

Progetto Modulo 6

La traccia

Malware Analysis

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

Analisi statica

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.

La traccia

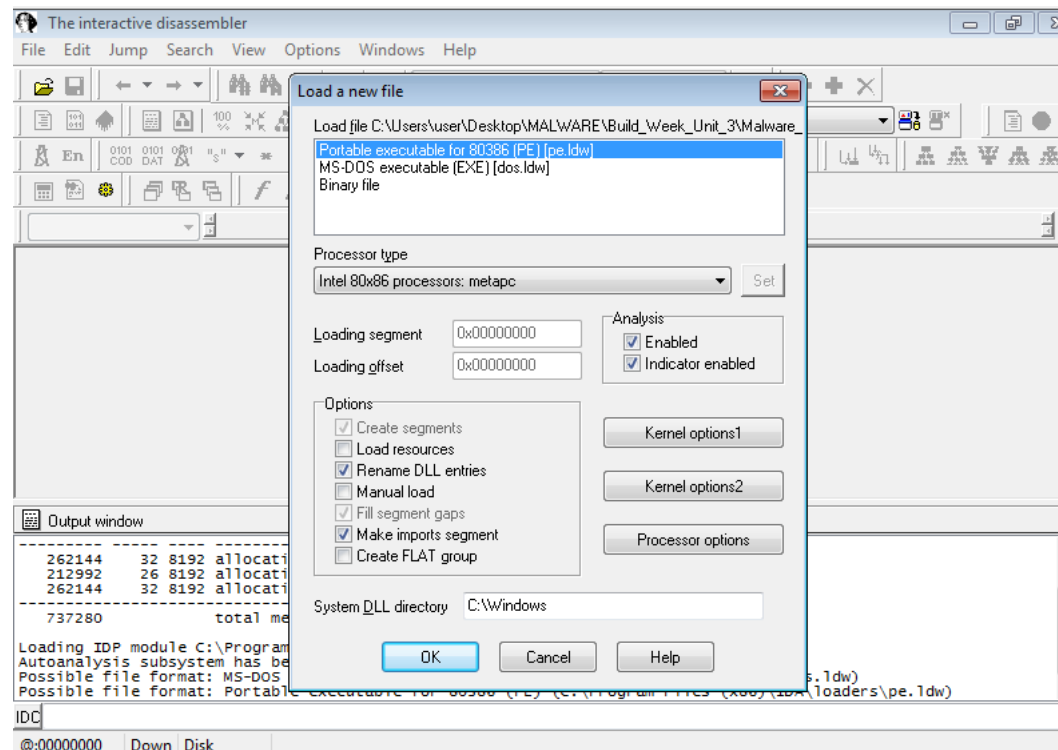
Malware Analysis

Con riferimento al Malware in 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**.
- ❑ Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C.
- ❑ Valutate ora la chiamata alla locazione **00401047**, qual è il valore del parametro «ValueName»?

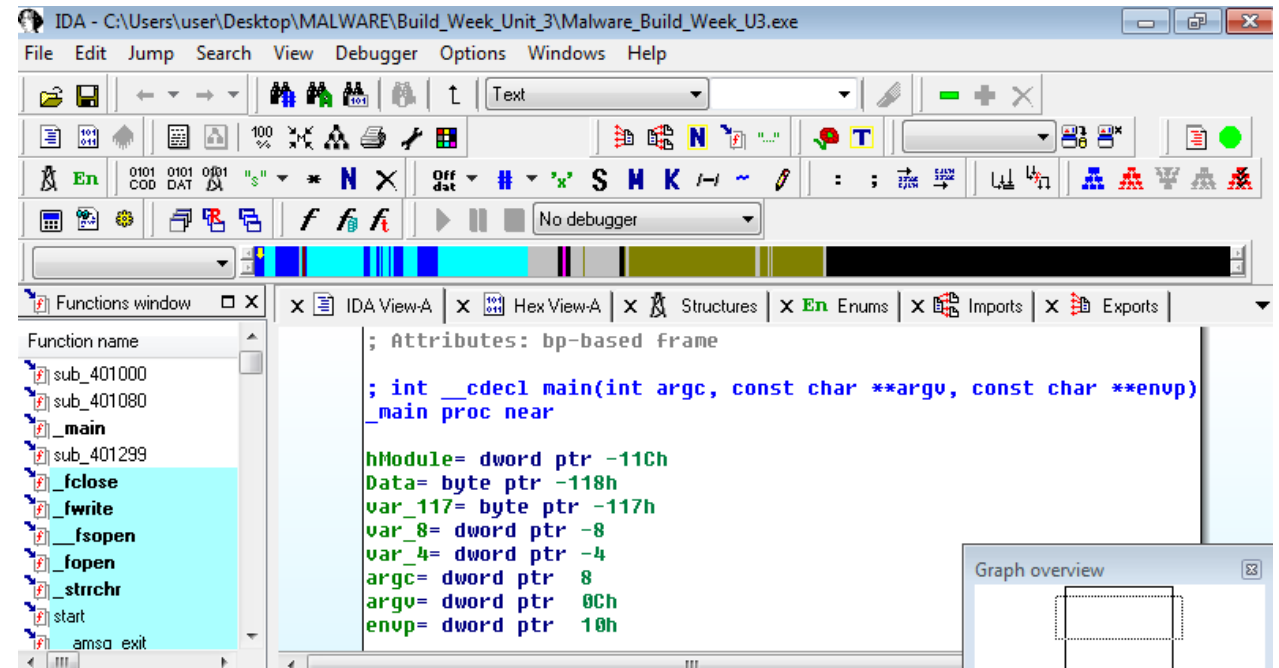
Analisi Statica

Per svolgere l'analisi statica del Malware ci serviamo del software IDA, dalla schermata iniziale apriamo il file.



Analisi Statica

A destra possiamo osservare dalla schermata grafica del Malware in analisi quali sono le variabili e i parametri passati alla funzione Main(), possiamo distinguerli in base all'offset rispetto al puntatore dello stack che nel caso delle variabili è negativo (ad esempio hModule = -11Ch), mentre nel caso dei parametri è positivo.



```
IDA - C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\Malware_Build_Week_U3.exe
File Edit Jump Search View Debugger Options Windows Help

; 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_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
```

Analisi Statica

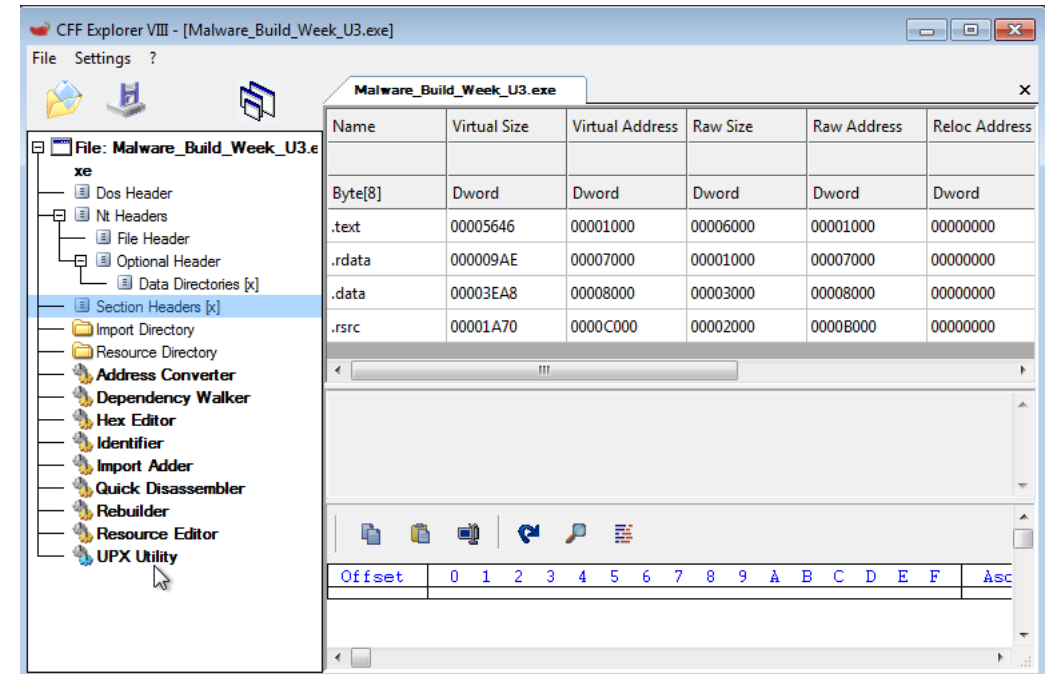
Le sezioni del MW sono queste rappresentate a dx: .text , .r.data ,.data , .rsrc .

.text: Questa sezione contiene il codice eseguibile del programma, ovvero le istruzioni in linguaggio macchina che vengono eseguite dalla CPU. È la parte del programma che contiene le istruzioni per svolgere le operazioni desiderate, come le funzioni e le istruzioni di controllo del flusso.

.rdata (Read-Only Data): Questa sezione contiene dati che sono accessibili *in sola lettura* durante l'esecuzione del programma. Solitamente include costanti, stringhe di testo e altri dati che non vengono modificati durante l'esecuzione del programma.

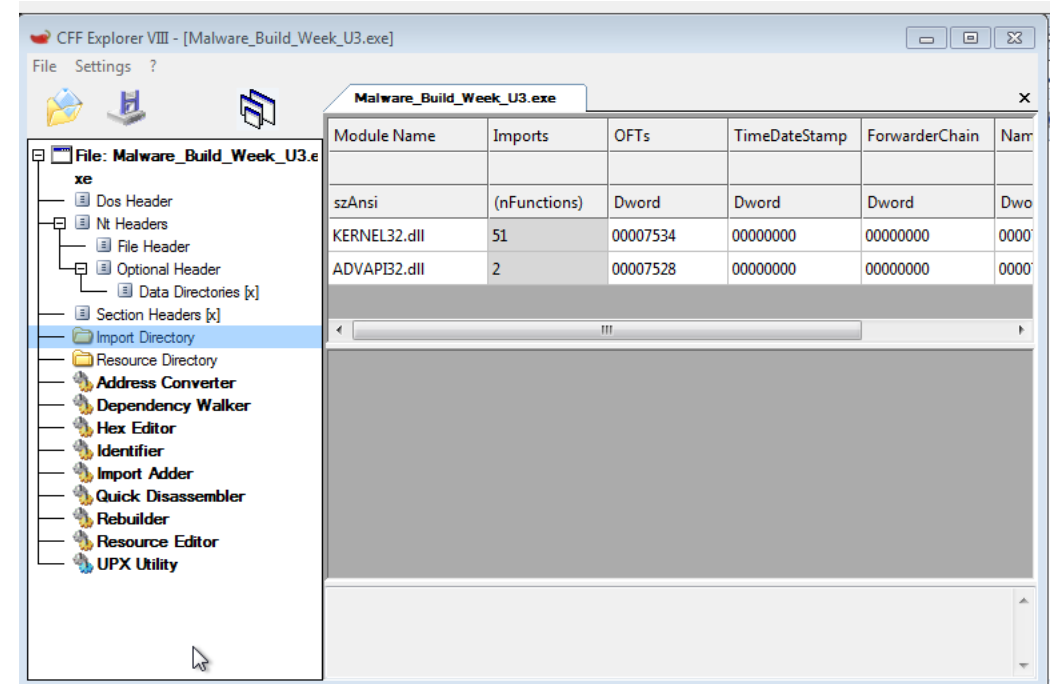
.data: Questa sezione contiene dati modificabili durante l'esecuzione del programma. Qui vengono memorizzate variabili globali e statiche che possono essere lette e scritte durante l'esecuzione del programma.

.rsrc (Resource): Questa sezione contiene le risorse del programma, come icone, immagini, stringhe localizzate, ecc. È utilizzata per memorizzare dati non eseguibili che il programma può utilizzare durante l'esecuzione, ad esempio per l'interfaccia utente o altri scopi.



Analisi Statica: visualizzazione librerie

Grazie a CFF Explorer posso vedere le librerie importate, in questo caso sono KERNEL32.dll e ADVAPI32.dll.



Analisi Statica: ipotesi sulle funzioni importate

Nella libreria KERNEL32.dll troviamo funzioni come 'LockResource' e 'LoadResource' che sono funzioni tipicamente utilizzate dai Dropper (una categoria di Malware) le quali servono a estrarre una risorsa (dalla sezione .rsrc), per poi (in questo caso) essere salvate e utilizzate in un secondo momento tramite le funzioni 'CreateFileA' e 'WriteFile'.

Le due funzioni riguardanti le chiavi di registro le troviamo nella libreria ADVAPI32.dll, probabilmente servono ad effettuare delle modifiche per ottenere la persistenza nel sistema.

OFTs	FTs (IAT)	Hint	Name
00007538	00007010	00007644	00007646
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc

000078C0	000078C0	00AA	FlushFileBuffers
000078D4	000078D4	026A	SetFilePointer
000078E6	000078E6	0034	CreateFileA
000078F4	000078F4	00BF	GetCPLInfo

0000770C	0000770C	019F	HeapFree
00007718	00007718	011A	GetLastError
00007728	00007728	02DF	WriteFile
00007734	00007734	029E	TerminateProcess

000076AC	000076AC	0186	RegSetValueExA
000076BE	000076BE	015F	RegCreateKeyExA

Analisi Statica: : istruzioni in memoria

00401021: tale istruzione serve a creare una chiave di sistema, i parametri vengono passati tramite delle istruzioni di push come l'handle della chiave.

```
.text:0040101d      push    0                ; ipclass
.text:00401015      push    0                ; Reserved
.text:00401017      push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows N
.text:0040101c      push    80000002h        ; hKey
.text:00401021      call    ds:RegCreateKeyExA
.text:00401027      test    eax, eax
.text:00401029      jz      short loc_401032
.text:0040102B      mov     eax, 1
```

00401017: tale istruzione serve a passare il path della chiave

```
* .text:00401017      push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
```

00401027-29: questa sezione di codice effettua prima il test tra i due registri facendo una sub tra i due valori contenuti, essendo i valori uguali darà come zero flag 1, l'istruzione successiva è un jump alla locazione di memoria indicata che viene effettuato se lo zero flag che risulta dall'istruzione precedente è 1, per cui ragionando in linguaggio C le due istruzioni si configurano come un if/then.

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

Analisi Statica: istruzioni in memoria

00401047: il valore del parametro 'ValueName' passato alla funzione come riportato sotto è: 'GinaDLL'.

```
IDA View-A | Program Segmentation | Hex View-A | Structures | Enums | Imports | Exports
.text:00401032 ; -----
.text:00401032
.text:00401032 loc_401032:
.text:00401032      mov     ecx, [ebp+cbData]      ; CODE XREF: sub_401000+29↑j
.text:00401035      push    ecx                    ; cbData
.text:00401036      mov     edx, [ebp+lpData]      ; 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

00001047  00401047: sub_401000+47
```

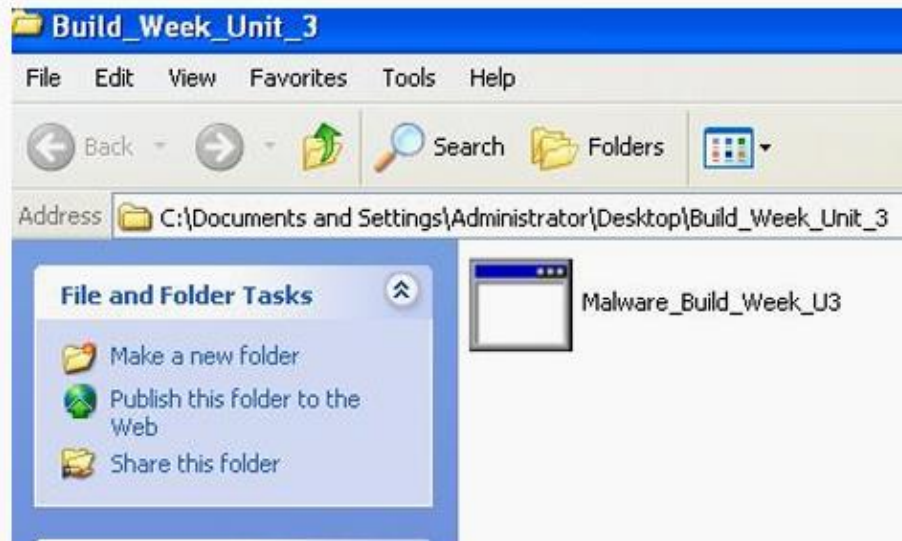
Analisi Dinamica

La traccia

Malware Analysis

Analisi dinamica

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



La traccia

Malware Analysis

- 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.



La traccia

Malware Analysis

Filtrate includendo solamente l'attività sul registro di Windows.

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

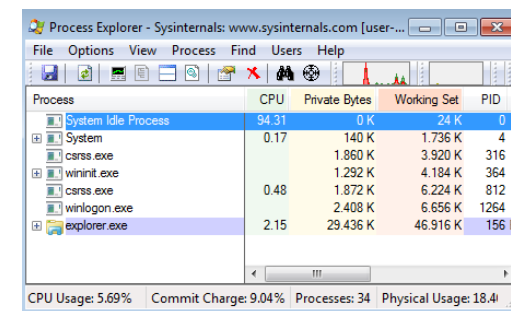
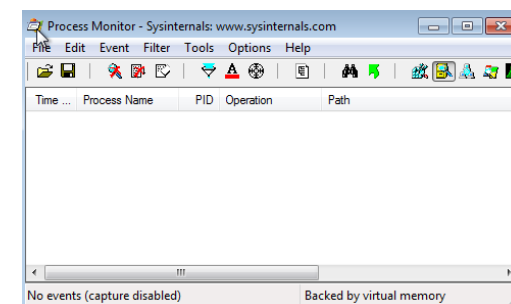
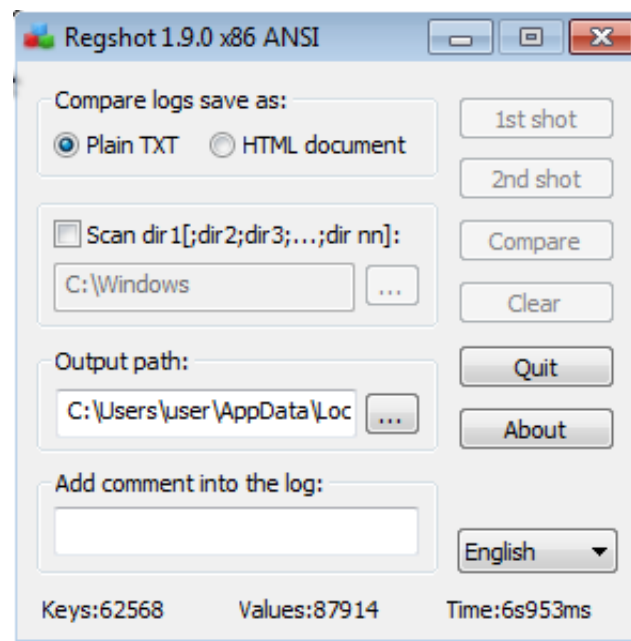
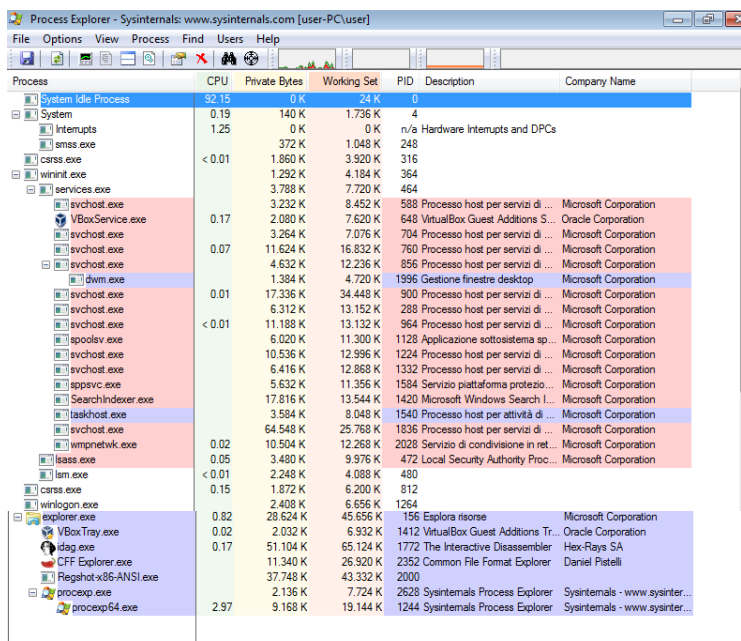
Passate ora alla visualizzazione dell'attività sul file system.

- 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.

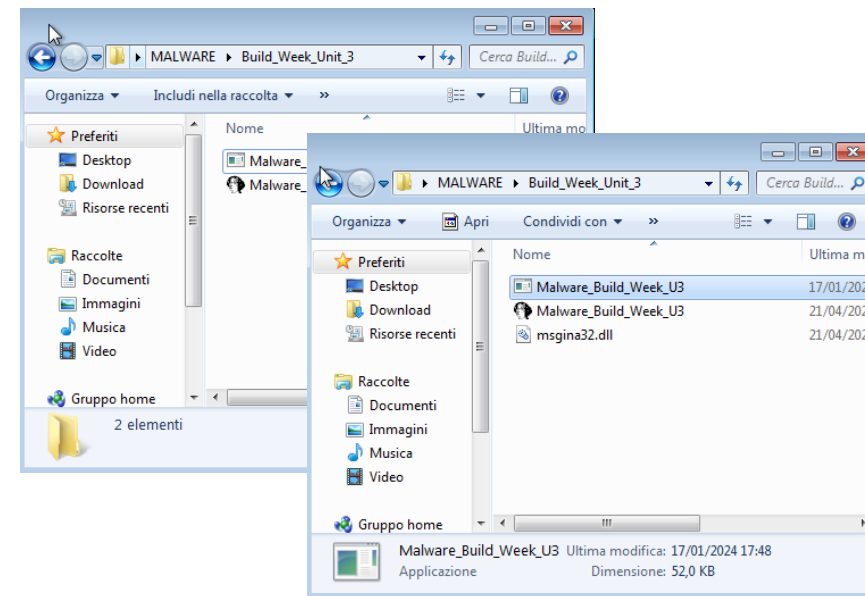
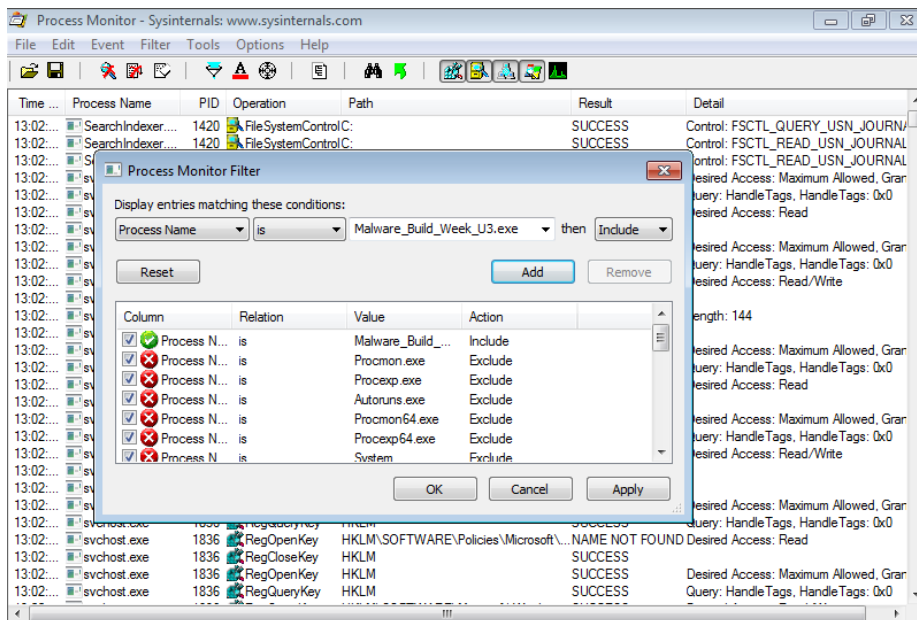
Analisi Dinamica: predisposizione tool pre-lancio del Malware

Prima di lanciare il Malware avvio ProcMon resettando i filtri e facendo il clear della schermata principale. Avvio Regshot in modo da avere una fotografia del registro di Windows e vedo i processi in esecuzione al momento sul sistema.



Analisi Dinamica: esecuzione Malware

La prima cosa che osserviamo una volta eseguito il Malware è la creazione del file msgina32.dll nella cartella del virus, intanto su Procmon attiviamo un filtro per isolare l'attività del Malware sul sistema. La creazione del file .dll potrebbe aver confermato la natura del Malware come dropper in quanto ha estratto la risorsa e creato appunto un nuovo file, troveremo eventualmente conferme di ciò in Procmon.



Analisi Dinamica: identificazione nuova chiave di registro su Procmon

Quello che tenta di fare il Malware da quanto osserviamo nell'attività di registro è creare una chiave di registro per avviare automaticamente il file .dll menzionato prima infatti

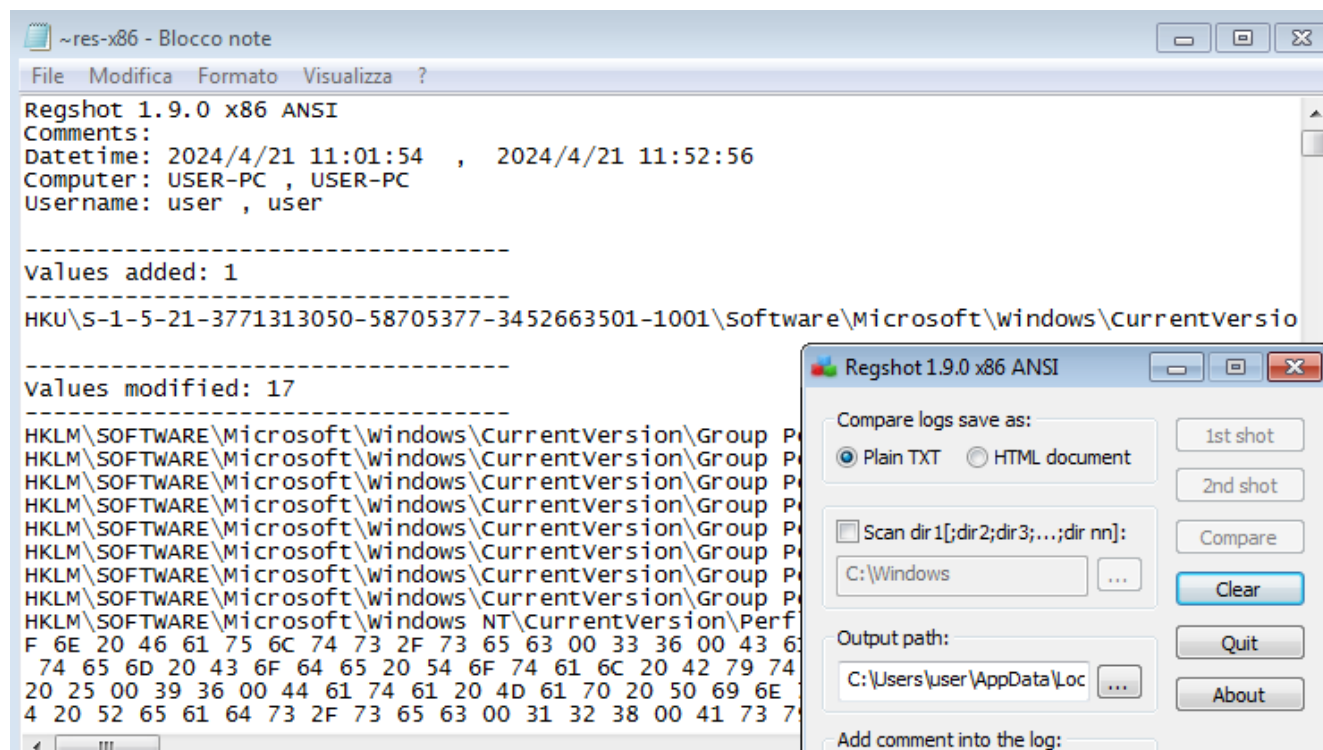
'HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon', è una posizione nel registro di sistema di Windows utilizzata per archiviare informazioni relative alla configurazione del processo di accesso (login) del sistema operativo.

Nel quarto evento osserviamo come viene settato il valore all'interno della chiave.

13:02:...	Malware_Build_...	656	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
13:02:...	Malware_Build_...	656	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
13:02:...	Malware_Build_...	656	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
13:02:...	Malware_Build_...	656	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
13:02:...	Malware_Build_...	656	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon









Analisi Dinamica: identificazione nuova chiave di registro su Regshot

La creazione della chiave è anche confermata dal 'secondo shot' su Regshot dopo l'avvio del malware, dal 'compare' rileviamo la creazione di una nuova chiave. Tramite la voce 'Value Added'.



Analisi Dinamica: identificazione chiamate di sistema su Procmon

Sotto possiamo invece osservare le chiamate di sistema che hanno permesso la creazione del file .dll e che quindi ha modificato il contenuto della cartella del malware.

13:02:...	 Malware_Build_...	656	 CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SI
13:02:...	 Malware_Build_...	656	 WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SI
13:02:...	 Malware_Build_...	656	 WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SI
13:02:...	 Malware_Build_...	656	 CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SI

Conclusioni

Per analizzare tale Malware ci siamo serviti di diversi tool sia per osservare la struttura del codice e più in generale dell'eseguibile del Malware (analisi statica) e il suo comportamento tramite l'analisi dinamica.

Grazie alla prima abbiamo osservato il contenuto delle librerie importate e le funzioni tipiche di un Dropper che estrae una risorsa creando poi un file sul sistema attaccato.

Tali ipotesi sono state confermate anche dall'analisi dinamica dove abbiamo analizzato le chiamate di sistema effettuate a tale scopo. Inoltre grazie all'attività dei registri abbiamo anche evidenziato la persistenza del virus sul sistema.