Esercizio 8
Modulo 6

Malware Analysis

Malware analizzato: Malware_Build_Week_U3

Oggi andremo ad analizzare un malware sia con l'analisi statica, sia con l'analisi dinamica.

Analisi Statica:
Viene eseguita senza eseguire il codice o l'applicazione. L'analisi si basa sui file sorgenti, il codice sorgente o i byte code senza eseguire realmente il programma.

Con riferimento a quello scritto sopra andiamo ad aprire ed analizzare il malware (Build_Week_Unit_3) con IDA.
IDA (Interactive Disassembler): È un software che consente di esaminare il codice binario di un programma o di un file eseguibile, andando a tradurlo in linguaggio assembly. Questo ci permette di leggere e capire come sono fatti i malware.

Quanti parametri sono passati alla funzione Main()?

```
; int __cdecl main(int argc,const char **argv,const char *envp)
_main proc near

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h
```

Nei Rettangolo evidenziato in giallo possiamo vedere i parametri passati alla funzione main() che sono: **argc, argv e envp**, i quali vengono anche specificati tra parentesi dopo il nome della funzione.
Nota: avviando IDA con all'interno il malware ci si apriranno alcune videate con molte informazioni. In questo caso lo screen viene effettuato nella prima videata con all'interno tutto il codice primario.

Quante variabili sono dichiarate all'interno della funzione Main()?

```
; int __cdecl main(int argc,const char **argv,const char *envp)
_main proc near

hModule= dword ptr -11Ch
Data= byte ptr -118h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h
```

Nel rettangolo in giallo possiamo vedere le variabili dichiarate all'interno della funzione Main() che sono: **hModule, Data, var_8 e var_4**

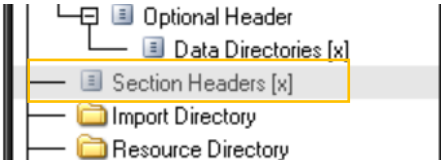
Nota:
I parametri dalle variabili vengono distinti grazie al valore dell'offset in base al registro EBP.

Quindi sappiamo che:
I parametri si troveranno ad un offset positivo;
Le variabili si troveranno ad un offset negativo;

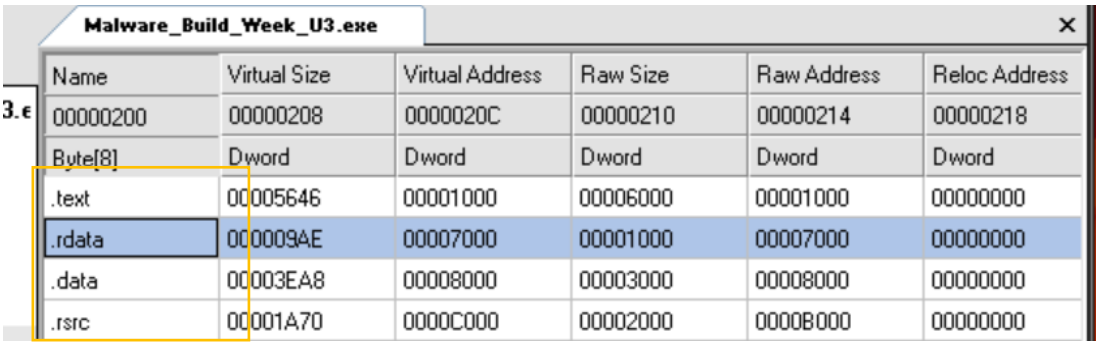
Quali sezioni sono presenti all'interno del file eseguibile?

Per analizzare le selezioni andiamo ad utilizzare un altro software, **CFF Explorer.**
Un software che ci permette di analizzare la struttura interna di file eseguibili. Tra le varie funzionalità, abbiamo l'esplorazione delle sezioni dei file, la gestione delle risorse e la visualizzazione delle intestazioni PE.

Avviato il tool e andati ad inserire il malware andiamo a vedere in quante sezioni si divide.



Tramite il menù a tendina andiamo a selezionare **Section Headers.**



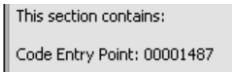
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
00000200	00000208	0000020C	00000210	00000214	00000218
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	00005646	00001000	00006000	00001000	00000000
.rdata	000009AE	00007000	00001000	00007000	00000000
.data	00003EA8	00008000	00003000	00008000	00000000
.rsrc	00001A70	0000C000	00002000	0000B000	00000000

Il risultato sarà una tabella con all'interno tutte le sezioni trovate nel malware.
Dove le colonne ci mostrano il loro nome e altre informazioni aggiuntive.

Descrizione delle sezioni trovate:

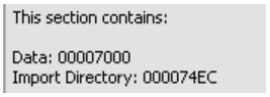
.text

La sezione .text in assembly contiene il codice eseguibile di un programma. Le istruzioni di assembly al suo interno vengono eseguite sequenzialmente quando il programma viene avviato. La sezione è di sola lettura e rappresenta la parte principale del codice macchina che guida l'esecuzione del programma.



.rdata

La sezione .rdata in assembly contiene dati di sola lettura, come costanti e stringhe immutabili, utilizzati dal programma durante l'esecuzione.

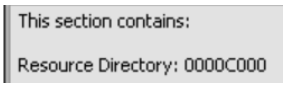


.data

La sezione .data in assembly è utilizzata per conservare dati modificabili e variabili globali che possono essere letti e scritti durante l'esecuzione del programma.

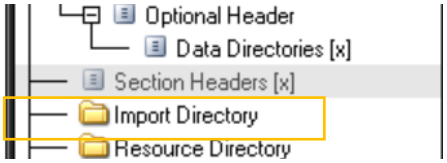
.rsrc

In sintesi, la sezione .rsrc in assembly è destinata a conservare risorse non eseguibili, come immagini o stringhe, che possono essere richiamate e utilizzate dal programma durante l'esecuzione.

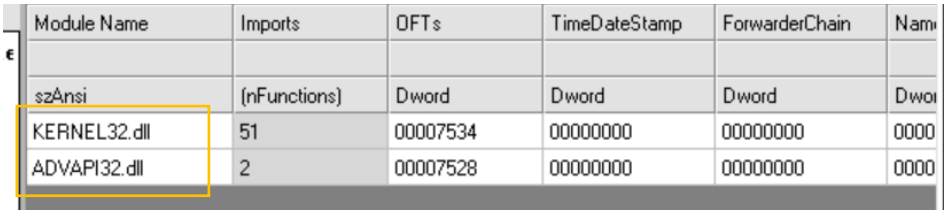


Quali librerie importa il malware?

Per andare a verificare quali librerie importa il malware, possiamo continuare ad utilizzare CFF Explorer e spostarci nella sezione “Import Directory”

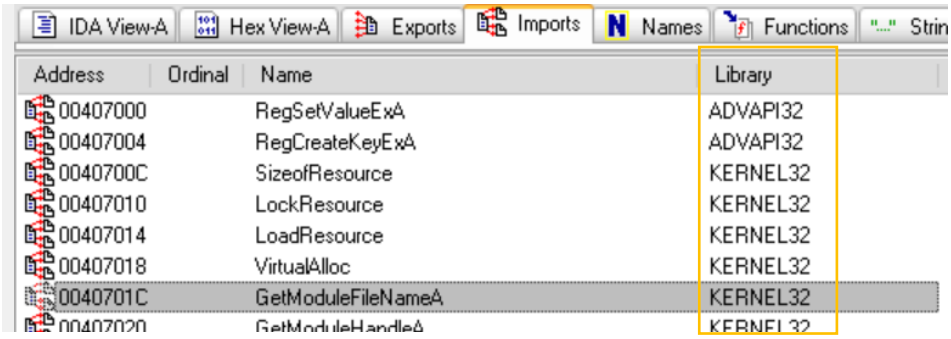


E ci verranno elencate le librerie che il malware va ad importare.
In questo caso vediamo come le librerie importate sono KERNEL32 e ADVAPI.



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Nam
szAnsi	(nFunctions)	Dword	Dword	Dword	Dwo
KERNEL32.dll	51	00007534	00000000	00000000	0000
ADVAPI32.dll	2	00007528	00000000	00000000	0000

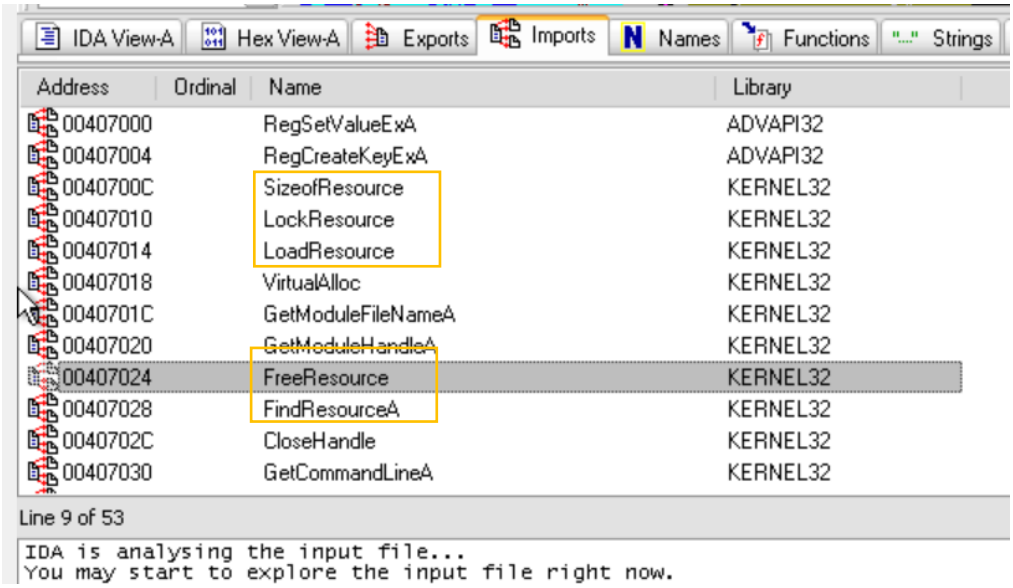
Anche con IDA avremmo potuto effettuare la stessa ricerca, andando a elencare i vari nomi delle funzioni che utilizzano le varie librerie.



Ipotesi sull'utilizzo che il malware va a fare delle librerie:

KERNEL32.dll:

La libreria KERNEL32 è una componente fondamentale nei sistemi operativi Windows. Essa fornisce una vasta gamma di funzioni di basso livello necessarie per il funzionamento delle applicazioni, inclusi processi, memoria, gestione di file, tempo e altri servizi di sistema.



Ipotesi:

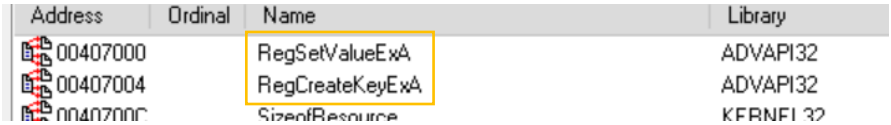
Queste risorse potrebbero contenere codice aggiuntivo, configurazioni, stringhe crittografiche o altri dati che il malware intende sfruttare durante la sua esecuzione**. La manipolazione delle risorse può essere una tattica utilizzata per nascondere il codice malevolo o per implementare funzionalità specifiche all'interno del programma infetto.**

In base a questo possiamo ipotizzare che il malware sia un "Dropper", un malware con all'interno un altro malware.

Possiamo ipotizzare anche un salvataggio del malware sul disco grazie alle funzioni WriteFile e CreateFileA. Infatti il dropper potrebbe andare a salvarsi sul disco.

ADVAPI.dll

La libreria ADVAPI32.dll in sistemi operativi Windows fornisce funzionalità avanzate di API per la gestione della sicurezza, dei servizi e del Registro di sistema. Essa offre funzioni per la crittografia, l'autenticazione, l'autorizzazione, la gestione degli eventi di sicurezza, la creazione e gestione dei servizi di Windows, nonché l'accesso e la manipolazione del Registro di sistema.



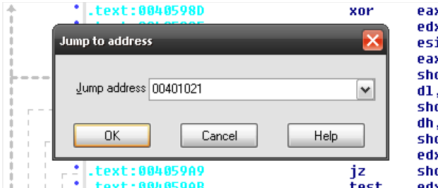
Ipotesi:

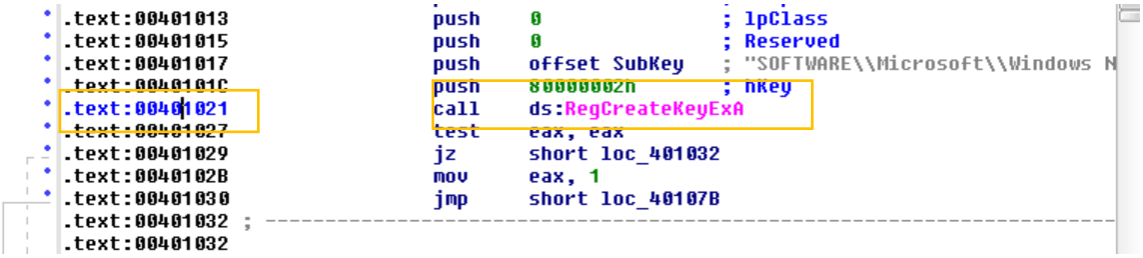
Il malware potrebbe utilizzare queste funzioni per creare o andare a modificare le chiavi di Registro ottenendo persistenza, la possibilità di essere avviato all'avvio del sistema operativo.
Infatti le funzioni richiamate sono:
RegSetValueExA: utilizzata per impostare i dati e il loro valore in una chiave del registro del sistema.
RegCreateKeyExA: utilizzata per creare o aprire le chiavi di registro specificata.

Malware Analysis

Scopo della funzione chiamata alla locazione di memoria 00401021?

Utilizzando sempre il software IDA, possiamo utilizzare una sua funzione per andare a fare un jump direttamente all'indirizzo della chiamata:





A questo indirizzo troviamo una chiamata alla funzione "RegCreateKeyExA", fa parte della libreria Kernel32.dll e serve per creare o aprire una chiave del registro del sistema.

Come vengono passati i parametri dalla funzione alla locazione 00401021?

Andando su microsoft learn, possiamo vedere i parametri passati per la seguente funzione che ricordiamo è utilizzata per modificare le chiavi di registro:

Syntax

C++ Copy

```
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
);
```

hKey (Handle della chiave padre):
Questo è l'handle della chiave padre sotto la quale verrà creata o aperta la nuova chiave.

lpSubKey (Nome della chiave):
La sottochiave da creare o aprire. È una stringa.

Reserved (Valore riservato):
Un valore riservato che deve essere 0.

lpClass (Classe della chiave):
La classe della chiave. Di solito, è una stringa vuota o null.

dwOptions (Opzioni):
Opzioni aggiuntive per la creazione o l'apertura della chiave.

samDesired (Accesso desiderato):
Il livello di accesso desiderato alla chiave.

lpSecurityAttributes (Attributi di sicurezza):
Attributi di sicurezza per la nuova chiave.

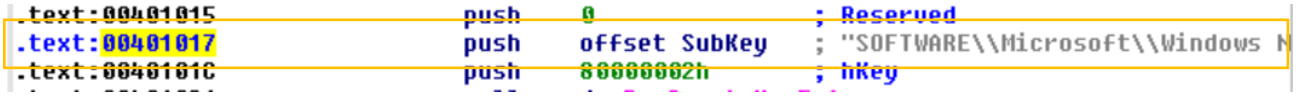
phkResult (Puntatore all'handle della nuova chiave):
Un puntatore a una variabile in cui verrà restituito l'handle della nuova chiave creata o aperta.

Visualizzazione IDA dei parametri passati con l'istruzione push:

```
push    0               ; lpdwDisposition
lea     eax, [ebp+hObject]
push    eax             ; phkResult
push    0               ; lpSecurityAttributes
push    0F003Fh         ; samDesired
push    0               ; dwOptions
push    0               ; lpClass
push    0               ; Reserved
push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows N
push    80000002h       ; hKey
call    ds:RegCreateKeyExA
```

Che oggetto rappresenta il parametro alla locazione 00401017?

Andando alla locazione richiesta, vediamo che viene passato il parametro Subkey che Inserirà nell'area dello stack l'indirizzo di memoria in cui inizia la stringa **"SOFTWARE\MICROSOFT\WINDOWS NT\CURRENT VERSION"**. Questo indirizzo verrà poi utilizzato come argomento quando verrà richiamata la funzione.
Qui troveremo informazioni di configurazione, riguardanti la versione del sistema operativo.

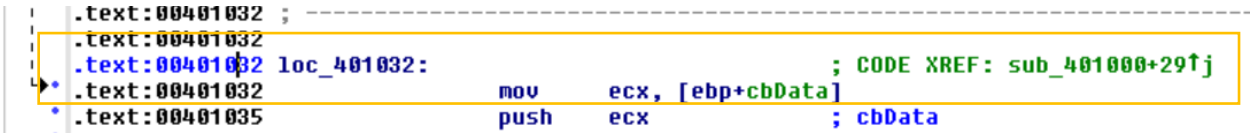


Significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029:

test eax, eax:
L'istruzione test esegue un'operazione di AND logico tra il valore del registro eax e se stesso. Questo non modifica il valore di eax, ma imposta le bandiere del processore in base al risultato dell'operazione. In particolare, il flag zero (ZF) viene impostato se il risultato dell'operazione è zero, altrimenti ZF viene azzerato.

jz short loc_401032:
Questa è un'istruzione di salto condizionato. L'istruzione jz (jump if zero) eseguirà un salto alla destinazione specificata (loc_401032) solo se il flag zero (ZF) è impostato. Se il risultato dell'operazione test eax, eax è zero (il valore di eax era zero), allora verrà eseguito il salto a loc_401032. Altrimenti, se ZF non è impostato (il valore di eax non era zero), l'esecuzione continuerà con l'istruzione successiva.

Quindi, il codice controlla se il valore nel registro eax è zero. Se eax è zero, il controllo salta a un'altra parte del codice (etichettata con loc_401032). Se eax non è zero, il flusso del programma continua normalmente senza effettuare il salto.



Qui visualizziamo il jump condizionale che va a spostare il contenuto della memoria.

Conversione linguaggio Assembly in linguaggio C:

```
int main(){

int a;        //eax
int b;        // ecx
int c;        // [ebp + cbData]

 if (a ==0){
        b = c;
} else {
        a = 1;
}
return 0; }
```

Valore del parametro “ValueName” chiamato alla locazione 00401047?

```
.text:00401046                push    eax             ; hKey
.text:00401047                call    ds:RegSetValueExA
.text:0040104D                test    eax, eax
```

Vediamo che è stata chiamata la funzione **RegSetValueExA**, la quale permette di impostare i dati e il tipo del loro valore specificata in una chiave di registro del sistema.

ValueName è un parametro della funzione che va a specificare il nome del valore da impostare per la chiave di registro per la funzione che si sta andando a creare/modificare.

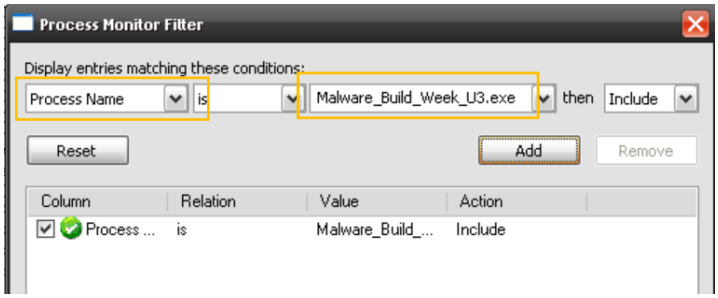
```
.text:0040103C                push    0               ; Reserved
.text:0040103E                push    offset ValueName ; "GinaDLL"
.text:00401043                mov     eax, [ebp+hObject]
.text:00401046                push    eax             ; hKey
```

Come vediamo, nel nostro caso il parametro ValueName è chiamato “GINADLL”.

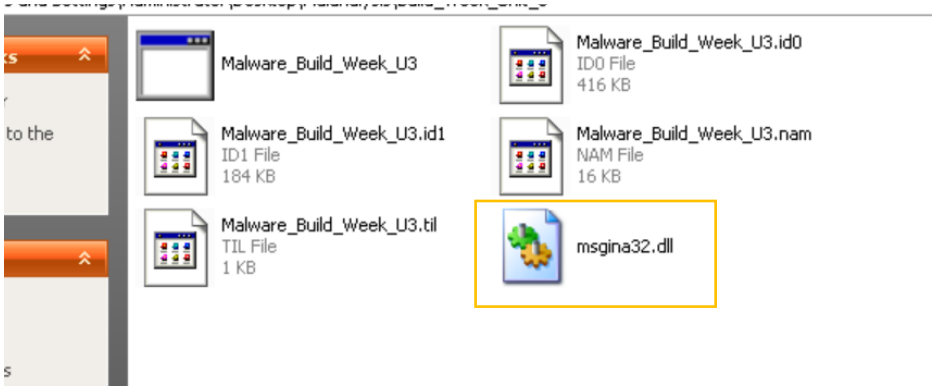
Analisi Dinamica

Viene eseguita in tempo reale mentre il programma è in esecuzione. L'analisi si concentra sul comportamento effettivo dell'applicazione durante l'esecuzione.

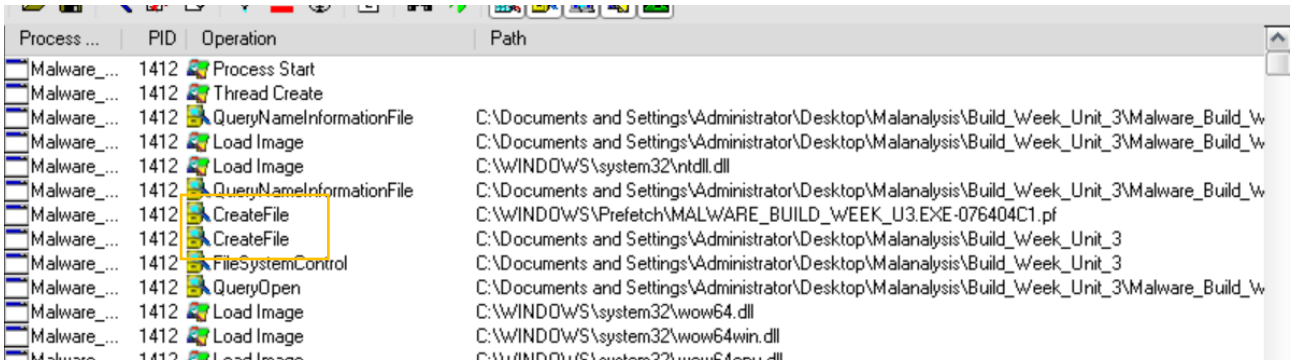
Per l’analisi statica utilizzeremo Process Monitor, dove andremo ad impostare i filtri come consigliato dalla traccia e andando a rimuovere tutti quelli che erano già preimpostati.



In questo modo Process Monitor verrà avviato senza nessun filtro.
Avviando il malware ci accorgiamo che nella sua cartella di provenienza si è creato un nuovo file, **msgina32.dll**.

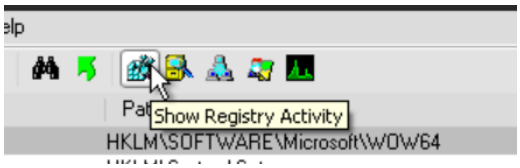


Avviata la scansione avremo come risultati una serie di operazioni mirate alla mappatura del sistema.
Nel print infatti riconosciamo il CreateFile:

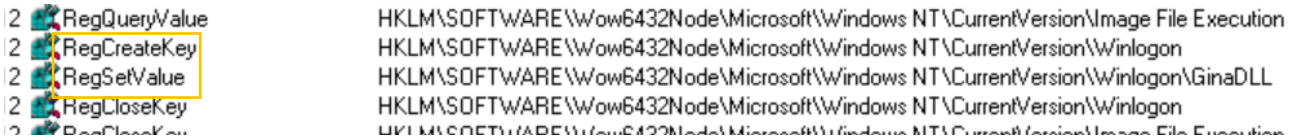


Ovviamente scorrendo possiamo ritrovare operazioni anche già viste come RegOpenKey e RegQueryValue esaminate anche con l'analisi statica.

Ora andiamo a filtrare le attività lasciando visibili solo le operazioni fatte dal malware sui registri di sistema:

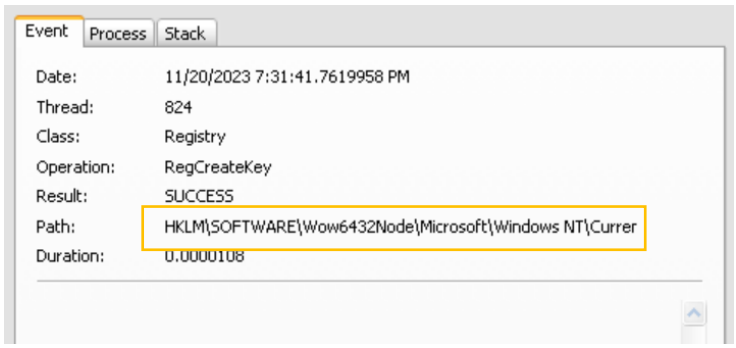


Come risultato noteremo come il malware punta ad ottenere informazioni sulle chiavi di registro e a modificarle.
A valorizzare la nostra tesi vediamo in fondo alla lista RegCreateKey e RegSetValue che ricordiamo vengono utilizzate per la manipolazione del registro di sistema.

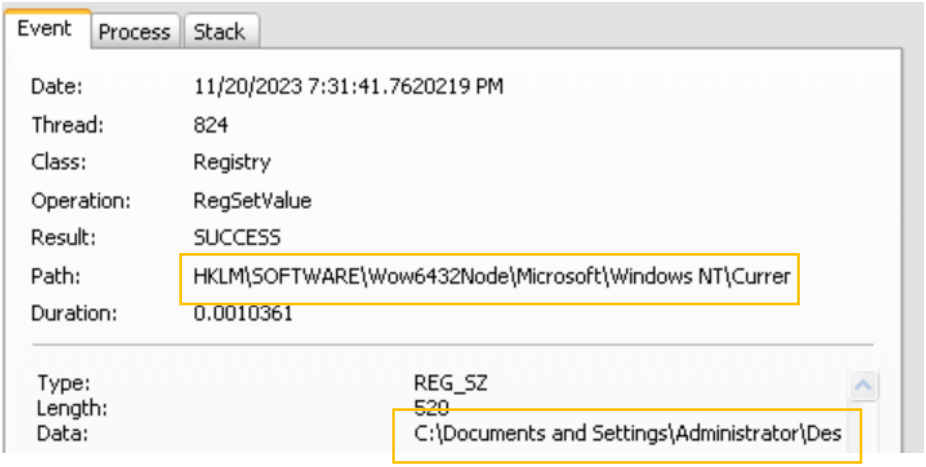


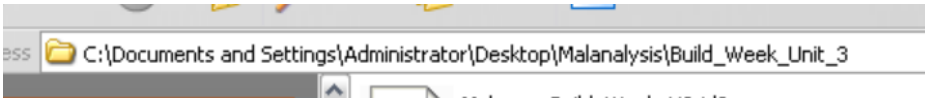
Andando ad osservare nel dettaglio Cosa vanno a fare ci rendiamo conto che:

RegCreateKey è stata utilizzata per modificare la chiave di registro che conteneva le info alla procedura di accesso dell'utente.



RegSetValue invece ha modificato la chiave andando ad inserire un nuovo valore, corrispondente al percorso del file msgina32.dll creato nella nostra cartella.

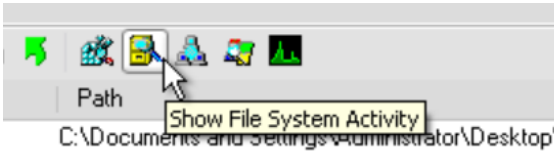




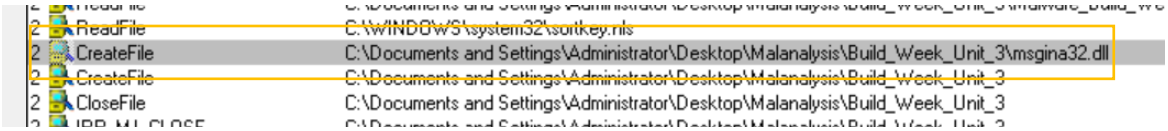
Nota* nello screen manca la parte finale con: \msgina32.dll

Visualizzazione dell'attività del file system:

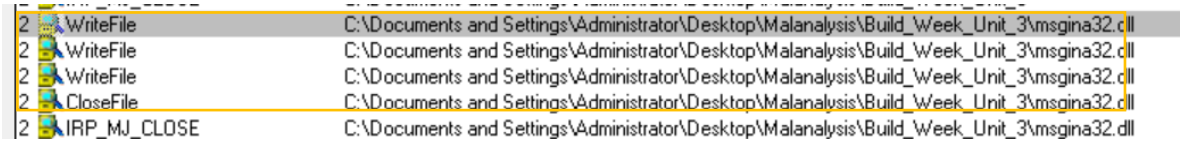
Andiamo a filtrare la nostra ricerca per file system con l'apposito filtro.



Oltre a vedere le attività di mappatura viste anche in precedenza, notiamo un'attività di CreateFile nel percorso del malware (msgina32).



E continuando a scorrere troveremo anche attività di lettura e scrittura di questo file, andando ovviamente a modificarne il contenuto.



Possiamo quindi concludere che il malware sia andato a creare e scrivere un file che poi ha inserito i una directory ben specifica.

Spiegazione Funzionamento del Malware:

Dopo aver terminato le nostre analisi possiamo confermare che il malware analizzato va a modificare le impostazioni di configurazione di accesso al sistema.

Il malware crea un file chiamato msgina32.dll che va a sfruttare una chiave di autenticazione di windows chiamata GinaDLL, imposta il suo percorso di creazione e va a leggerlo e riscriverlo.
Un malware così otterrà l'accesso automatico alla macchina o permetterà azioni non autorizzate.
Ricordando che al suo interno potrebbe contenere un altro malware se dovessimo ipotizzare la casistica del dropper.