

Malware Analysis

Sommario

Introduzione.....	2
Analisi Statica	3
Analisi dei parametri e delle variabili locali	3
Analisi della funzione Main()	5
Quanti parametri sono passati alla funzione Main()?.....	6
Quante variabili sono dichiarate all'interno della funzione Main()?	6
Analisi delle sezioni del file eseguibile.....	7
Quali sezioni sono presenti all'interno del file eseguibile?.....	9
Analisi delle librerie importate dal malware	10
Quali librerie importa il Malware?	10
Ipotesi su funzionalità del malware	10
Malware Analysis	12
Dettagli della funzione alla locazione di memoria 00401021.....	13
Locazione di memoria 00401017	14
Locazioni di memoria 00401027 - 00401029	15
Locazione di memoria 00401047	16
Analisi Dinamica.....	17
Analisi risultati di Process Monitor	19
Funzionamento del malware	23

Introduzione

Malware da analizzare: Malware_Build_Week_U3

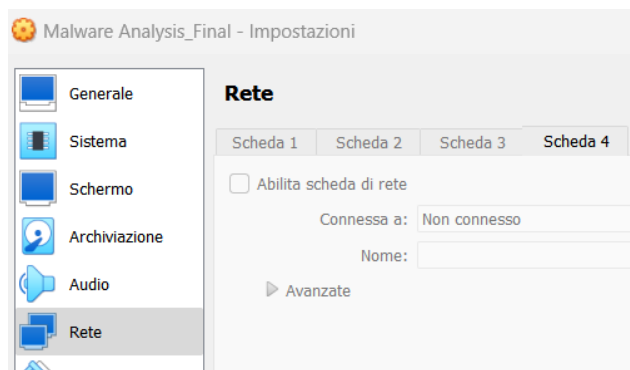
Note generali:

Il malware oggetto di analisi si trova nella macchina virtuale Windows XP "Malware Analysis_Final" installata su VirtualBox.

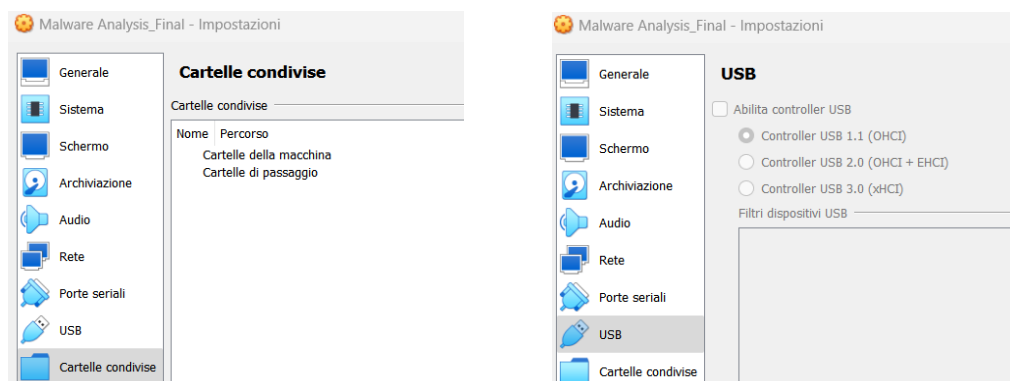
Ai fini della malware analysis, specialmente dell'analisi dinamica, che presuppone l'esecuzione del malware, è opportuno assicurarsi di operare in un ambiente di test isolato, in modo da assicurarsi che le operazioni eseguite dal malware non si propaghino ad altri dispositivi attraverso la rete.

In particolare, verifichiamo che:

La macchina virtuale Windows XP su VirtualBox non abbia schede di rete abilitate



Non ci siano cartelle condivise con Windows XP né dispositivi USB connessi a Windows XP



Inoltre, è buona pratica creare su VirtualBox delle istantanee della macchina virtuale nel suo stato iniziale, prima di iniziare tutte le analisi, in modo tale da ripristinarlo qualora ce ne fosse bisogno.

Nome	Creata
✓ Istantanea 1	23/10/2023 19:40 (26 giorni fa)
✓ Istantanea 2	09/11/2023 19:27 (9 giorni fa)
➔ Stato attuale (modificato)	

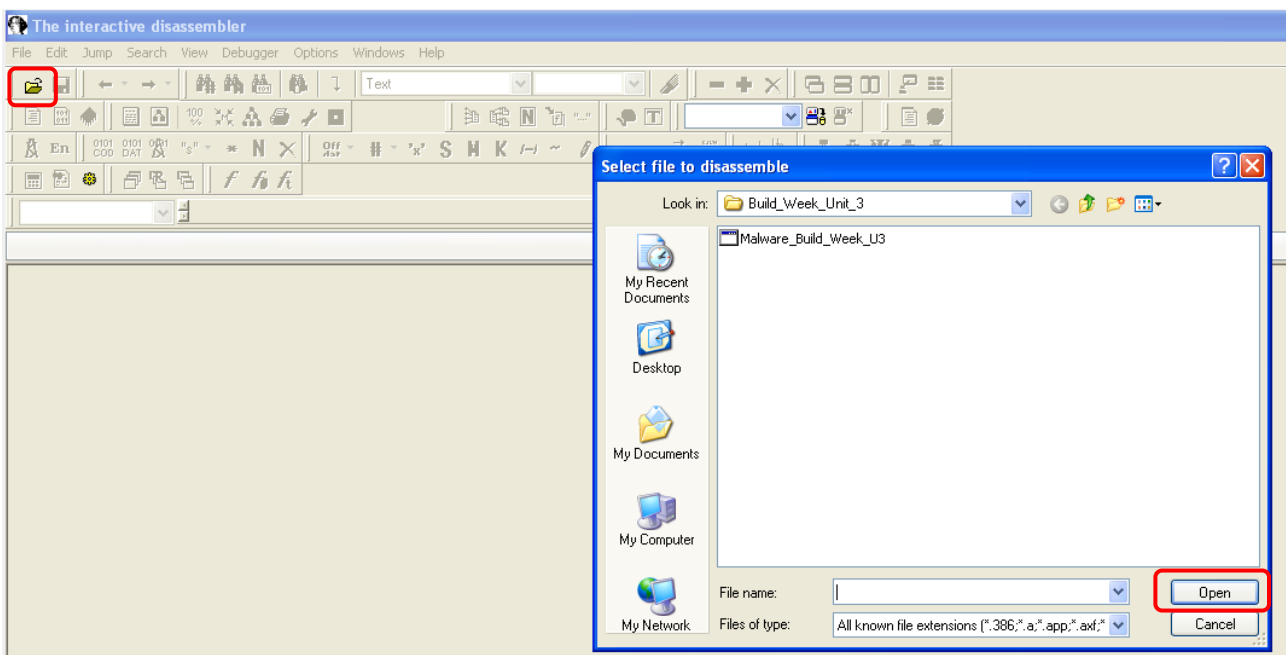
Analisi Statica

Analisi dei parametri e delle variabili locali

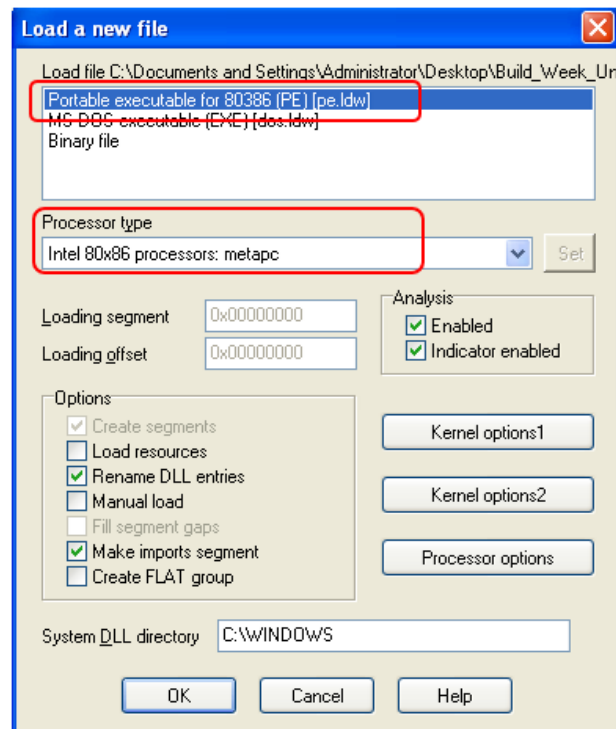
Un tool importante per l'analisi statica di un malware è IDA.

IDA, acronimo di "Interactive DisAssembler," è un potente strumento di disassemblaggio utilizzato principalmente nell'ambito della sicurezza informatica e dell'analisi dei malware. Questo software consente agli analisti di esaminare il codice binario di un programma o di un file eseguibile, traducendolo in linguaggio assembly comprensibile.

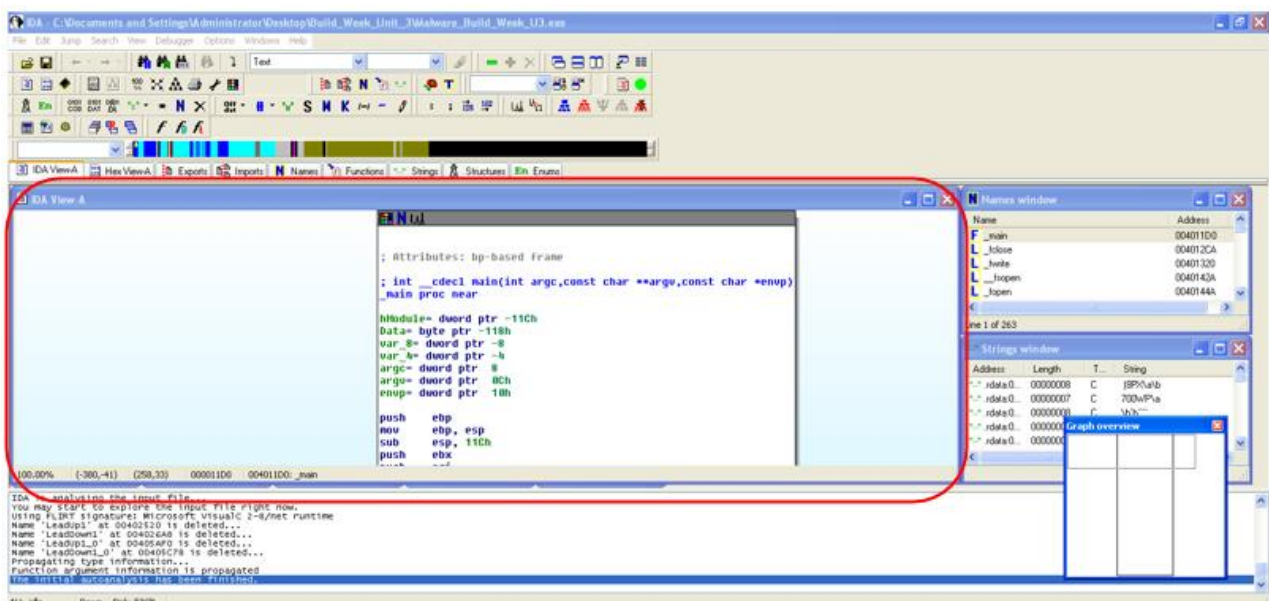
Avviamo IDA, installata sulla macchina Windows XP, e cliccando sull'icona a forma di cartella, navighiamo fino al file relativo al malware da analizzare. Selezioniamolo e clicchiamo su "Open" per aprirlo.



In questo caso è stato identificato un eseguibile in formato PE e un processore Intel 80x86. Possiamo quindi cliccare su "OK" e aprire il file.

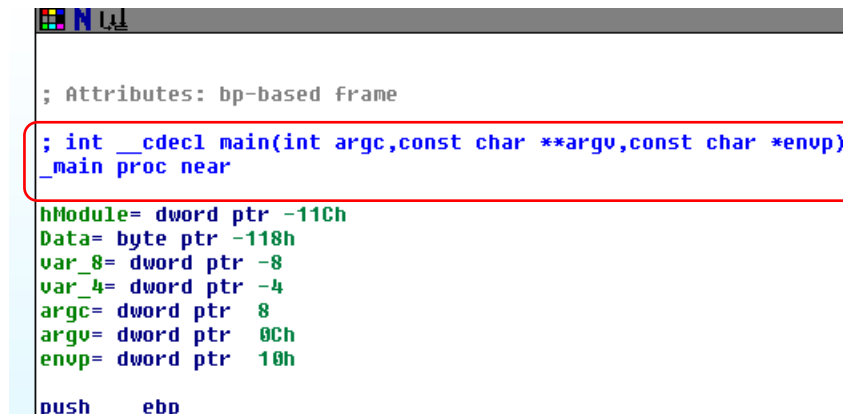


All'apertura viene mostrata l'interfaccia di analisi di IDA. Il pannello centrale che viene visualizzato è chiamato Disassembly Panel, e mostra alcune caratteristiche fondamentali del codice Assembly dell'eseguibile, come le funzioni definite, le variabili locali e i parametri, e i salti effettuati.



Analisi della funzione Main()

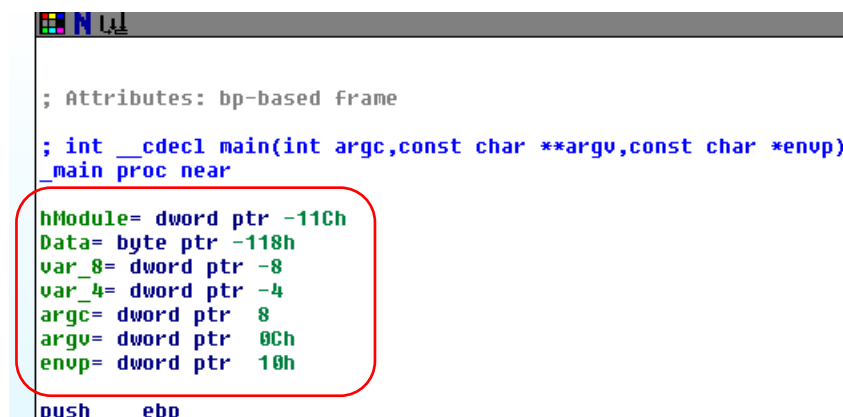
Nello specifico, vediamo che nel primo riquadro mostrato nell'interfaccia è stata identificata la funzione `int Main()`:



The screenshot shows the IDA Pro interface with the function signature of `main` highlighted by a red rectangle. The signature is `int __cdecl main(int argc, const char **argv, const char *envp)`. Below the signature, the local variables are listed: `hModule= dword ptr -11Ch`, `Data= byte ptr -118h`, `var_8= dword ptr -8`, `var_4= dword ptr -4`, `argc= dword ptr 8`, `argv= dword ptr 0Ch`, and `envp= dword ptr 10h`. The assembly instruction `push ebp` is also visible.

```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char *envp)
_main proc near
hModule= dword ptr -11Ch
Data= byte ptr -118h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
push    ebp
```

Subito sotto, sono riportati i parametri e le variabili locali utilizzati dalla funzione:



The screenshot shows the IDA Pro interface with the local variables and parameters of `main` highlighted by a red rectangle. The variables are: `hModule= dword ptr -11Ch`, `Data= byte ptr -118h`, `var_8= dword ptr -8`, `var_4= dword ptr -4`, `argc= dword ptr 8`, `argv= dword ptr 0Ch`, and `envp= dword ptr 10h`. The assembly instruction `push ebp` is also visible.

```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char *envp)
_main proc near
hModule= dword ptr -11Ch
Data= byte ptr -118h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
push    ebp
```

Per i parametri e le variabili la struttura sintattica utilizzata da IDA è la seguente:

<code><Nome parametro/variabile locale> = <formato> <offset rispetto al registro EBP></code>
--

L' "offset rispetto al registro EBP" indica, in formato esadecimale, la distanza in byte dal registro EBP.

Il registro EBP è il registro "Extended Base Pointer" e viene spesso utilizzato per creare un punto di riferimento, un "base pointer," alla base dello stack della funzione

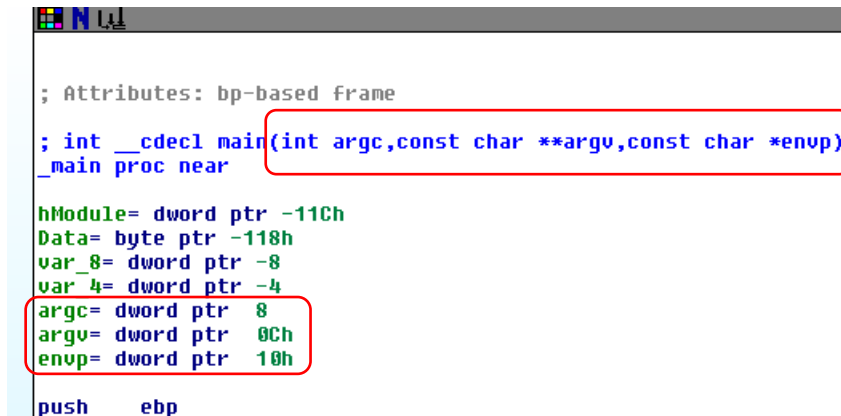
La convenzione adottata per distinguere le variabili locali dai parametri della funzione è la seguente:

- Le variabili locali si trovano ad un offset negativo rispetto ad EBP
- I parametri si trovano ad un offset positivo rispetto ad EBP

Quanti parametri sono passati alla funzione Main()?

Abbiamo detto che i parametri si trovano ad un offset positivo rispetto a EBP. I parametri passati alla funzione Main() sono quindi 3: **argc**, **argv** e **envp**.

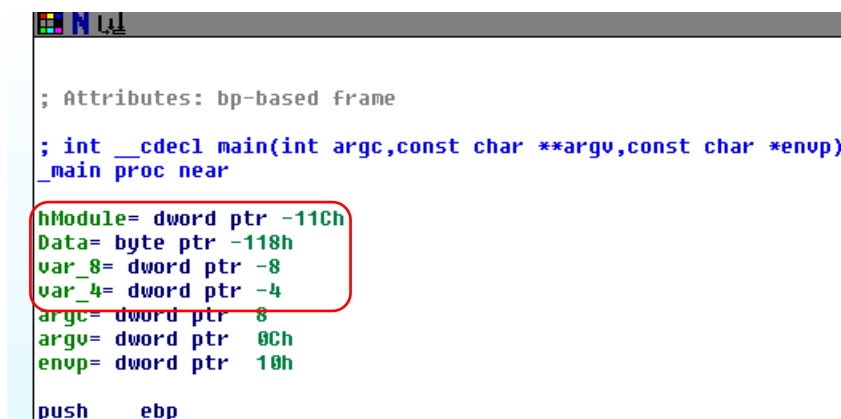
Possiamo vedere che i parametri in questo caso sono anche specificati tra parentesi dopo il nome della funzione.



```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char *envp)
_main proc near
    hModule= dword ptr -11Ch
    Data= byte ptr -118h
    var_8= dword ptr -8
    var_4= dword ptr -4
    argc= dword ptr 8
    argv= dword ptr 0Ch
    envp= dword ptr 10h
    push    ebp
```

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

Le variabili locali si trovano ad un offset negativo rispetto a EBP. Le variabili dichiarate all'interno della funzione Main() sono 4: **hModule**, **Data**, **var_8** e **var_4**.



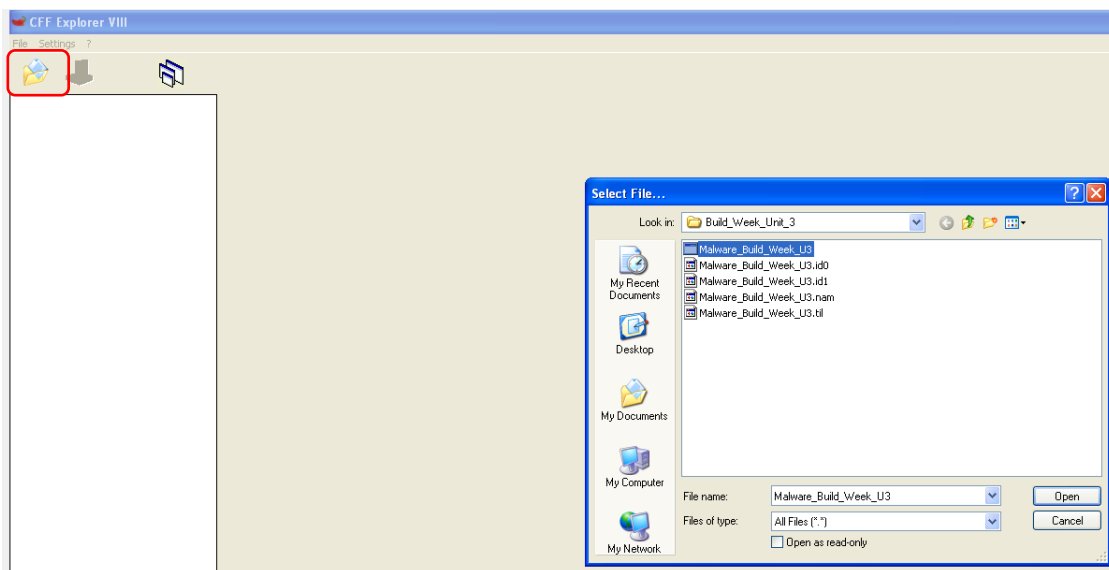
```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char *envp)
_main proc near
    hModule= dword ptr -11Ch
    Data= byte ptr -118h
    var_8= dword ptr -8
    var_4= dword ptr -4
    argc= dword ptr 8
    argv= dword ptr 0Ch
    envp= dword ptr 10h
    push    ebp
```

Analisi delle sezioni del file eseguibile

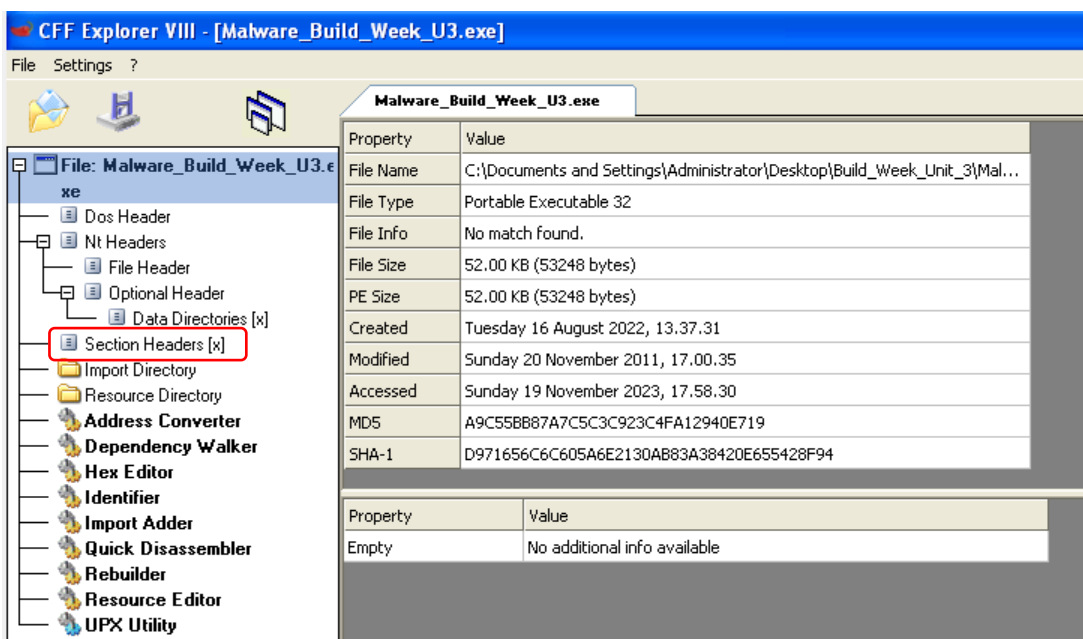
Analizziamo adesso le sezioni del file eseguibile. Per questa operazione possiamo utilizzare un altro tool: CFF Explorer.

CFF Explorer è uno strumento avanzato di analisi e disassemblaggio progettato per esaminare e comprendere file binari, in particolare file eseguibili come quelli con estensioni .exe, .dll e altri.

Avviamo quindi CFF Explorer, installato anch'esso sulla macchina Windows XP, e cliccando sull'icona a forma di cartella, navighiamo fino al file relativo al malware da analizzare. Selezioniamolo e clicchiamo su "Open" per aprirlo.



Dall'interfaccia principale, selezioniamo la voce "Section Headers [x]" nel pannello a sinistra per visualizzare le sezioni del file eseguibile.



Viene mostrata una tabella nelle cui righe sono presenti le sezioni presenti nel malware. Il nome della sezione è riportato nella colonna "Name", ci sono poi altre colonne che riportano, per ciascuna sezione, informazioni aggiuntive.

CFF Explorer VIII - [Malware_Build_Week_U3.exe]

File Settings ?

Malware_Build_Week_U3.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	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

File: Malware_Build_Week_U3.exe

- File Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [x]
- Section Headers [x]
- Import Directory

Selezionando una sezione, CFF Explorer mostra nei riquadri sottostanti le informazioni di dettaglio di quella sezione.

Malware_Build_Week_U3.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
000001D8	000001E0	000001E4	000001E8	000001EC	000001F0	000001F4	000001F8	000001FA	00000000
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

This section contains:

Code Entry Point: 00001487

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	55	8B	EC	51	6A	00	8D	45	FC	50	6A	00	68	3F	00	0F	U iQj. EuPj.h?.
00000010	00	6A	00	6A	00	6A	00	68	54	80	40	00	68	02	00	00	.j.j.j.hT!@.h...
00000020	80	FF	15	04	70	40	00	85	C0	74	07	B8	01	00	00	00	!y!lp@. At
00000030	EB	49	8B	4D	0C	51	8B	55	08	52	6A	01	6A	00	68	4C	è! M Q U Rj j.hL
00000040	80	40	00	8B	45	FC	50	FF	15	00	70	40	00	85	C0	74	!@. EuPy .p@. At
00000050	11	8B	4D	FC	51	FF	15	2C	70	40	00	B8	01	00	00	00	! MuQy .p@. ...
00000060	EB	19	68	48	80	40	00	E8	2D	02	00	00	83	C4	04	8B	è hH!@.è-
00000070	55	FC	52	FF	15	2C	70	40	00	33	C0	8B	E5	5D	C3	CC	UuRy .p@.3A á]A!
00000080	55	8B	EC	83	EC	18	56	57	C7	45	EC	00	00	00	00	C7	U i i VWÇEi...Ç
00000090	45	E8	00	00	00	C7	45	F8	00	00	00	00	C7	45	F0		Eè...ÇEè...ÇEè
000000A0	00	00	00	00	C7	45	F4	00	00	00	00	83	7D	08	00	75	...ÇEó... } u

Quali sezioni sono presenti all'interno del file eseguibile?

Vediamo da CFF Explorer che le sezioni all'interno del file eseguibile sono:

.text

la sezione ".text" contiene il codice eseguibile del programma. In questa sezione è riportato l'indirizzo dell'entry point dell'eseguibile, ovvero la prima istruzione che viene eseguita dalla CPU.

CFF Explorer indica che l'entry point dell'eseguibile si trova all'indirizzo 00001487.

This section contains:

Code Entry Point: 00001487

.rdata

La sezione ".rdata" contiene dati di sola lettura, ossia dati che il programma può leggere ma non modificare durante l'esecuzione.

CFF Explorer per questa sezione indica che:

- la sezione .rdata inizia all'indirizzo 00007000
- l'indirizzo di memoria della Import Directory (la directory delle librerie e funzioni richieste dal programma durante l'esecuzione) è 000074EC
- l'indirizzo di memoria della Address Table Directory (tabella che contiene gli indirizzi delle funzioni specifiche all'interno delle librerie dinamiche (DLL) che un programma importa durante l'esecuzione) è 00007000, lo stesso indirizzo a cui inizia la sezione.

This section contains:

Data: 00007000

Import Directory: 000074EC

Import Address Table Directory: 00007000

.data

La sezione ".data" contiene dati variabili e inizializzati che il programma può leggere e modificare durante l'esecuzione. In particolare, contiene le variabili globali definite nel codice eseguibile.

Non vengono mostrate informazioni descrittive aggiuntive da CFF Explorer per questa sezione.

.rsrc

La sezione ".rsrc" contiene risorse, che sono dati incorporati nel programma e utilizzati per scopi specifici durante l'esecuzione dell'applicazione.

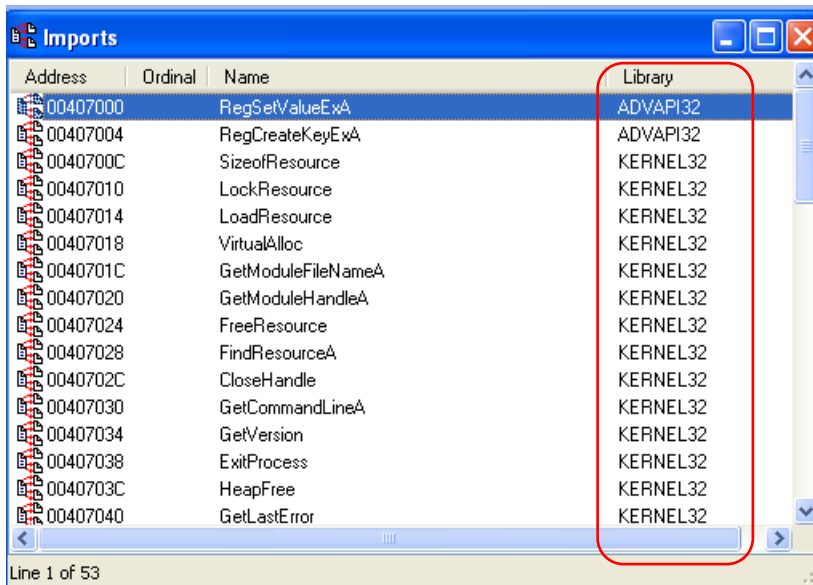
CFF Explorer indica che la Resource Directory (directory delle risorse) si trova all'indirizzo di memoria 0000C000.

This section contains:

Resource Directory: 0000C000

Analisi delle librerie importate dal malware

CFF Explorer permette di analizzare anche le librerie e le funzioni importate, ma a questo scopo è possibile utilizzare anche IDA: il dettaglio delle funzioni importate con indirizzo della funzione, nome della funzione e libreria, sono visualizzate nella scheda "Imports".



Quali librerie importa il Malware?

I risultati dell'analisi con CFF Explorer e IDA mostrano che il malware importa le librerie:

ADVAPI32

KERNEL32

Ipotesi su funzionalità del malware

ADVAPI32.dll contiene funzioni relative ai servizi avanzati di Windows. Questa libreria è coinvolta in operazioni di gestione di servizi, sicurezza, gestione degli account utente e altri servizi avanzati del sistema operativo. Ad esempio, contiene funzioni per manipolare il Registro di sistema, autenticazione e autorizzazione, e crittografia.

Ipotesi: il malware potrebbe utilizzare questa libreria per modificare chiavi di registro in modo da ottenere la persistenza, ovvero per fare in modo che il malware venga avviato all'avvio del sistema.

Le funzioni richiamate all'interno di questa libreria sono infatti:

RegCreateKeyExA, che crea, o apre se già esiste, la chiave del Registro di sistema specificata, e RegSetValueExA, che imposta i dati e il tipo di un valore specificato in una chiave del Registro di sistema.

Name	Library
RegSetValueExA	ADVAPI32
RegCreateKeyExA	ADVAPI32

KERNEL32.dll è una delle librerie di sistema principali in ambienti Windows. Essa contiene funzioni di basso livello coinvolte nell'interazione diretta con l'hardware e la gestione dei processi. Tra le funzioni incluse ci sono l'allocazione di memoria, la gestione dei file, la creazione di processi e altro ancora. È essenziale per l'esecuzione delle applicazioni Windows.

Ipotesi: il malware potrebbe essere un dropper, ovvero un malware che contiene al suo interno un altro malware. Generalmente, il malware incluso nel dropper è contenuto nella sezione risorse (in questo caso «.rsrc ») dell'eseguibile.

Nell'elenco delle funzioni richiamate all'interno della libreria KERNEL32 sono comprese infatti le funzioni evidenziate di seguito

Name	Library
LCMapStringW	KERNEL32
GetStringTypeA	KERNEL32
GetStringTypeW	KERNEL32
SizeofResource	KERNEL32
LockResource	KERNEL32
LoadResource	KERNEL32
VirtualAlloc	KERNEL32
GetModuleFileNameA	KERNEL32
GetModuleHandleA	KERNEL32
FreeResource	KERNEL32
FindResourceA	KERNEL32
CloseHandle	KERNEL32
GetCommandLineA	KERNEL32
GetVersion	KERNEL32

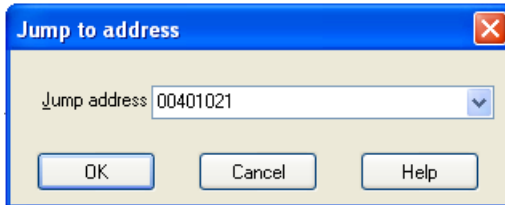
che sono le funzioni tipicamente utilizzate dai dropper per estrarre il malware contenuto nella sezione risorse.

Inoltre, utilizzando la coppia di funzioni `CreateFileA` e `WriteFile`, il dropper potrebbe salvare sul disco il malware al suo interno.

Name	Library
SetEndOfFile	KERNEL32
SetFilePointer	KERNEL32
SetHandleCount	KERNEL32
SetStdHandle	KERNEL32
SizeofResource	KERNEL32
TerminateProcess	KERNEL32
UnhandledExceptionFilter	KERNEL32
VirtualAlloc	KERNEL32
VirtualFree	KERNEL32
WideCharToMultiByte	KERNEL32
WriteFile	KERNEL32
CloseHandle	KERNEL32
CreateFileA	KERNEL32
ExitProcess	KERNEL32
FindResourceA	KERNEL32

Malware Analysis

Cliccando con la barra spaziatrice del Disassembly Panel di IDA, visualizziamo la versione testuale del codice. Da questa visualizzazione, clicchiamo col tasto destro del mouse e selezioniamo poi l'opzione "Jump to address" per saltare all'indirizzo di memoria 00401021.



Vediamo che a questo indirizzo di memoria viene chiamata la funzione RegCreateKeyExA

```
.text:00401011      push    0           ; dwOptions
.text:00401013      push    0           ; lpClass
.text:00401015      push    0           ; Reserved
.text:00401017      push    offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040101C      push    0000002h     ; hKey
.text:00401021      call    ds:RegCreateKeyExA
.text:00401027      test    eax, eax
.text:00401029      jz      short loc_401032
.text:0040102B      mov     eax, 1
.text:00401030      jmp     short loc_40107B
.text:00401032      ; -----
.text:00401032      loc_401032:          ; CODE XREF: sub_401000+29↑j
.text:00401032      mov     ecx, [ebp+cbData]
.text:00401035      push    ecx          ; cbData
.text:00401036      mov     edx, [ebp+lpData]
.text:00401039      push    edx          ; lpData
.text:0040103A      push    1            ; dwType
.text:0040103C      push    0            ; Reserved
.text:0040103E      push    offset ValueName ; "GinaDLL"
```

Dettagli della funzione alla locazione di memoria 00401021

Scopo

La funzione `RegCreateKeyExA` ha lo scopo di creare una specifica chiave del Registro di sistema o, se a chiave già esiste, di aprirla.

Parametri

La funzione richiede i parametri mostrati nell'immagine di seguito

```
LSTATUS RegCreateKeyExA(  
    [in]          HKEY          hKey,  
    [in]          LPCSTR       lpSubKey,  
    [in]          DWORD         Reserved,  
    [in, optional] LPSTR       lpClass,  
    [in]          DWORD         dwOptions,  
    [in]          REGSAM        samDesired,  
    [in, optional] const LPSECURITY_ATTRIBUTES lpSecurityAttributes,  
    [out]          PHKEY        phkResult,  
    [out, optional] LPDWORD     lpdwDisposition  
);
```

Nel codice Assembly mostrato da IDA vediamo che questi stessi parametri sono passati alla funzione con l'istruzione `push`.

```
.text:00401003      push     ecx  
.text:00401004      push     0                ; lpdwDisposition  
.text:00401006      lea     eax, [ebp+hObject]  
.text:00401009      push     eax              ; phkResult  
.text:0040100A      push     0                ; lpSecurityAttributes  
.text:0040100C      push     0F003Fh          ; samDesired  
.text:00401011      push     0                ; dwOptions  
.text:00401013      push     0                ; lpClass  
.text:00401015      push     0                ; Reserved  
.text:00401017      push     offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...  
.text:0040101C      push     80000002h        ; hKey  
.text:00401021      call    ds:RegCreateKeyExA
```

Locazione di memoria 00401017

Alla locazione di memoria 00401017 vediamo che con l'istruzione push viene passato il parametro lpSubKey della funzione RegCreateKeyExA, che specifica la sottochiave di registro che la funzione va a creare o modificare.

```
.text:00401003      push     ecx
.text:00401004      push     0                ; lpdwDisposition
.text:00401006      lea      eax, [ebp+hObject]
.text:00401009      push     eax                ; phkResult
.text:0040100A      push     0                ; lpSecurityAttributes
.text:0040100C      push     0F003Fh           ; samDesired
.text:00401011      push     0                ; dwOptions
.text:00401013      push     0                ; lpClass
.text:00401015      push     0                ; Reserved
.text:00401017      push     offset SubKey     ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"..
.text:0040101C      push     80000002h         ; hKey
.text:00401021      call     ds:RegCreateKeyExA
```

Una «subkey», o sottochiave di registro, è una cartella del registro contenuta nella cartella principale di una chiave di registro.

Il valore passato al parametro indica che la funzione sta andando a creare o modificare una chiave di registro della macrocategoria HKLM (HKEY_LOCAL_MACHINE), dove sono contenuti i record e le configurazioni della macchina.

In particolare, vediamo che la funzione modifica una chiave nella cartella di registro HKLM\SOFTWARE\\Microsoft\\Windows\\CurrentVersion, che contiene molte informazioni di configurazione relative alla versione corrente del sistema operativo Windows installato.

La funzione RegCreateKeyExA è una delle funzioni spesso utilizzate dai malware per modificare chiavi di registro in modo da ottenere persistenza, ovvero fare in modo che il malware venga avviato automaticamente ad ogni avvio del sistema.

Locazioni di memoria 00401027 - 00401029

Alle locazioni di memoria 00401027 e 00401029 abbiamo un codice Assembly che riproduce un costrutto If in linguaggio C. Vediamole in dettaglio:

```
.text:00401027          test     eax, eax
.text:00401029          jz       short loc_401032
```

La prima istruzione, **test eax, eax**, esegue un AND logico del valore del registro eax con se stesso.

Diversamente dall'operando AND, l'operatore test non modifica il valore del registro eax, ma modifica il flag ZF (Zero Flag) del registro EFLAGS, che viene settato ad 1 se e solo se il risultato dell'AND è 0.

Trattandosi dell'AND di un valore con se stesso, il risultato è 0 se e solo se il valore è 0. Questa operazione infatti viene spesso utilizzata per testare se un valore è uguale a 0.

La seconda istruzione, **jz short loc_401032**, esegue un salto condizionale, ovvero un salto ad un indirizzo di memoria, in questo caso loc_401032, se una certa condizione è soddisfatta. L'operatore jz, nello specifico, esegue il salto se il flag ZF (Zero Flag) del registro EFLAGS = 1.

- Se il valore di eax è uguale a 0, viene quindi eseguito il salto (anche evidenziato da IDA con la freccia tratteggiata a lato) alla locazione di memoria 00401032 e viene eseguita l'istruzione `mov ecx, [ebp+cbData]`, che sposta il contenuto della memoria all'indirizzo specificato da `[ebp+cbData]` nel registro ecx.:

```
.text:00401029          jz       short loc_401032
.text:0040102B          mov     eax, 1
.text:00401030          jmp     short loc_40107B
.text:00401032 ; -----
.text:00401032          loc_401032: ; CODE XREF: sub_401000+29↑j
.text:00401032          mov     ecx, [ebp+cbData]
```

- Se, al contrario, il valore di eax è diverso da 0, viene eseguita l'istruzione successiva, `mov eax, 1`, che imposta il registro eax con il valore 1:

```
.text:00401027          test     eax, eax
.text:00401029          jz       short loc_401032
.text:0040102B          mov     eax, 1
```

Possiamo tradurre questo codice in linguaggio C come:

```
int a;           // eax
int c;           // ecx
int d;           // [ebp+cbData]

//codice intermedio che assegna un valore alle variabili sopra dichiarate

if (a == 0) {
    c = d;
} else {
    a = 1;
}
```

Locazione di memoria 00401047

Alla locazione di memoria 00101047 viene chiamata la funzione `RegSetValueExA`. Questa funzione permette di impostare i dati e il tipo di un valore specificato in una chiave del Registro di sistema.

```
.text:00401032      mov     ecx, [ebp+cbData]
.text:00401035      push    ecx                ; cbData
.text:00401036      mov     edx, [ebp+lpData]
.text:00401039      push    edx                ; lpData
.text:0040103A      push    1                  ; dwType
.text:0040103C      push    0                  ; Reserved
.text:0040103E      push    offset ValueName ; "GinaDLL"
.text:00401043      mov     eax, [ebp+hObject]
.text:00401046      push    eax                ; hKey
.text:00401047      call   ds:RegSetValueExA
```

Come vediamo dall'immagine di seguito, `lpValueName` è un parametro della funzione che specifica il nome del valore da impostare per la chiave di Registro che la funzione sta creando/modificando.

```
LSTATUS RegSetValueExA(
    [in] HKEY hKey,
    [in, optional] LPCSTR lpValueName,
    DWORD Reserved,
    [in] DWORD dwType,
    [in] const BYTE *lpData,
    [in] DWORD cbData
);
```

Vediamo dallo screenshot che il valore del parametro `ValueName` è "GinaDLL":

```
:0040103E      push    offset ValueName ; "GinaDLL"
```

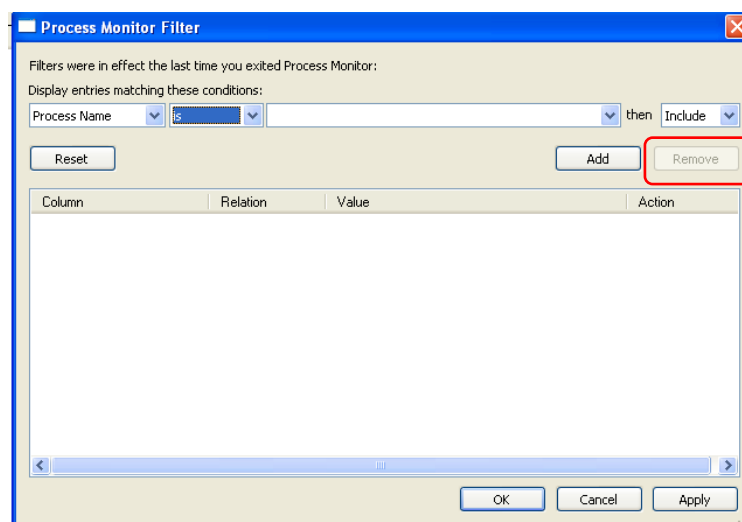

Analisi Dinamica

Per l'analisi dinamica del malware utilizziamo Process Monitor. Process Monitor è un'utilità di sistema fornita da Microsoft che monitora e registra l'attività del sistema operativo Windows in tempo reale. Questa applicazione fornisce informazioni dettagliate sulle operazioni eseguite dai processi in esecuzione, inclusi i processi di sistema e le applicazioni utente.

Avviamo Process Monitor, anch'esso presente sulla macchina virtuale Windows XP.

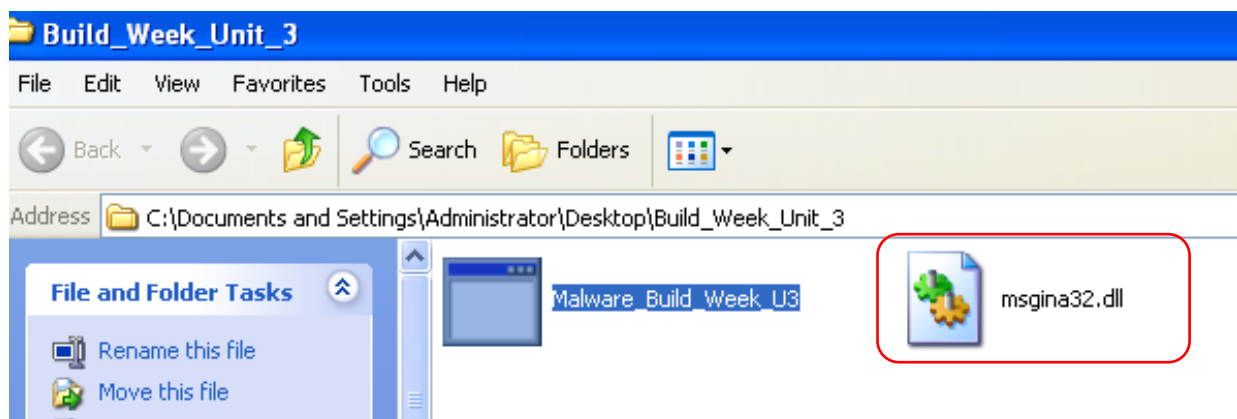
Vediamo che prima dell'avvio ci viene prospettata una finestra dalla quale è possibile impostare vari filtri, ad esempio sui processi da monitorare, sull'architettura del sistema e così via.

Selezioniamo i filtro eventualmente presenti nell'area centrale e clicchiamo su "Remove" per rimuoverli, e quindi su "Apply".



In questo modo Process Monitor viene avviato senza alcun filtro applicato.

Avviamo anche il malware. Notiamo che nella cartella in cui è contenuto il malware è stato creato il file **msgina32.dll**.



In fase di analisi statica con IDA abbiamo visto che il malware utilizza le funzioni `RegCreateKeyExA`, e `RegSetValueExA`, per creare/modificare una chiave del Registro di sistema.

Abbiamo anche visto che `RegCreateKeyExA` va a creare/modificare una sottochiave di registro nella cartella di registro `Software\\Microsoft\\Windows\\CurrentVersion\\`, che contiene molte informazioni di configurazione relative alla versione corrente del sistema operativo Windows installato.

```
.text:00401003      push     ecx
.text:00401004      push     0                ; lpdwDisposition
.text:00401006      lea      eax, [ebp+hObject]
.text:00401009      push     eax                ; phkResult
.text:0040100A      push     0                ; lpSecurityAttributes
.text:0040100C      push     0F003Fh           ; samDesired
.text:00401011      push     0                ; dwOptions
.text:00401013      push     0                ; lpClass
.text:00401015      push     0                ; Reserved
.text:00401017      push     offset SubKey     ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"..
.text:0040101C      push     80000002h         ; hKey
.text:00401021      call     ds:RegCreateKeyExA
```

Inoltre, l'analisi dei parametri passati per la chiamata alla funzione `RegSetValueExA` ci ha mostrato, inoltre, che il nome del valore della chiave creata/modificata è "GinaDLL".

```
.text:00401032      mov      ecx, [ebp+cbData]
.text:00401035      push     ecx                ; cbData
.text:00401036      mov      edx, [ebp+lpData]
.text:00401039      push     edx                ; lpData
.text:0040103A      push     1                ; dwType
.text:0040103C      push     0                ; Reserved
.text:0040103E      push     offset ValueName  ; "GinaDLL"
.text:00401043      mov      eax, [ebp+hObject]
.text:00401046      push     eax                ; hKey
.text:00401047      call     ds:RegSetValueExA
```

Per quanto riguarda l'analisi statica delle attività del malware sul file system, tra le funzioni utilizzate dal malware abbiamo trovato `CreateFileA` e `WriteFile`.

WideCharToMultiByte	KERNEL32
WriteFile	KERNEL32
CloseHandle	KERNEL32
CreateFileA	KERNEL32
ExitProcess	KERNEL32

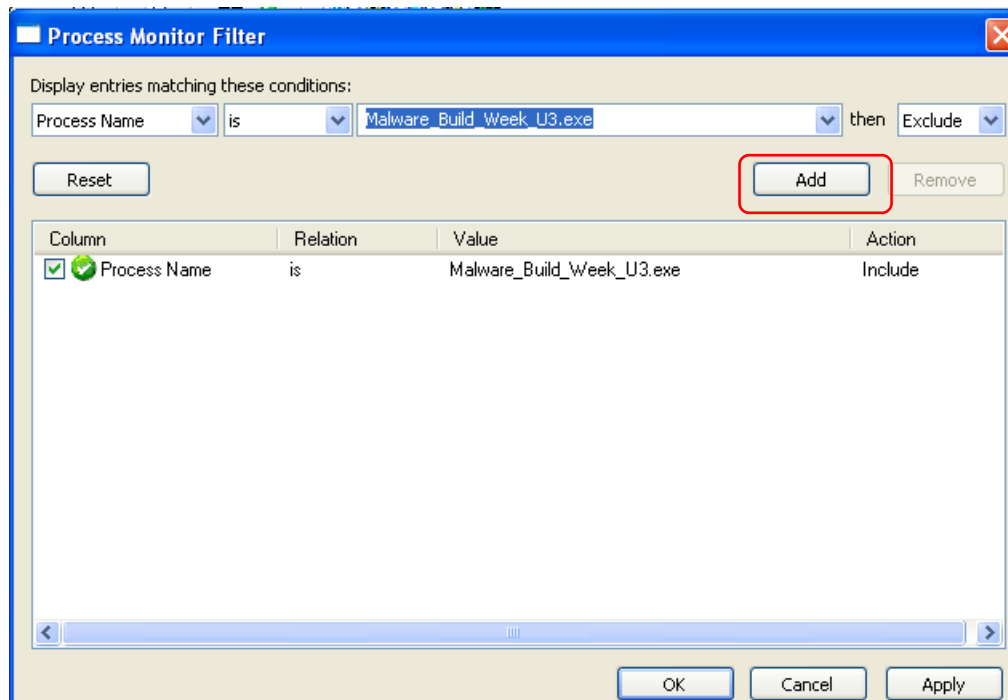
I dati raccolti fino ad adesso ci mostrano che il malware può aver utilizzato le funzioni `CreateFileA` e `WriteFile` per creare il file `msgina32.dll`.

La chiave di registro creata/modificata dal malware, `GinaDLL`, è associata all'interfaccia di accesso (GINA, Graphical Identification and Authentication) e può essere utilizzata per estendere o sostituire la procedura di accesso di Windows.

Il file `msgina32.dll` potrebbe essere in qualche modo legato alla chiave di registro `GinaDLL` creata/modificata dal malware, e potrebbe essere utilizzato per attività malevole legate agli accessi, come ottenere le credenziali degli utenti per eseguire azioni non autorizzate.

Analisi risultati di Process Monitor

Impostiamo adesso il filtro di Process Monitor per fare in modo che vengano visualizzate solo le azioni del malware in analisi. Clicchiamo poi su "Add" e "Apply".



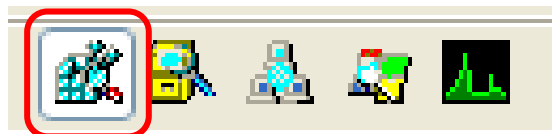
I primi risultati ci mostrano una serie di operazioni mirate alla mappatura dei file di sistema e delle directory, in cui sono presenti operazioni come `CreateFile`, `ReadFile`, `QueryStandardInformationFile`, `QueryDirectory` e `CloseFile` mirate rispettivamente, all'apertura, lettura, richiesta informazioni e chiusura di file e directory.

1592	CreateFile	C:\WINDOWS\Prefetch\MALWARE_BUILD_WEEK_U3.EXE-0E171D0F.pf	SUCCESS
1592	QueryStandardInformationFile	C:\WINDOWS\Prefetch\MALWARE_BUILD_WEEK_U3.EXE-0E171D0F.pf	SUCCESS
1592	ReadFile	C:\WINDOWS\Prefetch\MALWARE_BUILD_WEEK_U3.EXE-0E171D0F.pf	SUCCESS
1592	CloseFile	C:\WINDOWS\Prefetch\MALWARE_BUILD_WEEK_U3.EXE-0E171D0F.pf	SUCCESS
1592	CreateFile	C:	SUCCESS
1592	QueryInformationVolume	C:	SUCCESS
1592	FileSystemControl	C:	SUCCESS
1592	CreateFile	C:\	SUCCESS
1592	QueryDirectory	C:\	SUCCESS
1592	QueryDirectory	C:\	NO MORE FILES
1592	CloseFile	C:\	SUCCESS
1592	IRP_MJ_CLOSE	C:\	SUCCESS
1592	CreateFile	C:\DOCUMENTS AND SETTINGS	SUCCESS
1592	QueryDirectory	C:\Documents and Settings	SUCCESS
1592	QueryDirectory	C:\Documents and Settings	NO MORE FILES
1592	CloseFile	C:\Documents and Settings	SUCCESS
1592	IRP_MJ_CLOSE	C:\Documents and Settings	SUCCESS
1592	CreateFile	C:\Documents and Settings\ADMINISTRATOR	SUCCESS
1592	QueryDirectory	C:\Documents and Settings\Administrator	SUCCESS
1592	QueryDirectory	C:\Documents and Settings\Administrator	NO MORE FILES
1592	CloseFile	C:\Documents and Settings\Administrator	SUCCESS
1592	IRP_MJ_CLOSE	C:\Documents and Settings\Administrator	SUCCESS
1592	CreateFile	C:\Documents and Settings\Administrator\Desktop	SUCCESS
1592	QueryDirectory	C:\Documents and Settings\Administrator\Desktop	SUCCESS
1592	QueryDirectory	C:\Documents and Settings\Administrator\Desktop	NO MORE FILES
1592	CloseFile	C:\Documents and Settings\Administrator\Desktop	SUCCESS
1592	IRP_MJ_CLOSE	C:\Documents and Settings\Administrator\Desktop	SUCCESS
1592	CreateFile	C:\Documents and Settings\Administrator\Desktop\BUILD_WEEK_UNIT_3	SUCCESS
1592	QueryDirectory	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
1592	QueryDirectory	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	NO MORE FILES
1592	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS

Scorrendo più in basso, abbiamo una serie di operazioni RegOpenKey, RegQueryValue e RegCloseKey mirate, rispettivamente, all'apertura, richiesta informazioni e chiusura di chiavi di registro di sistema.

1592	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Malware_Build_Week_U3.exe	NAME NOT FOUND
1592	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
1592	FileSystemControl	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
1592	QueryOpen	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe.Local	NAME NOT FOUND
1592	Load Image	C:\WINDOWS\system32\kernel32.dll	SUCCESS
1592	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS
1592	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS
1592	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS
1592	Load Image	C:\WINDOWS\system32\advapi32.dll	SUCCESS
1592	Load Image	C:\WINDOWS\system32\rpcrt4.dll	SUCCESS
1592	Load Image	C:\WINDOWS\system32\secur32.dll	SUCCESS
1592	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS
1592	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS
1592	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS
1592	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Secur32.dll	NAME NOT FOUND
1592	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\RPCRT4.dll	NAME NOT FOUND
1592	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ADVAPI32.dll	NAME NOT FOUND
1592	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS
1592	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS
1592	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSUserEnabled	SUCCESS
1592	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS
1592	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
1592	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTrack	NAME NOT FOUND
1592	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
1592	RegOpenKey	HKLM	SUCCESS
1592	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics	NAME NOT FOUND
1592	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll	NAME NOT FOUND
1592	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll	NAME NOT FOUND
1592	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS

Filtriamo ora le attività sul registro di sistema lasciando attivo nella barra in alto di Process Monitor solo il relativo pulsante.



Come notato dall'analisi sommaria effettuata in precedenza, la maggior parte delle operazioni effettuate dal malware sono mirate all'ottenimento di informazioni su chiavi di registro.

In fondo alla lista notiamo però le operazioni RegCreateKey e RegSetValue:

1592	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
1592	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL

Facendo doppio click su `RegCreateKey` vediamo che il malware ha modificato la chiave

`HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon`




Questa chiave contiene informazioni relative alla procedura di accesso e autenticazione degli utenti.

Date:	11/19/2023 10:57:57 PM
Thread:	2968
Class:	Registry
Operation:	RegCreateKey
Result:	SUCCESS
Path:	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
Duration:	0.0000120
Desired Access:	All Access

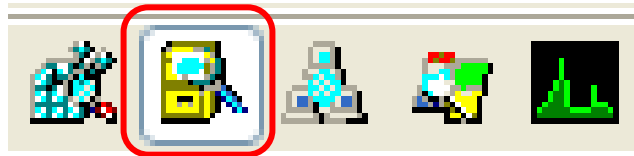
Facendo doppio click su `RegSetValue` vediamo che il malware ha modificato la chiave inserendo un nuovo valore, che corrisponde al percorso al file `msgina32.dll` (creato nella stessa cartella del malware, come visto in precedenza), e come nome del valore `GinaDLL`, come visto nell'analisi statica.

Date:	11/19/2023 10:57:57 PM
Thread:	2968
Class:	Registry
Operation:	RegSetValue
Result:	SUCCESS
Path:	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
Duration:	0.0000084
Type:	REG_SZ
Length:	520
Data:	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll

Aprendo il registro di sistema con il comando `regedit` troviamo infatti il nuovo valore inserito:

	<code>forceunlocklogon</code>	REG_DWORD	0x00000000 (0)
	<code>GinaDLL</code>	REG_SZ	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll...
	<code>hibernationPreviouslyEnabled</code>	REG_DWORD	0x00000001 (1)

Filtriamo ora le attività sul file system lasciando attivo nella barra in alto di Process Monitor solo il relativo pulsante.



Oltre alle attività di mappatura e richiesta informazioni sui file e le directory individuate nell'analisi sommaria effettuata in precedenza, vediamo un'attività di `CreateFile` nel path della directory del malware:

`C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll`

In questo caso l'operazione `CreateFile` non è servita ad aprire file esistenti, come nei casi evidenziati in precedenza ma a creare il file `msgina32.dll`, che prima dell'avvio del malware non esisteva.

2	Load Image	C:\WINDOWS\system32\secur32.dll
2	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
2	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll

Sotto troviamo infatti anche attività di scrittura e chiusura di questo file.

2	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
2	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
2	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
2	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll

Queste attività su File System hanno modificato il contenuto della cartella nella quale è contenuto il malware.

Funzionamento del malware

I risultati dell'analisi statica e dinamica hanno confermato che il malware ha creato il file msgina32.dll nel percorso:

```
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\
```

Inoltre ha creato la chiave di registro "GinaDLL" con valore

```
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
```

nella cartella di registro

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
```

Questo conferma che il file msgina32.dll è effettivamente legato alla chiave di registro GinaDLL.

Come già detto, la chiave GinaDLL è associata all'interfaccia di accesso (GINA, Graphical Identification and Authentication) e può essere utilizzata per estendere o sostituire la procedura di accesso di Windows.

Il file msgina32.dll impostato nella chiave di registro "GinaDLL" se specificamente configurato, potrebbe essere utilizzato dal malware per ottenere un accesso automatico personalizzato, oppure per ottenere le credenziali degli utenti per eseguire azioni non autorizzate.

In conclusione, la principale attività del malware analizzato è la modifica delle impostazioni di configurazione dell'interfaccia di accesso al sistema.