PROGETTO MODULO 6

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.

Malware Analysis

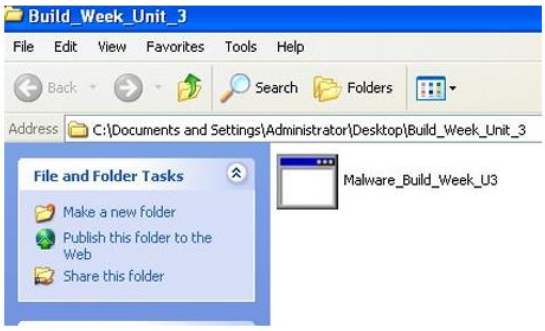
Con riferimento al Malware in analisi, spiegare:

- 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 costrutto C.
- Valutate ora la chiamata alla locazione **00401047**, qual è il valore del parametro «ValueName»?

Malware Analysis

Analisi dinamica

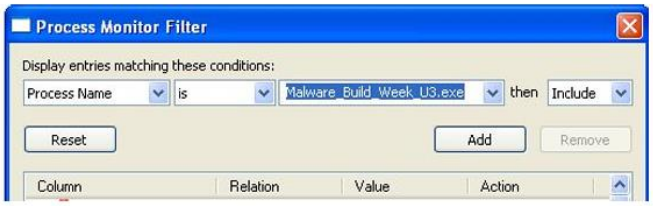
Preparate l'ambiente ed i tool per l'esecuzione del Malware (suggerimento: avviate principalmente Process Monitor ed assicurate di eliminare ogni filtro cliccando sul tasto «reset» quando richiesto in fase di avvio). Eseguite il Malware, facendo doppio click sull'icona dell'eseguibile



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.



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 statica

1. Aprendo il codice con il disassembler IDA Pro, si evince che i parametri passati alla funzione **main** sono 3 e sono evidenziati nella sequente figura. Infatti sono indicati con un offset positivo rispetto a EBP:

```
.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
```

2. Le variabili dichiarate all'interno della funzione **main** sono 5, ovvero quelle che hanno offset negativo rispetto a EBP:

```
.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
```

3. Attraverso il tool CFF Explorer, si visualizzano le sezioni di cui è composto l'eseguibile alla voce "Section Headers":

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	00003EA8	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040

   a. La sezione **.text** è quella in cui si trovano le istruzioni vere e proprie che la CPU esegue quando viene avviato l'eseguibile;
   b. La sezione .**rdata** contiene le funzioni e le librerie importate ed esportate;
   c. La sezione **.data** contiene dati e variabili globali del programma;
   d. La sezione **.rsrc** contiene le risorse usate dall'eseguibile come icone, immagini, menù e stringhe.

4. Da una analisi statica fatta con IDA Pro e CFF, le librerie che risultano importate sono:
   a. **KERNEL32.dll**: è una libreria che contiene le funzioni principali per interagire con il sistema operativo, come per la manipolazione dei file e la gestione della memoria;
   b. **ADVAPI32.dll**: è la libreria che contiene le funzioni per interagire con i servizi ed i registri del sistema operativo Microsoft.

IDA View-A | Hex View-A | Structures | Enums | Imports | Exports

Address	Ordinal	Name	Library
00407000		RegSetValueExA	ADVAPI32
00407004		RegCreateKeyExA	ADVAPI32
00407074		GetStartupInfoA	KERNEL32
00407078		GetEnvironmentVariableA	KERNEL32
0040707C		GetVersionExA	KERNEL32
00407080		HeapDestroy	KERNEL32
00407084		HeapCreate	KERNEL32
00407088		VirtualFree	KERNEL32
0040708C		RtlUnwind	KERNEL32
00407090		HeapAlloc	KERNEL32
00407094		HeapReAlloc	KERNEL32
00407098		SetStdHandle	KERNEL32

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

Analizzando le funzioni contenute all'interno della libreria **ADVAPI32.dll**, che sono **RegCreateKeyExA** e **RegSetValueExA**, possiamo da una semplice analisi statica ipotizzare che il malware in questione cerchi di ottenere la persistenza modificando una chiave di registro.
La persistenza è quel metodo che il malware usa per indurre il sistema operativo ad avviare il malware all'avvio del sistema , senza che esso debba essere avviato da un'azione da parte di un utente. Quindi con la funzione RegCreateKeyExA, il malware crea una nuova chiave di registro oppure ne apre una già esistente e poi con la funzione RegSetValueExA, ne modifica il valore.

Le funzioni che invece sono all'interno della libreria KERNEL32.dll, sono molteplici. Inizialmente, a catturare l'attenzione sono le funzioni **FindResourceA**, **LoadResource**, **LockResource**, **SizeofResource**. Queste

funzioni infatti sono di solito usate dai malware della famiglia dei **dropper** ovvero programmi malevoli che contengono al loro interno un malware. Di solito questo malware è contenuto nella sezione resource (.rss/.rsc) dell'eseguibile, pertanto il programma si serve delle funzioni sopracitate per cercare il malware in questa sezione, estrarlo, caricarlo in memoria per eseguirlo e eventualmente salvarlo per un utilizzo futuro.
La presenza della funzione **VirtualAlloc** fa pensare alla possibilità che il programma tenti di allocare memoria virtuale per il processo che servirà ad eseguire il malware. La **CreateFileA** e la **WriteFile** potrebbero essere usate per salvare il malware sul pc infetto per un utilizzo futuro.
Ci sono altre funzioni interessanti che fanno pensare ad una manipolazione della memoria e dei file.

5. La funzione chiamata alla locazione di memoria 00401021 è la **RegCreateKeyExA**. Come detto in precedenza, questa funzione crea una nuova chiave di registro o ne apre una esistente.
In particolare accetta in input tali parametri:

```
LSTATUS RegCreateKeyExA(
  [in]            HKEY                        hKey,
  [in]            LPCSTR                      lpSubKey,
                  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
);
```

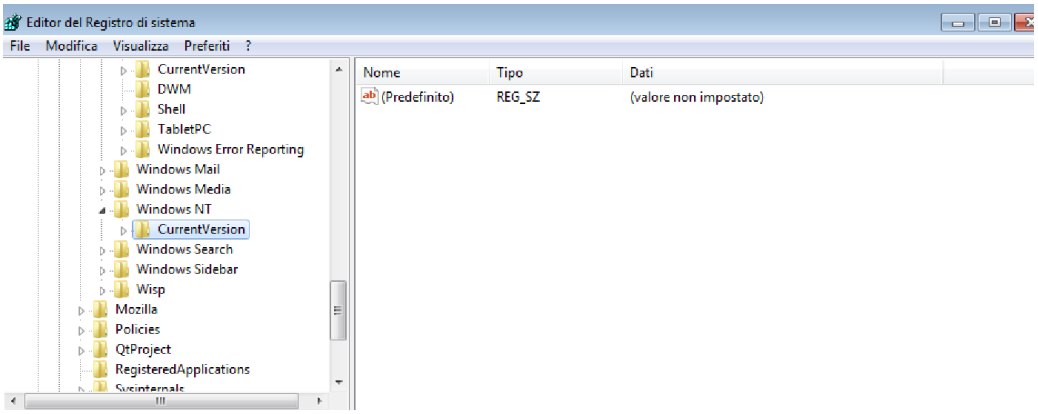
Questi parametri vengono spinti sullo stack con un'operazione di **push** visibile da codice:

```
.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
.text:00401027    test    eax, eax
```

Da qui si evince che viene richiesto in ingresso un handle per una chiave del registro di sistema, in questo caso pari a 80000002h e la sottochiave "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon" che è quella che deve essere creata o aperta. L'input dwOptions pari a 0 indica l'impostazione di default di archiviare le informazioni in un file e mantenerle al riavvio del sistema.
Il valore di samDesired pari a 0F003Fh indica il tipo di accesso richiesto, in questo caso si richiede accesso completo alla chiave del registro di sistema, ovvero il permesso di leggere, scrivere, eliminare e modificare la chiave.

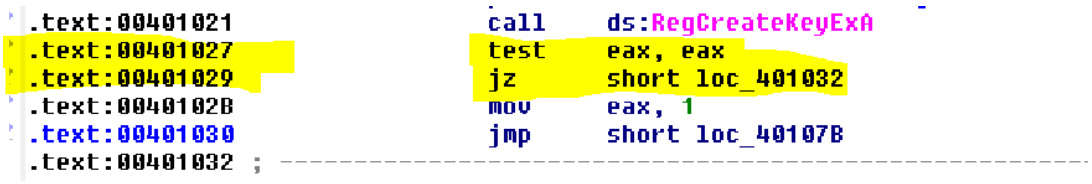
6. La funzione precedentemente analizzata riceve i parametri sullo stack attraverso delle istruzioni di **push**. Questa operazione è l'equivalente di aggiungere un piatto alla cima di una pila di piatti che compongono lo stack.

7. L'oggetto alla locazione 00401017 è un puntatore a una stringa che rappresenta la sottochiave da creare o aprire, "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon". Attraverso **regedit** si può ricavare il percorso e capire che la root key di interesse è **HKEY_LOCAL_MACHINE,** che contiene le configurazioni della macchina.



Questo percorso contiene informazioni importanti relative alla versione del sistema operativo Windows NT installato sul computer. Questa chiave è fondamentale per molte operazioni di configurazione del sistema e per i programmi che devono interagire con il sistema operativo.

8. Il codice compreso tra 00401027 e 00401029 è il seguente:



C'è dunque una istruzione condizionale seguita da un jump condizionale di tipo jz. In particolare viene eseguita l'operazione di **test** tra il registro accumulatore **eax** e se stesso. Questa operazione è equivalente a eseguire l'AND bit a bit tra eax e se stesso, aggiornando poi il bit 6 del registro **EFLAGS** che si chiama **ZERO FLAG(ZF)**. Questo flag si valorizza a 1 se il risultato di un'operazione, in questo caso l'AND bit a bit, risulta 0.
Quindi l'unico caso per avere ZF=1 come risultato di un AND bit a bit tra un numero e se stesso, è che il numero stesso sia 0. Di fatto questa operazione di test è spesso usata per verificare se un valore è 0.
Il conditional jump presente **jz** indica che il salto alla locazione 00401032, avviene se lo zero flag è valorizzato a 1.
Quindi se eax è pari a 0 (ZF=1), si salta alla locazione 00401032, altrimenti si prosegue con le istruzioni alla locazione 0040102B.

9. Queste istruzioni Assembly equivalgono al costrutto IF-ELSE in C. Si riporta uno pseudo-codice:

```
if(eax==0)
{
  //salta alla locazione 00401032 e chiama la funzione
RegSetValueExA();
}
else
{
 eax=1;
//salta alla locazione 0040107B dove viene pulito lo stack
 return;
}
```

Ovviamente non si avrà eax, ma la rispettiva variabile da controllare.

10. All’indirizzo 00401047 è chiamata la funzione RegSetValueExA. Il valore di ValueName è “GinaDLL”.

Analisi Dinamica

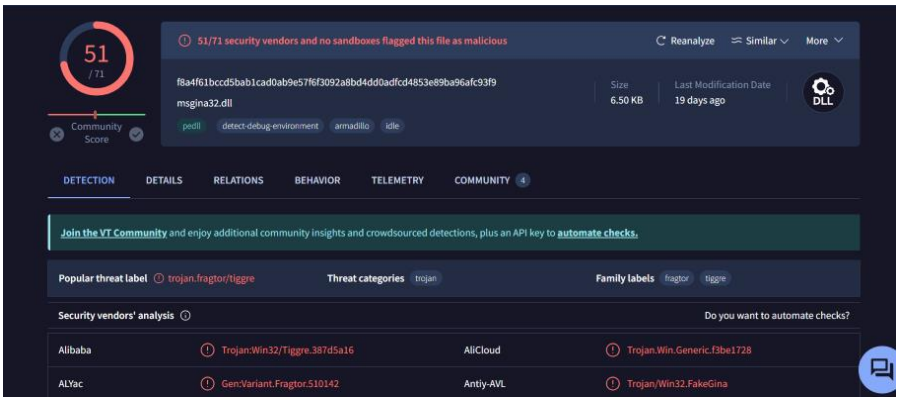
Per l’analisi dinamica, sono stati avviati prima il tool **Process Monitor**, è stata fatta una prima istantanea dei registri a monte dell’esecuzione del malware con **RegShot** ed è stato anche avviato **Process Explorer**.

È stato eseguito il malware e si è notato che nella cartella dove è contenuto il suo eseguibile, è stato creato un file **msgina32.dll** che è omonimo di un componente del sistema operativo Windows che gestisce l'interfaccia tra il sistema operativo e il processo di autenticazione degli utenti. In particolare, è responsabile della schermata di accesso e del processo di login.

Dunque abbiamo con un’analisi dinamica in ambiente controllato, verificato che effettivamente il programma ha salvato un malware che ha precedentemente estratto, in un’altra locazione di memoria e che desidera che venga eseguito all’avvio del sistema.

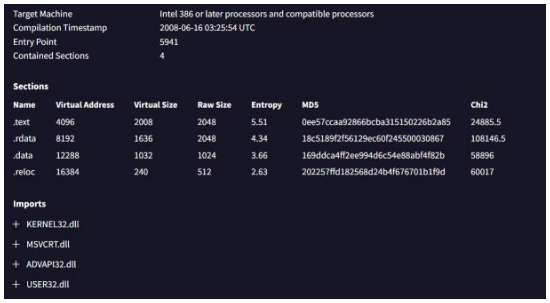
L’intento potrebbe essere quello di sostituire il file lecito con il file malevolo e ottenere accesso non autorizzato al sistema e informazioni su di esso o comunque camuffarsi da componente lecito del sistema.

La verifica che il file sia malevolo è stata eseguita calcolando l’hash di tale file con il tool **md5deep** e poi ricercato su **VirusTotal:**



Molti dei vendor di sicurezza analizzati da VirusTotal, definiscono questo file come un Trojan, alcuni specificando che sia una backdoor.

Altre informazioni utili, sono legate alla macchina target, numero di sezioni, librerie importate:



Domini e indirizzi IP che cerca di contattare:

Contacted Domains (3)

Domain	Detections	Created	Registrar
fp2e7a.wpc.2be4.phicdn.net	1 / 90	2014-11-14	GoDaddy.com, LLC
fp2e7a.wpc.phicdn.net	0 / 90	2014-11-14	GoDaddy.com, LLC
login.live.com	0 / 90	1994-12-28	CSC CORPORATE DOMAINS, INC.

Contacted IP addresses (9)

IP	Detections	Autonomous System	Country
192.229.221.95	1 / 90	15133	US
20.190.159.0	1 / 90	8075	IE
20.190.159.23	1 / 90	8075	IE
20.190.159.4	1 / 90	8075	IE
20.190.159.64	1 / 90	8075	IE
20.190.159.68	1 / 90	8075	IE
40.126.31.67	0 / 90	8075	IE
40.126.31.71	0 / 90	8075	IE
40.126.31.73	0 / 90	8075	IE

Tecniche di attacco:

MITRE ATT&CK Tactics and Techniques

- Execution TA0002
- Persistence TA0003
- Privilege Escalation TA0004
- Defense Evasion TA0005
- Discovery TA0007

Azioni sul file system:

File system actions

Files Opened

- C:\Program Files (x86)\Common Files\Oracle\Java\javapath\
- C:\Users\user\Desktop\library.dll
- C:\Users\user\Desktop\library.dll.123.Manifest
- C:\Users\user\Desktop\library.dll.124.Manifest
- C:\Users\user\Desktop\library.dll.2.Manifest
- C:\Windows\AppPatch\sysmain.sdb
- C:\Windows\Globalization\Sorting\sortdefault.nls
- C:\Windows\SYSTEM32\ole32.dll
- C:\Windows\SysWOW64\
- C:\Windows\SysWOW64\ADVAPI32.dll

Files Written

- \Device\ConDrv\\Connect

Azioni sui registri:

Registry actions

Registry Keys Opened

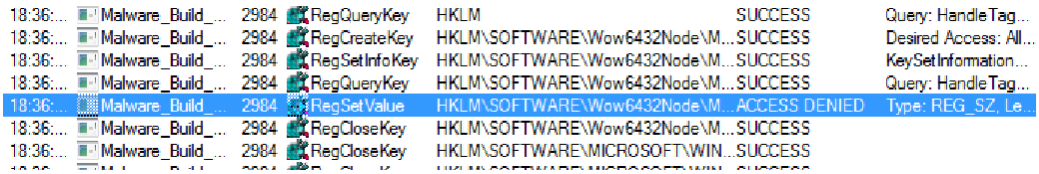
- HKEY_CURRENT_USER\Software\Microsoft\CTF\DirectSwitchHotkeys
- HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion
- HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\AppModel\Lookaside\Packages
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\FontLink\SystemLink
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\DllNXOptions
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\cmd.exe
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\loaddll32.exe
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\rundll32.exe

Su **Procmon** viene attivato un filtro per visualizzare solo il processo inerente il malware sotto analisi. Dalle operation che vengono eseguite, si evince che il malware fa partire il processo e crea un thread. Con le operazioni di **Load Image** carica dei file (un .exe e vari .dll).

Seguono poi delle operazioni sui registri e sul file system.

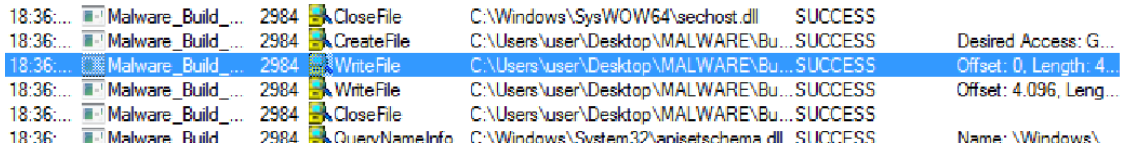
1. Filtrando per avere visione sulle sole attività di registro, si evince che vengono fatte diverse aperture di chiavi di registro, query su determinati valori, chiusure di registri. Sono tutte inerenti la root key HKLM che continee le configurazioni della macchina. L'unica chiamata dell'operazione **RegCreateKey** è quella che serve a creare la chiave HKLM\SOFTWARE\Wow6432Node\Microsoft\WindowsNT\CurrentVersion\Winlogon, con accesso richiesto di tipo "All access".
2. Il valore associato alla chiave creata è "GinaDLL". Lo si vede dall'operation **RegSetValue** HKLM\SOFTWARE\Wow6432Node\Microsoft\WindowsNT\CurrentVersion\Winlogon\GinaDLL.

   Nel particolare, nella sezione "Detail", si vedono i dettagli di ciò che è stato inserito:
   - **Type: REG_SZ**: indica il tipo di dato che viene impostato nel registro. In questo caso, "REG_SZ" indica che si tratta di una stringa di testo;
   - **Length 520:** indica la lunghezza dei dati che vengono inseriti nel valore del registro;
   - **Data C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll**: percorso del file che il processo sta cercando di inserire nel valore del registro.



   Le chiavi di registro modificate si possono anche verificare con la seconda istantanea su Regshot da comparare con la precedente.

3. Filtrando per avere visione sulle sole attività che agiscono sul file system, si nota che le chiamate che hanno modificato il contenuto della cartella dove è contenuto l'eseguibile del malware sono la **CreateFile** che richiede un accesso in lettura e scrittura e la **WriteFile** che scrive con successo 4096 bytes nel file:



Dall'analisi statica e dinamica eseguite si può concludere di avere a che fare con un malware della famiglia dei **dropper.**

In particolare questo programma inizialmente modifica le chiavi di registro, aggiungendone una nuova chiave con la funzione **RegCreateKeyExA** e ne modifica il valore all'interno con la **RegSetValueExA.** Il valore che vi inserisce è relativo al path del file che intende far eseguire all'avvio del sistema operativo, senza che sia l'utente a doverci cliccare sopra, ottenendo così la persistenza. Tale file è **msgina32.dll**, che sembra essere un Trojan che contatta anche dei domini esterni, che viene estratto dalla sezione risorse del dropper, attraverso le funzioni **FindResourceA, LoadResource, LockResource** e **SizeofResource** per poi copiarlo nella stessa directory del file eseguibile attraverso le funzioni **CreateFile** e **WriteFile**. Dunque il file si camuffa da file lecito, avendo lo stesso nome di un file noto del sistema operativo Windows che fornisce funzionalità per l'interfaccia grafica utente (GUI) utilizzata durante il processo di accesso e autenticazione degli utenti sul sistema Windows, gestisce la schermata di accesso, dove gli utenti inseriscono le proprie credenziali per accedere al sistema e fornisce i mezzi per personalizzare e configurare l'aspetto e il comportamento della schermata di accesso. Questo è un comportamento di molti malware, che dunque si fingono file innocui ma che invece, una volta eseguiti, generano un danno al sistema.

N.B: il percorso della chiave di registro visibile su IDAPro è leggermente diverso da quello visibile con ProcMon. Questo dimostra come l'analisi statica porta a ipotesi che vanno poi sempre confrontate e confermate con l'analisi dinamica.