



MALWARE ANALYSIS REPORT

08/05/2024

Prepared By:
Olivier Team

Prepared For :
Epic Education srl

INDICE

1 - Introduzione

2 - Generalità del malware

- 2.1 - Analisi dei Parametri
- 2.2 - Analisi delle variabili
- 2.3 - Analisi delle sezioni
- 2.4 - Analisi delle librerie importate

3 - Analisi delle routine tra le locazioni di memoria 00401017 e 00401047

- 3.1 - Scopo della funzione chiamata alla locazione di memoria 00401021
- 3.2 - Parametri passati alla funzione in locazione 00401021
- 3.3 - L'oggetto rappresentato nel parametro in locazione 00401017
- 3.4 - Le istruzioni comprese tra gli indirizzi 00401027 e 00401029
- 3.5 - Traduzione del codice assembly in loc. 00401027/00401029 in costrutto C
- 3.6 - Valutazione della chiamata alla locazione 00401047, valore ValueName
- 3.7 - Valutazione complessiva dell'estratto di codice

4 - Analisi delle routine tra le locazioni di memoria 00401080 e 00401128

- 4.1 - valore del parametro «ResourceName»
- 4.2 - Funzionalità implementate dal malware nella sezione in esame
- 4.3 - Possibilità di identificazione delle funzioni con l'analisi statica basica
- 4.4 - Conclusioni sull'analisi statica basica
- 4.5 - Diagramma di flusso

5 - Esecuzione del malware / Analisi dinamica

- 5.1 - Analisi con Process Monitor
- 5.2 - Analisi chiave del registro
- 5.3 - Analisi valori chiave del registro
- 5.4 - Analisi del File System
- 5.5 - Conclusioni sul comportamento del malware

6 - Funzionalità del malware

- 6.1 - Graphical Identification and Authentication
- 6.2 - Grafico del funzionamento

7 - Glossario



1 - Introduzione

Questo report risponde ai quesiti posti utilizzando i tool e le tecniche apprese nelle lezioni teoriche.

Con riferimento al file eseguibile `Malware_Build_Week_U3`, caricato su un sistema operativo Windows 7, andremo a studiare approfonditamente il file eseguibile.

2 - Generalità del malware

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()`?; **(2.1)**
- Quante variabili sono dichiarate all'interno della funzione `Main()`?; **(2.2)**
- Quali sezioni sono presenti all'interno del file eseguibile? **(2.3)**
Descrivete brevemente almeno 2 di quelle identificate;

- Quali librerie importa il Malware? **(2.4)**

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;

2.1 - Analisi dei parametri

Sono stati passati 3 parametri alla funzione `Main()`, vedi figura sottostante:

argc

argv

envp

Il parametro è l'argomento che viene passato alla funzione.

Riconosciamo il parametro perché si trova ad un offset (1) positivo rispetto al registro di riferimento EBP (2)

```
; 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
```



2.2 - Analisi delle variabili

Sono state dichiarate 5 variabili all'interno della funzione main(), vedi figura sottostante:

hModule

Data

var_117

var_8

var_4

Le variabili sono definite localmente nel contesto di una funzione.

Possiamo riconoscere le variabili perché sono ad un offset negativo rispetto al registro di riferimento.

```
; 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
```

2.3 Analisi delle sezioni

Il file eseguibile presenta le seguenti sezioni (3): **.text**, **.rdata**, **.data**, **.rsrc**.

La sezione **.text** contiene il codice eseguibile del programma.

La sezione **.rdata** contiene le informazioni di lettura dei dati, come le stringhe di testo e le costanti.

La sezione **.data** contiene i dati inizializzati e non inizializzati utilizzati dal programma.

La sezione **.rsrc** contiene le risorse come icone, menu e dialoghi utilizzati dall'applicazione.

Siamo risaliti alle sezioni dell'eseguibile grazie ad un Tool per l'analisi statica dei malware, CFF Explorer, vedi figura sottostante.

Malware_Build_Week_U3.exe									
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	00003EAB	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040



2.4 Analisi delle librerie importate

Il Malware importa le seguenti librerie (4): ADVAPI32.dll, KERNEL32.dll.

ADVAPI32.dll: Questa libreria è comunemente utilizzata per le operazioni avanzate di servizio come la gestione del registro e la sicurezza.

Il Malware potrebbe utilizzare questa libreria per manipolare il registro o alterare le impostazioni di sicurezza.

Le funzioni importate da questa libreria sono rispettivamente la RegSetValueA e

RegCreateKeySetA, ne possiamo dedurre che il malware può utilizzare il registro e quindi ottenere la persistenza (5), andremo ad approfondire questo argomento durante l'analisi dinamica dell'eseguibile.

KERNEL32.dll: Questa libreria fornisce accesso a funzioni di basso livello come la gestione della memoria, le operazioni di input/output e le interruzioni.

Il Malware potrebbe utilizzare questa libreria per manipolare la memoria o eseguire operazioni di I/O.

Le funzioni importate sono 51, e tramite lo studio delle stesse possiamo dedurre che:

- Il malware può risolvere e importare dinamicamente risorse esterne (GetProcAddress e LoadLibraryA).
- È probabile che il malware utilizzi risorse(FindResourceA , FreeResource , LoadResource , LockResource e SizeofResource)
- Esistono molte importazioni che danno al malware la capacità di operare con i file. (WriteFile , CreateFileA , ReadFile , ecc.)
- Inoltre, il malware potrebbe allocare spazio per eseguire codice malevolo, nascondere dati sensibili o manipolare il funzionamento di altre applicazioni tramite queste funzioni di gestione della memoria.(VirtualAlloc , HeapAlloc , VirtualFree , HeapFree , ecc.)

Nella figura sottostante vediamo, tramite CFF Explorer, le librerie importate dal malware

Module Name	Imports	OFTs	TimeDateStamp	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

3 - Analisi delle routine tra le locazioni di memoria 00401017 e 00401047

Con riferimento al Malware in analisi, spiegare:

- Lo scopo della funzione chiamata alla locazione di memoria 00401021; **(3.1)**
- Come vengono passati i parametri alla funzione alla locazione 00401021; **(3.2)**
- Che oggetto rappresenta il parametro alla locazione 00401017; **(3.3)**
- Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029. (se serve, valutate anche un'altra o altre due righe assembly); **(3.4)**
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C ; **(3.5)**
- Valutate ora la chiamata alla locazione 00401047, qual'è il valore del parametro «ValueName»?; **(3.6)**
- Nel complesso delle due funzionalità appena viste, spiegate quale funzionalità sta implementando il Malware in questa sezione; **(3.7)**



3.1- Scopo della funzione chiamata alla locazione di memoria 00401021

La funzione chiamata è la **RegCreateKeyExA**: Questa funzione crea una nuova chiave nel registro di sistema. Permette di specificare il nome della chiave, i diritti di accesso, un oggetto di sicurezza opzionale che determina l'accesso alla chiave, e altri parametri opzionali. Nello specifico, la funzione crea un nuovo registro al path indicato alla locazione 00401017, andremo a studiare come il malware modifica i registri con l'analisi dinamica.

```
.text:00401017      push     offset SubKey      ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040101C      push     80000002h         ; hKey
.text:00401021      call    ds:RegCreateKeyExA
```

3.2 - Parametri passati alla funzione in locazione 00401021

La locazione **00401021** corrisponde all'istruzione `call ds:RegCreateKeyExA`.

Quando viene eseguita questa istruzione, vengono passati i parametri alla funzione `RegCreateKeyExA` nello stack e nei registri, seguendo la convenzione di chiamata della piattaforma, in questo caso di tipo LIFO.

Nel dettaglio, i parametri vengono passati in questo modo:

I parametri vengono solitamente passati nei registri dedicati per i parametri (come ECX, EDI, e così via) e nello stack (6), nell'ordine in cui vengono richiesti dalla funzione chiamata.

Per la funzione **RegCreateKeyExA**, l'ordine dei parametri è il seguente:

hKey (handle della chiave di registro padre)

lpSubKey (puntatore alla stringa che specifica il nome della sottochiave)

Reserved (generalmente usato come zero)

lpClass (puntatore alla stringa che contiene il nome della classe del valore da creare o NULL)

dwOptions (opzioni per l'apertura/creazione della chiave)

samDesired (accesso desiderato alla chiave)

lpSecurityAttributes (puntatore a strutture di sicurezza, solitamente NULL)

phkResult (puntatore a una variabile dove viene restituito l'handle della chiave aperta o creata)

lpdwDisposition (indica se una chiave di registro è stata creata nuova o aperta se già esistente, in questo è stata creata.)

```
.text:00401021      push     0                  ; lpdwDisposition
.text:00401024      lea     eax, [ebp+hObject]
.text:00401027      push     eax               ; phkResult
.text:0040102A      push     0                 ; lpSecurityAttributes
.text:0040102D      push     0F003Fh           ; samDesired
.text:00401030      push     0                 ; dwOptions
.text:00401033      push     0                 ; lpClass
.text:00401036      push     0                 ; Reserved
.text:00401039      push     offset SubKey     ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040103C      push     80000002h         ; hKey
.text:0040103F      call    ds:RegCreateKeyExA
```



3.3 - L'oggetto rappresentato nel parametro in locazione 00401017

Alla locazione 00401017, il parametro passato è l'offset del nome della sottochiave nel Registro di sistema di Windows. Questo parametro è un puntatore al nome della sottochiave che la funzione RegCreateKeyEx utilizzerà per creare o aprire la chiave di registro specificata.

La sottochiave rappresenta il percorso all'interno del Registro di sistema dove verrà creata o aperta la chiave. In questo caso specifico, il nome della sottochiave sembra essere "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon".

```
; char SubKey[]
SubKey      db 'SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon',0
              ; DATA XREF: sub_401000+17fo
```

3.4 - Le istruzioni comprese tra gli indirizzi 00401027 e 00401029

Le istruzioni comprese tra gli indirizzi **00401027** e **00401029** svolgono la funzione di costruito if, utilizzando come operando l'istruzione test e come condizione l'istruzione jz.

```
test     eax, eax
jz       short loc_401032
```

3.5 - Traduzione del codice assembly in loc. 00401027/00401029 in costruito C

Nell'immagine sottostante riportiamo l'istruzione tradotta in linguaggio C

```
if (eax == 0) {
    // Blocco di codice da eseguire se eax è zero
    goto loc_401032;
}
```

3.6 - Valutazione della chiamata alla locazione 00401047, valore ValueName

Il valore del parametro ValueName è "GinaDLL"

```
*.text:0040103E      push     offset ValueName ; "GinaDLL"
*.text:00401043      mov     eax, [ebp+hObject]
*.text:00401046      push     eax              ; hKey
*.text:00401047      call    ds:RegSetValueExA
```

```
; char ValueName[]
ValueName    db 'GinaDLL',0          ; DATA XREF: sub_401000+3Efo
```



3.7 - Valutazione complessiva dell'estratto di codice

Dalle funzioni appena analizzate, possiamo dedurre che il malware sia stato progettato per modificare il Registro di sistema al fine di ottenere un accesso persistente e per eseguire azioni dannose all'interno del sistema operativo Windows.

Inoltre, il fatto che il malware stia tentando di impostare un valore di registro chiamato "GinaDLL" indica un tentativo di sostituire il componente DLL del processo di autenticazione di Windows, che potrebbe consentire al malware di raccogliere credenziali di accesso o di ottenere accesso non autorizzato al sistema.

4 - Analisi delle routine tra le locazioni di memoria 00401080 e 00401128

- Qual è il valore del parametro «ResourceName» passato alla funzione FindResourceA(); **(4.1)**
- 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?; **(4.2)**
- È possibile identificare questa funzionalità utilizzando l'analisi statica basica ? (dal giorno 1 in pratica); **(4.3)**
- 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(); **(4.4)**
- Disegnare un diagramma di flusso che comprenda le tre funzioni; **(4.5)**

4.1- Valore del parametro ResourceName

Il Valore del parametro passato alla funzione FindResourceA è contenuto nel registro ECX al momento della chiamata. "TGAD" è il parametro lpName passato

```
* .text:004010BE          mov     ecx, lpName
* .text:004010C4          push    ecx           ; lpName

; LPCSTR lpName
lpName dd offset aTgad          ; DATA XREF: sub_401080+3E↑r
; "TGAD"
```



4.2 - Funzionalità implementate dal malware nella sezione in esame

La funzione **FindResourceA** appartiene alla libreria di sistema Windows KERNEL32.dll ed è utilizzata per individuare una risorsa all'interno di un modulo specificato, come un file eseguibile o una DLL. La ricerca della risorsa avviene tramite il tipo e il nome della risorsa specificati come parametri.

La funzione **LoadResource** di KERNEL32.dll viene utilizzata per caricare una risorsa individuata da FindResourceA in memoria.

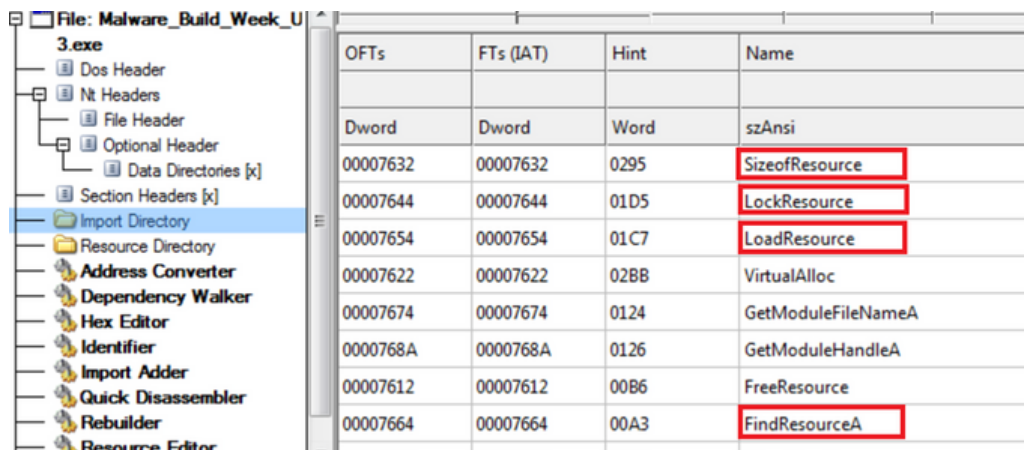
La funzione **LockResource** di KERNEL32.dll viene utilizzata per ottenere un puntatore alla risorsa caricata in memoria tramite LoadResource.

La funzione **SizeofResource** di KERNEL32.dll viene utilizzata per ottenere la dimensione di una risorsa individuata da FindResourceA.

Queste funzioni quindi permettono di localizzare all'interno della sezione risorse il malware da estrarre e di caricare o salvare sul disco l'esecuzione futura. Queste tipo di funzionalità sono caratteristiche necessarie ad un tipo di malware, il Dropper, che è un programma malevolo che contiene un malware. Quando viene eseguito il dropper estrae il malware e lo salva sul disco. Generalmente il malware è contenuto nella sezione .ress dell'eseguibile (la sezione risorse).

4.3 - Possibilità di identificazione delle funzioni con l'analisi statica basica

E' possibile identificare le funzioni trovate tramite **IDA** viste nel paragrafo 4.2 attraverso un tool usato nell'analisi statica basica, **CFF Explorer**.



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

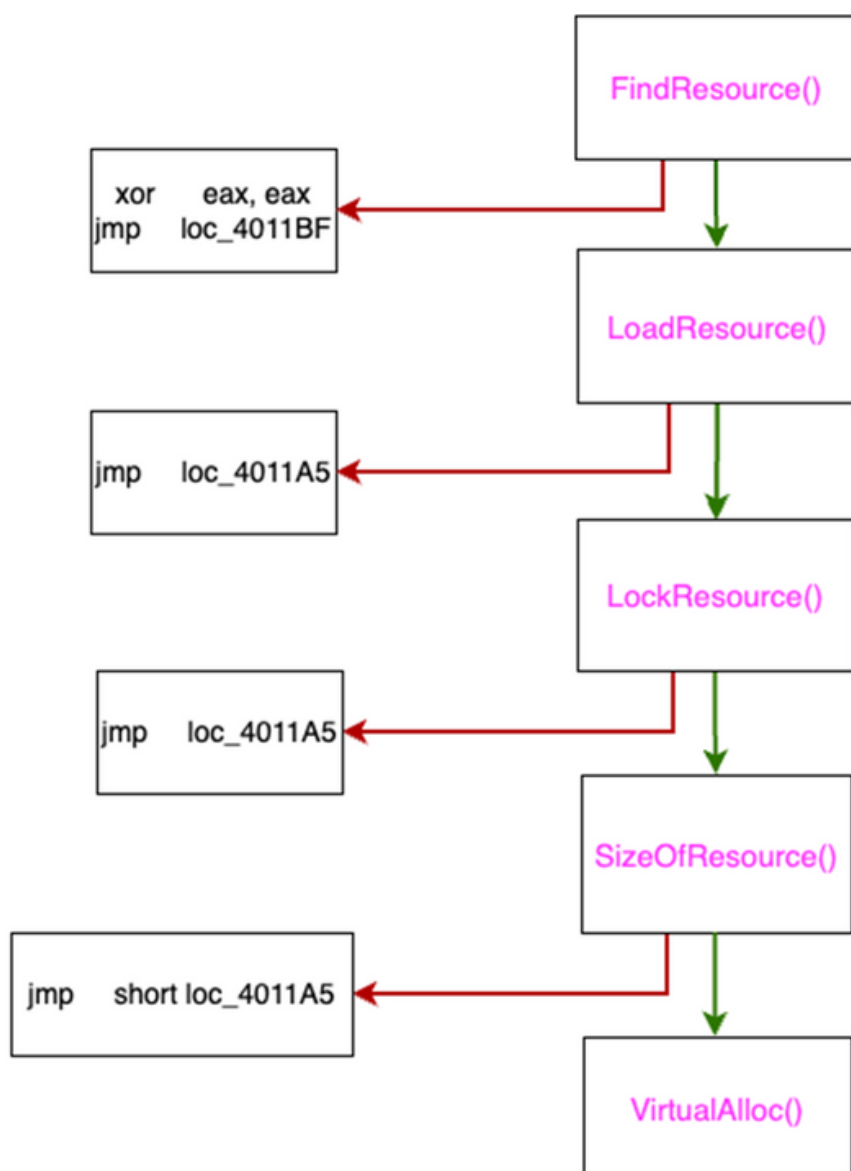
```
* .text:004010C9      call     ds:FindResourceA
* .text:004010E7      call     ds:LoadResource
* .text:004010FF      call     ds:LockResource
* .text:0040111B      call     ds:SizeofResource
```

4.4 - Conclusioni sull'analisi statica basica

Con l'analisi statica basica possiamo identificare quali sono le funzioni importate dal malware, ma non abbiamo modo di capire come queste funzioni vengano usate. Possiamo quindi affermare che l'analisi statica fornisca un'idea parziale di come lavorino le funzioni all'interno del malware

4.5 - Diagramma di flusso

Nella figura sottostante viene data un'idea di come operano le funzioni appena descritte.



5 - Esecuzione del malware / 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

-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; **(5.1)**

Filtrate includendo solamente l'attività sul registro di Windows .

-Quale chiave di registro viene creata?; **(5.2)**

-Quale valore viene associato alla chiave di registro creata?; **(5.3)**

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?; **(5.4)**

Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware; **(5.5)**

5.1 - Analisi con Process Monitor

Dopo che il malware viene eseguito, crea un nuovo file all'interno della propria directory attraverso il susseguirsi delle funzioni analizzate in precedenza e contenute nella subroutine_401080 **(4.2)**.

```
loc_4010B8:
mov     eax, lpType
push    eax                ; lpType
mov     ecx, lpName
push    ecx                ; lpName
mov     edx, [ebp+hModule]
push    edx                ; hModule
call    ds:FindResourceA
mov     [ebp+hResInfo], eax
cmp     [ebp+hResInfo], 0
jnz     short loc_4010DF
```

```
loc_4010DF:
mov     eax, [ebp+hResInfo]
push    eax                ; hResInfo
mov     ecx, [ebp+hModule]
push    ecx                ; hModule
call    ds:LoadResource
mov     [ebp+hResData], eax
cmp     [ebp+hResData], 0
jnz     short loc_4010FB
```

```
loc_4010FB:
mov     edx, [ebp+hResData]
push    edx                ; hResData
call    ds:LockResource
mov     [ebp+Str], eax
cmp     [ebp+Str], 0
jnz     short loc_401113
```

```
loc_401113:
mov     eax, [ebp+hResInfo]
push    eax                ; hResInfo
mov     ecx, [ebp+hModule]
push    ecx                ; hModule
call    ds:SizeofResource
mov     [ebp+Count], eax
cmp     [ebp+Count], 0
ja      short loc_40112C
```



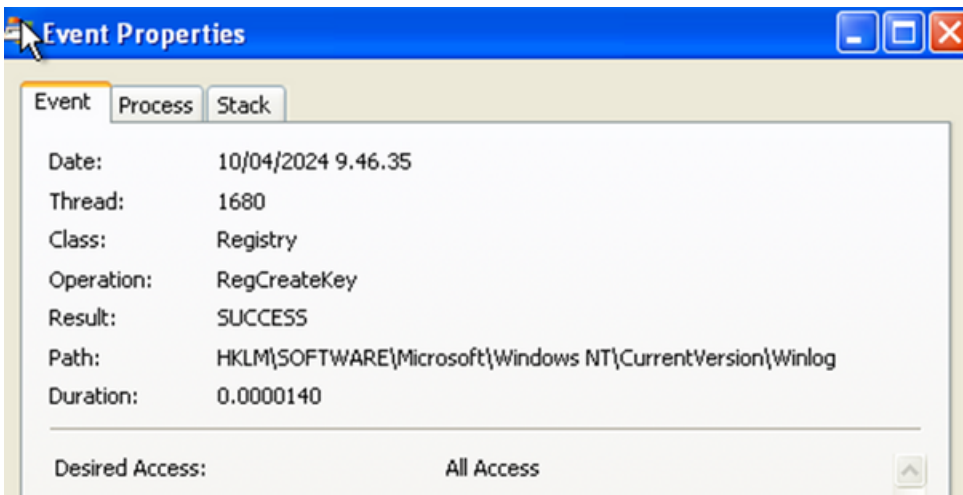
Nell'immagine sottostante è mostrata la directory del file eseguibile dove è stato creato il nuovo file (msgina32.dll), i 4 file .id0, .id1, .nam, .til sono stati creati da IDA dopo che il malware è stato caricato sul tool.

Malware_Build_Week_U3	17/01/2024 17:48	Applicazione	52 KB
Malware_Build_Week_U3.id0	09/04/2024 14:17	File ID0	384 KB
Malware_Build_Week_U3.id1	09/04/2024 14:17	File ID1	184 KB
Malware_Build_Week_U3.nam	09/04/2024 14:17	File NAM	16 KB
Malware_Build_Week_U3.til	09/04/2024 09:20	File TIL	1 KB
msgina32.dll	09/04/2024 14:24	Estensione dell'ap...	7 KB

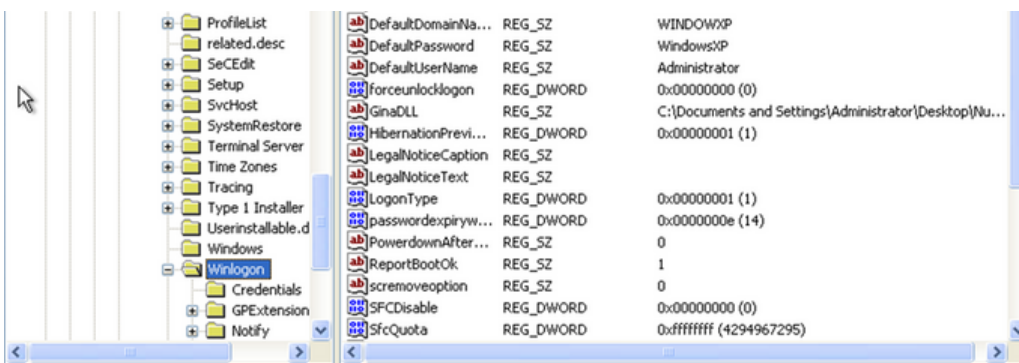
5.2 - Analisi chiave del registro

Nell'immagine sottostante viene mostrata la nuova chiave di registro creata dal file malevolo (Winlogon (7)). Possiamo vedere la nuova chiave di registro creata su Process monitor, filtrando solamente le attività sul registro di Windows.

Time of Day	Process Name	PID	Operation	Path
12:24:56.3182070	Malware_Build_...	2904	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon



Nell'immagine sottostante vediamo la nuova chiave creata su Regedit.



5.3 - Analisi valori chiave del registro

Il valore associato alla nuova chiave di registro e il percorso

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\GinaDLL.

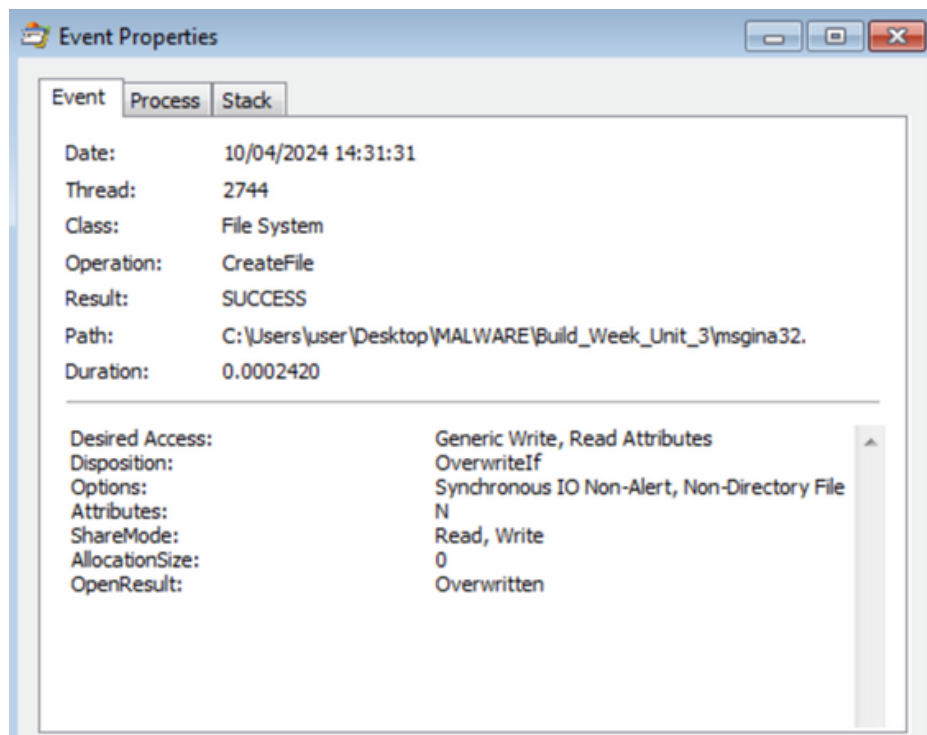
La stringa ci indica il percorso della libreria che viene eseguita all'autenticazione dell'utente (winlogon.exe).

```
12:24:56.3182343 Malware_Build_... 2904 RegSetValue HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
12:24:56.3182628 Malware_Build_... 2904 RegCloseKey HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
```

5.4 - Analisi del File System

Nell'immagine sottostante viene mostrata la sezione che riguarda i File System, su Process Monitor, dove viene creato il file **msgina32.dll**

```
CreateFile C:\Users\user\Desktop\MALWARE\Bu... SUCCESS Desired Access: G...
WriteFile C:\Users\user\Desktop\MALWARE\Bu... SUCCESS Offset: 0, Length: 4...
WriteFile C:\Users\user\Desktop\MALWARE\Bu... SUCCESS Offset: 4.096, Leng...
CloseFile C:\Users\user\Desktop\MALWARE\Bu... SUCCESS
```



5.5 - Conclusioni sul comportamento del malware

Dopo uno studio approfondito del malware, siamo giunti alla conclusione che si tratta di un Dropper che successivamente scarica all'interno della macchina target un credential stealer (msgina32.dll).

Il "dropper" in questione installa segretamente il credential stealer sul sistema informatico compromesso. Il suo scopo principale è "droppare" (da qui il termine) o rilasciare il componente dannoso (**msgina32.dll**) all'interno del sistema senza che l'utente ne sia consapevole. Ecco spiegato passo passo il suo funzionamento:



Fase 1: Infiltrazione

Il dropper si fa strada nel sistema di destinazione attraverso tecniche come:

- Download ingannevoli
- Allegati email malevoli
- File condivisi
- Exploit di vulnerabilità del software
- Infezione del sistema tramite dispositivi USB

Fase 2: Evasione e Persistenza

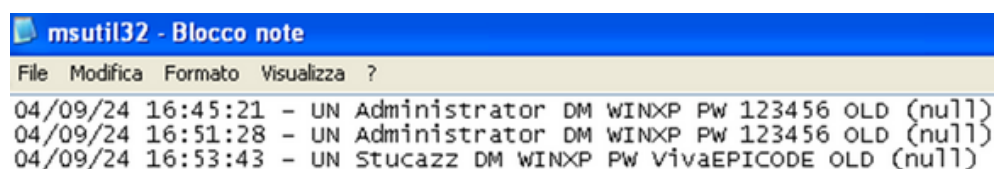
Una volta eseguito, il dropper evade il rilevamento da parte dell'antivirus (proprio perché il file eseguibile non viene riconosciuto come malevolo) e stabilisce la persistenza sul sistema, tramite la modifica delle chiavi di registro per eseguirsi automaticamente all'avvio del sistema operativo.

Fase 3: Esecuzione del Credential Stealer

Il dropper importa la libreria DLL malevola progettata per funzionare con l'interfaccia di accesso di Windows, nota come **GINA (Graphical Identification and Authentication)**. GINA è stata usata in versioni di Windows fino a Windows Vista per gestire le schermate di login. Il DLL malevolo modifica il comportamento di GINA per catturare le credenziali utente quando vengono inserite durante il processo di login.

Fase 4: Creazione del file di sistema ed esfiltrazione dei dati

Il file "msgina32.dll" crea il file "msutil32.sys" dove andrà a salvare le credenziali rubate. Il path per accedere a quel file è il seguente: **%SystemRoot%\System32\msutil32.sys**.



```
msutil32 - Blocco note
File Modifica Formato Visualizza ?
04/09/24 16:45:21 - UN Administrator DM WINXP PW 123456 OLD (null)
04/09/24 16:51:28 - UN Administrator DM WINXP PW 123456 OLD (null)
04/09/24 16:53:43 - UN Stucazz DM WINXP PW VivaEPICODE OLD (null)
```

Successivamente, questo file può essere trasmesso al criminale informatico tramite internet, completando così il furto di informazioni.

È importante notare che le versioni moderne di Windows hanno sostituito GINA con Credential Provider, ma il concetto di intercettazione delle credenziali mediante componenti malevoli rimane rilevante. Le difese contro tali minacce includono l'uso di software antivirus aggiornato, firewall, l'attuazione di politiche di sicurezza solide, e la formazione degli utenti sull'identificazione delle tattiche di phishing e sui rischi di scaricare software da fonti non affidabili.

6 - Funzionalità del malware

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 username e 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 .dll malevolo, che intercetta i dati inseriti?; **(6.1)**

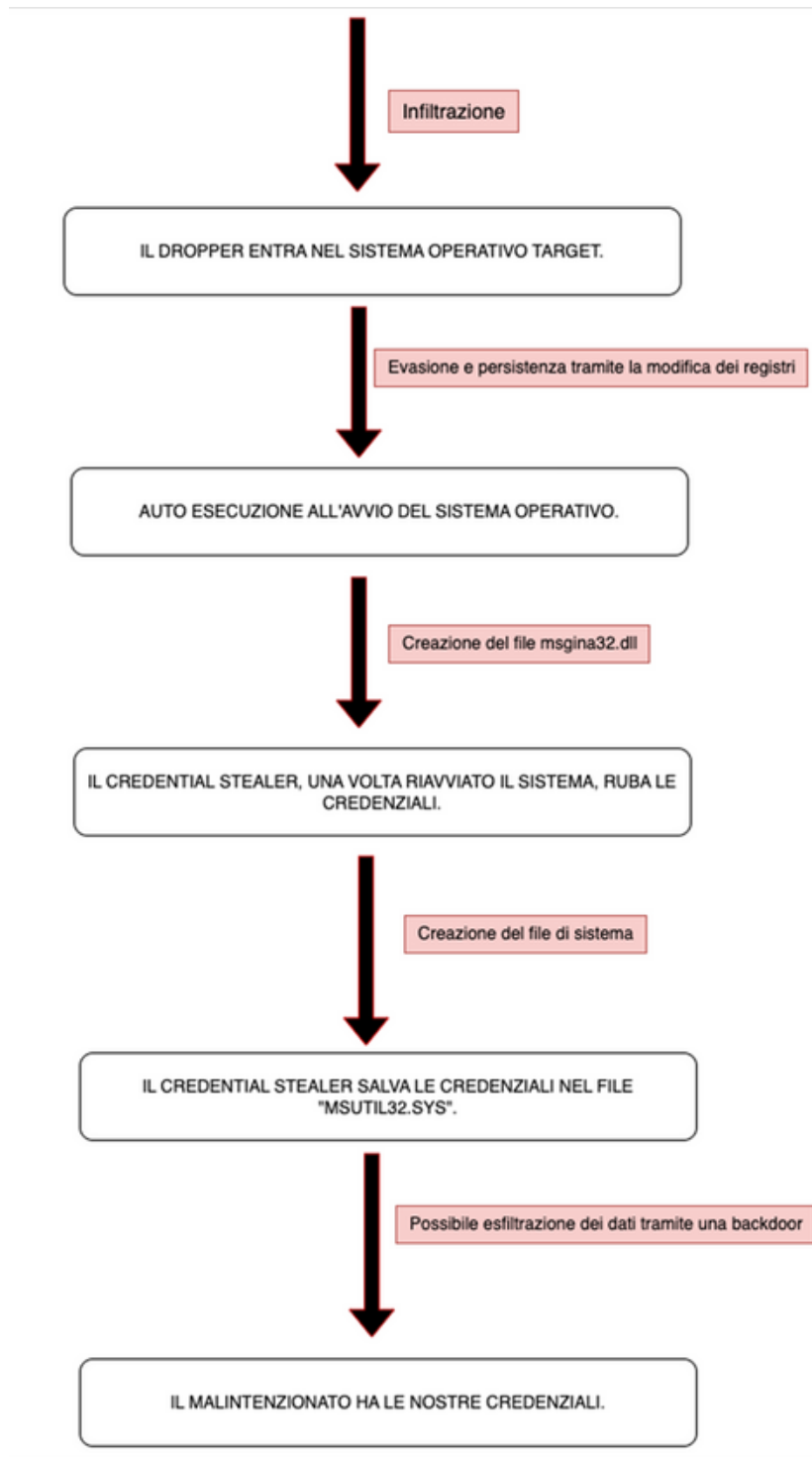
Sulla base della risposta sopra, delineate il profilo del Malware e delle sue funzionalità. Unite tutti i punti per creare un grafico che ne rappresenti lo scopo ad alto livello; **(6.2)**



6.1 - Graphical Identification and Authentication

Quando viene importato, il malware sostituisce il file .dll lecito con un file .dll malevolo (msgina32.DLL) che va ad intercettare le credenziali inserite dall'utente per poi inserirle nel file al path **%SystemRoot%\System32\msutil32.sys**.

6.2 - Grafico del funzionamento



7 - Glossario

OFFSET (1) - Differenza di indirizzo tra due punti, solitamente utilizzato per accedere a variabili, dati o istruzioni di salto. È essenzialmente una distanza calcolata rispetto a un punto di riferimento.

EBP (2) - Il registro EBP, noto anche come Base Pointer, è un registro utilizzato nelle architetture dei processori x86. Il suo ruolo principale è quello di puntare alla base dello stack frame corrente in un programma. Quando una funzione viene chiamata, l'indirizzo al quale ritornare e i parametri della funzione sono spesso memorizzati nello stack. EBP viene utilizzato per mantenere un punto di riferimento costante, così che tutti i parametri e le variabili locali possano essere acceduti usando un offset rispetto a EBP, indipendentemente da quanti altri dati siano stati messi sullo stack.

SEZIONI (3) - Un file eseguibile nel formato PE (Portable Executable), è diviso in diverse sezioni con scopi specifici. Queste sezioni organizzano il file in modo che il sistema operativo possa caricarlo in memoria ed eseguirlo efficacemente.

LIBRERIE (4) - Le librerie di Windows sono collezioni di codice pre-scritto che gli sviluppatori possono utilizzare nelle loro applicazioni per eseguire funzionalità specifiche senza dover scrivere tutto il codice da zero. Ci sono due formati principali:

- **DLL (Dynamic Link Libraries)**: Sono librerie condivise che contengono funzioni, dati e risorse che le applicazioni di Windows possono utilizzare. Le DLL aiutano a promuovere la modularità del codice, il riutilizzo del codice, l'efficienza della memoria e la riduzione dello spazio su disco. Quando un'applicazione viene eseguita, le funzioni di una DLL non vengono caricate nella memoria finché non vengono effettivamente chiamate (linking dinamico).
- **Static Libraries (.lib)**: A differenza delle DLL, le librerie statiche vengono incorporate direttamente nell'eseguibile dell'applicazione al momento della compilazione. Ciò significa che ogni applicazione ha la propria copia del codice di libreria, portando a un aumento della dimensione dell'eseguibile ma riducendo le dipendenze esterne e i tempi di caricamento all'avvio.

•
PERSISTENZA (5) - La persistenza, nel contesto dell'informatica e della sicurezza informatica, si riferisce alla capacità di un elemento, come un file, un processo o un software, di rimanere attivo o persistere nel sistema anche dopo che l'utente ha tentato di eliminarlo o dopo un riavvio del sistema. Questo concetto è particolarmente rilevante quando si parla di malware, poiché molti malware cercano di garantire la loro persistenza nel sistema al fine di mantenere l'accesso e continuare a danneggiare o rubare dati senza essere facilmente rilevati o rimossi dagli utenti o dai software antivirus. In sostanza, la persistenza è la capacità di un elemento di "restare in vita" nel sistema, nonostante gli sforzi per eliminarlo.



STACK (6) - Lo stack è una struttura dati fondamentale utilizzata in informatica per l'organizzazione e la gestione delle informazioni. In breve, si tratta di una collezione di elementi ordinati secondo un principio di accesso **LIFO** (Last In, First Out), il che significa che l'ultimo elemento inserito è il primo ad essere estratto.

Nel contesto dei programmi informatici, lo stack è spesso utilizzato per gestire le chiamate di funzioni e le variabili locali durante l'esecuzione del codice. Quando una funzione viene chiamata, le sue variabili locali vengono allocate nello stack e vengono rimosse quando la funzione ritorna. Questa gestione a stack delle chiamate di funzione consente di mantenere un tracciamento dell'esecuzione del programma e di gestire in modo efficiente le variabili locali.

WINLOGON (7) - Winlogon è un componente critico del sistema operativo Windows, responsabile per la gestione dell'accesso e della disconnessione degli utenti. Il suo nome completo è Windows Logon Process, e si tratta di un processo che opera in background per gestire varie funzioni legate alla sicurezza e all'accesso degli utenti.





GRAZIE PER L'ATTENZIONE

08/05/2024

Prepared By:

Oliviero Camarota
Pignatello Giuseppe
Vitale Francesco
Scopece Francesco Pio

Prepared For :

Epic Education srl