

# Report – M6W24D5

## Malware Analysis

### Indice generale

Introduzione e Parole Chiave.....	2
Analisi Statica.....	3
Funzione Main( ).....	4
Sezioni dell'eseguibile.....	4
Librerie importate.....	5
Locazione di memoria 00401021.....	5
Locazione di memoria 00401017.....	6
Locazione di memoria 00401027 e 00401029.....	6
Locazione di memoria 00401047.....	6
Analisi Dinamica.....	7
Esecuzione del malware.....	8
Analisi Procmon – registro.....	9
Analisi Procmon – file system.....	9
Conclusioni.....	10

# Introduzione e Parole Chiave

Viene richiesto di applicare tutte le conoscenze sui malware e sulle loro tecniche di analisi maturate nelle ultime settimane per effettuare uno studio il più approfondito possibile di uno specifico malware fornito.

Prima di iniziare richiamiamo alcuni concetti fondamentali:

- **Malware** si tratta di una categoria di software molto ampia, con funzionalità diversissime gli uni dagli altri, dalla criptazione di file al furto di informazioni sensibili. Quello che hanno in comune è la capacità di nuocere al sistema e agli utenti con cui vengono a contatto.
- **Analisi statica** consiste nell'utilizzo di strumenti che consentono la lettura del codice malevolo, applicando le conoscenze acquisite si è in grado di inferire il funzionamento del malware prima che questo venga eseguito.
- **Analisi dinamica** consiste nell'utilizzo di strumenti che consentono di studiare il comportamento del malware in esecuzione in un ambiente controllato, confrontando lo stato del sistema e gli effetti del malware prima, durante e dopo il suo avvio.

# 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? Descrivere brevemente almeno due 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. Utilizzare le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.*
- *Qual è 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?*
- *Qual è il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029?*
- *Con riferimento al quesito precedente, tradurre il codice Assembly nel corrispondente costruito C.*
- *Valutare ora la chiamata alla locazione 00401047, qual è il valore del parametro "ValueName"?*

## Funzione Main( )

La funzione principale del malware è presente all'indirizzo di memoria 004011D0.

Le sono passati 3 parametri (argc, argv, envp) e dichiara 5 variabili locali (hModule, Data, var\_117, var\_8, var\_4); possiamo distinguerli grazie al loro offset rispetto al puntatore EBP, se positivo infatti fa riferimento a locazioni di memoria che precedono EBP e quindi facenti parte dello stack della funzione chiamante, se l'offset è negativo fanno parte delle locazioni di memoria assegnate allo stack appena creato per la nuova funzione chiamata.

```
.text:004011D0 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:004011D0 _main proc near ; CODE XREF: start+AF↓p
.text:004011D0
.text:004011D0 hModule = dword ptr -11Ch
.text:004011D0 Data = byte ptr -118h
.text:004011D0 var_117 = byte ptr -117h
.text:004011D0 var_8 = dword ptr -8
.text:004011D0 var_4 = dword ptr -4
.text:004011D0 argc = dword ptr 8
.text:004011D0 argv = dword ptr 0Ch
.text:004011D0 envp = dword ptr 10h
.text:004011D0
```

## Sezioni dell'eseguibile

Il codice di un file Portable Executable o PE (cioè i molto comuni .exe) è suddiviso in più sezioni per facilitarne la lettura, analizzando questo specifico malware ne sono state identificate quattro:

- **.text** si tratta della parte principale del programma che contiene il codice eseguibile
- **.idata** in questa sezione vengono mantenute le funzioni e informazioni importate
- **.rdata** informazioni di sola lettura come stringhe e costanti
- **.data** contiene le variabili globali accessibili a tutte le funzioni del programma

```
.text:004065CA ; [
.text:00406640 ; [
.text:00406646 te
.text:00406646 [
.idata:00407000 ;
.idata:00407000 ;
.idata:00407000 ;
.idata:00407000 ;
.idata:00407000 ;
.idata:00407000 ;
.idata:00407000 ;
.idata:00407000 ;
.rdata:0040798H
.rdata:0040798C
.rdata:0040799B
.rdata:0040799C
.rdata:0040799E
.rdata:004079AD
.rdata:004079AD
.rdata:004079AD
.data:00408000 ;
.data:00408000 ;
.data:00408000 ;
.data:00408000 ;
.data:00408000 ;
.data:00408000 ;
```

## Librerie importate

I programmi, compresi i malware, utilizzano delle librerie messe a disposizione dal sistema operativo per importare funzioni standard senza doverle avere già memorizzate.

Questo malware sfrutta le funzioni di due librerie:

- ADVAPI32 o Advanced Windows 32 Base API, fa parte delle librerie di servizio avanzate che affincano la libreria Kernel, contengono funzioni per la gestione dei registri di windows, accensione e spegnimento del sistema, gestione degli utenti e creazione/start/stop dei servizi windows.

Le funzioni caricate in particolare sono “RegSetValueExA” e “RegCreateKeyExA”, per creare una nuova chiave di registro e assegnarle un valore, probabilmente per ottenere la persistenza del malware, cioè il suo avvio automatico assieme al sistema operativo.

- KERNEL32 è una libreria fondamentale che permette la gestione della memoria, gestisce le operazioni di I/O (input/output), la creazione di processi e thread e funzioni di sincronizzazione.

Tra le funzioni caricate notiamo “CreateFileA”, “ReadFile” e “WriteFile”, presumibilmente il malware crea o modifica dei file per garantire il suo funzionamento.

Address	Ordinal	Name	Library
00407000		RegSetValueExA	ADVAPI32
00407004		RegCreateKeyExA	ADVAPI32
0040700C		SizeofResource	KERNEL32
00407010		LockResource	KERNEL32
00407014		LoadResource	KERNEL32
00407018		VirtualAlloc	KERNEL32

## Locazione di memoria 00401021

La funzione presente alla locazione di memoria specificata è “RegCreateKeyExA” e serve per creare una nuova chiave di registro vuota alla quale verrà successivamente assegnato un valore.

I parametri necessari al corretto funzionamento della funzione vengono caricati sullo stack dalla funzione chiamante in un ordine ben preciso secondo la metodologia LIFO (last-in first-out).

```
.text:00401000
* .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\\CurrentVe"...
* .text:0040101C      push    80000002h     ; hKey
* .text:00401021      call    ds:RegCreateKeyExA
```

## Locazione di memoria 00401017

In questa locazione di memoria è contenuto uno dei parametri della funzione discussa sopra: “lpSubKey”; già dalle prime due lettere del nome (“lp”) intuivamo che sia un puntatore e andando a consultare la documentazione della libreria si evince che punti ad una sottochiave del registro specificato da un altro dei parametri (“hKey”), questa sottochiave sarà effettivamente quella che andremo a creare.

```
.text:00401015      push    0                ; reserved
.text:00401017      push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040101C      push    80000002h        ; hKey
```

## Locazione di memoria 00401027 e 00401029

Queste due locazioni memoria corrispondono ad un salto condizionale: la prima “test” compara i suoi parametri con una operazione AND su ogni bit e setta di conseguenza ZeroFlag e CarryFlag del registro EFLAGS, poi questi due flag vengono consultati dalla funzione successiva per decidere se effettuare o no il salto.

In questo caso la funzione è “jz” e questa esegue il salto se la ZeroFlag è settata a 1, e questo è possibile solo se entrambi i parametri passati a “test” (in questo caso sono uguali) sono uguali a 0.

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

Una reinterpretazione di questo codice assembly nel linguaggio C potrebbe essere:

```
int a = eax;
If (a==0) {
    Funzione_401032( );
}
```

## Locazione di memoria 00401047

In questa locazione di memoria troviamo la funzione “RegSetValueExA” che assegna un valore alla chiave di registro specificata, il suo parametro “ValueName” va ricercato nella righe che la precedono, in particolare all’indirizzo 0040103e dove possiamo leggere il valore assegnato al parametro che è “GinaDLL”.

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

## Analisi Dinamica

*Preparare l'ambiente ed i tool per l'esecuzione del malware (suggerimento: avviare 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 gli indizi che avete raccolto finora per rispondere alla domanda.*

*Analizzare ora i risultati di Process Monitor (consiglio: utilizzare il filtro con il nome dell'eseguibile per estrarre solo le modifiche apportate al sistema da parte del malware, fate click su "Add" e poi "Apply" come abbiamo visto durante la lezione di teoria).*

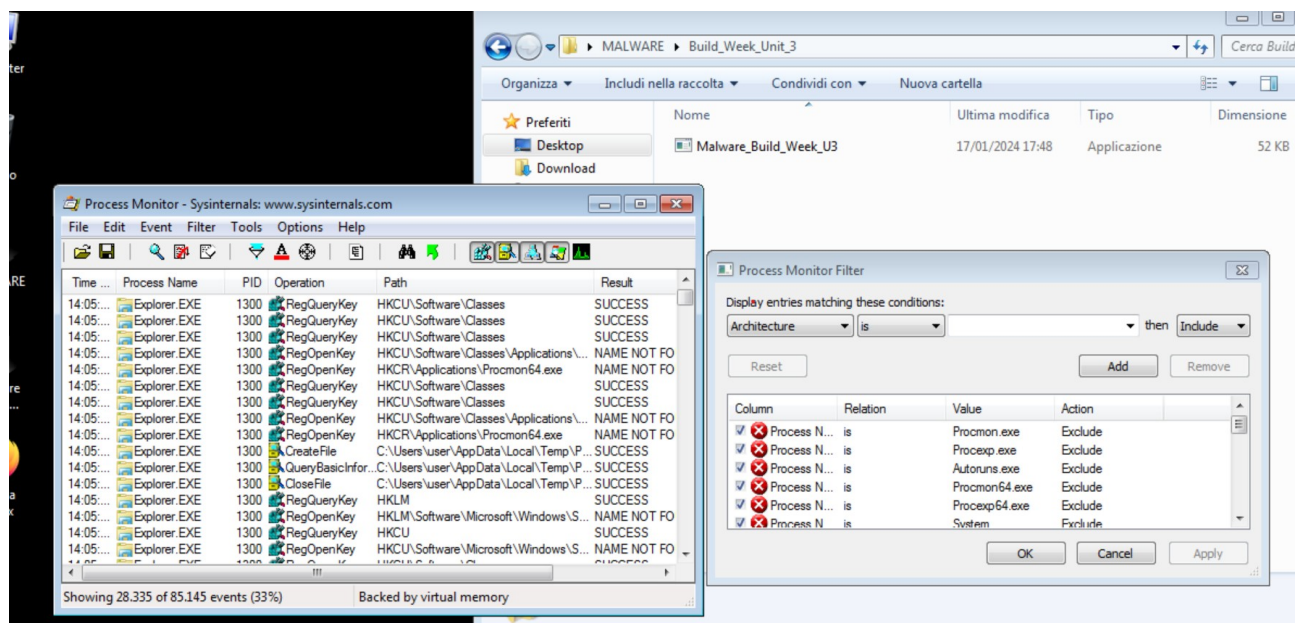
*Filtrate includendo solamente l'attività sul registro di Windows:*

- *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?*

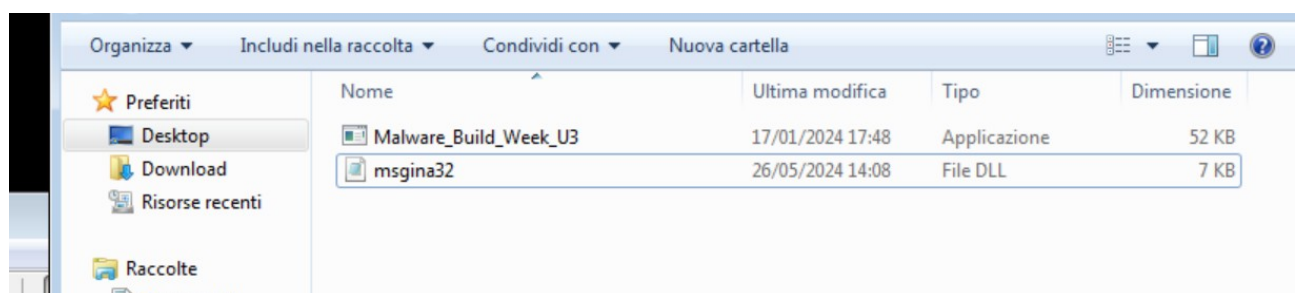
Iniziamo come suggerito con l'apertura del tool Procmon resettandone i filtri, nella cartella di origine del malware possiamo notare che non sono presenti altri file oltre al suo eseguibile.



## Esecuzione del malware

Il cambiamento è lampante dopo aver eseguito il malware: è stato creato un nuovo file “msgina.dll”. Analizziamolo: l'estensione “.dll” ci indica si tratti di una libreria, inoltre il nome “gina.dll” è stato già visto durante l'analisi statica durante l'analisi delle funzioni che creano una nuova chiave di registro windows.

Risulta ragionevole pensare che questo file sia una copia del malware che verrà eseguito con il riavvio del sistema.

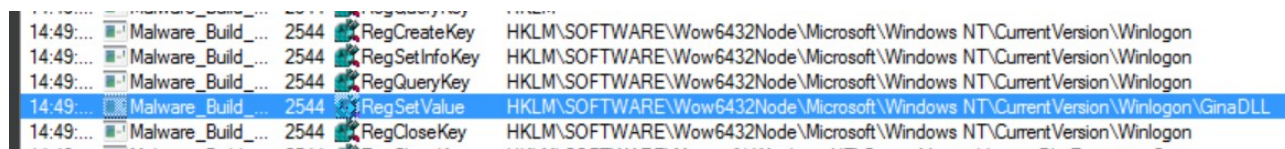




## Analisi Procmon – registro

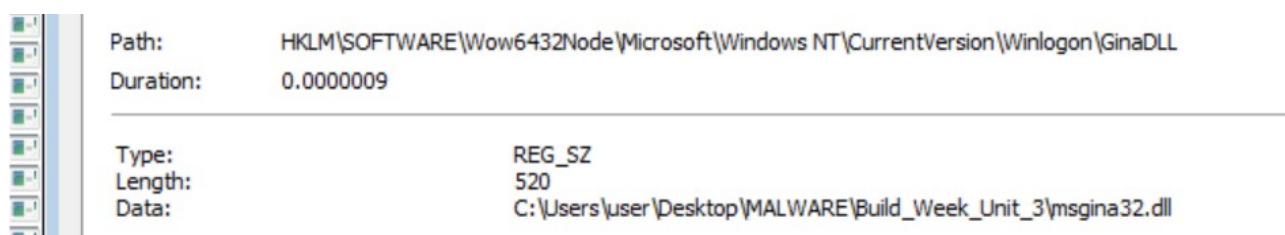
Filtrando le azioni sulle chiavi di registro effettuate dal solo malware notiamo che la maggior parte sono funzioni di lettura, c'è una sola voce che viene modificata ed è esattamente quella identificata durante la fase di analisi statica, cioè "...\\Winlogon\\GinaDLL".

Le librerie Winlogon gestiscono l'accesso al sistema interagendo con un provider di rete e delle librerie di collegamento dinamico di identificazione grafica e autenticazione che sono le GinaDLL.



14:49:...	Malware_Build_...	2544	RegCreateKey	HKLM\\SOFTWARE\\Wow6432Node\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon
14:49:...	Malware_Build_...	2544	RegSetInfoKey	HKLM\\SOFTWARE\\Wow6432Node\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon
14:49:...	Malware_Build_...	2544	RegQueryKey	HKLM\\SOFTWARE\\Wow6432Node\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon
14:49:...	Malware_Build_...	2544	RegSetValue	HKLM\\SOFTWARE\\Wow6432Node\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\GinaDLL
14:49:...	Malware_Build_...	2544	RegCloseKey	HKLM\\SOFTWARE\\Wow6432Node\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon

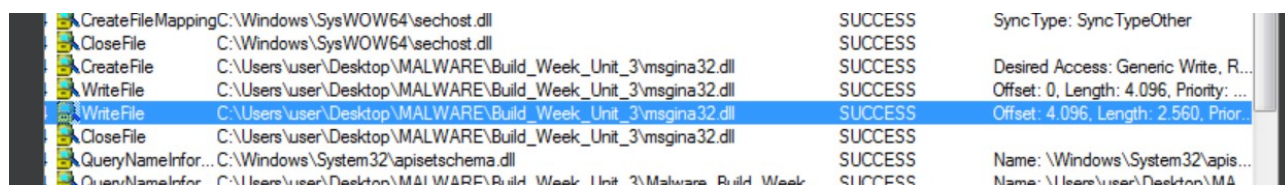
Quindi la chiave di registro viene modificata e il suo valore modificato, in questo caso cambiando il path della libreria al file creato dal malware:



Path:	HKLM\\SOFTWARE\\Wow6432Node\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\GinaDLL
Duration:	0.0000009
Type:	REG_SZ
Length:	520
Data:	C:\\Users\\user\\Desktop\\MALWARE\\Build_Week_Unit_3\\msgina32.dll

## Analisi Procmon – file system

Filtrando le azioni sulle modifiche al file system ci rendiamo conto che ricalcano quelle precedenti sulle chiavi di registro: vengono aperti e letti molti file ma solamente uno viene creato e modificato ed è proprio il file "msgina.dll" che ritroviamo nella cartella del malware.



CreateFileMapping	C:\\Windows\\SysWOW64\\sechost.dll	SUCCESS	SyncType: SyncTypeOther
CloseFile	C:\\Windows\\SysWOW64\\sechost.dll	SUCCESS	
CreateFile	C:\\Users\\user\\Desktop\\MALWARE\\Build_Week_Unit_3\\msgina32.dll	SUCCESS	Desired Access: Generic Write, R...
WriteFile	C:\\Users\\user\\Desktop\\MALWARE\\Build_Week_Unit_3\\msgina32.dll	SUCCESS	Offset: 0, Length: 4.096, Priority: ...
WriteFile	C:\\Users\\user\\Desktop\\MALWARE\\Build_Week_Unit_3\\msgina32.dll	SUCCESS	Offset: 4.096, Length: 2.560, Prior...
CloseFile	C:\\Users\\user\\Desktop\\MALWARE\\Build_Week_Unit_3\\msgina32.dll	SUCCESS	
QueryNameInfor...	C:\\Windows\\System32\\apisetschema.dll	SUCCESS	Name: \\Windows\\System32\\apis...
QueryNameInfor...	C:\\Users\\user\\Desktop\\MALWARE\\Build_Week_Unit_3\\Malware_Build_Week	SUCCESS	Name: \\Users\\user\\Desktop\\MA...

## Conclusioni

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

Il malware ha modificato una chiave di registro associata a Winlogon sostituendo la libreria GinaDLL con una sua versione malevola.

Considerando che Winlogon si occupa della gestione degli accessi al sistema il malware va a compromettere questa funzionalità consentendo l'accesso a utenti non autorizzati.