

Esercitazione Fine modulo 6

Malware Analysis

Progetto

Fabio Benevento - 19/04/2024

Traccia

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_U3, rispondere ai seguenti quesiti utilizzando i tool e le tecniche apprese nelle lezioni teoriche:

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

Con riferimento al Malware in analisi, spiegare:

- 5) Lo scopo della funzione chiamata alla locazione di memoria 00401021
- 6) Come vengono passati i parametri alla funzione alla locazione 00401021
- 7) Che oggetto rappresenta il parametro alla locazione 00401017
- 8) Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.
- 9) Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C.
- 10) Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

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

- 1) 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.

- 2) Quale chiave di registro viene creata?
- 3) Quale valore viene associato alla chiave di registro creata?

Passate ora alla visualizzazione dell'attività sul file system.

- 4) 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.

Svolgimento

Analisi statica

1) Quanti parametri sono passati alla funzione Main()?

La funzione main presenta 3 parametri, nello specifico argc, argv e envp come mostrato nell'immagine in figura estratta tramite il tool di analisi statica avanzata IdaPro.

Si tratta dei parametri tipici di una delle forme del main in linguaggio C/C++-

In IdaPro, i parametri di una funzione sono individuabili in quanto presentano un offset positivo rispetto allo stack base pointer (EBP).

Nel caso specifico della funzione main, il tutto è reso ancora più semplice in quanto il tool riesce anche ad identificare in maniera autonoma la firma 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_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) Quante variabili sono dichiarate all'interno della funzione Main()?

Le variabili in IdaPro presentano al contrario un offset negativo rispetto ad EBP. Facendo quindi riferimento alla figura precedente, nel main sono utilizzate 5 variabili, hmodule, Data, var_117, var_8 e var_4

3) Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate

Effettuando l'analisi con un tool di analisi statica basica, come CFF Explorer VIII, nella specifica sezione Section Header, mostrata in figura, è possibile individuare 4 sezioni,

descritte brevemente di seguito:

- .text: è la sezione che contiene il codice (le istruzioni) del programma
- .data: è la sezione dove sono collocate le variabili globali del programma, ovvero quelle variabili definite al di fuori di una funzione e per questo motivo accessibili da qualsiasi funzione (scope globale) e la cui durata è pari all'intero tempo di esecuzione del programma.
- .rdata: contiene informazioni sulle librerie importate/esportate dal programma
- .rsrc: contiene le risorse usate dall'applicazione, come ad esempio immagini, icone, stringhe, etc..

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Word	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

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

Il malware importa 2 librerie, come è possibile vedere dalla sezione Import Directory di CFF Explorer, advapi32.dll e kernel32.dll.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

Per quanto riguarda advap32.dll, il malware utilizza 2 sole funzioni, RegCreateKeyExA e RegSetValueExA.

CFF Explorer VIII - [Malware_Build_Week_U3.exe]

File Settings ?

File: Malware_Build_Week_U3.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [x]
- Section Headers [x]
- Import Directory
- Resource Directory
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler
- Rebuilder
- Resource Editor
- UPX Utility

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
000076D0	N/A	00007500	00007504	00007508	0000750C	00007510
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

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

Si tratta di funzioni utilizzate per manipolare il registro di sistema, nello specifico per creare una nuova chiave di registro (RegCreateKeyExA) e per settare un valore ad una chiave (RegSetValueExA).

Tipicamente queste funzioni vengono sfruttate dal malware per ottenere la persistenza.

CFF Explorer VIII - [Malware_Build_Week_U3.exe]

File Settings ?

File: Malware_Build_Week_U3.exe

- xo
- Dos Header
- Nt Headers
- File Header
- Optional Header
- Data Directories [x]
- Section Headers [x]
- Import Directory
- Resource Directory
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler
- Rebuilder
- Resource Editor
- UPX Utility

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

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	0295	SizedResource	
00007644	0105	LockResource	
00007654	01C7	LoadResource	
00007622	02B8	VirtualAlloc	
00007674	0124	GetModuleFileNameA	
0000768A	0126	GetModuleHandleA	
00007612	0086	FreeResource	
00007664	00A3	FindResourceA	
00007604	001B	CloseHandle	
000076DE	00CA	GetCommandLineA	
000076F0	0174	GetVersion	
000076F6	007D	ExitProcess	
0000770C	010F	HeapFree	
00007718	011A	GetLastError	
00007728	02DF	WriteFile	
00007734	029E	TerminateProcess	
00007748	00F7	GetCurrentProcess	
0000775C	02AD	UnhandledExceptionFilter	
00007778	00B2	FreeEnvironmentStringsA	
00007792	00B3	FreeEnvironmentStringsW	
000077AC	02D2	WideCharToMultiByte	
000077C2	0106	GetEnvironmentStrings	
000077DA	0108	GetEnvironmentStringsW	
000077F4	02BD	SetHandleCount	
00007806	0152	GetStdHandle	
00007816	0115	GetFileType	
00007824	0150	GetStartupInfoA	
00007836	0109	GetEnvironmentVariableA	
00007850	0175	GetVersionExA	

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

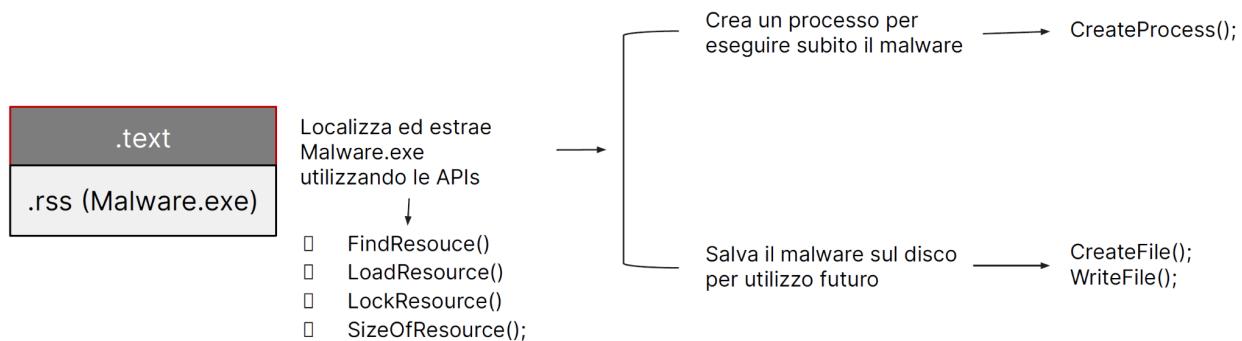
OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000077DA	000077DA	0108	GetEnvironmentStringsW
000077F4	000077F4	0260	SetHandleCount
00007806	00007806	0152	GetStdHandle
00007816	00007816	0115	GetFileType
00007824	00007824	0150	GetStartupInfoA
00007836	00007836	0109	GetEnvironmentVariableA
00007850	00007850	0175	GetVersionExA
00007860	00007860	019D	HeapDestroy
00007866	00007866	0198	HeapCreate
0000787C	0000787C	02BF	VirtualFree
0000788A	0000788A	02F2	RtlUnwind
00007896	00007896	0199	HeapAlloc
000078A2	000078A2	01A2	HeapReAlloc
000078B0	000078B0	027C	SetStdHandle
000078C0	000078C0	00AA	FlushFileBuffers
000078D4	000078D4	026A	SetFilePointer
000078E6	000078E6	0034	CreateFileA
000078F4	000078F4	00BF	GetPInfo
00007900	00007900	0089	GetACP
0000790A	0000790A	0131	GetOEMCP
00007916	00007916	013E	GetProcAddress
00007928	00007928	01C2	LoadLibraryA
00007938	00007938	0261	SetEndOfFile
00007948	00007948	0218	ReadFile
00007954	00007954	01E4	MultiByteToWideChar
0000796A	0000796A	01BF	LCMapStringA
0000797A	0000797A	01C0	LCMapStringW
0000798A	0000798A	0153	GetStringTypeA
0000799C	0000799C	0156	GetStringTypeW

La libreria Kernel32.dll contiene invece una vasta serie di funzioni per interagire con il sistema operativo, come ad esempio la manipolazione di file, la gestione della memoria e la creazione di thread/processi.

Il malware utilizza 51 funzioni di questa libreria. Tra esse spiccano all'occhio le funzioni:

- CreateFile, ReadFile, WriteFile che servono per la manipolazione di un file, per cui, molto probabilmente il malware creerà in una data locazione un file per i suoi scopi
- FindResource, LoadResource, LockResource, SizeOfResource utilizzate dal malware per l'accesso a risorse immagazzinate nella sezione risorse (.rsc o .rss)

La combinazione di questo tipo di funzione è tipica dei malware della categoria "dropper", in cui il malware vero e proprio viene nascosto nel segmento risorse e all'avvio dell'applicazione malevola esso viene estratto per essere eseguito immediatamente tramite la CreateProcess (non presente nella lista in esame), o salvato su disco per l'esecuzione futura con le funzioni CreateFile, WriteFile individuate.



5) Lo scopo della funzione chiamata alla locazione di memoria 00401021

La funzione chiamata all'indirizzo 00401021 è la funzione RegCreateKeyExA, che come visto in precedenza si trova nella libreria advapi32.dll e serve per creare una nuova chiave di registro.

Nello specifico la chiave viene creata sotto il path

“Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon”

```
* .text:00401000      push    ebp
* .text:00401001      mov     ebp, esp
* .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\\CurrentVersion\\Winlogon"
* .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
* .text:00401030      jmp    short loc_40107B
```

6) Come vengono passati i parametri alla funzione alla locazione 00401021

I parametri vengono passati nello stack tramite istruzioni “push” in ordine inverso rispetto alla firma della funzione mostrata in figura

```
C++
```

```
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
);
```

7) Che oggetto rappresenta il parametro alla locazione 00401017

Il parametro alla locazione 00401017 è una stringa che rappresenta il percorso nel registro (path) della chiave da creare.

8) Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.

```
> .text:0040101C          push    80000002h      ; hKey
> .text:00401021          call    ds:RegCreateKeyExA
> .text:00401027          test    eax, eax
> .text:00401029          jz     short loc_401032
```

Il codice nelle locazioni indicate controlla il risultato della chiamata alla funzione RegCreateKeyExA, memorizzato nel registro EAX. La funzione RegCreateKeyExA restituisce il valore 0 (**ERROR_SUCCESS**) ad indicare che l'operazione è stata completata con successo, mentre altri valori rappresentano diversi codici di errore che indicano il motivo del fallimento. Tramite l'istruzione `test` viene quindi per l'appunto testato questo valore senza la modifica del valore del registro (come per l'istruzione `cmp`) ma aggiornando di conseguenza il solo registro dei flag. Proseguendo viene utilizzata l'istruzione di `jump jz` in maniera che se il test dà come risultato 0, ad identificare che la creazione della chiave è andata a buon fine, viene effettuato un jump ad una sezione di codice per il proseguimento dell'applicazione.

9) Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C.

La traduzione in linguaggio C del codice Assembly indicato in precedenza può essere la seguente:

```
int retVal = RegCreateKeyExA(...);
if (retVal == 0) {
    ...
}
```

10) Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

Il valore del parametro ValueName viene valutato ed indicato in maniera molto immediata da IdaPro ed è pari a "GinaDLL". Esso corrisponde al parametro Hkey che rappresenta il nome della chiave di registro da creare.

```

...:00401032 loc_401032:          ; CODE XREF: sub_401000+29↑j
.00401032      mov     ecx, [ebp+cbData]    ; cbData
.00401035      push    ecx                 ; cbData
.00401036      mov     edx, [ebp+lpData]   ; lpData
.00401039      push    edx                 ; dwType
.0040103A      push    1                  ; Reserved
.0040103C      push    0                  ; Reserved
.0040103E      push    offset ValueName ; "GinaDLL"
.00401043      mov     eax, [ebp+hObject]
.00401046      push    eax                 ; hKey
.00401047      call    ds:RegSetValueExA
.0040104D      test    eax, eax
.0040104F      jz     short loc_401062
.00401051      mov     ecx, [ebp+hObject]
.00401054      push    ecx                 ; hObject
.00401055      call    ds:CloseHandle
.0040105B      mov     eax, 1
.00401060      jmp    short loc_40107B

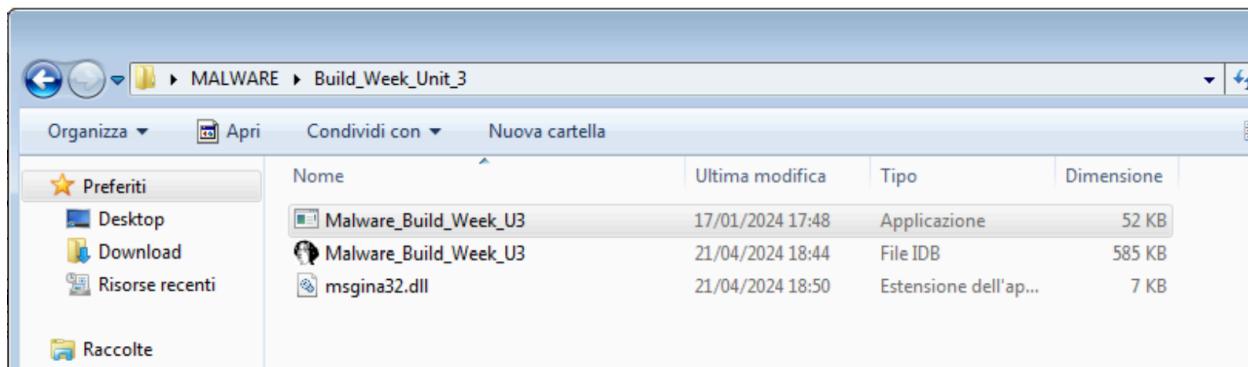
```

Analisi dinamica

1) 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

Il malware ha creato un nuovo file (msgina32.dll) all'interno della cartella, in precedenza non presente. Il nome di questa dll corrisponde al nome del parametro visto in precedenza che viene utilizzato nella creazione della chiave di registro, utilizzata molto probabilmente per la persistenza del malware.



2) Quale chiave di registro viene creata?

La chiave che viene creata è quella emersa già in precedenza durante l'analisi statica, ovvero la chiave GinaDLL sotto il path

HKEY\Software\Wow6432Node\Microsoft\WindowsNT\CurrentVersion\Winlogon

La chiave visualizzata in questo caso è leggermente diversa (presenta il campo Wow6432Node) ma si tratta della medesima chiave in quanto il sistema, per compatibilità delle applicazioni 32 bit che girano su sistemi a 64 bit, ridireziona automaticamente la chiave verso il percorso

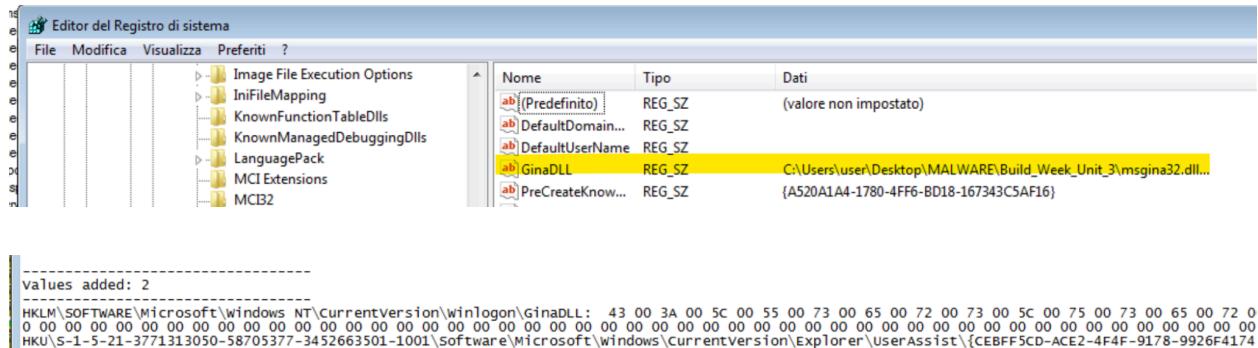
HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon precedentemente individuato.

18:50...	Malware_Build...	2512	RegCreateKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All Access, Disposition: REG_OPENED_EXISTING_KEY
18:50...	Malware_Build...	2512	RegSetKeyValue	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	KeySetInformationClass: KeySetGetInfo, KeyInformation: 0x400
18:50...	Malware_Build...	2512	RegGetValue	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	SUCCESS	Query: HKEY_TAGS, HandleType: 0x400, Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll

3) Quale valore viene associato alla chiave di registro creata?

Il valore associato alla chiave è di tipo stringa (REG_SZ), ed ha come valore il path C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll (campo Data della chiamata RegSetValue nella figura precedente).

La chiave è effettivamente presente nel registro di sistema, visualizzato tramite il tool regedit e la modifica è rilevata dal tool Regshot come mostrato in figura, avendo cura di prendere le due scansioni del registro per la comparazione prima e dopo l'avvio del malware



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

Le chiamate di sistema utilizzate per modificare il contenuto della cartella dove è presente l'eseguibile del file con il file msgina32.dll sono le chiamate di sistema CreateFile per la sua creazione, WriteFile per la scrittura del contenuto e CloseFile per l'operazione di chiusura finale del file. Queste chiamate sono presenti nella libreria di sistema Kernel32.dll come visto in precedenza

```

18:50: [!] Malware_Build_ 2512 [!] RegOpenKey HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Diagnos...
18:50: [!] Malware_Build_ 2512 [!] CreateFile C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
18:50: [!] Malware_Build_ 2512 [!] WriteFile C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
18:50: [!] Malware_Build_ 2512 [!] WriteFile C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
18:50: [!] Malware_Build_ 2512 [!] CloseFile C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll

```

Delineamento funzionalità malware

Calcolando l'hash tramite un tool come md5deep e dandolo in pasto ad un servizio come VirusTotal esso viene effettivamente rilevato ed identificato con grande probabilità come malware

```
C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
md5deep: WARNING: You are running a 32-bit program on a 64-bit system.
md5deep: You probably want to use the 64-bit version of this program.
7ce4f799946f0fa44e5b2b5e6a702f27 C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
```

Security vendor	Detection	Family
Alibaba	Trojan:Win32/Tiggre.387d5a16	AliCloud
AIYac	Gen:Variant:Fragtor.510142	Anti-AVL
Arcabit	Trojan:Fragtor.D7C8BE	Avast
AVG	Win32:Trojan-gen	Avira (no cloud)
BitDefender	Gen:Variant:Fragtor.510142	BitDefenderTheta
Bkav Pro	W32.Common.1467CBBC	ClamAV
CrowdStrike Falcon	Win/malicious_confidence_100% (W)	Cylance
Cynet	Malicious (score: 100)	DeepInstinct
		Trojan.Win.Generic.f3be1728
		Trojan/Win32.FakeGina
		Win32:Trojan-gen
		HEUR/AGEN.1326250
		Gen:NN.ZedlaF.36802.aq4@0clrOb
		Win.Trojan.Agent.595082
		Unsafe
		MALICIOUS

L'analisi dinamica conferma che siamo in presenza di un malware della categoria "dropper".

Il malware infatti non effettua attività di rete e non sono individuabili chiamate alla libreria Winsock, utilizzate ad esempio dai malware di tipo "downloader". Sono invece individuabili le chiamate FindResource, LoadResource, LockResource, SizeOfResource, per cui il file msgina.dll che viene creato all'avvio, viene estratto dalla sezione risorse

dell'applicazione e salvato sul disco.

Viene utilizzato poi il percorso dove viene salvata la dll per creare/modificare la chiave di registro GINADLL.

Effettuando una ricerca più approfondita, la libreria msgina.dll è una libreria di sistema effettivamente presente nei sistemi Windows più datati ed utilizzata per gestire l'interfaccia utente di accesso e autenticazione (login screen). Il suo nome sta per "Microsoft Graphical Identification and Authentication".

Il malware nel caso specifico effettua un attacco denominato "**GINA Interception**" tramite il quale il malware intercetta e sostituisce la libreria msgina.dll con una sua versione modificata tramite la quale può effettuare varie attività dannose, come ad esempio:

- Intercettare le credenziali di accesso: la libreria modificata può acquisire il nome utente e la password inseriti dall'utente durante il processo di login e inviarli all'attaccante.
- Modificare il comportamento di login: l'attaccante può modificare il comportamento del login, ad esempio permettendo l'accesso a utenti non autorizzati o impedendo l'accesso a utenti legittimi.
- Mantenere il controllo sul sistema: La libreria modificata può consentire all'attaccante di mantenere il controllo sul sistema, anche dopo che l'utente ha effettuato il login.

La libreria msgina.dll è stata comunque deprecata e sostituita nei sistemi operativi Windows Vista e superiori con sistemi di autenticazione più moderni e sicuri.