

# **MALWARE ANALYSIS**

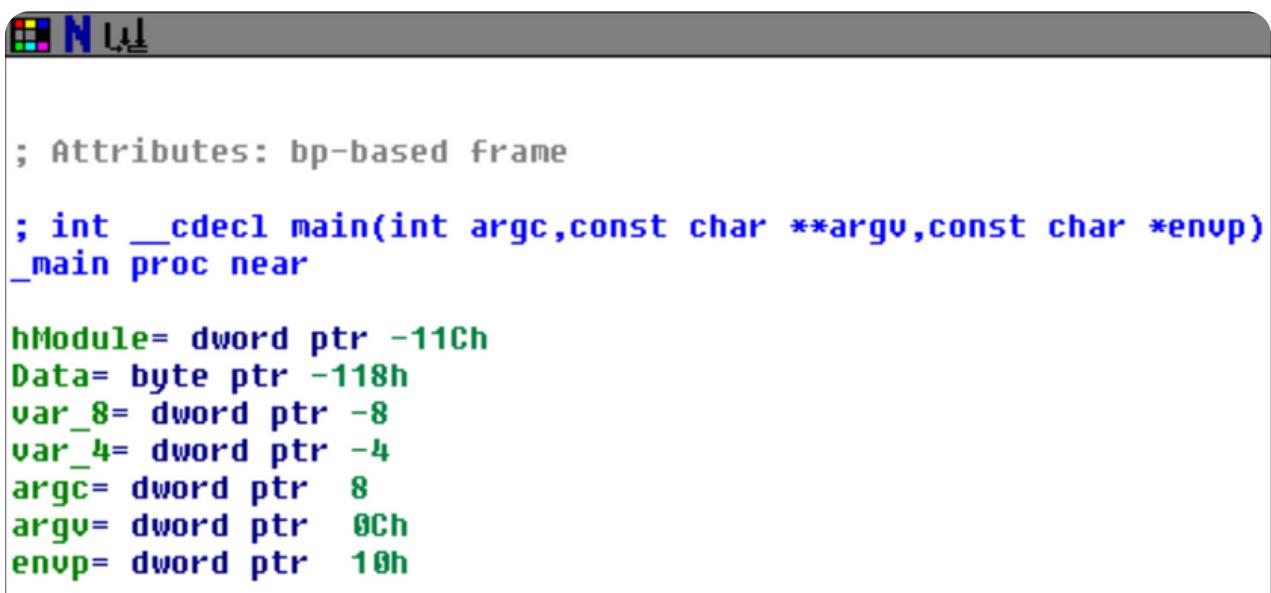
## **BUILD WEEK III**

# GIORNO 1

Per analizzare il malware in oggetto abbiamo utilizzato **IDA Pro**, un software disassembler che analizza un malware e restituisce in output il suo codice in linguaggio assembly (nel nostro caso assembly dell'architettura X86).

Il codice disassemblato verrà rappresentato da una barra con diversi colori, quello che a noi interessa è il colore blu poiché rappresenta il codice del malware.

Una volta che IDA avrà terminato le sue operazioni, sullo schermo comparirà l'entry point dell'applicazione, ovvero il punto da cui il programma comincia ad eseguire il suo codice. E' possibile vederlo grazie allo screenshot sottostante:



```
; Attributes: bp-based frame
; 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
```

La **funzione main** quindi rappresenta l'**entry point** e grazie ad una veloce analisi possiamo individuare quali sono i parametri della funzione nonché le variabili dichiarate al suo interno (sempre visibili nello screenshot), di seguito le abbiamo elencate:

## Parametri

Sulla base dei dati forniti dalla traccia dell'esercizio, è stato possibile rilevare tre parametri: **argc**; **argv**; **envp**.

## Variabili

Le variabili riscontrate sono quattro e sono individuabili tramite l'offset negativo: **hModule**; **Data**; **var\_8**; **var\_4**.

Malware_Build_Week_U3.exe						
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000076D0	N/A	00007500	00007504	00007508	0000750C	00007510
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

## Librerie

Le librerie visualizzate tramite il tool CFF Explorer sono due: **KERNEL32.dll** e **ADVAPI32.dll**

- **KERNEL32.dll**: questa libreria contiene le funzioni principali per interagire con il sistema operativo, ad esempio gestione della memoria e manipolazione file.
- **ADVAPI32.dll**: contiene le funzioni per interagire con i registri del sistema operativo.

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

Tra le funzioni utilizzate dalle librerie quelle che vale la pena evidenziare per **KERNEL32.dll** sono quelle presenti nello screenshot sopra, si potrebbe ipotizzare che il malware sia un **dropper** che carica una risorsa malevola una volta sul pc target. Per la libreria **ADVAPI32.dll** invece abbiamo due funzioni (mostrate sotto) che permettono di creare e settare una chiave del registro, molto probabilmente per ottenere la **persistenza** necessaria.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000076AC	000076AC	0186	RegSetValueExA
000076BE	000076BE	015F	RegCreateKeyExA

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characterist
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

## Sezioni

Le sezioni visualizzate tramite CFF Explorer sono quattro: **.text**; **.rdata**; **.data**; **.rsrc**.

- **.text**: la sezione in questione contiene il codice eseguibile dal programma, ovvero le istruzioni in Assembly che vengono eseguite dal processore nel momento in cui il programma è in esecuzione.
- **.rdata**: questa sezione contiene dati di sola lettura (read only data) i quali rimangono costanti e non vengono modificati durante l'esecuzione del programma.
- **.data**: a differenza della sezione precedente, qui i dati possono essere modificati durante l'esecuzione. Spesso sono incluse variabili globali e altri valori che possono essere cambiati dinamicamente.
- **.rsrc**: quest'ultima sezione contiene immagini, icone, stringhe e altri dati riconducibili all'interfaccia dell'utente e alle funzionalità del programma.

# GIORNO 2

```
    .text:00401009      push    eax          ; phkResult
    .text:0040100A      push    0             ; lpSecurityAttributes
    .text:0040100C      push    0F 003Fh   ; sanDesired
    .text:00401011      push    0             ; dwOptions
    .text:00401013      push    0             ; lpClass
    .text:00401015      push    0             ; Reserved
    .text:00401017      push    offset SubKey  ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
    .text:0040101C      push    80000002h  ; hKey
    .text:00401021      call    ds:RegCreateKeyExA
    .text:00401027      test   eax, eax
    .text:00401029      jz     short loc_401032
    .text:00401028      mov    eax, 1
    .text:00401030      jmp    short loc_40107B
.loc_401032 ; -----
    .text:00401032      mov    ecx, [ebp+cbData] ; CODE XREF: sub_401000+29↑j
    .text:00401032      mov    edx, [ebp+lpData]
    .text:00401035      push   ecx          ; cbData
    .text:00401036      push   edx          ; lpData
    .text:00401039      push   1             ; dwType
    .text:0040103A      push   0             ; Reserved
    .text:0040103C
```

Nella locazione di memoria **00401021** è presente la funzione **RegCreateKeyExA**. Essa ha il compito di creare una specifica chiave di registro, se è già presente, viene aperta.

I parametri vengono passati tramite una serie di istruzioni push.

L'oggetto che rappresenta il parametro alla locazione **00401017** è la stringa **"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"** che corrisponde al percorso della chiave di registro di sistema che il programma sta cercando di creare utilizzando la funzione "RegCreateKeyExA". Il codice coinvolge la chiamata alla funzione "RegCreateKeyExA" che è una funzione di Windows API utilizzata per creare o aprire una chiave di registro di sistema, e questa stringa rappresenta, dunque, il nome della chiave di registro stesso.

Le istruzioni comprese tra **00401027** e **00401029** fanno riferimento ai codici mnemonici **test** e **jz**.

- **Test:** questa istruzione è utilizzata per effettuare un'operazione logica tra due operandi, tramite un confronto bit a bit. Il registro EAX viene confrontato con se stesso tramite un'operazione di AND. Il risultato non viene memorizzato ma gli effetti dell'operazioni sono riscontrabili su eventuali EFLAGS. La ZF verrà impostata ad 1 se il risultato della comparazione restituirà zero, in caso contrario la ZF avrà valore 0 se il risultato della comparazione sarà diverso da zero.
- **jz:** l'istruzione jz fa parte della famiglia dei jmp, in questo caso assume il valore di un jump condizionale. La condizione necessaria per attuare il salto è rappresentata dal valore della ZF. "Jump if Zero", il salto verrà effettuato se la ZF restituirà 1 nella comparazione precedente.

Le istruzioni **00401027** e **00401029** possono essere tradotte in linguaggio C in un modo simile a questo:

```
if (eax==0) {  
    //salto alla locazione 401032  
} else {  
    //proseguimento del codice  
}
```

```
00401027 test    eax, eax  
00401029 jz     short loc_401032  
0040102B mov    eax, 1  
0040102D jmp    short loc_40107B  
00401032 ;  
00401032 loc_401032:  
00401032 mov    ecx, [ebp+cbData] ; cbData  
00401035 push   ecx  
00401036 mov    edx, [ebp+lpData] ; lpData  
00401039 push   edx  
0040103A push   1 ; dwType  
0040103C push   0 ; Reserved  
0040103E push   offset ValueName ; "GinaDLL"  
00401043 mov    eax, [ebp+hObject]  
00401046 push   eax ; hKey  
00401047 call   ds:RegSetValueExA |  
0040104D test   eax, eax  
00401062 jz     short loc_401062  
00401051 mov    ecx, [ebp+hObject]  
00401054 push   ecx ; hObject  
00401055 call   ds:CloseHandle
```

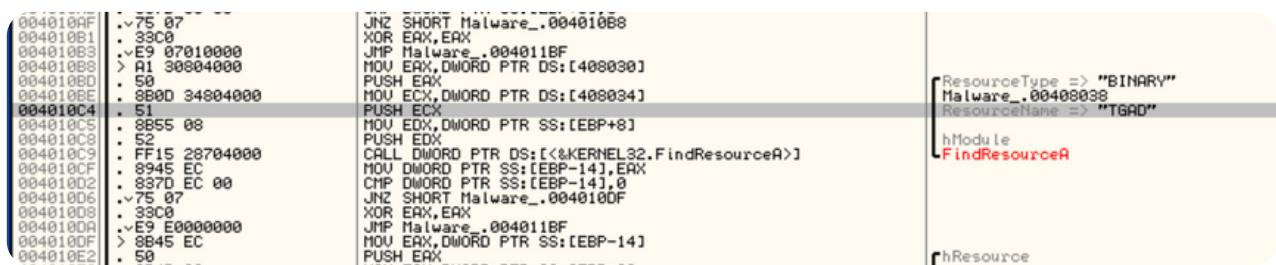
La chiamata alla locazione **00401047** è una chiamata alla funzione **RegSetValueExA**. Il valore del parametro ValueName è **GinaDLL**.

```
0040103E push   offset ValueName ; "GinaDLL"  
00401043 mov    eax, [ebp+hObject]  
00401046 push   eax ; hKey  
00401047 call   ds:RegSetValueExA |  
0040104D test   eax, eax  
00401062 jz     short loc_401062  
00401051 mov    ecx, [ebp+hObject]  
00401054 push   ecx ; hObject  
00401055 call   ds:CloseHandle
```

In questo estratto del codice il malware sta cercando di inserire un valore associato ad una chiave di registro di sistema che è comunemente utilizzato per configurare il caricamento di DLL. Poichè GINA (Graphical Identification and Authentication) gestisce l'interfaccia di autenticazione, **probabilmente il malware sta utilizzando il valore "GinaDLL" per manipolare il comportamento del sistema operativo in maniera da influenzare il processo di autenticazione.**

# GIORNO 3

Abbiamo analizzato la subroutine compresa tra gli indirizzi di memoria **00401080** e **0401128**, in particolare la funzione **FindResourceA()**. Uno dei parametri passati a questa funzione è **lpName** (ResourceName nella traccia dell'esercizio), e utilizzando OllyDB possiamo determinarne il valore che corrisponde a "**TGAD**". Di seguito è possibile vedere uno screenshot con evidenziato il valore:



Il malware implementa delle funzionalità appartenenti alla famiglia dei dropper. Quest'ultimo è un malware che contiene all'interno un file malevolo da estrarre per essere eseguito.

Le funzioni che ci fanno capire che si tratti di un dropper sono:

- **FindResource();**
- **LoadResource();**
- **LockResource();**
- **SizeOfResource();**

Queste particolari funzioni permettono di estrarre il file malevolo.

The screenshot displays the assembly code for four related functions:

- FindResourceA()**: Located at **00401080**, this function takes parameters **lpName** (EAX) and **hModule** (EDX). It calls **ds::FindResource()** and returns **hResInfo** (EAX).
- LoadResource()**: Located at **004010DF**, this function takes parameters **hResInfo** (EAX) and **hModule** (ECX). It calls **ds::LoadResource()** and returns **hResData** (EAX).
- LockResource()**: Located at **004010F0**, this function takes parameters **hResData** (EAX) and **var\_8** (ECX). It calls **ds::LockResource()** and returns **hResData** (EAX).
- SizeOfResource()**: Located at **004010F8**, this function takes parameters **hResInfo** (EAX) and **hModule** (ECX). It calls **ds::SizeOfResource()** and returns **dwSize** (EAX).

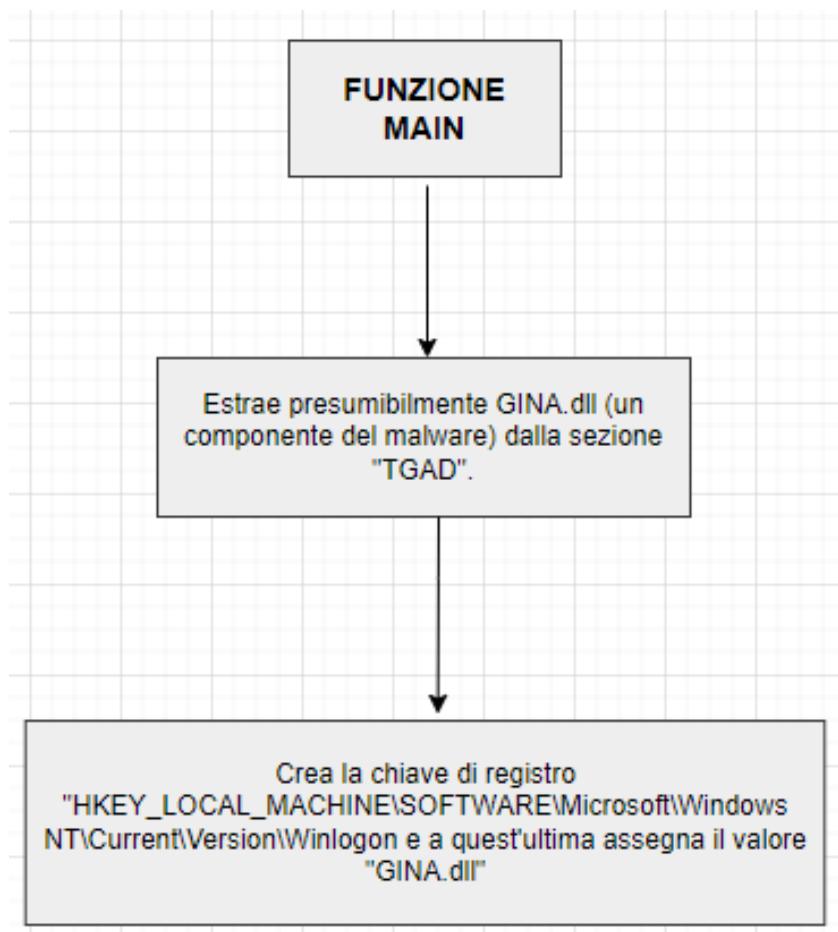
Questa funzionalità è possibile identificarla utilizzando l'analisi statica basica perché tramite il tool "CFF Explorer" si possono vedere le funzioni usate dal dropper nella libreria **Kernel32.dll**.

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

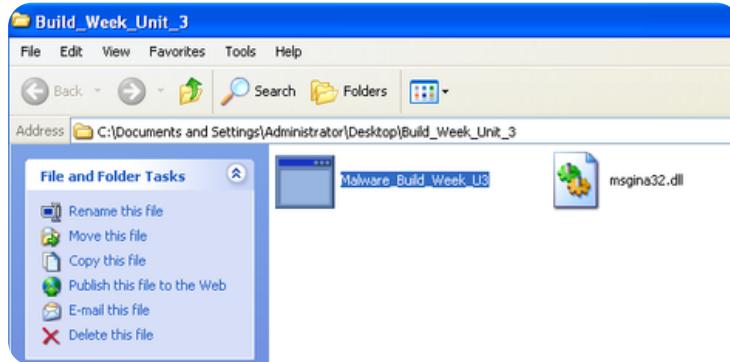
OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02B8	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	0018	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion
000076FE	000076FE	007D	ExitProcess

Di seguito le quattro funzioni rappresentate da un diagramma di flusso:



# GIORNO 4

Una volta eseguito il malware, ritroveremo sulla stessa cartella dell'eseguibile un file dll nominato **msgina32.dll**. Il file creato potrebbe corrispondere ad un rifacimento malevolo della libreria **GINA.dll**, componente lecito di Windows.



La chiave di registro creata è:

**HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon.**

Il valore associato alla chiave di registro è **GinaDLL**.

```
237:13.11651... [Malware_Build_Week_U3....] 1452 [RegCreateKey  
237:13.11659... [Malware_Build_Week_U3....] 1452 [RegSetValue  
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon  
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
```

Le chiamate alla funzione **CreateFile** sono le responsabili della creazione del file **msgina32.dll** nella cartella Malware\_Build\_Week\_U3.

```
237:13.10783... [Malware_Build_Week_U3....] 1452 [CreateFile  
37:13.10822... [Malware_Build_Week_U3....] 1452 [Createfile  
37:13.10982... [Malware_Build_Week_U3....] 1452 [CloseFile  
37:13.11225... [Malware_Build_Week_U3....] 1452 [ReadFile  
37:13.11403... [Malware_Build_Week_U3....] 1452 [WriteFile  
37:13.11523... [Malware_Build_Week_U3....] 1452 [WriteFile  
37:13.11576... [Malware_Build_Week_U3....] 1452 [CloseFile  
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll  
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3  
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3  
C:  
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll  
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll  
C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
```

Tramite le esecuzioni delle tecniche di analisi statica e dinamica, abbiamo delineato il funzionamento del Malware.

A seguito dell'installazione del file sulla macchina vittima e la conseguente esecuzione da parte dell'utente, il Malware opererà a livello software per creare una chiave di registro e settare quel valore come **GINA.DLL**. Quest'ultimo verrà sostituito al componente **GINA** originale.

# Giorno 5

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

Rimpiazzare un file **dll lecito** con un file **dll malevolo** che intercetti i dati inseriti, avrebbe come conseguenza la possibilità, da parte dell'attaccante, di ricevere i dati per il login degli utenti. Ciò che è stato analizzato nel punto 2 si ricollega a questa funzione. Sostituendo la componente lecita GINA con un componente GINA fittizio, è possibile entrare in possesso dei dati di login.

## Grafico del Malware

