importa il Malware? Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

Giorno 1

Per l'analisi del malware in questione, abbiamo impiegato **IDA Pro**, un software che esamina il contenuto del malware e restituisce il suo codice in linguaggio assembly. Il codice disassemblato viene visualizzato tramite una barra colorata; ciò che ci interessa è il colore blu, che rappresenta il codice del malware. Una volta completate le operazioni da parte di IDA, sullo schermo comparirà il punto di ingresso dell'applicazione, cioè il punto in cui il programma inizia ad eseguire il suo codice.

Quanti parametri sono passati alla funzione Main()? 3

Nel main della funzione le variabili dichiarate sono tutte quelle che hanno i numeri negativi in verde.

Invece, nella parentesi nera sono presenti i parametri, che hanno valore positivo.

```
.text:004011D0 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:004011D0 _main          proc near          ; CODE XREF: start+AF↓p
.text:004011D0
.text:004011D0 hModule          = dword ptr -11Ch
.text:004011D0 Data            = byte ptr -118h
.text:004011D0 var_117         = byte ptr -117h
.text:004011D0 var_8           = dword ptr -8
.text:004011D0 var_4           = dword ptr -4
.text:004011D0 argc            = dword ptr 8
.text:004011D0 argv            = dword ptr 0Ch
.text:004011D0 envp            = dword ptr 10h
.text:004011D0
```

Non so come spiegarlo

Quante variabili sono dichiarate all'interno della funzione Main()? 5

Quali sezioni sono presenti all'interno del file eseguibile?

Descrivete brevemente almeno 2 di quelle identificate

L'**analisi statica basica** consiste nell'esaminare un eseguibile senza vedere le istruzioni che lo compongono. Lo scopo è quello di confermare se un dato file è malevolo e fornire informazioni generiche circa le sue funzionalità. Questa metodologia è sicuramente la più intuitiva e semplice da mettere in pratica, ma risulta anche essere la più inefficiente soprattutto contro malware sofisticati.

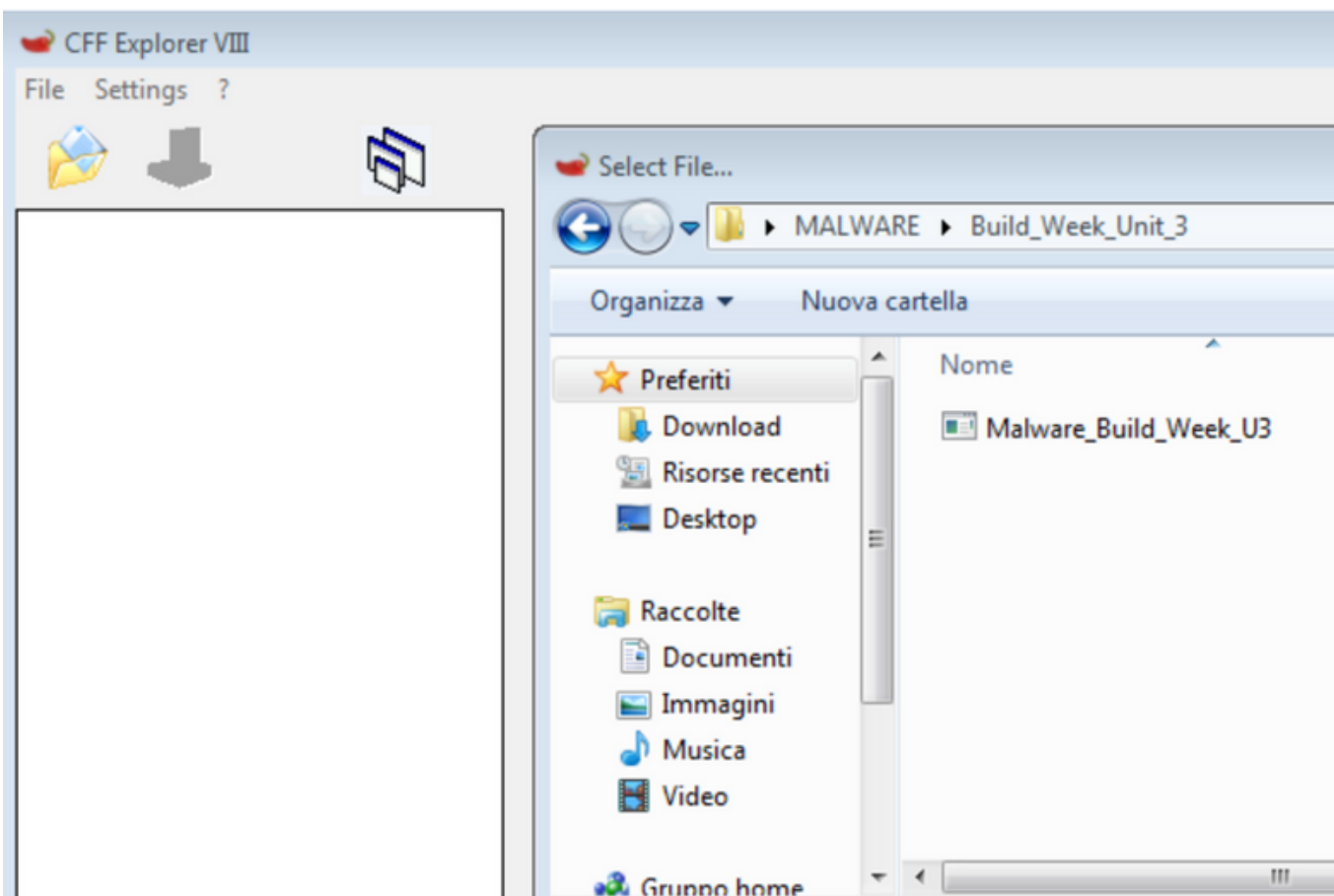
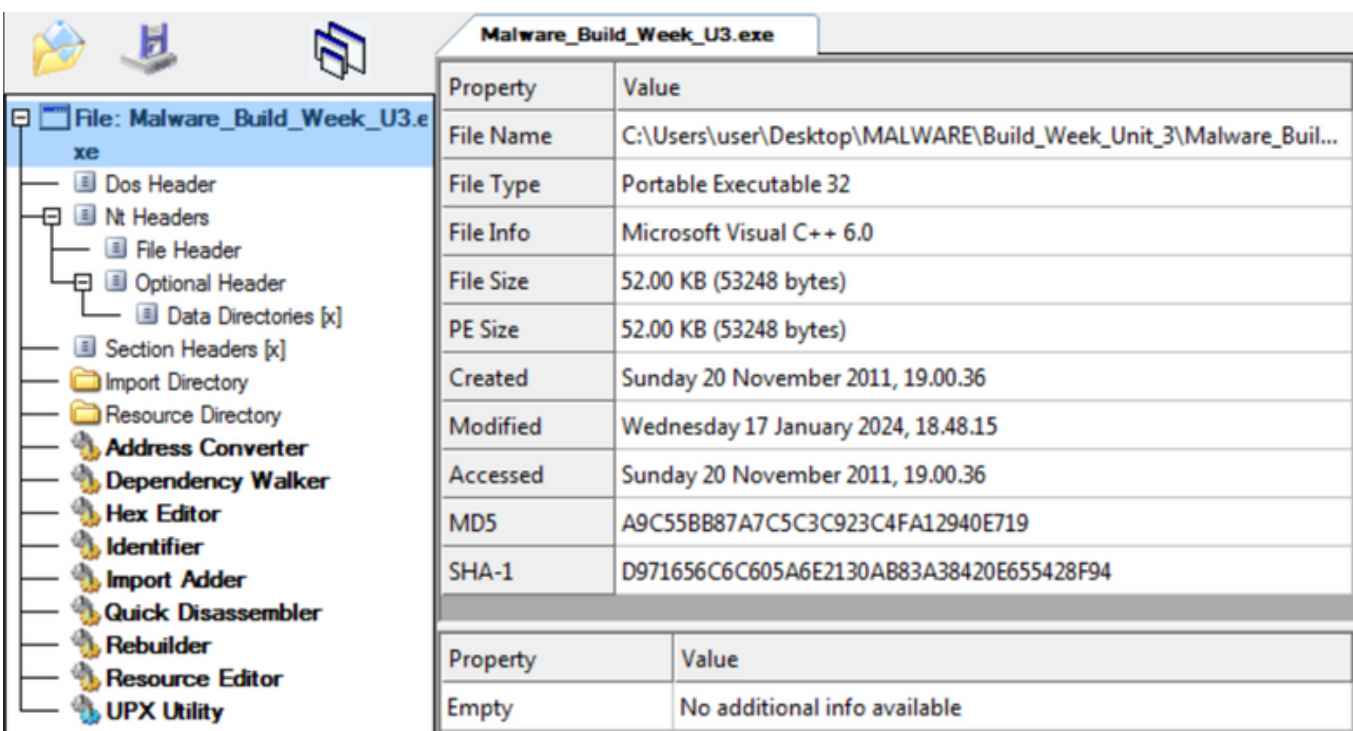
Windows utilizza per la maggior parte dei file eseguibili il formato **PE**, "**PortableExecutable**". Il formato PE al suo interno contiene delle informazioni necessarie al sistema operativo per capire come gestire il codice del file, come ad esempio le librerie e funzioni.

Quando un programma ha bisogno di una funzione "chiama" una libreria al cui interno è definita la funzione necessaria. Oltre alle funzioni importate, un file eseguibile può esportare funzioni. Ovvero, può mettere a disposizione di altri programmi o dell'utente delle funzioni da "chiamare".

L'header del formato PE contiene anche un elenco delle funzioni esportate da un eseguibile. Per controllare le funzioni importate ed esportate da un malware, possiamo utilizzare il tool **CFF Explorer**.

CFF Explorer è uno strumento software progettato per analizzare e modificare i file eseguibili di Windows, come i file EXE, DLL, OCX e molti altri. Il suo nome "**CFF**" sta per "**Cavalry File Format**", che è un formato di file proprietario introdotto da lui stesso. Le principali funzionalità di CFF Explorer includono l'analisi dei file, la modifica dei file, il supporto per plugin ed estensioni, l'esportazione di dati e il controllo della sicurezza.

Questo software è utilizzato principalmente dagli sviluppatori software, dagli analisti di sicurezza e dagli esperti di reverse engineering per analizzare, modificare e comprendere i file eseguibili di Windows.

Interfaccia
DLL explorerInterfaccia
del Malware

Possiamo notare come CFF Explorer analizza tutto il file che abbiamo caricato restituendoci molte informazioni su esso.

Quali librerie importa il Malware? Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

Per controllare le librerie e le funzioni importate, ci spostiamo su «import directory» nel menù a sinistra. Il pannello darà informazioni sulle librerie importate dall'eseguibile.

In questo caso il malware che abbiamo analizzato importa 2 librerie:

- **Kernel32.dll**: contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria;
- **Advapi32.dll**: contiene le funzioni per interagire con i servizi ed i registri del sistema operativo;

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
0000769E	N/A	000074EC	000074F0	000074F4	000074F8	000074FC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion

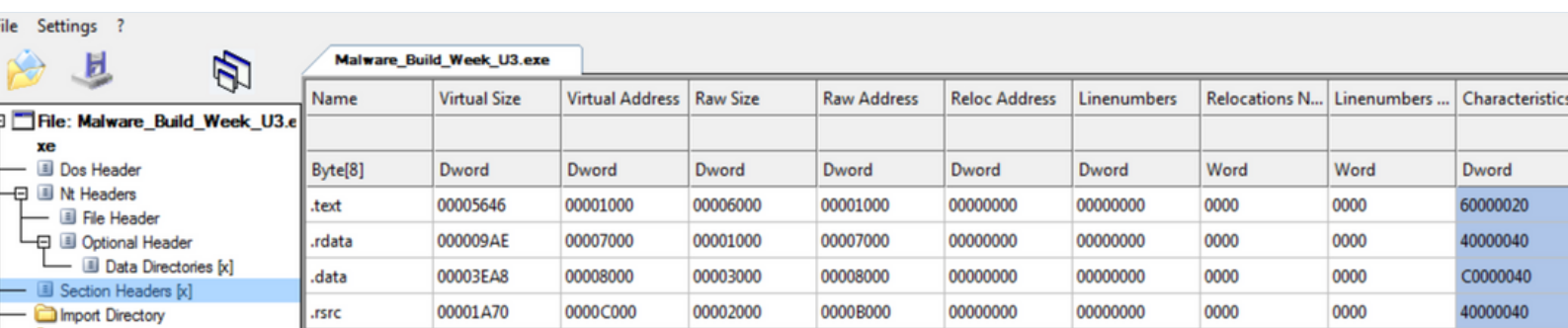
Andando poi a selezionare queste librerie ci verrà mostrato nel pannello inferiore una lista delle funzioni richieste all'interno della libreria selezionata.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000076AC	000076AC	0186	RegSetValueExA
000076BE	000076BE	015F	RegCreateKeyExA

Queste sono comunemente utilizzate in un malware per eseguire per ottenere il controllo del sistema. Ecco cosa si può dedurre da ciascuna di esse:

- **szAnsi**: Nome personalizzato assegnato dall'autore del malware.
- **SizeOfResource**: Restituisce la dimensione, in byte, di una risorsa specificata all'interno di un file eseguibile.
- **LockResource**: Blocca una risorsa specificata in memoria e restituisce un puntatore a essa.
- **LoadResource**: Carica una risorsa specificata in memoria.
- **VirtualAlloc**: Alloca memoria in un processo specificato.
- **GetModuleFileNameA**: Restituisce il percorso completo del file eseguibile per il modulo specificato.
- **GetModuleHandleA**: Restituisce un handle per il modulo caricato nel processo chiamante.
- **FreeResource**: Rilascia una risorsa precedentemente caricata in memoria.
- **FindResourceA**: Ricerca una risorsa specificata all'interno di un file eseguibile.
- **CloseHandle**: Chiude un handle aperto.
- **GetCommandLineA**: Restituisce una stringa contenente gli argomenti della riga di comando del programma.
- **GetVersion**: Restituisce informazioni sulla versione del sistema operativo in esecuzione.
- **RegSetValueExA**: Imposta il valore di un'entry nel Registro di sistema di Windows.
- **RegCreateKeyExA**: Crea o apre una chiave nel Registro di sistema.

Per controllare le sezioni di un file eseguibile spostiamoci nel pannello a sinistra nella sezione «section headers». Il pannello principale a destra mostrerà le informazioni circa le sezioni di cui si compone l'eseguibile.



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	00005646	00001000	00006000	00001000	00000000	00000000	0000	0000	60000020
.rdata	000009AE	00007000	00001000	00007000	00000000	00000000	0000	0000	40000040
.data	00003EA8	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040

bho

La figura riporta non solo il nome delle sezioni ma anche altre importanti informazioni, come ad esempio:

- **Virtual size:** indica lo spazio allocato per la sezione durante il processo di caricamento dell'eseguibile in memoria.
- **Rawsize:** indica lo spazio occupato dalla sezione quando è sul disco.

L'header del formato PE fornisce molte altre informazioni importanti oltre alle funzioni/librerie importate ed esportate. Ogni sezione ha un preciso scopo, e conoscerle è una preziosa informazione per le analisi. In questo caso le sezioni del formato PE sono:

- **.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.
- **.rsrc:** include le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell'eseguibile stesso.

Aprile 2024



**InfiniTech
S.p.A.**

BUILD WEEK

GRAZIE PER L'ATTENZIONE

TEAM LEADER:

Francesco Perticaroli

TEAM MEMBERS:

Manuel Buonanno

Bruno Falconi

Flaviano Sedici

Jacopo Trovato