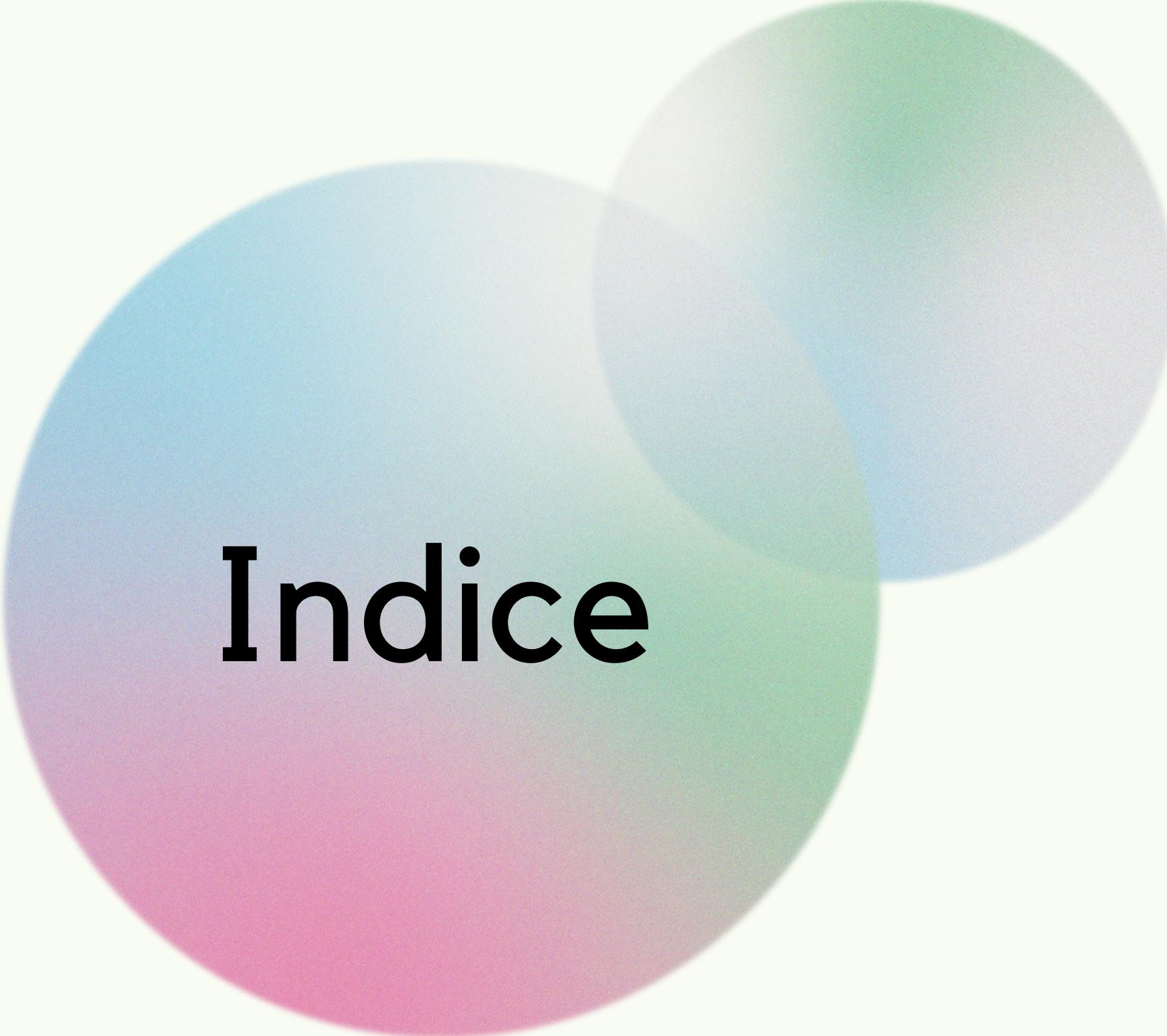


Malware Analysis

Build Week 3

TEAM DI TURO

08-12 aprile 2024



Indice

- Presentazione Tem
- Traccia e Requisiti
- Giorno 1
- Giorno 2
- Giorno 3
- Giorno 4
- Giorno 5
- Conclusioni

Presentazione Team

Team leader - Davide Di Turo

Team:

- Alessio D'Ottavio
- Manuel Di Gangi
- Verdiana Germani
- Francesco Valcavi
- Michael Robert Antonio Di Leggio



Traccia e requisiti

[Torna all'indice](#)

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:

- Quanti parametri sono passati alla funzione Main()?
- Quante variabili sono dichiarate all'interno della funzione Main()?
- Quali sezioni sono presenti all'interno del file eseguibile?
Descrivete brevemente almeno 2 di quelle identificate
- 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 3

Riprendete l'analisi del codice, analizzando le routine tra le locazioni di memoria 00401080 e 00401128:

- Qual è il valore del parametro «ResourceName» passato alla funzione FindResourceA();
- Il susseguirsi delle chiamate di funzione che effettua il Malware in questa sezione di codice l'abbiamo visto durante le lezioni teoriche.
Che funzionalità sta implementando il Malware?
- È possibile identificare questa funzionalità utilizzando l'analisi statica basica ? (dal giorno 1 in pratica)
- In caso di risposta affermativa, elencare le evidenze a supporto.

Entrambe le funzionalità principali del Malware viste finora sono richiamate all'interno della funzione Main(). Disegnare un diagramma di flusso (inserite all'interno dei box solo le informazioni circa le funzionalità principali) che comprenda le 3 funzioni.

Giorno 2

Con riferimento al Malwarein analisi, spiegare:

- Lo scopo della funzione chiamata alla locazione di memoria 00401021
- Come vengono passati i parametri alla funzione alla locazione 00401021;
- Che oggetto rappresenta il parametro alla locazione 00401017
- Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029. (se serve, valutate anche un'altra o altre due righe assembly)
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C .
- Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

Nel complesso delle due funzionalità appena viste, spiegate quale funzionalità sta implementando il Malwarein questa sezione.

Traccia e requisiti

[Torna all'indice](#)

Giorno 4

Preparate l'ambiente ed i tool per l'esecuzione del Malware(suggerimento: avviate principalmente Process Monitor ed assicurate di eliminare ogni filtro cliccando sul tasto «reset» quando richiesto in fase di avvio). Eseguite il Malware, facendo doppio click sull'icona dell'eseguibile

- Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda

Analizzate ora i risultati di Process Monitor (consiglio: utilizzate il filtro come in figura sotto per estrarre solo le modifiche apportate al sistema da parte del Malware). Fate click su «ADD» poi su «Apply» come abbiamo visto nella lezione teorica.

Filtrate includendo solamente l'attività sul registro di Windows

- Quale chiave di registro viene creata?
- Quale valore viene associato alla chiave di registro creata?

Passate ora alla visualizzazione dell'attività sul File System.

- Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?

Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

Giorno 5

GINA (Graphical identification and authentication) è un componente lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica -ovvero permette agli utenti di inserire usernamee password nel classico riquadro Windows, come quello in figura a destra che usate anche voi per accedere alla macchina virtuale.

- Cosa può succedere se il file .dll lecito viene sostituito con un file .dllmalevolo, che intercetta i dati inseriti?

Sulla base della risposta sopra, delineate il profilo del Malwaree delle sue funzionalità.

Giorno 1

IDA PRO

IDA (Interactive Disassembler) è un software ampiamente utilizzato per il disassemblaggio e il debugging.

Le principali caratteristiche di IDA includono:

- **Disassemblaggio:** IDA può disassemblare eseguibili binari in codice assembly, fornendo una rappresentazione leggibile dall'uomo del codice macchina.
- **Rappresentazione Grafica:** Offre un'interfaccia grafica per navigare attraverso il codice disassemblato, che aiuta a comprendere le strutture di programma complesse.
- **Analisi:** IDA esegue un'analisi statica del binario, identificando funzioni, variabili e strutture di flusso di controllo. Questo aiuta a comprendere la funzionalità del software.

The screenshot shows the IDA Pro interface with the assembly view active. The assembly code for the `_main` function is displayed, starting with the declaration:

```
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near
```

Local variables are defined:

```
hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

The assembly instructions for the prologue are:

```
push    ebp
mov     ebp, esp
sub    esp, 11Ch
push    ebx
push    esi
```

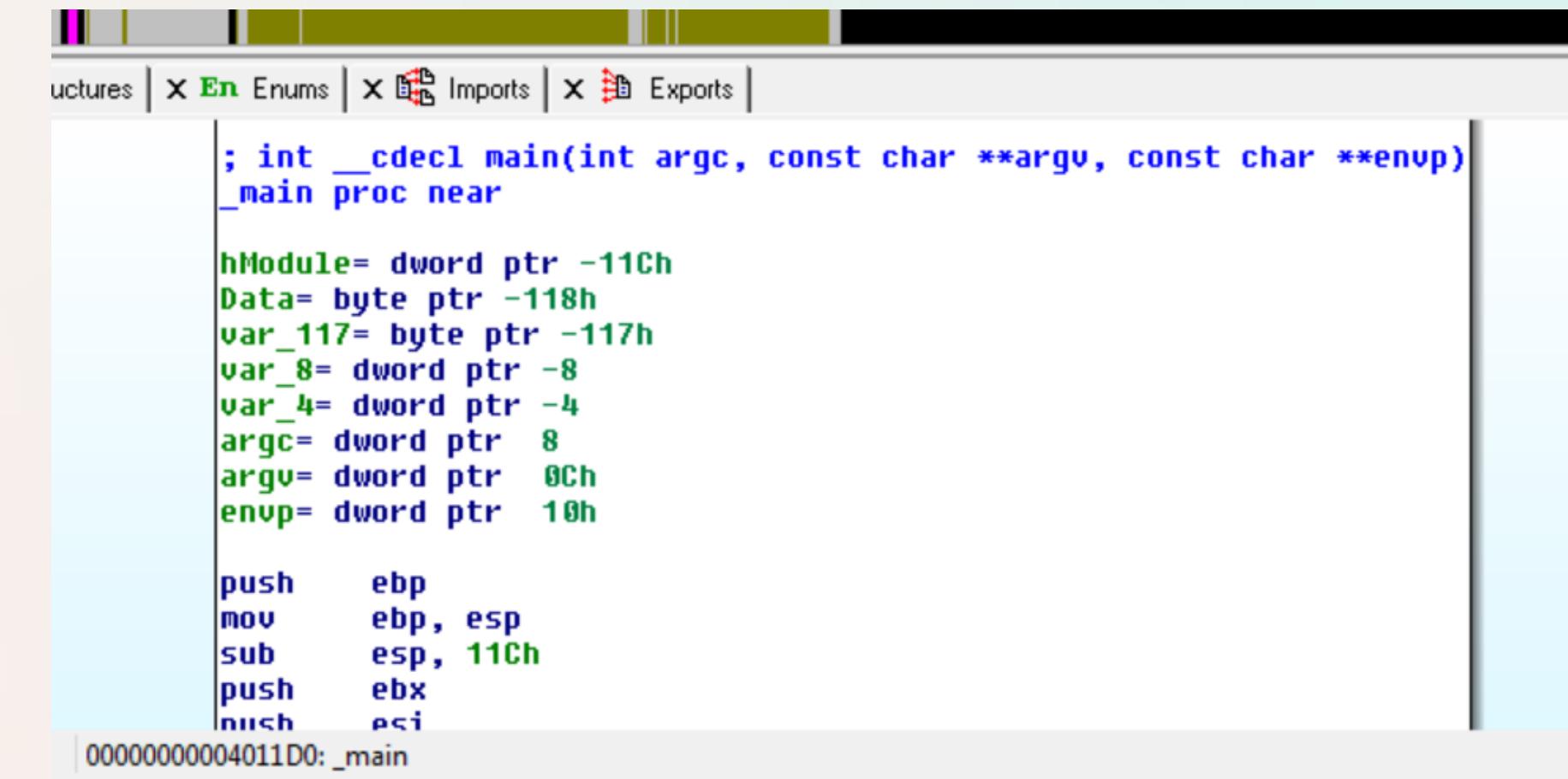
The instruction address is shown as `000000000004011D0: _main`.

- **Debugging:** Fornisce funzionalità di debugging, consentendo agli utenti di passare attraverso il codice disassemblato, impostare punti di interruzione, ispezionare la memoria e analizzare il comportamento in fase di esecuzione.
- **Supporto per lo Scripting:** IDA supporta lo scripting utilizzando vari linguaggi di programmazione come IDC (IDA C), Python e IDC Script. Ciò consente di automatizzare compiti e personalizzare il processo di analisi.
- **Ecosistema dei Plugin:** IDA può essere esteso attraverso plugin, che forniscono funzionalità aggiuntive adattate a specifici casi d'uso o compiti di analisi.

Parametri e Variabili

Argomenti nella funzione main:

argc	Indica il numero di entrate nell'array argv
argv	Un array di puntatori ad una stringa che contengono gli argomenti del programma
envp	Un array di puntatori ad una stringa che definisce l'ambiente del programma



```

; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h

push    ebp
mov     ebp, esp
sub    esp, 11Ch
push    ebx
push    esi
push    edi

00000000000401D0: _main

```

- La funzione main è fornita dall'user ed è dove l'esecuzione del programma ha inizio. La linea di comando al programma è suddivisa in sequenze di tokens o separate da blanks; e sono passate al main come un array di puntatori a stringhe/strings in argv. Il numero di argomenti trovati passa al parametro argc.
- L'argomento argv è un puntatore ad un carattere string contenente il nome del programma. L'ultimo elemento dell'array puntato ad argv è NULL. Gli argomenti contenenti blanks possono essere passati al main racchiudendoli in una citazione (che viene rimossa da quel elemento nel vettore argv).
- L'argomento envp punta ad un array di puntatori a caratteri strings che sono l'ambiente delle stringhe per il corrente processo. Questo valore è identico alla variabile environ definita nel file header <stdlib.h>.

Parametri e Variabili

Nella funzione main troviamo **5 variabili**

```
push    ebx  
push    esi  
push    edi
```

hModule	Questa è una variabile usata nel SO Windows per rappresentare un identificatore (handle) associato ad un modulo
Data	Si tratta di una variabile flessibile adatta per memorizzare dati di qualsiasi tipo senza specificare un formato particolare
var_117	Questa potrebbe rappresentare un'area di memoria usata per memorizzare dati temporanei o risultati non definitivi durante l'esecuzione di un programma
var_8	Questa è una variabile che può contenere diversi tipi di dati
var_4	Questa è una variabile che può contenere diversi tipi di dati

```
hModule= dword ptr -11Ch  
Data= byte ptr -118h  
var_117= byte ptr -117h  
var_8= dword ptr -8  
var_4= dword ptr -4
```

Giorno 1

Parametri e Variabili

Nella funzione main troviamo 5 variabili

push	ebx
push	esi
push	edi

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_117= byte ptr -117h
var_8= dword ptr -8
var_4= dword ptr -4

hModule	<ul style="list-style-type: none">• Tipo di dato: dword ptr (puntatore a dword, un dword è una parola di 32 bit)• Offset: -11Ch rispetto al punto di riferimento nella memoria
Data	<ul style="list-style-type: none">• Tipo di dato: byte ptr (puntatore a byte)• Offset: -118h rispetto al punto di riferimento nella memoria
var_117	<ul style="list-style-type: none">• Tipo di dato: byte ptr (puntatore a byte)• Offset: -117h rispetto al punto di riferimento nella memoria
var_8	<ul style="list-style-type: none">• Tipo di dato: dword ptr (puntatore a dword)• Offset: -8 rispetto al punto di riferimento nella memoria
var_4	<ul style="list-style-type: none">• Tipo di dato: dword ptr (puntatore a dword)• Offset: -4 rispetto al punto di riferimento nella memoria

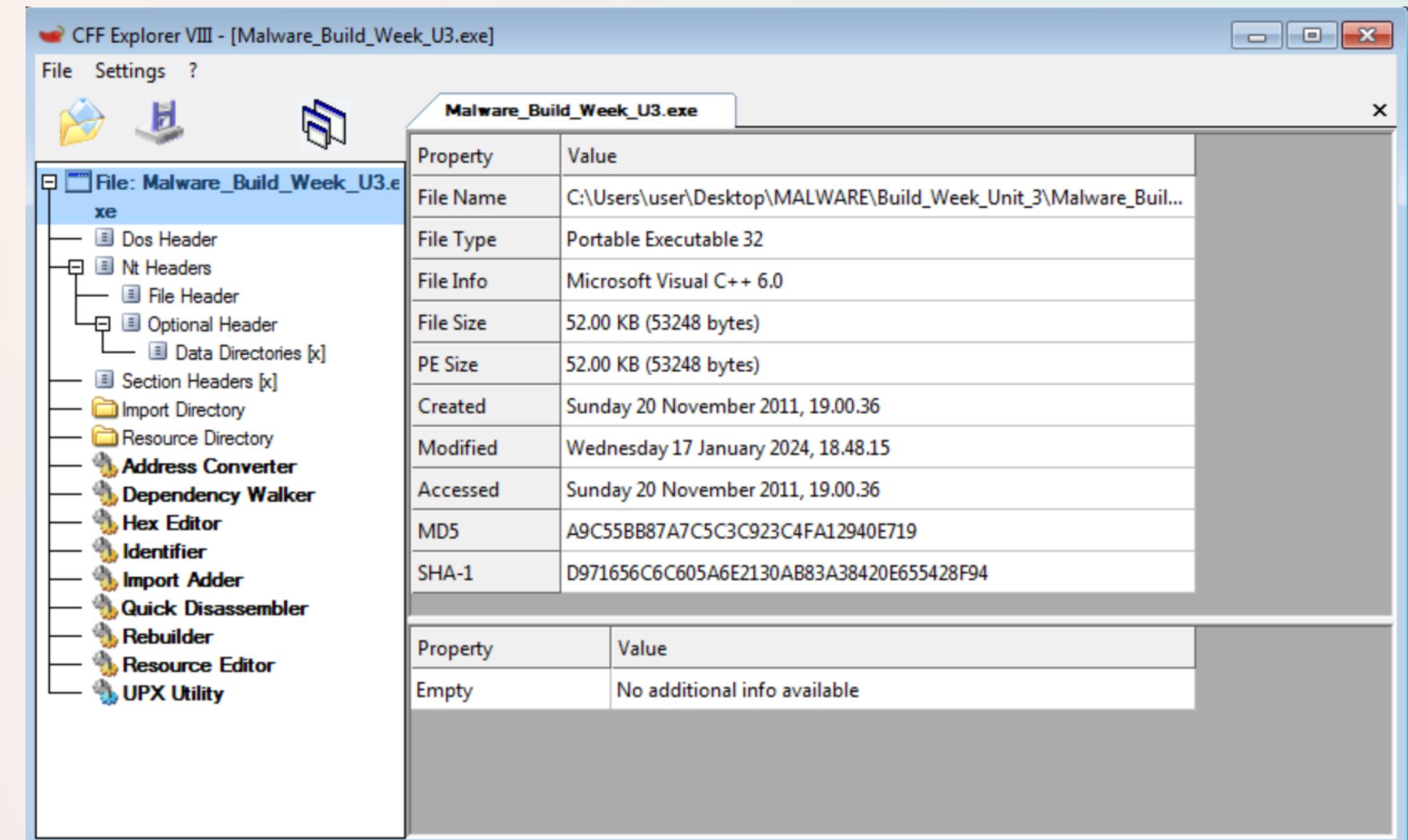
Nella descrizione "ptr" sta per puntatore, e dword e byte sono tipi di dato che rappresentano rispettivamente un doppio word (una parola di 32 bit) e un byte (8 bit). Gli offset negativi indicano che le variabili sono posizionate in memoria a una distanza negativa rispetto al punto di riferimento. Questo è un formato comune per la rappresentazione delle variabili in linguaggi di programmazione a basso livello o nell'assembly.

Giorno 1

Sezioni

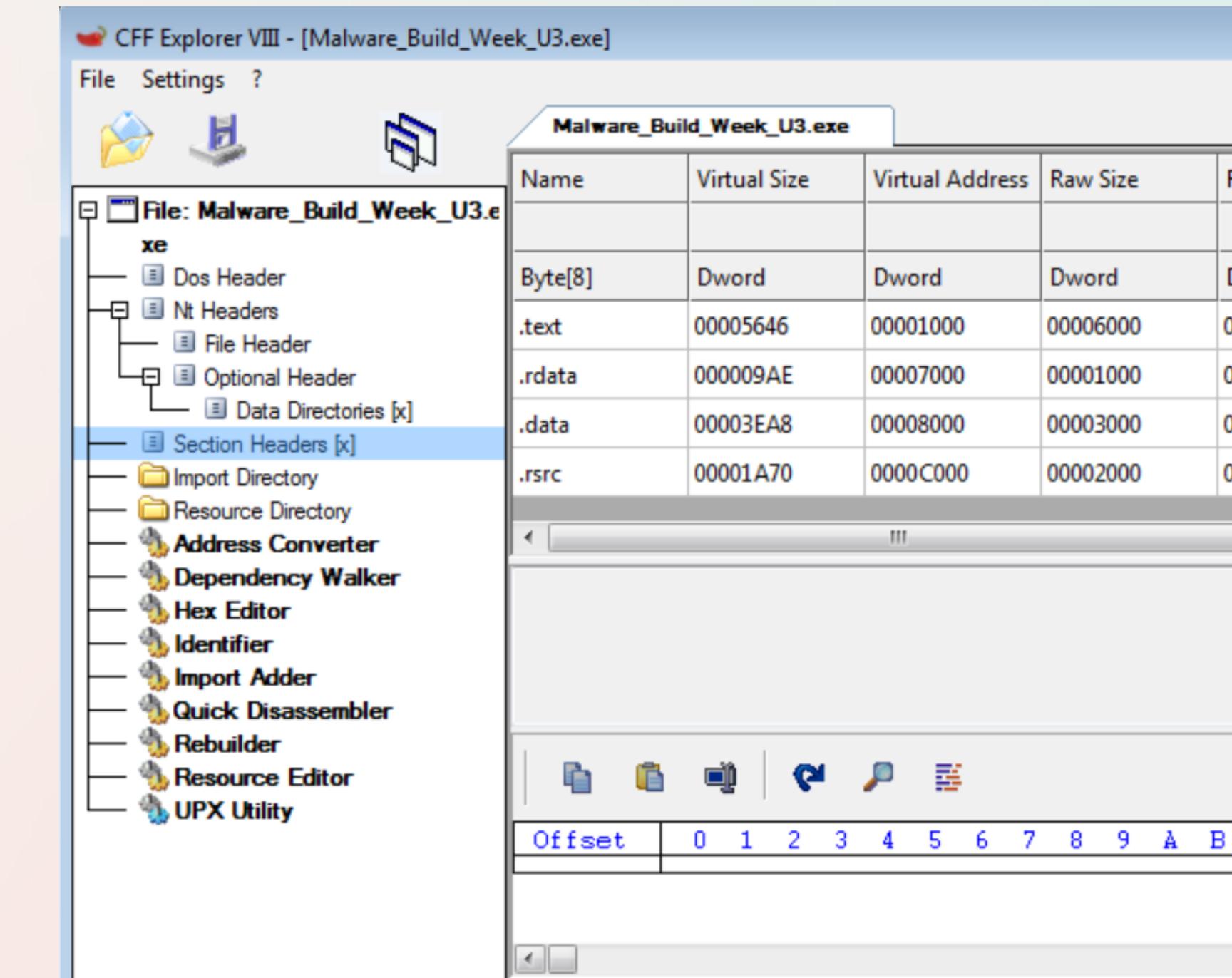
[Torna all'indice](#)

Utilizziamo CFF Explorer per l'analisi dei file eseguibili su Windows; possiamo trovare al suo interno varie funzionalità che ci permettono di analizzare le funzioni, le risorse, le librerie etc.



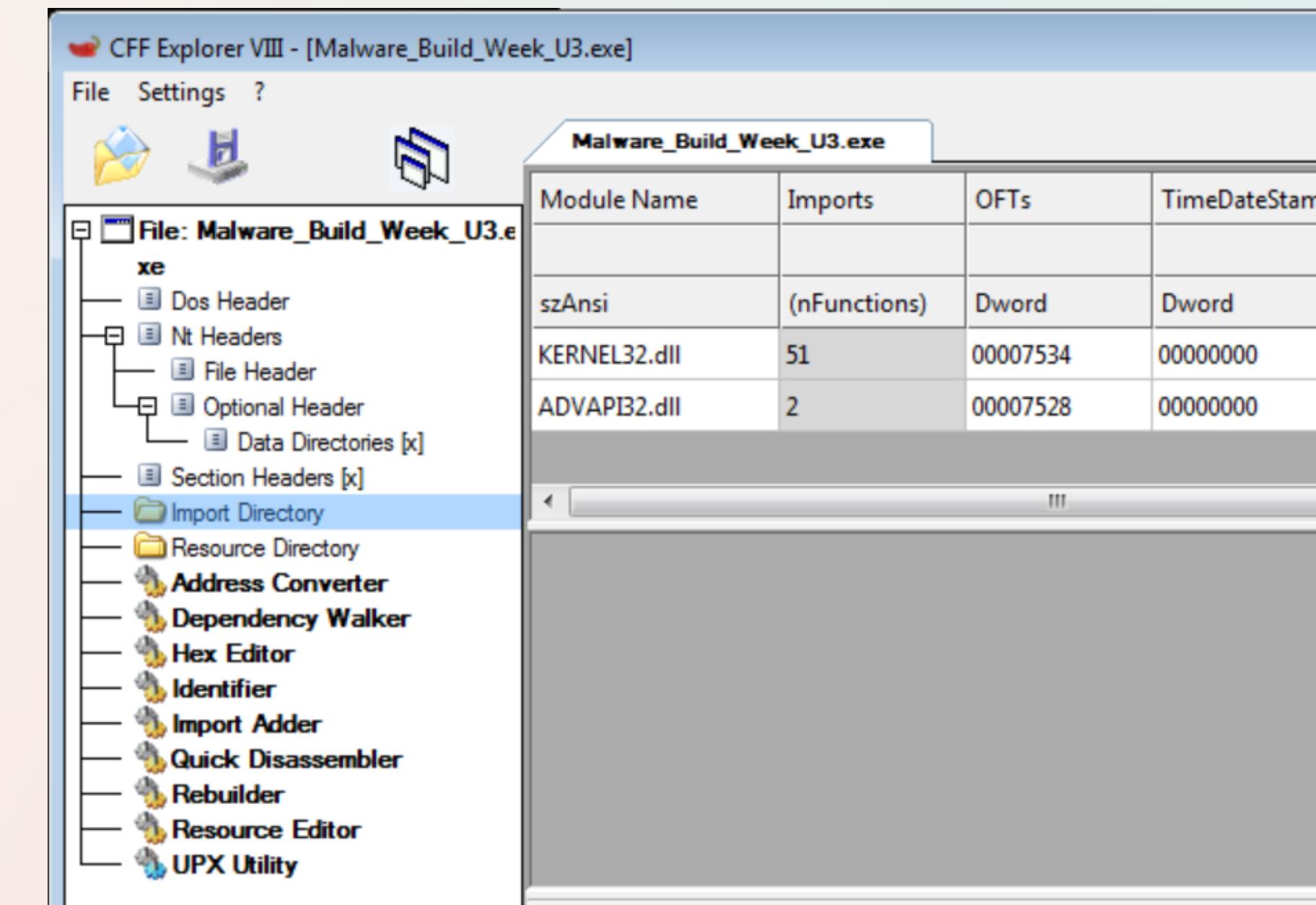
Sezioni

.text	Contiene le istruzioni che il processore eseguirà una volta che il software verrà avviato. Questa è, generalmente, 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 le informazioni circa le librerie e le funzioni importate ed esportate dal file eseguibile
.data	Contiene le variabili globali del programma eseguibile; si tratta di variabili non definite all'interno di un contesto di una funzione, ma bensì globalmente dichiarate e di conseguenza accessibili 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



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



Funzioni

Tra le funzioni importate dal malware possiamo vederne alcune del Kernel32.dll:

- **SizeofResource**: essa riporta la dimensione di una risorsa all'interno di un eseguibile o dll.
- **LockResource**: essa viene usata per acquisire un puntatore al cui interno sono presenti i dati di una risorsa bloccata.
- **LoadResource**: carica una risorsa definita all'interno di un modulo.
- **CreateFileA**: viene usata per creare file
- **LoadLibraryA**: viene usata per caricare un dll
- **GetProcAddress**: permette al programma di ricavare l'indirizzo di una funzione da un dll.

Possiamo ora analizzare le funzioni di Advapi32.dll:

- **RegSetValueExA**: permette di aggiungere un nuovo valore all'interno del registro e di settare i rispettivi dati. Accetta come parametri la chiave, la sottochiave e il dato da inserire. Ciò implica che il malware tenta di modificare o aggiungere un valore all'interno del Registro di sistema per poter persistere nel sistema o modificare le configurazioni
- **RegCreateKeyExA**: viene usata per creare una nuova chiave o aprire una chiave esistente nel Registro di sistema. Potrebbe voler dire che il malware tenta di stabilire una persistenza all'interno del sistema o tenta di modificare le impostazioni di sistema.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource

000078E6	000078E6	0034	CreateFileA
000078F4	000078F4	00BF	GetCPInfo
00007900	00007900	00B9	GetACP
0000790A	0000790A	0131	GetOEMCP
00007916	00007916	013E	GetProcAddress
00007928	00007928	01C2	LoadLibraryA

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

Funzioni

Attraverso queste funzioni possiamo ipotizzare che il comportamento del malware riguardi il caricamento e l'uso di risorse all'interno del sistema, questo può farci pensare al caricamento di dati, asset e alla manipolazione delle risorse di sistema per l'esecuzione di codice malevolo o l'installazione di una backdoor. Inoltre, le funzioni ci permettono anche di comprendere che il malware interagisce con il registro di sistema e usa le funzioni di sistema per acquisire informazioni o probabilmente per modificare le configurazioni di sistema. Dunque, l'analisi effettuata ci porta ad ipotizzare un comportamento tipico di un malware che tenta di ottenere l'accesso e il controllo del sistema attraverso la manipolazione del registro di sistema e l'uso delle risorse di sistema; infatti il registro di Windows è utilizzato per salvare informazioni sul sistema operativo come ad esempio configurazioni del sistema stesso o delle applicazioni che girano sul sistema e i malware lo usano spesso per ottenere la «persistenza».

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource

000078E6	000078E6	0034	CreateFileA
000078F4	000078F4	00BF	GetCPInfo
00007900	00007900	00B9	GetACP
0000790A	0000790A	0131	GetOEMCP
00007916	00007916	013E	GetProcAddress
00007928	00007928	01C2	LoadLibraryA

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