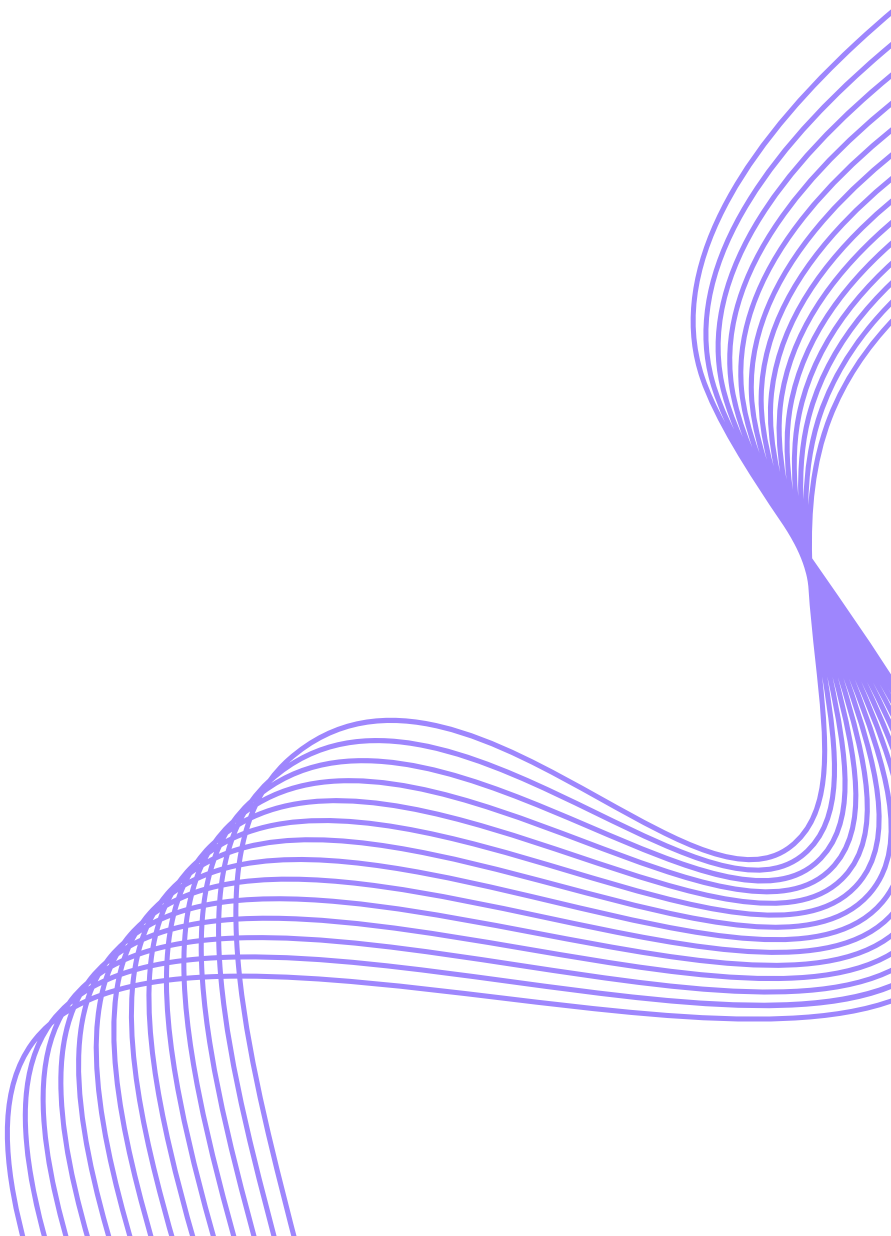
BUILD WEEK 3

MALWARE ANALYSIS AND REVERSE ENGINEERING

REPORTED BY:

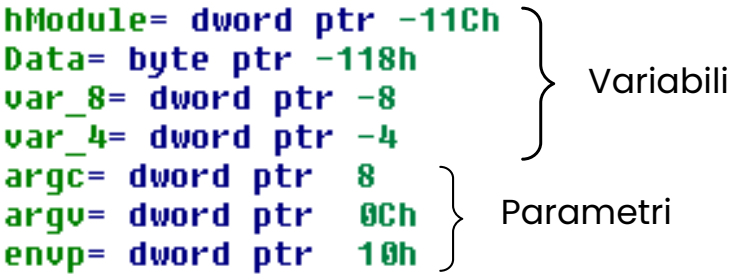
FERNANDO CATRAMBONE
ALESSANDRO MOSCETTI
MATTEO MURILLO
MICHAEL POGGIALI
BENEDETTA FORESTIERI
LUCA GALLEANI
NATALINO IMBROGNO
DAVIDE DIGLIO



Day 1

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 nella funzione **Main**()?



Utilizzando il programma "**Ida pro**" siamo andati ad analizzare il codice malevolo (Malware_Build_Week_U3) alla funzione **Main**, questo perchè è la funzione di ingresso principale del programma che viene eseguita quando viene avviato.

Abbiamo esaminato la funzione **Main()** e notato che i suoi **tre parametri** sono individuati da offset positivo rispetto al registro EBP, indicando che i valori dei parametri sono allocati a una certa distanza in avanti rispetto a EBP.

Allo stesso tempo, abbiamo rilevato la presenza di **quattro variabili** all'interno della funzione **Main()**, ciascuna identificata da un offset negativo rispetto al registro EBP. Questo suggerisce che lo spazio di memoria assegnato a queste variabili si trova a una certa distanza all'indietro rispetto a EBP.

La differenza tra **parametro** e **variabile** sta nell'utilizzo durante l'esecuzione del programma: i parametri sono valori passati a una funzione quando viene chiamata, mentre le variabili sono spazi di memoria utilizzati per conservare dati all'interno della funzione. La distinzione è evidenziata dagli offset positivi per i parametri e dagli offset negativi per le variabili rispetto al registro EBP.

Day 1

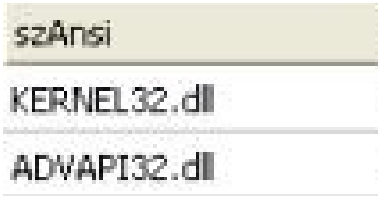
Con riferimento al file eseguibile **Malware_Build_Week_U3**, rispondere ai seguenti quesiti utilizzando i tool e le tecniche apprese nelle lezioni teoriche:

- Quali **sezioni** sono presenti all'interno del file eseguibile?
- Quali **librerie** importa il Malware?

L'analisi condotta attraverso **CFFExplorer** ha consentito di ottenere una panoramica più dettagliata delle attività che può effettuare il malware.
Le **sezioni** da cui è composto il malware sono:

- **.text**: contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato.
- **.data**: contiene i dati e le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.
- **.rdata**: contiene i dati disponibili in sola lettura come librerie o funzioni importate o esportate dal programma.
- **.rsrc**: include le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell'eseguibile stesso.

Il malware utilizza funzioni provenienti da due **librerie**:



- **Kernel32.dll**: contiene le funzioni principali per l'interazione con il sistema operativo.
- **Advapi32.dll**: sono presenti le funzioni necessarie per interagire con il registro di Windows.

In questo modo, il malware sfrutta le risorse di tali librerie per eseguire operazioni specifiche, coinvolgendo sia il sistema operativo che il registro di Windows.

Day 1

Ipotesi del comportamento del **Malware_Build_Week_U3** dalle informazioni trovate con **CFFExplorer**

1. L'analisi delle librerie di questo malware ha rivelato una serie di funzioni chiave che indicano un comportamento potenzialmente dannoso e orientato all'attacco.

   - L'uso di **GetProcAddress** indica una dinamicità nel caricamento di funzioni, suggerendo che il malware vada a caricare altre librerie e funzioni.

   - Le funzioni **RegSetValueExA** e **RegCreateKeyExA** suggeriscono che il malware potrebbe cercare di persistere nel sistema attraverso la modifica del Registro di Sistema.

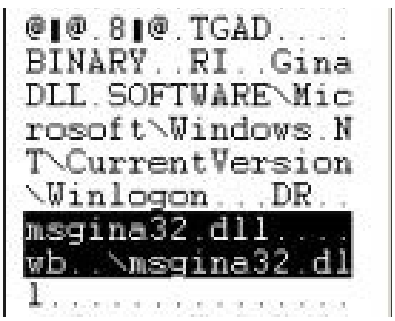
     

   - L'impiego di **LoadResource**, **LockResource**, e **SizeofResource** indica un interesse verso la manipolazione delle risorse presenti nell'eseguibile del malware.

Dalle funzioni emerse nelle librerie possiamo supporre che si tratti di un malware della famiglia dei **Dropper**.
Inoltre analizzando la sezione **.data,** che contiene i dati necessari al programma per funzionare, abbiamo individuato un file di nome **msgina32.dll** e un path che riguarda **winlogon**, che è un processo Windows che riguarda il logon interattivo.
Da questi elementi possiamo supporre che il malware tramite un componente malevolo interferisca con l'accesso per rubare le credenziali.

.data	00003EA8	00006000
.rsrc	00001A70	0000C000



Day 2

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**

```
push    0               ; lpSecurityAttributes
push    0F003Fh         ; samDesired
push    0               ; dwOptions
push    0               ; lpClass
push    0               ; Reserved
push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h       ; hKey
call    ds:RegCreateKeyExA
```

I valori necessari per questa funzione vengono trasmessi attraverso lo stack di memoria mediante l'operazione "**push**". Prima di eseguire la funzione, i parametri vengono posti nello stack in sequenza, e la funzione li preleva da lì durante l'esecuzione.
L'indirizzo **00401017** nel codice contiene la chiave il cui valore viene fornito come argomento a **RegCreateKeyExA**.
Questa funzione sta ad indicare che il programma sta cercando di creare o aprire una chiave del Registro di Sistema per scrivere o leggere informazioni al fine di manipolare il Registro di Sistema.
L'**oggetto** nella loc **00401017** contiene il percorso della chiave del Registro di Sistema che si desidera creare o aprire.

- Spiegare il significato delle istruzioni comprese tra gli indirizzi **00401027** e **00401029**
- Tradurre il codice Assembly nel corrispondente **costrutto C.**

```
00401027        test    eax, eax
00401029        jz      short loc_401032
```

L'istruzione **test eax, eax** è simile all'operatore logico **AND** ma a differenza che non memorizza il risultato in **eax.** In particolare, effettuerà un test bit a bit tra il registro **eax** e se stesso impostando cosi i flag di zero (**ZF**) a 0.
L'istruzione **jz short loc_401032** a questo punto effettuerà un salto alla locazione solo se il flag di zero (**ZF**) sarà impostato a zero.
Questo potrebbe essere una sua rappresentazione in **costrutto C**:

```
if (eax==0){
    nome_registro="GinaDLL"
}
else {
    return 1;
}
```

Day 2

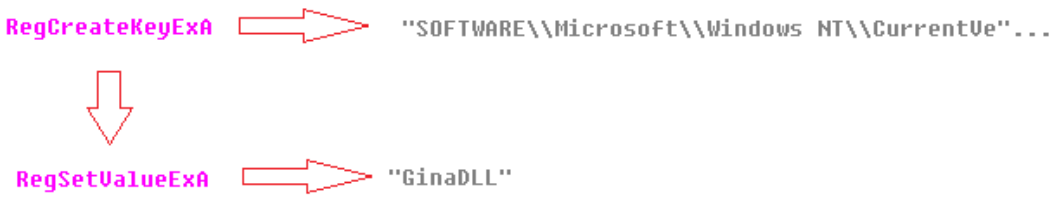
Con riferimento al **Malware** in analisi, spiegare:

```
push    0               ; lpSecurityAttributes
push    0F003Fh         ; samDesired
push    0               ; dwOptions
push    0               ; lpClass
push    0               ; Reserved
push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h       ; hKey
call    ds:RegCreateKeyExA

40103E                  push    offset ValueName ; "GinaDLL"
401043                  mov     eax, [ebp+hObject]
401046                  push    eax             ; hKey
401047                  call    ds:RegSetValueExA
```

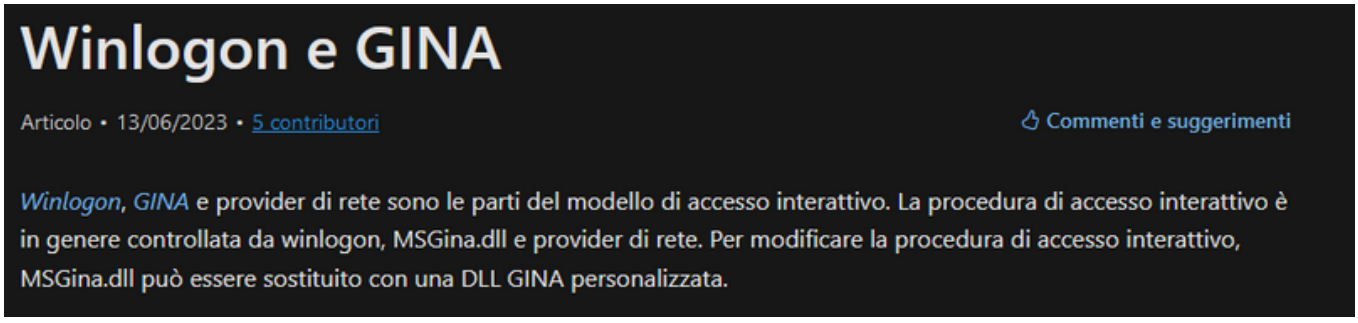
- Qual è il valore di «**ValueName**» alla locazione **00401047**

Il valore del parametro ValueName è "**GinaDLL**"

- Spiegate quale **funzionalità** sta implementando il Malware in queste sezione.



In queste sezione il malware sta creando una nuova chiave di registro **RegCreateKeyExA** e sta settando il suo nome a: "**GinaDLL**" utilizzando la funzione **RegSetValueExa** . Come possiamo vedere GINA e Winlogon servono per gestire la procedura di accesso.

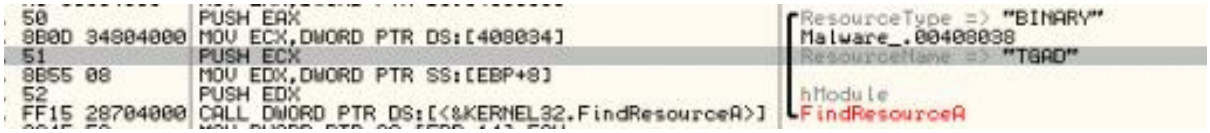


https://learn.microsoft.com/it-it/windows/win32/secauthn/winlogon-and-gina

Day 3

Analizzando le routine tra le locazioni di memoria **00401080** e **00401128**:

- Qual è il valore del parametro «**ResourceName**» passato alla funzione



Tramite l'utilizzo del software **"OllyDBG"** abbiamo individuato che il valore del parametro **"ResourceName"** è **"TGAD"**

- Il susseguirsi delle chiamate di funzione che effettua il Malware in questa sezione di codice che **funzionalità** sta implementando?

Dalle chiamate di funzione presenti in questa sezione di codice abbiamo una conferma che il malware sia un **dropper**, ovvero un software malevolo che svolge il ruolo di un trasportatore di altri malware, facilitando la loro introduzione e esecuzione nell'ambiente del computer infetto

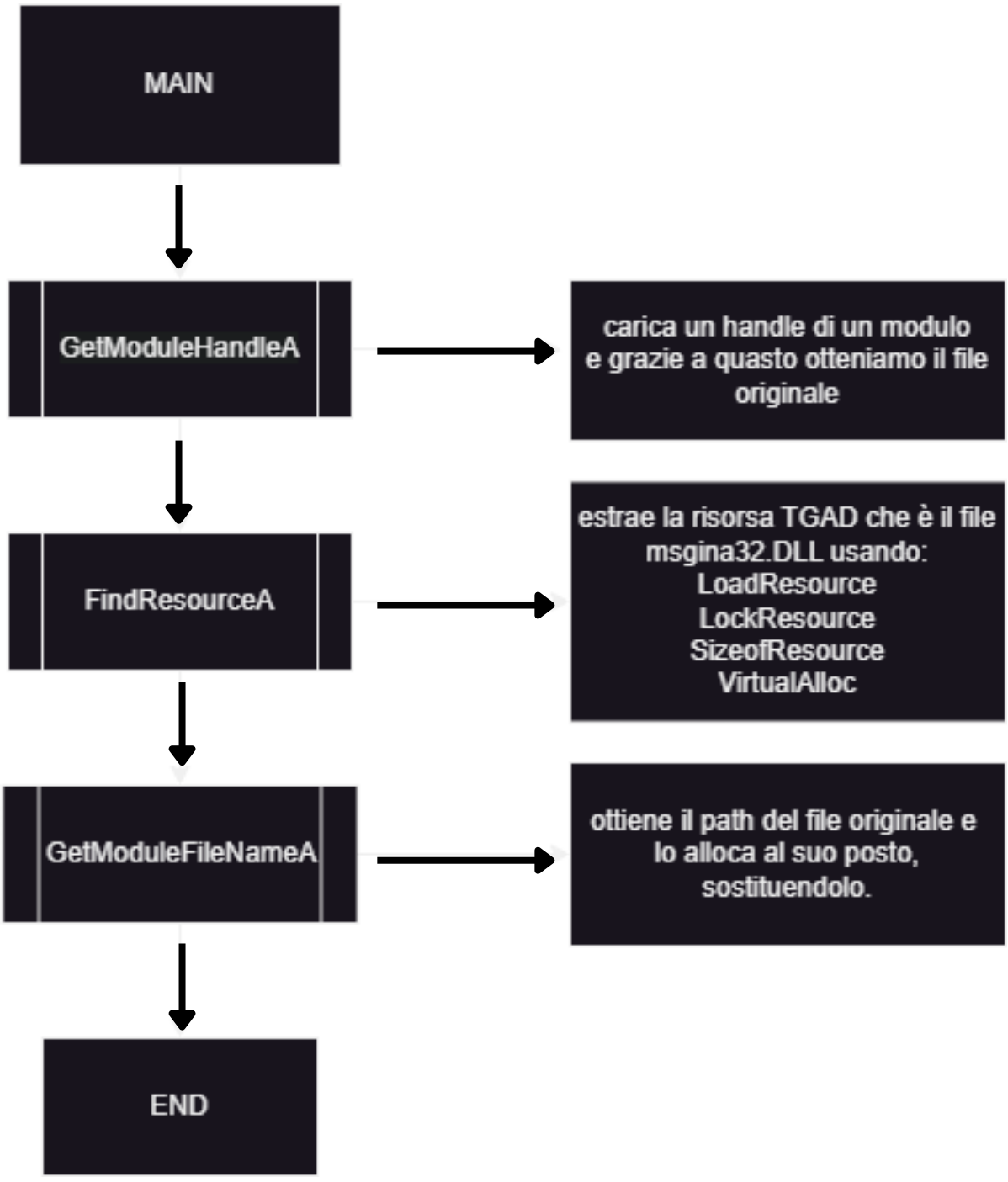
```
loc_4010DF:                             ; CODE XREF: sub_401080+56↑j
                mov     eax, [ebp+hResInfo]
                push    eax             ; hResInfo
                mov     ecx, [ebp+hModule]
                push    ecx             ; hModule
                call    ds:LoadResource
                mov     [ebp+hResData], eax
                cmp     [ebp+hResData], 0
                jnz     short loc_4010FB
                jmp     loc_4011A5
; ---------------------------------------------------------------------------

loc_4010FB:                             ; CODE XREF: sub_401080+74↑j
                mov     edx, [ebp+hResData]
                push    edx             ; hResData
                call    ds:LockResource
                mov     [ebp+var_8], eax
                cmp     [ebp+var_8], 0
                jnz     short loc_401113
                jmp     loc_4011A5
; ---------------------------------------------------------------------------

loc_401113:                             ; CODE XREF: sub_401080+8C↑j
                mov     eax, [ebp+hResInfo]
                push    eax             ; hResInfo
                mov     ecx, [ebp+hModule]
                push    ecx             ; hModule
                call    ds:SizeofResource
```

- È possibile identificare questa funzionalità utilizzando l'analisi **statica basica**? (elencare le evidenze a supporto).

Già dall'analisi **statica basica** avevamo intuito il possibile funzionamento del malware tramite la presenze delle funzioni **LoadResouce**, **Lock Resource** **SizeOfResource** all'interno della libreria **KERNEL32.dll** oltre che alla presenza della sezione **.rsrc** che contiene le ulteoriori risorse che il **dropper** va a caricare nella macchina vittima.

Day 3

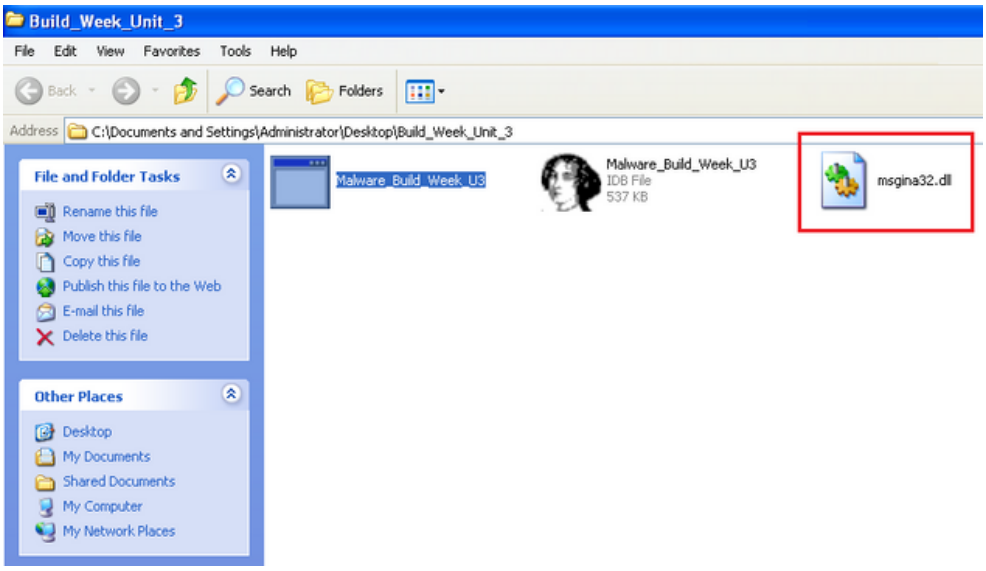
- Disegnare un diagramma di flusso che comprenda le tre funzioni che descrivono le funzionalità appena viste del malware.



Come si può vedere in figura questo diagramma semplificato mostra il comportamento di queste tre funzioni all'interno del **main** ed il loro utilizzo.

Day 4

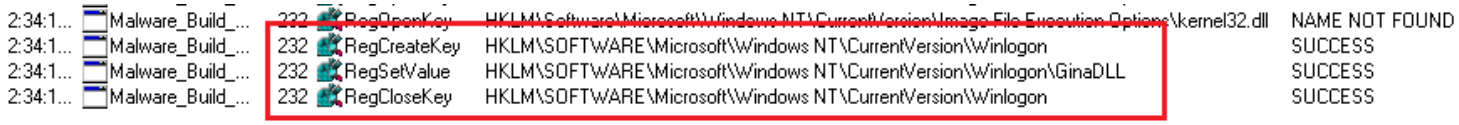
- Preparate l'ambiente ed i tool per l'esecuzione del Malware
- Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware?



Una volta preparato il nostro laboratorio, isolando la nostra macchina virtuale abbiamo avviato il malware, all'interno della stessa cartella viene creato un file di nome "**msgina32.dll**". Questa evidenza ci da conferma sulle ipotesi precedentemente fatte ovvero che si tratta di un **dropper** che estrae un file **.dll**

Filtrate includendo solamente l'attività sul registro di Windows.

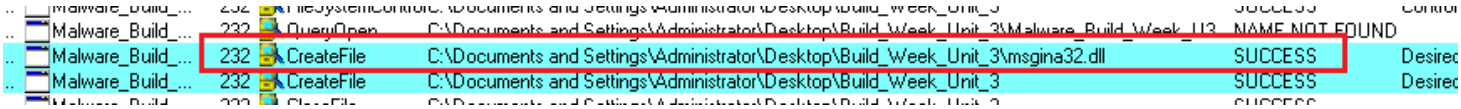
- Quale chiave di registro viene creata?
- Quale valore viene associato alla chiave di registro creata?



Viene creata una chiave di registro all'indirizzo di **Winlogon**, a questa chiave di registro viene associato il valore **"GinaDLL"** che abbiamo già visto nelle giornate precedenti.

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?



La funzione chiamata è **"CreateFile"** come possiamo vedere in figura crea il nostro file **msgina32.dll**.

Day 5

GINA (Graphicauthentication& authentication) è un componente 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.

- Cosa può succedere se il file **.dll lecito** viene sostituito con un file **.dll malevolo** che intercetta i dati inseriti?

Sapendo che GINA è un componente di Windows che permette l'autenticazione tramite interfaccia grafica nel caso in cui un .dll lecito venga sostituito con un file .dll malevolo c'è il rischio che le credenziali degli utenti vengano rubate.

- Delineate il **profilo** del Malware e delle sue funzionalità.

Possiamo concludere che il comportamento del malware è così descritto:

- Estrae una componente malevola di nome "GinaDLL" dalle sue risorse all'interno della cartella dove si trova il malware.
- Crea una chiave di registro "Winlogon" che è parte del sistema operativo Windows e serve per l'autenticazione interattiva.
- Sostituisce il componente .dll legittimo con quello malevolo estratto precedentemente.
- Il componente malevolo viene utilizzato per rubare le credenziali.

Day 5

- Unite tutti i punti visti fino ad esso per creare un grafico che ne rappresenti lo scopo ad alto livello.