Gruppo 2

Build Week 3

4-8 Marzo 2024

Danilo Teresa
Stefano Carlini
Sara Hizdër
Matteo Palozza
Davide Salvatore
Pablo Balbuena

Malware Analysis

Il Malware da analizzare è nella cartella Build_Week_Unit_3 presente sul desktop della macchina virtuale dedicata.

Giorno 1 - Quesito 1

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_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
```

Mediante l'impiego dello strumento **IDAPro**, emerge chiaramente che la funzione **main()** è implementata con tre parametri standard denominati "parametri di riga di comando".
Questi sono identificati come: **argc, argv** e **envp**.

Giorno 1 - Quesito 2

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_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
```

È possibile osservare anche che la funzione main() è definita con cinque variabili locali, ciascuna contrassegnata da un valore offset negativo.
Le variabili sono: **hModule**, **Data**, **var_117**, **var_8**, **var_4**.

Giorno 1 - Quesito 3

Quali sezioni sono presenti all’interno del file eseguibile?
Descrizione breve di almeno 2 di quelle identificate

IDaProo

Byte[8]	Dword	Dword
.text	00005646	00001000
.rdata	000009AE	00007000
.data	00003EA8	00008000
.rsrc	00001A70	0000C000

CFF Explorer

Name	Start	End
.text	0000000000401000	0000000000407000
.idata	0000000000407000	00000000004070DC
.rdata	00000000004070DC	0000000000408000
.data	0000000000408000	000000000040C000

- **.text**: Contiene il codice eseguibile del programma, rappresentando il cuore dell'esecuzione con la sequenza di istruzioni che il processore seguirà per implementare le operazioni specificate nel programma.
- **.rdata**: Ospita dati di sola lettura, solitamente costanti o inizializzati, che mantengono la loro immutabilità durante l'esecuzione del programma.
- **.data**: Custodisce dati globali e variabili inizializzate che possono essere modificati durante l'esecuzione del programma, fungendo da area di memoria per informazioni dinamiche.
- **.rsrc**: Racchiude le risorse del programma, quali immagini, icone, stringhe localizzate e altri dati non eseguibili che il programma potrebbe richiedere durante il suo funzionamento.
- **.idata**: Contiene le informazioni di importazione, specificando quali funzioni o librerie esterne il programma utilizzerà durante l'esecuzione.

Giorno 1 - Quesito 4

Quali librerie importa il Malware?

Ipotizzare sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare.

Name	Library
RegSetValueExA	ADVAPI32
RegCreateKeyExA	ADVAPI32

Le funzioni ***RegSetValueExA()*** e ***RegCreateKeyExA()*** sono entrambe funzioni della libreria di Windows **ADVAPI32.dll** che gestiscono operazioni nel registro di sistema.
Il Malware potrebbe utilizzare quelle funzioni per creare nuove chiavi nel registro o per modificare o aggiungere voci al registro di sistema.

Name	Library
SizeofResource	KERNEL32
LockResource	KERNEL32
LoadResource	KERNEL32
VirtualAlloc	KERNEL32
GetModuleFileNameA	KERNEL32
GetModuleHandleA	KERNEL32
FreeResource	KERNEL32
FindResourceA	KERNEL32

Le funzioni ***FindResourceA()***, ***LoadResource()***, ***LockResource()***, e ***SizeOfResource()*** appartengono alla libreria di Windows **KERNEL32.dll**, e vengono comunemente impiegate per la gestione di risorse in applicazioni Windows. Tuttavia, quando adoperate in modo malevolo, queste funzioni potrebbero facilitare attività dannose, come nel caso dei **Dropper**. Un **Dropper** è un tipo di malware progettato per rilasciare e installare ulteriori componenti dannosi sul sistema bersaglio.

Giorno 2 - Quesito 1

Lo scopo della funzione chiamata alla locazione di memoria <u>00401021</u>

```
:00401015                       push    0                       ; Reserved
:00401017                       push    offset SubKey           ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe".
:0040101C                       push    80000002h               ; hKey
:00401021                       call    ds:RegCreateKeyExA
:00401027                       test    eax, eax
:00401029                       jz      short loc_401032
:0040102B                       mov     eax, 1
```

La funzione ***RegCreateKeyExA*** situata alla locazione di memoria ***0x00401021***, è una delle Windows APIs e gestisce le chiavi di registro in Windows. Può creare nuove sottochiavi o aprire quelle esistenti sotto una chiave principale specificata

Giorno 2 - Quesito 2 e 3

- **Come vengono passati i parametri alla funzione alla locazione 00401021**
- **Che oggetto rappresenta il parametro alla locazione 00401017**

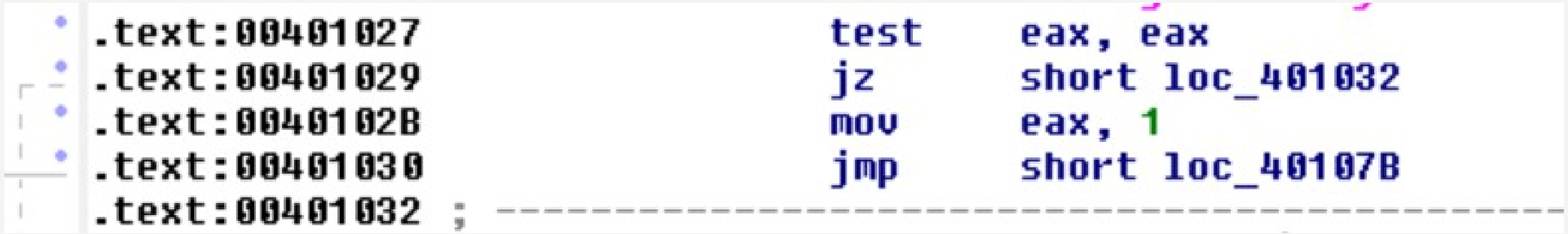
```
:00401015   push    0                   ; Reserved
:00401017   push    offset SubKey       ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
:0040101C   push    80000002h           ; hKey
:00401021   call    ds:RegCreateKeyExA
:00401027   test    eax, eax
:00401029   jz      short loc_401032
:0040102B   mov     eax, 1
```

Durante la chiamata alla funzione RegCreateKeyExA, i ***parametri*** vengono passati tramite l'istruzione ***push***. Questi includono:

- **h*key***: Handle a una chiave di registro principale, come HKEY_LOCAL_MACHINE, che contiene informazioni principali di configurazione del computer.
- ***SubKey***: Puntatore al percorso della sottochiave da aprire, nel nostro caso il percorso è *"SOFTWARE\Microsoft\Windows NT\CurrentVersion"*. Questa sottochiave è comunemente utilizzata per contenere informazioni sulla versione del sistema operativo Windows.
- ***Reserved***: Valore DWORD che specifica le opzioni di creazione/apertura della chiave di registro. Nel codice è passato il valore 0, dunque senza opzioni speciali. Questo valore è generalmente riservato per usi futuri.

Giorno 2 - Quesito 4

Il significato delle istruzioni comprese tra gli indirizzi <u>00401027 e 00401029</u>



Le istruzioni comprese tra gli indirizzi ***00401027*** e ***00401029*** svolgono un ***controllo condizionale*** e un' azione conseguente, in base al risultato del test:

- "***test eax, eax***": Simila un'operazione logica ***AND*** bit per bit tra il registro eax e se stesso. Il test verifica se eax è zero o meno. Il risultato non viene memorizzato, ma imposta i flag del processore in base al risultato.
- "***jz short loc_401032***": Verifica se lo **ZERO FLAG** (***ZF***) è impostato dopo il test precedente. Se il flag ZF è impostato (cioè se eax è zero), il programma salta alla locazione di memoria ***00401032***, altrimenti il flusso del programma continua normalmente.

Giorno 2 - Quesito 5

Tradurre il codice Assembly nel corrispondente costrutto C

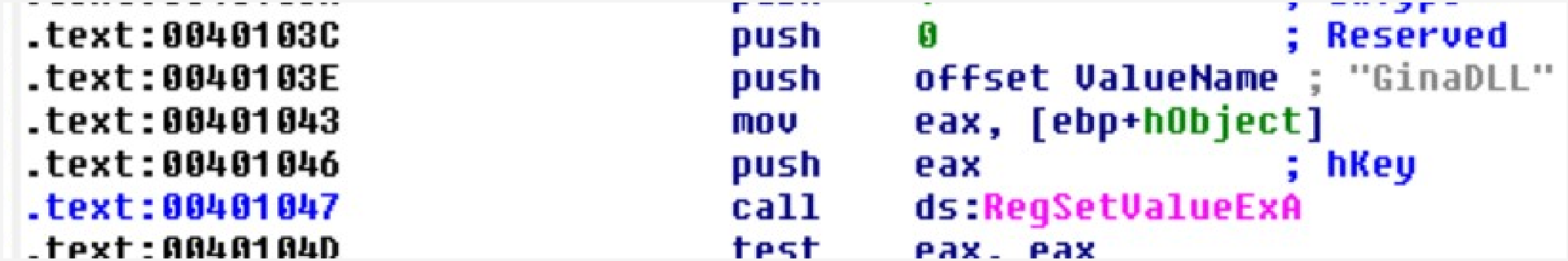
```
.text:00401027                  test    eax, eax
.text:00401029                  jz      short loc_401032
.text:0040102B                  mov     eax, 1
.text:00401030                  jmp     short loc_40107B
.text:00401032 ; ---------------------------------------------
```

In ***C***, le istruzioni comprese tra gli indirizzi ***00401027*** e ***00401029***, potrebbero essere rappresentate in questo modo:

```
if (eax == 0) {
  goto loc_401032;
}
eax = 1;
goto loc_40107B;
```

Giorno 2 - Quesito 6

Valutate la chiamata alla locazione 00401047 e qual è il valore del parametro «ValueName»?



Il valore del parametro ValueName è ***GinaDLL***. Questo viene passato come argomento alla funzione **RegSetValueExA**, che viene chiamata alla locazione di memoria ***00401047***.

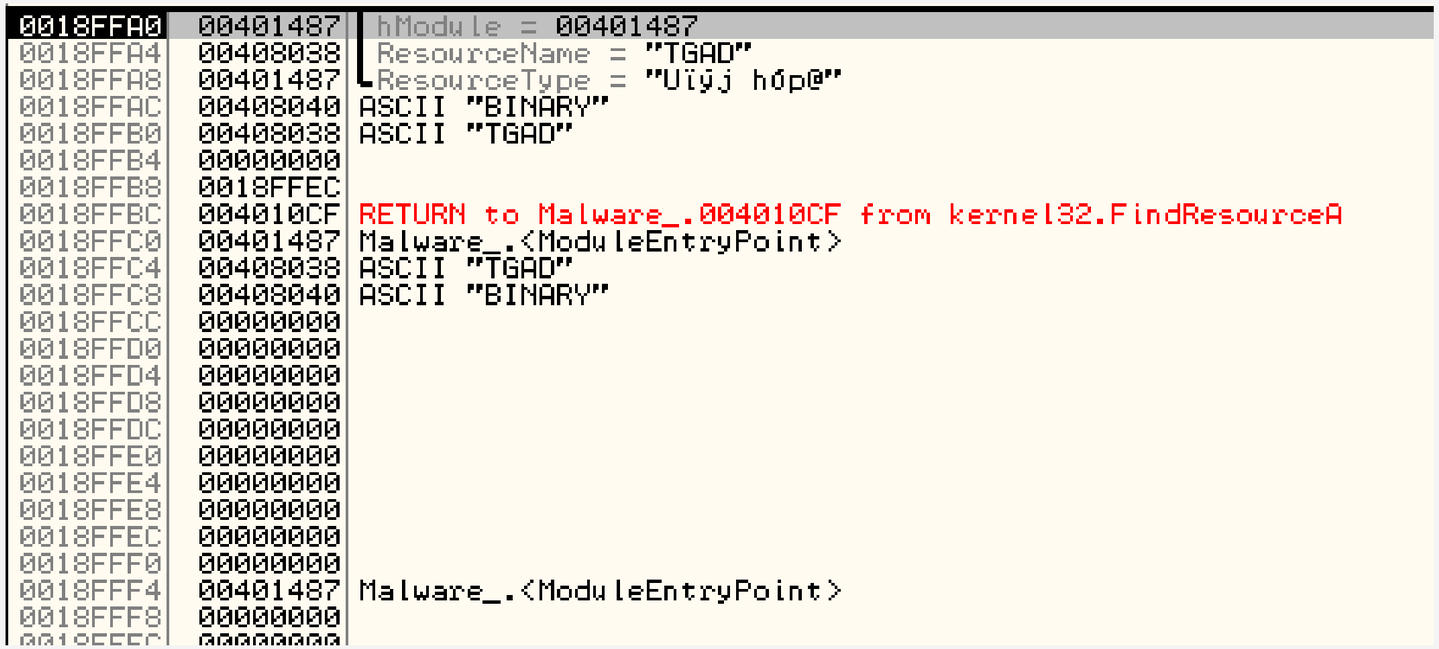
Considerazioni

La creazione di una nuova chiave di registro con il valore "***GinaDLL***" suggerisce un tentativo del malware di ***manipolare il processo di login*** standard di Windows. Questo permette di compromettere le credenziali dell'utente durante il login, consentendo l'accesso non autorizzato al sistema o altre azioni dannose.

Giorno 3 - Quesito 1

Qual’è il valore del parametro “ResourceName” tra le locazioni di memoria 00401080 e 00401128 ?

Il valore del parametro “**ResourceName**” è uguale a “**TGAD**”. Esso è possibile trovarlo tramite Ollydbg nel modulo malware.



Giorno 3 - Quesito 2

Che funzionalità sta implementando il malware?

Il malware è un dropper, cioè un malware con al suo interno un altro malware. Quando viene eseguito utilizza API come “**FindResource**”, “**LoadResource**”, “**LockResource**” e “**SizeOfResource**” per cercare ed estrarre il malware situato all’interno della sezione risorse (.**rss** o .**rsc**) del proprio codice. Inoltre cambia la chiave di registro tramite le funzioni “**Regsetvalue”** e “**Regcreatekey**“ per ottenere la persistenza.

Giorno 3 - Quesiti 3 e 4

È possibile identificare queste funzionalità tramite analisi statica basica?

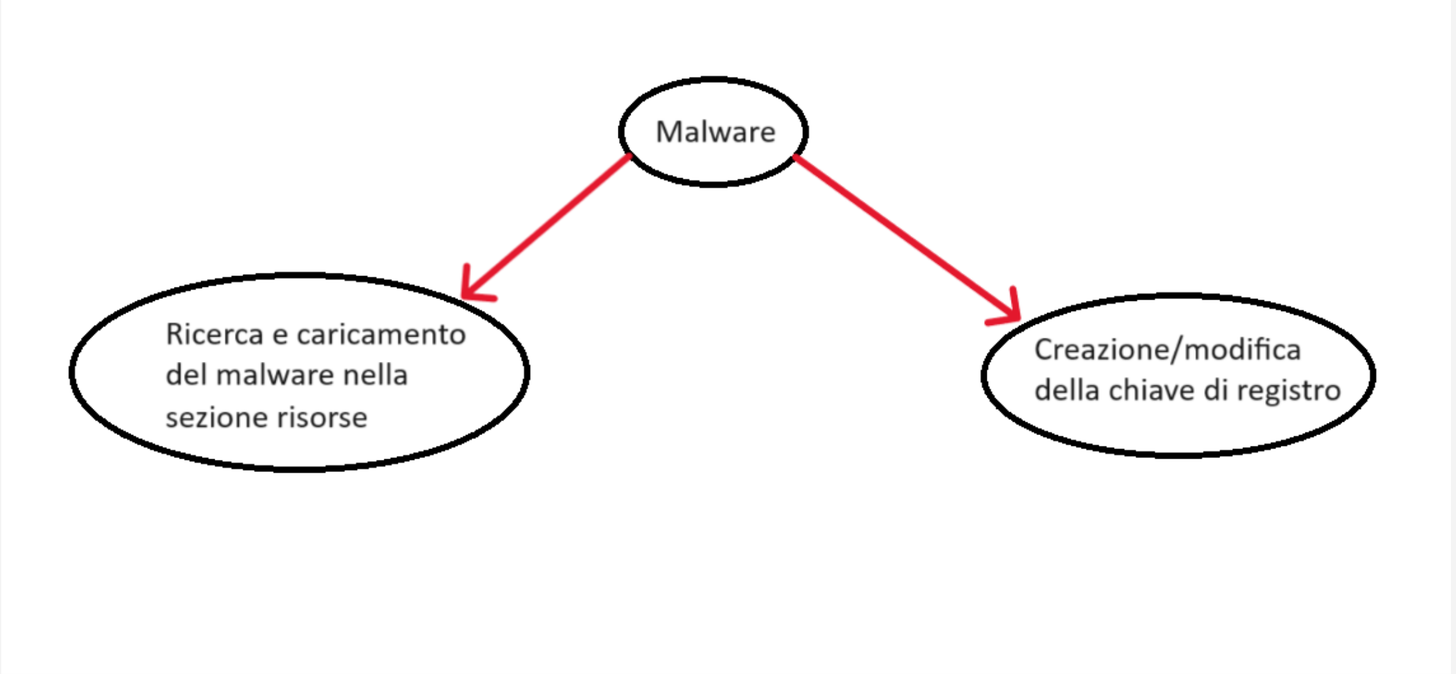
Anche con l'uso dell'analisi statica basica possiamo identificare le funzionalità e trovarle nella libreria **KERNEL32.DLL** come in figura

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA

Giorno 3: Quesito 5

Disegnare un diagramma di flusso

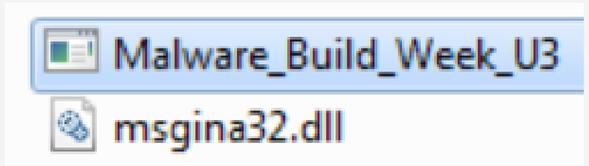
Nel diagramma a fianco viene schematizzato il funzionamento delle funzioni principali del malware.



Giorno 4 - Quesiti da 1 a 3

Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware? Spiegate cosa è avvenuto

Nella cartella del Malware, durante l'avvio, si nota la creazione del file "msgina32.dll". Dalle informazioni raccolte, possiamo presumere che si tratti di un dropper, in quanto andrà a localizzare ed estrarre tale risorsa dalle sue sezioni

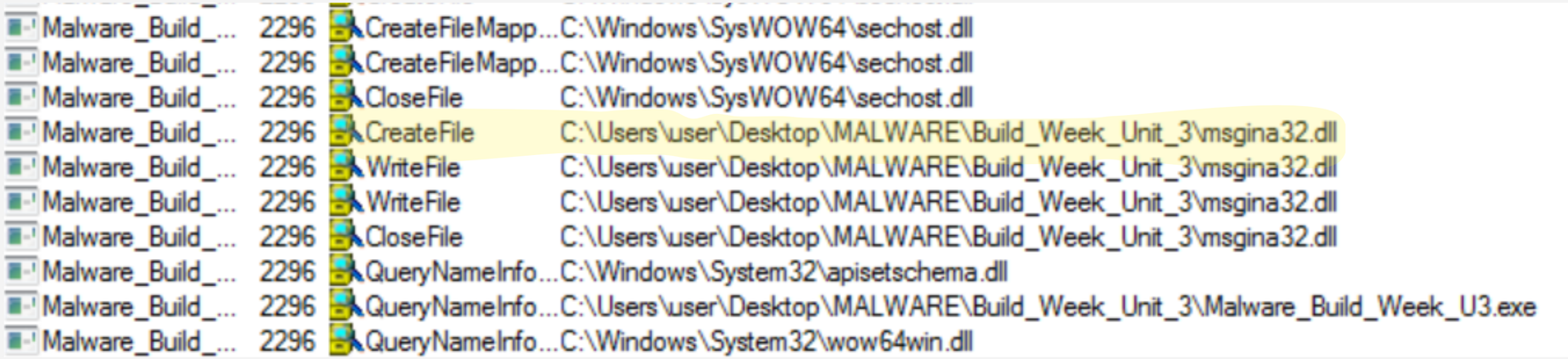


Dopo aver filtrato gli eventi generati dal malware con Procmon, possiamo esaminare le modifiche apportate alle chiavi di registro. Notiamo che inizialmente viene creata la chiave di registro **HKLM\SOFTWARE\Wow6432Node\Microsoft\WindowsNT\CurrentVersione\WinLogon** e successivamente le viene attribuito il valore di **GinaDLL**.

Malware_Build_...	2296	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
Malware_Build_...	2296	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\WINLOGON
Malware_Build_...	2296	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\WINLOGON
Malware_Build_...	2296	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\WINLOGON\GinaDLL
Malware_Build_...	2296	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\WINLOGON

Giorno 4 - Quesito 4

Analizando l’attività sul FIle System, possiamo osservare la presenza dell’istruzione con cui il file è stato generato all’interno della cartella del malware, ovvero attraverso la funzione **CreateFile**.



Malware_Build_...	2296	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	2296	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	2296	CloseFile	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	2296	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_...	2296	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_...	2296	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_...	2296	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_...	2296	QueryNameInfo...	C:\Windows\System32\apisetschema.dll
Malware_Build_...	2296	QueryNameInfo...	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\Malware_Build_Week_U3.exe
Malware_Build_...	2296	QueryNameInfo...	C:\Windows\System32\wow64win.dll

Possiamo definire il comportamento basandoci sulla sua attività. Al suo avvio andrà ad identificare la risorsa «**msgina32.dll**» nei suoi segmenti per poi estrarla all’interno della cartella in cui si trova.
Successivamente andrà a creare e impostare la chiave di registro come visto nelle precedenti slide.

Giorno 5 - Quesito 1 :

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

Se il componente GINA (Graphical Identification and Authentication) di Windows, responsabile dell'autenticazione degli utenti tramite interfaccia grafica, venisse sostituito da un file DLL malevolo, le potenziali conseguenze sarebbero gravi. Il file DLL malevolo andrebbe a intercettare i dati inseriti dagli utenti durante il processo di autenticazione, come username e password.

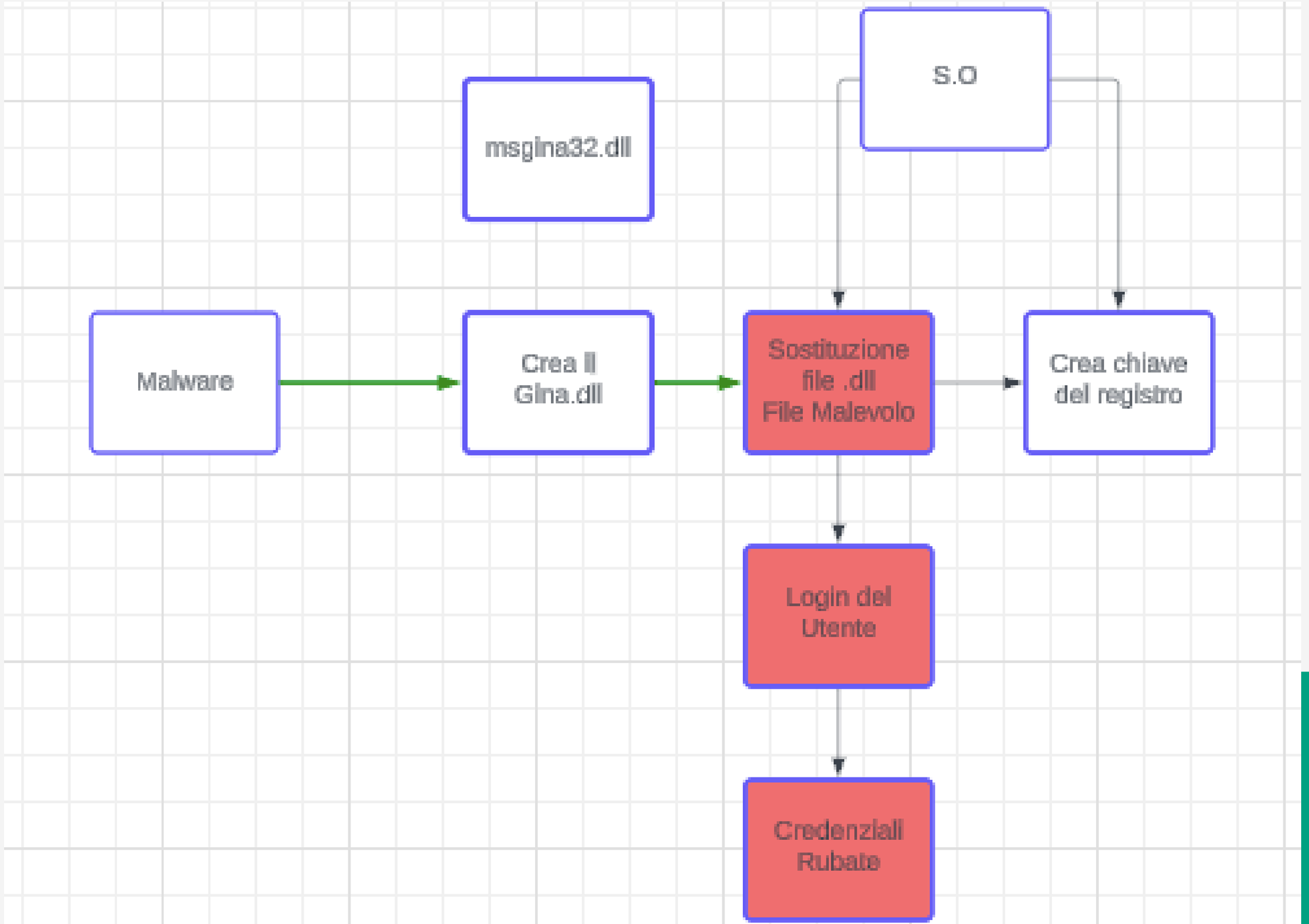
Profilo del Malware e delle sue funzionalità:

Individuazione del file legittimo e sostituzione : Il malware indentifica il file msgina32.dll e lo sostituisce con una versione malevola tramite le funzioni visti al Giorno 4

Manipolazione del registro: Il malware crea una chiave di registro la HKEY_LOCAL_MACHINE ed assegna il valore Gina.dll a questa chiave .

Rubare credenziali: Grazie alla sostituzione del GINA.dll, il malware intercetta e raccoglie le credenziali dell'utente durante il processo di login .

In sintesi, il malware agisce in modo mirato sostituendo il componente di autenticazione di Windows con una versione malevola, consentendo il furto delle credenziali degli utenti durante l'accesso al sistema operativo tramite l'interfaccia grafica di autenticazione.


S.O
msgina32.dll
Malware
Crea Il
Gina.dll
Sostituzione
file .dll
File Malevolo
Crea chiave
del registro
Login del
Utente
Credenziali
Rubate

Grazie