

# BUILD WEEK PT.3

**CREATED BY:**

Team leader: Ciaschini Giorgio  
Iannone Luca  
El Ashri Ahmed  
Rossetti Alessio  
Fasani Marco

# INDICE

**GIORNO 1**

**GIORNO 2**

**GIORNO 3**

**GIORNO 4**

**GIORNO 5**



# INDICE

## GIORNO 1:

- TRACCIA pag.4
- PARAMETRI pag.5
- VARIABILI pag.6
- CFF EPLOKER pag.7
- SEZIONI pag.8
- LIBRERIE pag.9
- FUNZIONI pag.10
- IPOTESI pag.12

# Giorno 1

## TRACCIA

CON RIFERIMENTO AL FILE ESEGUIBILE MALWARE\_BUILD\_WEEK\_U3, RISONDERE AI SEGUENTI QUESITI UTILIZZANDO I TOOL E LE TECNICHE APPRESE NELLE LEZIONI TEORICHE:

1. QUANTI PARAMETRI SONO PASSATI ALLA FUNZIONE MAIN()?
2. QUANTE VARIABILI SONO DICHIARATE ALL'INTERNO DELLA FUNZIONE MAIN()?
3. QUALI SEZIONI SONO PRESENTI ALL'INTERNO DEL FILE ESEGUIBILE? DESCRIVETE BREVEMENTE ALMENO 2 DI QUELLE IDENTIFICATE
4. QUALI LIBRERIE IMPORTA IL MALWARE? PER OGUNA DELLE LIBRERIE IMPORTATE, FATE DELLE IPOTESI SULLA BASE DELLA SOLA ANALISI STATICÀ DELLE FUNZIONALITÀ CHE IL MALWARE POTREBBE IMPLEMENTARE. UTILIZZATE LE FUNZIONI CHE SONO RICHIAMATE ALL'INTERNO DELLE LIBRERIE PER SUPPORTARE LE VOSTRE IPOTESI.

# PARAMETRI

ALL'INTERNO DELLA FUNZIONE MAIN() SONO PRESENTI TRE PARAMETRI IDENTIFICATI DALL'OFFSET POSITIVO RISPETTO AD EBP:

**ARGC:** Variabile utilizzata per rappresentare il numero di argomenti passati a un programma da riga di comando quando viene eseguito.

**ARGV:** Array di puntatori a stringhe utilizzato per contenere gli argomenti passati al programma da riga di comando quando viene eseguito.

**ENVP:** Array di puntatori a stringhe che contiene le variabili d'ambiente.

.text:004011D0	argc	= dword ptr	8
.text:004011D0	argv	= dword ptr	0Ch
.text:004011D0	envp	= dword ptr	10h

# VARIABILI

ALL'INTERNO DELLA FUNZIONE TROVIAMO 4 DIFFERENTI VARIABILI:

```
hModule= dword ptr -11Ch  
Data= byte ptr -118h  
var_117= byte ptr -117h  
var_8= dword ptr -8  
var_4= dword ptr -4
```

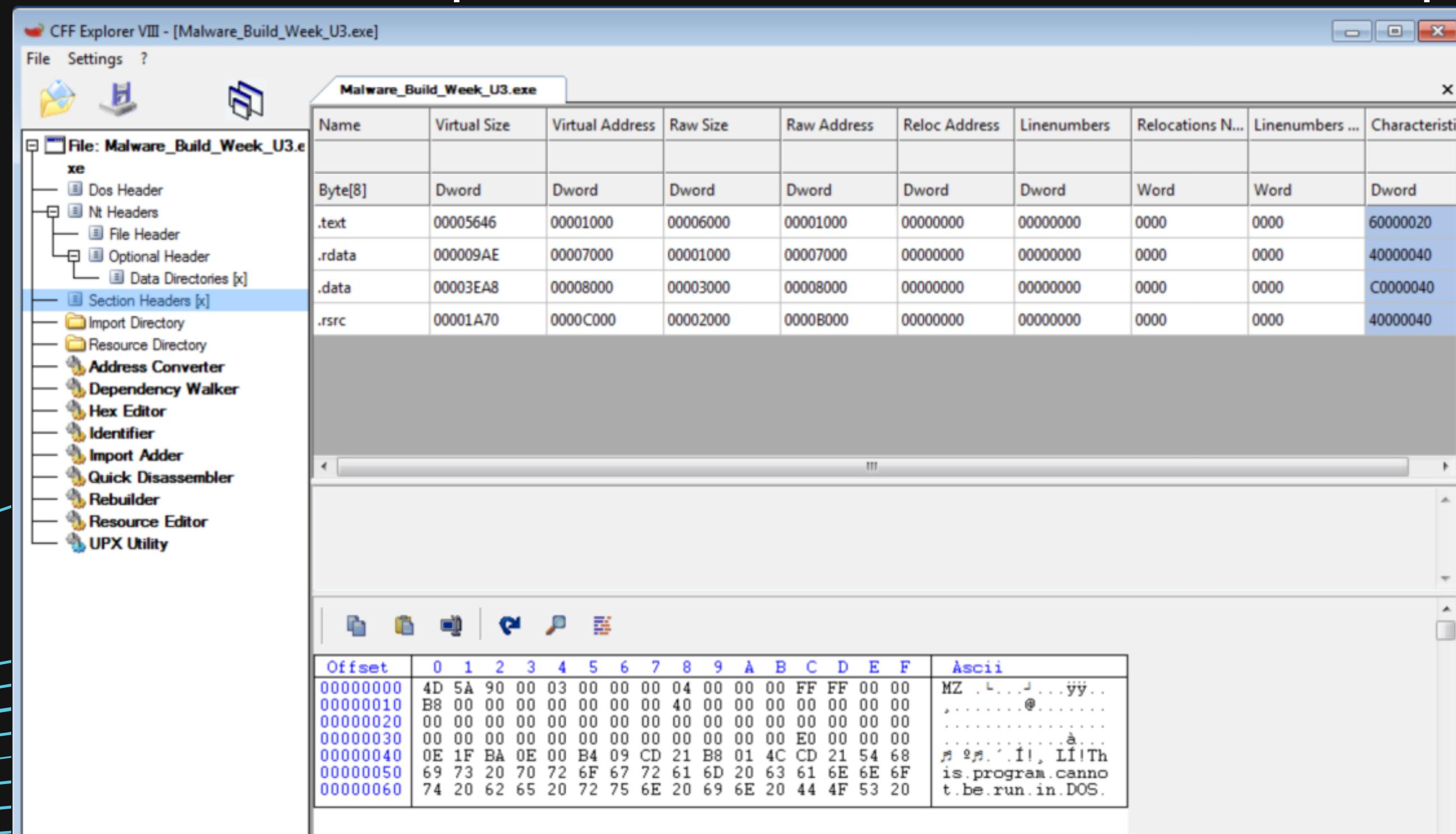
**HMODULE** è una variabile utilizzata per rappresentare un handle a un modulo, ovvero un puntatore a una risorsa che rappresenta un modulo (o un'eseguibile) caricato in memoria dal sistema operativo Windows. Gli handle sono spesso utilizzati per accedere e manipolare risorse di sistema come file, finestre, processi e moduli.

**DATA** è una direttiva utilizzata per definire variabili o costanti nella sezione dei dati del programma. Queste variabili possono essere utilizzate per memorizzare informazioni come numeri, stringhe o altre strutture dati necessarie per l'esecuzione del programma.

**VAR\_4 & VAR\_8 & VAR\_117** sono variabili locali

# CFF EXPLORER

CFF Explorer è un software per l'analisi e la modifica di file eseguibili Windows. Essenzialmente, permette di esaminare e manipolare elementi come la struttura interna dei file, le risorse, le intestazioni e altro ancora, utile per analizzare e modificare il funzionamento dei programmi.



# SEZIONI

All'interno dell'eseguibile sono presenti quattro sezioni:

**.TEXT:** Contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato.

**.RDATA:** Include le informazioni circa le librerie e le funzioni importate ed esportate dall' eseguibile.

**.DATA:** Contiene dati e variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.

**.RSRC:** Include le risorse utilizzate dall' eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell' eseguibile stesso, inoltre vi è la possibilità che contenga, nel caso dei DROPPER, il malware stesso.

# LIBRERIE

**Kernel32.dll:** Libreria che contiene le funzioni principali per interagire col sistema operativo, come per esempio la manipolazione di file e la gestione della memoria.

**Advapi32.dll:** Libreria che contiene le funzioni per interagire con i registri e i servizi del sistema operativo

Microsoft

The screenshot shows the CFF Explorer VIII interface with the title bar "CFF Explorer VIII - [Malware\_Build\_Week\_U3.exe]". The left sidebar displays the file structure of "File: Malware\_Build\_Week\_U3.e" with sections like Dos Header, Nt Headers, File Header, Optional Header, Data Directories, Section Headers, Import Directory (which is selected), Resource Directory, Address Converter, Dependency Walker, Hex Editor, and Identifier. The main pane shows a table for the executable "Malware\_Build\_Week\_U3.exe" with the following data:

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

# FUNZIONI

Nelle funzioni di libreria del Kernel32.dll troviamo 51 funzioni, prendiamo in considerazione per il nostro esercizio le seguenti funzioni:

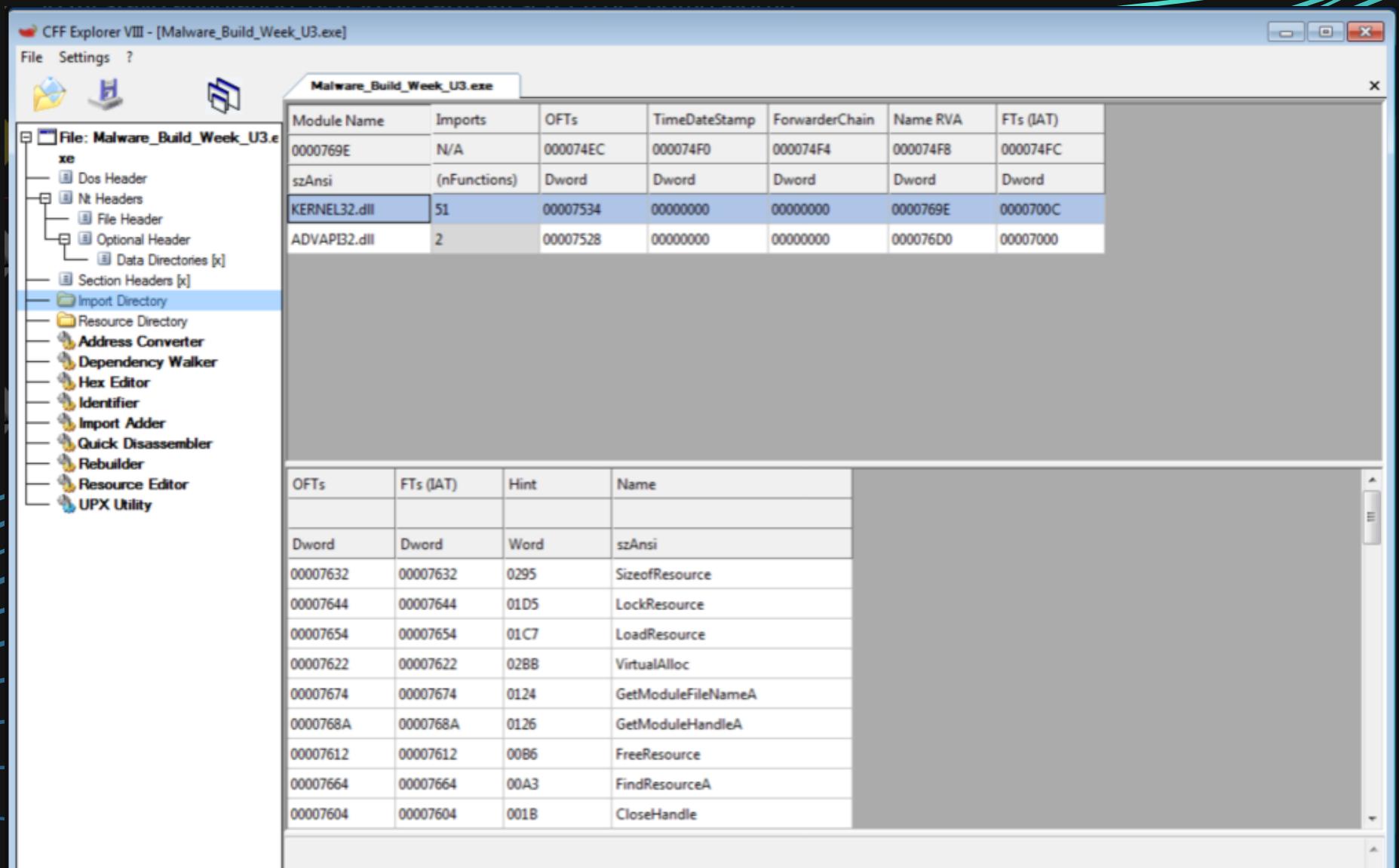
**FindResourceA()** : cerca la risorsa

**LoadResource()** : carica la risorsa

**SizeofResource()** : predisponde lo spazio per la risorsa

**LockResource()** : puntatore per le risorse selezionate

Generalmente utilizzate per recuperare altri file malevoli dalla sezione delle risorse ".rsrc".

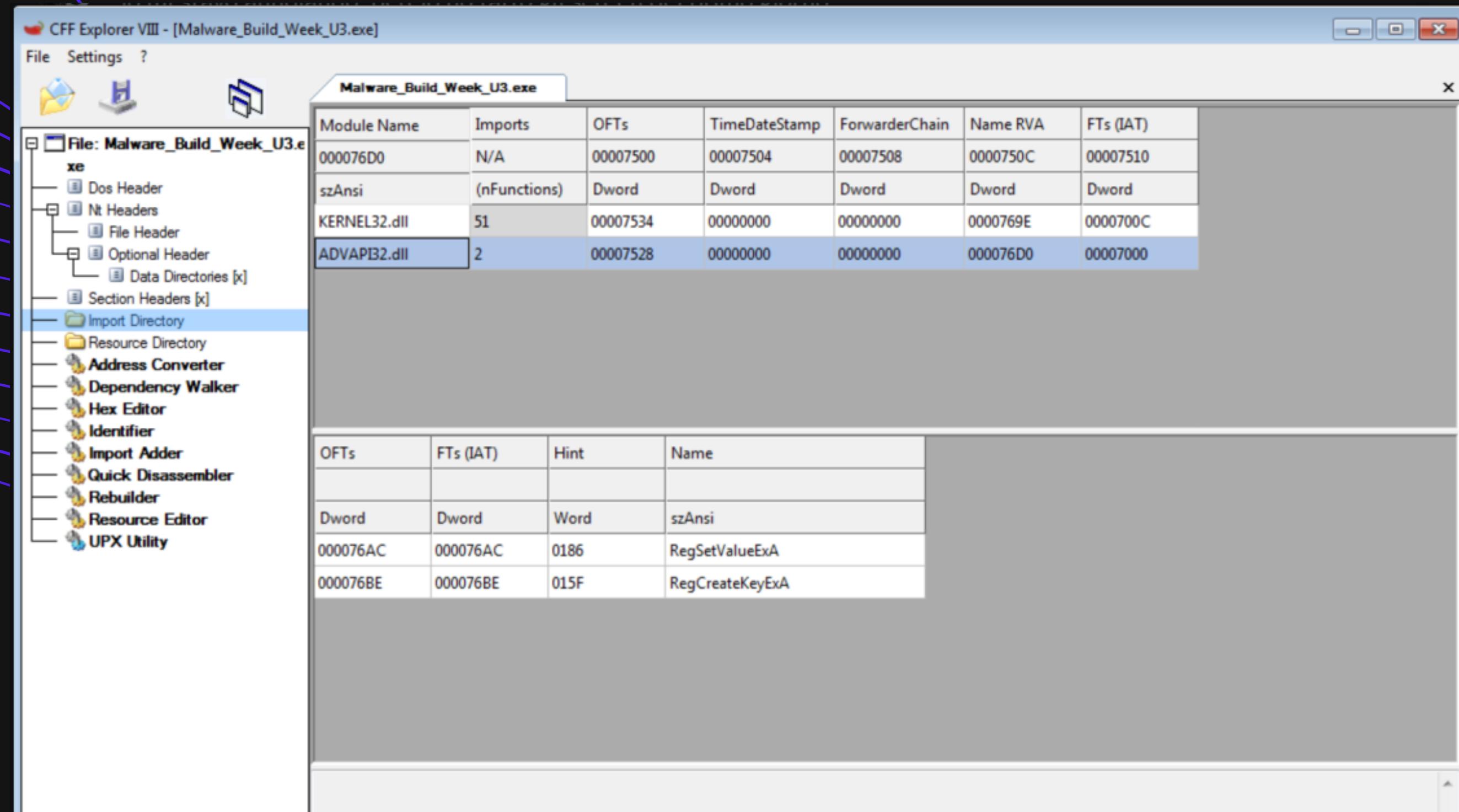


# FUNZIONI

Le 2 funzioni della libreria Advapi32.dll:

**RegCreateKeyExA()** : crea la chiave di registro

**RegSetValueExA()** : imposta i valori delle chiavi di registro



# IPOTESI

```
w1xshutdown  
w1xStartApplication  
w1xwkstaLockedSAS  
\MSGina  
ShellshutdownDialog  
w1xActivateUserShell  
w1xDisconnectNotify  
w1xDisplayLockedNotice  
w1xDisplaysASNotice  
w1xDisplayStatusMessage  
w1xGetConsoleSwitchCredentials  
w1xGetStatusMessage  
w1xInitialize  
w1xIsLockok  
w1xIsLogoffok  
w1xLoggedonSAS  
w1xLogoff  
w1xNegotiate  
w1xNetworkProviderLoad  
w1xReconnectNotify  
w1xRemoveStatusMessage  
w1xScreenSaverNotify  
w1xshutdown  
w1xStartApplication  
w1xwkstaLockedSAS  
GinaDLL  
Software\Microsoft\Windows NT\CurrentVersion\winlogon  
MSGina.dll  
Software\Microsoft\Windows NT\CurrentVersion\winlogon  
MSGina.dll  
GetStringGetTypeA  
GetStringTypeW  
dT@  
TGAD  
BINAY  
GinaDLL  
SOFTWARE\Microsoft\Windows NT\CurrentVersion\winlogon  
msgina32.dll  
\msgina32.dll  
Xq@  
Hq@
```

Da una prima analisi statica a basso livello si evince che il programma sia un **DROPPER**, come ben noto serve a nascondere un file malevolo più grande al suo interno per “passare inosservato”

Utilizzando strings:

Carica funzioni da 2 punti diversi del programma chiamando anche

**MSGina.dll** (che si occupa degli accessi sul sistema operativo).

# INDICE

## GIORNO 2:

- TRACCIA pag.12
- ANALISI STEP BY STEP pag.13
- TRASCRIZIONE IN C pag.16
- GINA.DLL pag.17
- CONCLUSIONI pag.18

## TRACCIA

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. (se serve, valutate anche un'altra o altre due righe assembly)
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C .

# ANALISI STEP BY STEP

All'indirizzo di memoria 401021, viene chiamata la funzione **RegCreateKeyExA** per creare una chiave di registro.

La funzione **RegCreateKeyExA** è una funzione dell'API di Windows utilizzata per creare o aprire una chiave nel Registro di sistema. In particolare, il suffisso "Ex" indica che questa è una versione estesa della funzione, che fornisce funzionalità aggiuntive rispetto alla versione di base. Serve a:

Creare una nuova chiave nel Registro di sistema se non esiste già.

Aprire una chiave esistente nel Registro di sistema per eseguire operazioni su di essa.

```
* .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\\CurrentVersion\\App Paths\\Windows Script Host\\Scripting"
* .text:0040101C          push    80000002h   ; hKey
* .text:00401021          call    ds:RegCreateKeyExA
```

# ANALISI STEP BY STEP

I parametri vengono passati attraverso il comando **push**, che inserisce i dati in cima allo stack.

Il parametro alla locazione 00401017 è la sottochiave “**SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon**” , dove risiedono informazioni riguardanti gli utenti e i programmi in avvio al login dell’utente.

```
* .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
```

# ANALISI STEP BY STEP

Con **test eax, eax**, **eax** viene verificato se il valore del registro è 0. Quindi se **eax** è uguale a 0, viene effettuato un salto alla locazione di memoria **00401032**. Quando **eax** è uguale a 0, la creazione della chiave è avvenuta con successo.

L'istruzione **test** è utilizzata per eseguire un'operazione logica tra due operandi senza modificarli.

L'operazione logica eseguita è l'**AND** bit a bit tra i due operandi.

L'istruzione **test** prende due operandi e setta le apposite flag del processore (come il flag Zero) in base al risultato dell'operazione AND bit a bit tra di essi. Tuttavia, a differenza dell'istruzione **and**, l'istruzione **test** non modifica il valore dei registri.

```
.text:00401027      test    eax, eax
.text:00401029      jz     short loc_401032
.text:0040102B      mov    eax, 1
.text:00401030      jmp    short loc_401032
.text:00401032      ; -----
.text:00401032      loc_401032:
.text:00401032      mov    ecx, [ebp+cbData]
.text:00401032      push   ecx
.text:00401032      mov    edx, [ebp+lpData]
.text:00401032      push   edx
.text:00401032      push   1
.text:00401032      push   0
.text:00401032      push   offset ValueName ; "GinaDLL"
00000103  0000000000401013: sub_401000+13
```

# TRASCRIZIONE IN C

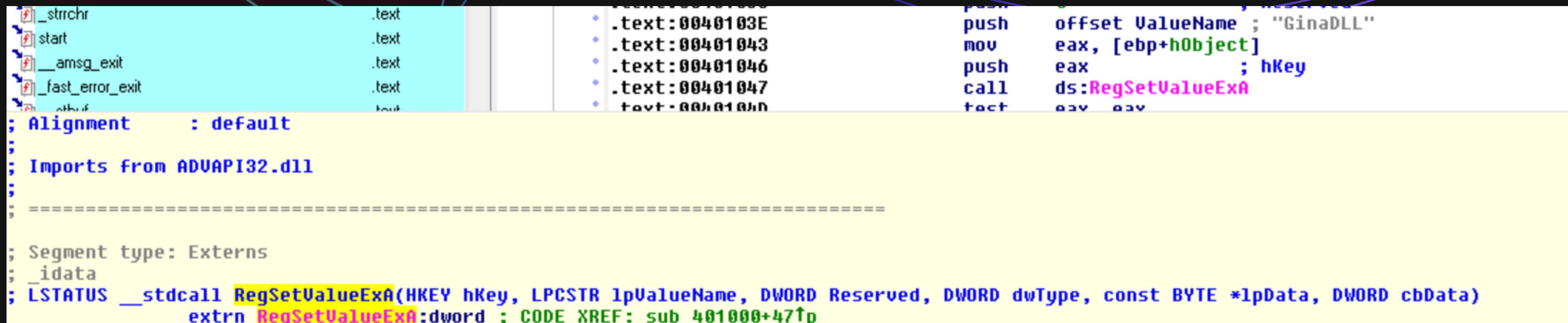
```
if a == 0:  
{  
    RegSetValue(cbData,lpData,dwType,Reserved,"GinaDLL");  
    //chiama RegSetValue();  
    closehandle(hObject);  
    a=1;  
}  
else:  
{  
    a=1;  
}  
return (a);
```

The screenshot shows a debugger interface with assembly code on the left and its C transcription on the right. The assembly code is in Intel syntax. A yellow box highlights the label `loc_401032`, which corresponds to the C code block above.

Assembly Address	Assembly Instruction	C Transcription
.text:00401027	test eax, eax	
.text:00401029	jz short loc_401032	
.text:0040102B	mov eax, 1	
.text:00401030	jmp short loc_401032	
.text:00401032	;	
.text:00401032 loc_401032:	mov ecx, [ebp+cbData] push ecx mov edx, [ebp+lpData] push edx push 1 push 0 push offset ValueName ; "GinaDLL"	RegSetValue(cbData,lpData,dwType,Reserved,"GinaDLL"); //chiama RegSetValue(); closehandle(hObject); a=1;
000001013	0000000000401013: sub_401000+13	

# GINA.DLL

Il valore di "ValueName" corrisponde a "GinaDLL", che rappresenta l'acronimo di Graphical Identification and Authentication DLL. Questa è una libreria di collegamento dinamico (DLL) che gestisce l'interfaccia grafica per l'autenticazione degli utenti in ambienti Windows.



```
file .text
    .text:0040103E push    offset ValueName ; "GinaDLL"
    .text:00401043 mov     eax, [ebp+hObject]
    .text:00401046 push    eax
    .text:00401047 call    ds:RegSetValueExA
    .text:0040104D test    eax, eax

; Alignment : default
;
; Imports from ADVAPI32.dll
;
=====
; Segment type: Externs
; _idata
; LSTATUS __stdcall RegSetValueExA(HKEY hKey, LPCSTR lpValueName, DWORD Reserved, DWORD dwType, const BYTE *lpData, DWORD cbData)
extrn RegSetValueExA:dword ; CODE XREF: sub_401000+47↑p
```

# CONCLUSIONI

I malware stanno diventando sempre più sofisticati nell'utilizzare le chiavi di registro per ottenere una persistenza nel sistema.

Le chiavi di registro, essendo parte fondamentale dei sistemi Windows, offrono un'opportunità conveniente per i malware di inserire il loro codice nel flusso di avvio del sistema.

Una volta stabilita la presenza, i malware possono essere difficili da individuare e rimuovere, poiché possono nascondersi nel registro e mascherare le loro attività. Utilizzando varie tecniche, come l'inserimento di voci nelle aree di avvio del registro, i malware possono garantire una presenza persistente.

Per mitigare questa minaccia, è necessario adottare pratiche di sicurezza informatica solide, come l'uso di software antivirus aggiornato e l'educazione degli utenti sulle minacce informatiche.

# INDICE

## GIORNO 3:

- TRACCIA pag.22
- TGAD pag.23
- IMPLEMENTAZIONI DEL MALWARE pag.24
- DIAGRAMMA DI FLUSSO pag.26
- ANALISI STATICÀ BASICA pag.31
- ANALISI DELLA FUNZIONE MAIN pag.34

## TRACCIA

Riprendete l'analisi del codice, analizzando le routine tra le locazioni di memoria  
00401080 e 00401128:

- Qual è il valore del parametro ««ResourceName»» passato alla funzione FindResourceA();
  - Il susseguirsi delle chiamate di funzione che effettua il Malware in questa sezione di codice l'abbiamo visto durante le lezioni teoriche. Che funzionalità sta implementando il Malware?
  - È possibile identificare questa funzionalità utilizzando l'analisi statica basica ? (dal giorno 1 in pratica)
    - In caso di risposta affermativa, elencare le evidenze a supporto.
- Entrambe le funzionalità principali del Malware viste finora sono richiamate all'interno della funzione Main().

# TGAD

Nel codice assembly indicato dalla traccia, il nome della risorsa viene caricato direttamente in ECX prima della chiamata a FindResourceA.

**004010C4 | . 51 PUSH ECX ; |ResourceName => "TGAD"**

Questo significa che il nome della risorsa passato alla funzione FindResourceA è il valore contenuto nel registro ECX al momento della chiamata, che nel nostro caso è "TGAD". Pertanto, "TGAD" è il parametro lpName passato alla funzione FindResourceA.

```
004010B3 | .vE9 07010000 JMP Malware_.004011BF
004010B8 > A1 30804000 MOU EAX,DWORD PTR DS:[408030]
004010BD . 50 PUSH EAX
004010BE . 8B0D 34804000 MOU ECX,DWORD PTR DS:[408034]
004010C4 . 51 PUSH ECX
004010C5 . 8B55 08 MOU EDX,DWORD PTR SS:[EBP+8]
004010C8 . 52 PUSH EDX
004010C9 . FF15 28704000 CALL DWORD PTR DS:[<&KERNEL32.FindResou:
004010CF . 8945 EC MOU DWORD PTR SS:[EBP-14],EAX
004010D2 . 837D EC 00 CMP DWORD PTR SS:[EBP-14],0
004010D6 .v75 07 JNZ SHORT Malware_.004010DF
004010D8 . 33C0 XOR EAX,EAX
004010DA .vE9 E0000000 JMP Malware_.004011BF
004010E0 > 004E EC MOU EAX,DWORD PTR SS:[EBP-14]
```

ResourceType => "BINARY"  
Malware\_.00408038  
ResourceName => "TGAD"  
hModule  
**FindResourceA**

# IMPLEMENTAZIONI DEL MALWARE:

## FindResourceA:

**Descrizione:** La funzione FindResourceA appartiene alla libreria di sistema Windows KERNEL32.dll ed è utilizzata per individuare una risorsa all'interno di un modulo specificato, come un file eseguibile o una DLL. La ricerca della risorsa avviene tramite il tipo e il nome della risorsa specificati come parametri.

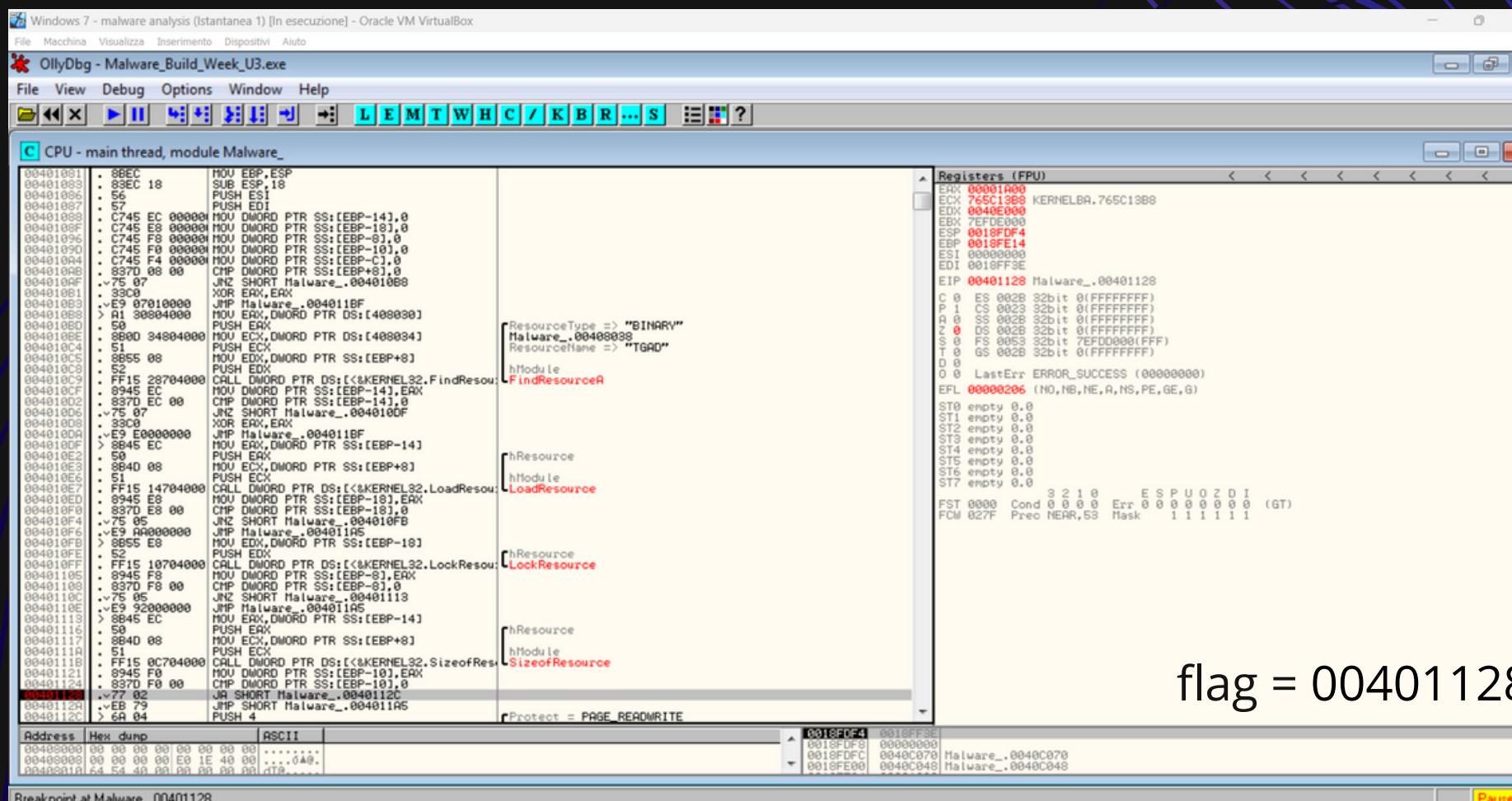
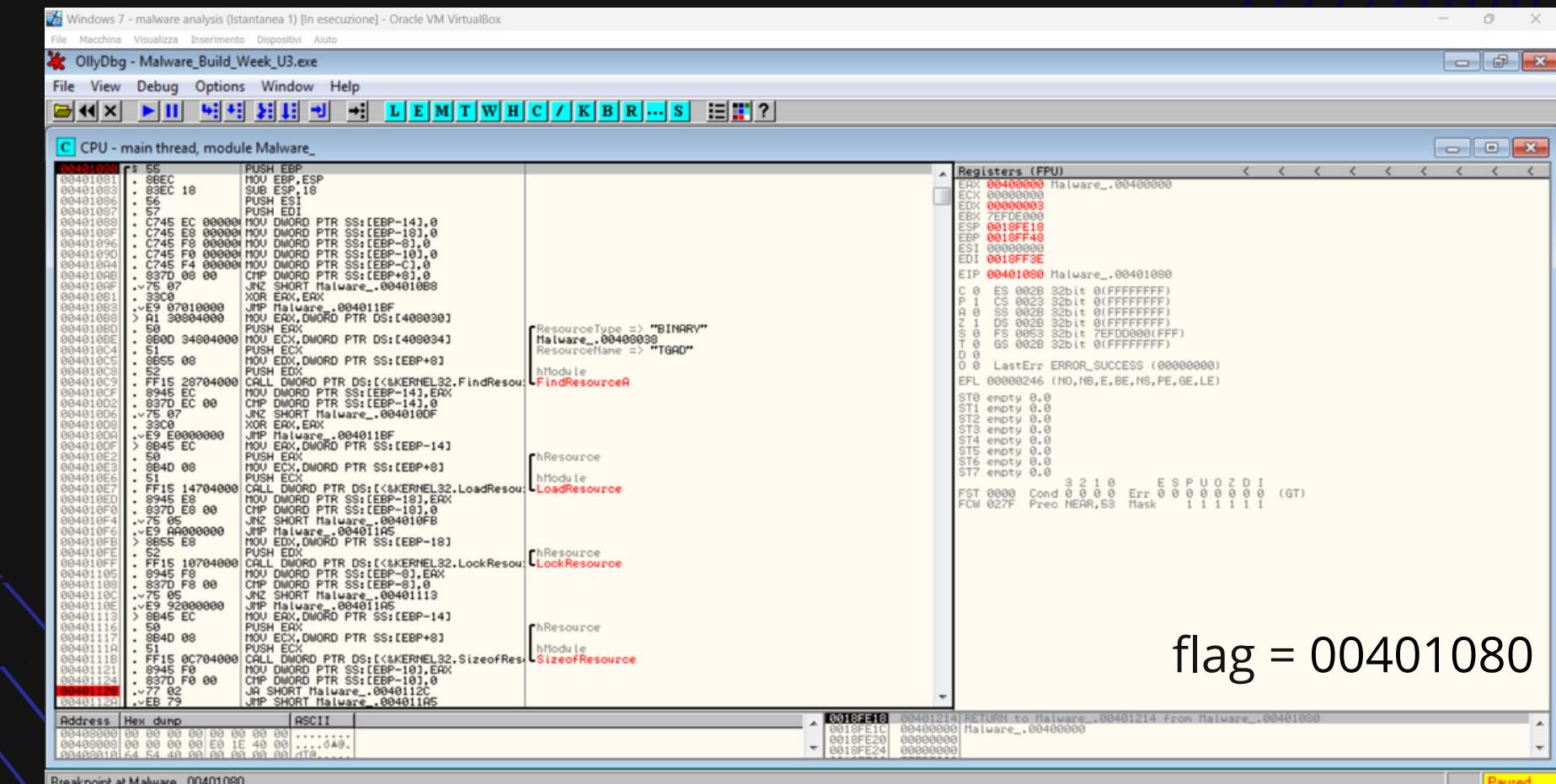
### Parametri:

**hModule:** Handle del modulo in cui cercare la risorsa.

**lpName:** Puntatore a una stringa che specifica il nome della risorsa.

**lpType:** Puntatore a una stringa che specifica il tipo della risorsa.

**Valore di ritorno:** Handle alla risorsa individuata, o NULL se la risorsa non viene trovata.



## LoadResource:

**Descrizione:** La funzione LoadResource di KERNEL32.dll viene utilizzata per caricare una risorsa individuata da FindResourceA in memoria.

### Parametri:

**hModule:** Handle del modulo in cui è stata individuata la risorsa.

**hResource:** Handle della risorsa individuata.

**Valore di ritorno:** Puntatore alla risorsa in memoria, o NULL in caso di errore.

# IMPLEMENTAZIONI DEL MALWARE:

## LockResource:

**Descrizione:** La funzione LockResource di KERNEL32.dll viene utilizzata per ottenere un puntatore alla risorsa caricata in memoria tramite LoadResource.

### Parametri:

**hResource:** Handle della risorsa caricata in memoria.

**Valore di ritorno:** Puntatore alla risorsa, o NULL in caso di errore.

```
Windows 7 - malware analysis (Istantanea 1) [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivo Aiuto
* OllyDbg - Malware_Build_Week_U3.exe
File View Debug Options Window Help
CPU - main thread, module Malware_
00401080 E 55 PUSH EBP
00401083 : 8BEC MOV EBP,ESP
00401085 : 83EC 18 SUB ESP,18
00401087 : 56 PUSH ES
00401088 : 57 PUSH EDI
00401089 C745 EC 000000 MOU DWORD PTR SS:[EBP-14],0
0040108C : 8BEC MOV EBP,ESP
0040108D : 83EC 18 SUB ESP,18
0040108F : 56 PUSH ES
00401090 : 57 PUSH EDI
00401091 C745 F8 000000 MOU DWORD PTR SS:[EBP-10],0
00401094 : 8BEC MOV EBP,ESP
00401095 : 83EC 18 SUB ESP,18
00401096 : 56 PUSH ES
00401097 : 57 PUSH EDI
00401098 C745 F0 000000 MOU DWORD PTR SS:[EBP-10],0
0040109B : 8BEC MOV EBP,ESP
0040109C : 83EC 18 SUB ESP,18
0040109D : 56 PUSH ES
0040109E : 57 PUSH EDI
0040109F C745 F4 000000 MOU DWORD PTR SS:[EBP-C1],0
004010A2 : 8B70 08 00 CMP DWORD PTR SS:[EBP+8],0
004010A5 : 77 07 JNZ SHORT Malware_..004010B8
004010A8 : E9 07010000 JMP Malware_..004010B8
004010B1 : A1 30004000 MOU EAX,DWORD PTR DS:[400030]
004010B4 : 8B00 34004000 PUSH ECX
004010B7 : 51 PUSH EDI
004010B8 : 8B55 08 MOU EDX,DWORD PTR SS:[EBP+8]
004010B9 : 8B45 EC PUSH ECX
004010B9 FF15 28704000 CALL DWORD PTR DS:[<&KERNEL32.FindResou
004010C2 : 8945 EC MOU DWORD PTR SS:[EBP-14],ERX
004010C5 : 8370 08 00 CMP DWORD PTR SS:[EBP+8],0
004010C8 : 77 07 JNZ SHORT Malware_..004010D6
004010CB : E9 E00000000 JMP Malware_..004010D6
004010D5 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-14]
004010D8 : 8370 08 00 CMP DWORD PTR SS:[EBP+8],0
004010DB : 77 05 JNZ SHORT Malware_..00401113
004010E0 : E9 92000000 CALL DWORD PTR DS:[<&KERNEL32.LockResou
004010E3 : 8945 F8 MOU DWORD PTR SS:[EBP-10],ERX
004010E6 : 8370 F8 00 CMP DWORD PTR SS:[EBP-10],0
004010E9 : 77 02 JNZ SHORT Malware_..00401121
004010FB : EB 79 JMP SHORT Malware_..00401145
00401113 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401116 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401119 : 8370 08 00 CMP DWORD PTR SS:[EBP+8],0
00401122 : 77 05 JNZ SHORT Malware_..00401145
00401125 : E9 92000000 CALL DWORD PTR DS:[<&KERNEL32.SizeofResou
00401128 : 8945 F8 MOU DWORD PTR SS:[EBP-10],ERX
00401131 : 8370 F8 00 CMP DWORD PTR SS:[EBP-10],0
00401134 : 77 02 JNZ SHORT Malware_..00401152
00401137 : EB 79 JMP SHORT Malware_..00401145
00401140 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401143 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401146 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401149 : 77 05 JNZ SHORT Malware_..00401145
00401152 : E9 92000000 CALL DWORD PTR DS:[<&KERNEL32.LockResou
00401155 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401158 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401161 : 77 02 JNZ SHORT Malware_..00401179
00401164 : EB 79 JMP SHORT Malware_..00401145
00401167 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401170 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401173 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401176 : 77 05 JNZ SHORT Malware_..00401145
00401179 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401182 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401185 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401188 : 77 02 JNZ SHORT Malware_..00401121
00401191 : EB 79 JMP SHORT Malware_..00401145
00401194 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401197 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401200 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401203 : 77 05 JNZ SHORT Malware_..00401145
00401206 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401209 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401212 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401215 : 77 02 JNZ SHORT Malware_..00401121
00401218 : EB 79 JMP SHORT Malware_..00401145
00401221 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401224 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401227 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401230 : 77 05 JNZ SHORT Malware_..00401145
00401233 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401236 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401239 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401242 : 77 02 JNZ SHORT Malware_..00401121
00401245 : EB 79 JMP SHORT Malware_..00401145
00401248 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401251 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401254 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401257 : 77 05 JNZ SHORT Malware_..00401145
00401260 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401263 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401266 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401269 : 77 02 JNZ SHORT Malware_..00401121
00401272 : EB 79 JMP SHORT Malware_..00401145
00401275 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401278 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401281 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401284 : 77 05 JNZ SHORT Malware_..00401145
00401287 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401290 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401293 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401296 : 77 02 JNZ SHORT Malware_..00401121
00401299 : EB 79 JMP SHORT Malware_..00401145
00401302 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401305 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401308 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401311 : 77 05 JNZ SHORT Malware_..00401145
00401314 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401317 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401320 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401323 : 77 02 JNZ SHORT Malware_..00401121
00401326 : EB 79 JMP SHORT Malware_..00401145
00401329 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401332 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401335 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401338 : 77 05 JNZ SHORT Malware_..00401145
00401341 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401344 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401347 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401350 : 77 02 JNZ SHORT Malware_..00401121
00401353 : EB 79 JMP SHORT Malware_..00401145
00401356 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401359 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401362 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401365 : 77 05 JNZ SHORT Malware_..00401145
00401368 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401371 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401374 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401377 : 77 02 JNZ SHORT Malware_..00401121
00401380 : EB 79 JMP SHORT Malware_..00401145
00401383 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401386 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401389 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401392 : 77 05 JNZ SHORT Malware_..00401145
00401395 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401398 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401401 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401404 : 77 02 JNZ SHORT Malware_..00401121
00401407 : EB 79 JMP SHORT Malware_..00401145
00401410 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401413 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401416 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401419 : 77 05 JNZ SHORT Malware_..00401145
00401422 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401425 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401428 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401431 : 77 02 JNZ SHORT Malware_..00401121
00401434 : EB 79 JMP SHORT Malware_..00401145
00401437 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401440 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401443 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401446 : 77 05 JNZ SHORT Malware_..00401145
00401449 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401452 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401455 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401458 : 77 02 JNZ SHORT Malware_..00401121
00401461 : EB 79 JMP SHORT Malware_..00401145
00401464 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401467 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401470 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401473 : 77 05 JNZ SHORT Malware_..00401145
00401476 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401479 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401482 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401485 : 77 02 JNZ SHORT Malware_..00401121
00401488 : EB 79 JMP SHORT Malware_..00401145
00401491 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401494 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401497 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401500 : 77 05 JNZ SHORT Malware_..00401145
00401503 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401506 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401509 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401512 : 77 02 JNZ SHORT Malware_..00401121
00401515 : EB 79 JMP SHORT Malware_..00401145
00401518 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401521 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401524 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401527 : 77 05 JNZ SHORT Malware_..00401145
00401530 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401533 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401536 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401539 : 77 02 JNZ SHORT Malware_..00401121
00401542 : EB 79 JMP SHORT Malware_..00401145
00401545 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401548 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401551 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401554 : 77 05 JNZ SHORT Malware_..00401145
00401557 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401560 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401563 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401566 : 77 02 JNZ SHORT Malware_..00401121
00401569 : EB 79 JMP SHORT Malware_..00401145
00401572 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401575 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401578 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401581 : 77 05 JNZ SHORT Malware_..00401145
00401584 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401587 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401590 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401593 : 77 02 JNZ SHORT Malware_..00401121
00401596 : EB 79 JMP SHORT Malware_..00401145
00401599 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401602 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401605 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401608 : 77 05 JNZ SHORT Malware_..00401145
00401611 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401614 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401617 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401620 : 77 02 JNZ SHORT Malware_..00401121
00401623 : EB 79 JMP SHORT Malware_..00401145
00401626 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401629 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401632 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401635 : 77 05 JNZ SHORT Malware_..00401145
00401638 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.LockResou
00401641 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401644 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401647 : 77 02 JNZ SHORT Malware_..00401121
00401650 : EB 79 JMP SHORT Malware_..00401145
00401653 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401656 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401659 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401662 : 77 05 JNZ SHORT Malware_..00401145
00401665 : E9 92000000 CALL DWOD PTR DS:[<&KERNEL32.SizeofResou
00401668 : 8945 F8 MOU DWOD PTR SS:[EBP-10],ERX
00401671 : 8370 F8 00 CMP DWOD PTR SS:[EBP-10],0
00401674 : 77 02 JNZ SHORT Malware_..00401121
00401677 : EB 79 JMP SHORT Malware_..00401145
00401680 : E9 87010000 MOU EAX,DWORD PTR SS:[EBP-18]
00401683 : 8B45 EC MOU EDX,DWORD PTR SS:[EBP-18]
00401686 : 8370 08 00 CMP DWOD PTR SS:[EBP+8],0
00401689 : 77 05
```

# lessCopy code

Ecco il diagramma di flusso del codice assembly fornito all'interno del malware preso in esame, insieme alle relative spiegazioni:

```
lessCopy code
Inizio | V PUSH EBP
          MOV EBP, ESP
          SUB ESP, 18
          PUSH ESI
          PUSH EDI
          MOV DWORD PTR SS:[EBP-14], 0
          MOV DWORD PTR SS:[EBP-18], 0
          MOV DWORD PTR SS:[EBP-8], 0
          MOV DWORD PTR SS:[EBP-10], 0
          MOV DWORD PTR SS:[EBP-C], 0
          CMP DWORD PTR SS:[EBP+8], 0
          JNZ -> Trovato_Handle
          XOR EAX, EAX
          JMP -> Fine
```

in sintesi:

Questa parte iniziale del codice imposta l'ambiente e le variabili necessarie per l'esecuzione successiva del programma. Viene controllato se è presente un handle, rappresentato dal valore nella posizione di stack [EBP+8]. Se è presente, il flusso del programma salta alla sezione "Trovato\_Handle"; altrimenti, viene eseguita l'istruzione XOR EAX, EAX per azzerare il registro EAX, seguita da un salto alla fine del programma.

# vbnetCopy code

vbnetCopy code

Trovato\_Handle: MOV EAX, DWORD PTR DS:[408030]

PUSH EAX

MOV ECX, DWORD PTR DS:[408034]

PUSH ECX

MOV EDX, DWORD PTR SS:[EBP+8]

PUSH EDX

CALL KERNEL32.FindResourceA

MOV DWORD PTR SS:[EBP-14], EAX

CMP DWORD PTR SS:[EBP-14], 0

JNZ -> Risorsa\_Trovata

XOR EAX, EAX

JMP -> Fine

Spiegazione:

Se è presente un handle, il programma chiama la funzione FindResourceA della libreria KERNEL32.dll per individuare la risorsa specificata dal tipo e nome forniti. L'handle del modulo viene caricato da [408030], il tipo di risorsa da [408034], e l'handle passato come parametro è quello presente nella posizione [EBP+8].

Se la chiamata a FindResourceA restituisce un handle valido, il flusso del programma salta alla sezione "Risorsa\_Trovata"; altrimenti, viene eseguita l'istruzione XOR EAX, EAX per azzerare il registro EAX, seguita da un salto alla fine del programma.

# objectivecCopy code

objectivecCopy code

Risorsa\_Trovata:

MOV EAX, DWORD PTR SS:[EBP-14]

PUSH EAX

MOV ECX, DWORD PTR SS:[EBP+8]

PUSH ECX CALL KERNEL32.LoadResource

MOV DWORD PTR SS:[EBP-18], EAX

CMP DWORD PTR SS:[EBP-18], 0

JNZ -> Risorsa\_Caricata

JMP -> Fine

Si esplica che:

Se FindResourceA restituisce un handle valido, il programma chiama la funzione LoadResource della libreria KERNEL32.dll per caricare la risorsa in memoria. L'handle della risorsa trovata è memorizzato in [EBP-14].

Se LoadResource ha successo nel caricare la risorsa, il flusso del programma salta alla sezione "Risorsa\_Caricata"; altrimenti, viene eseguita l'istruzione JMP per saltare alla fine del programma.

# objectivecCopy code

objectivecCopy code

Risorsa\_Caricata:

```
MOV EDX, DWORD PTR SS:[EBP-18]  
PUSH EDX CALL KERNEL32.LockResource
```

```
MOV DWORD PTR SS:[EBP-8], EAX  
CMP DWORD PTR SS:[EBP-8], 0
```

JNZ ->Risorsa\_Lockata

JMP -> Fine

Quindi:

Se LoadResource ha successo nel caricare la risorsa in memoria, il programma chiama la funzione LockResource della libreria KERNEL32.dll per ottenere un puntatore alla risorsa in memoria.

L'handle della risorsa caricata è memorizzato in [EBP-18].

Se LockResource ha successo nel bloccare la risorsa in memoria, il flusso del programma salta alla sezione "Risorsa\_Lockata"; altrimenti, viene eseguita l'istruzione JMP per saltare alla fine del programma.

# mathematicaCopy code

mathematicaCopy code

Risorsa\_Lockata:

MOV EAX, DWORD PTR SS:[EBP-14]

PUSH EAX

MOV ECX, DWORD PTR SS:[EBP+8]

PUSH ECX

CALL KERNEL32.SizeofResource

MOV DWORD PTR SS:[EBP-10], EAX

CMP DWORD PTR SS:[EBP-10], 0

JA -> Fine Fine

Di conseguenza:

Se LockResource ha successo nel bloccare la risorsa in memoria, il programma chiama la funzione SizeofResource della libreria KERNEL32.dll per ottenere la dimensione della risorsa. L'handle della risorsa è memorizzato in [EBP-14].

Se la dimensione della risorsa è maggiore di zero, il flusso del programma continua e raggiunge la fine del programma; altrimenti, il flusso del programma salta direttamente alla fine.

# ANALISI STATICÀ BASICA

## Preparazione dell'ambiente:

Le istruzioni PUSH EBP, MOV EBP, ESP, SUB ESP, 18, PUSH ESI, e PUSH EDI vengono utilizzate per preparare lo stack e i registri necessari per l'esecuzione del codice.

Le istruzioni MOV DWORD PTR SS:[EBP-14], 0, MOV DWORD PTR SS:[EBP-18], 0, MOV DWORD PTR SS:[EBP-8], 0, MOV DWORD PTR SS:[EBP-10], 0, e MOV DWORD PTR SS:[EBP-C], 0 inizializzano alcune variabili locali a zero.

## Controllo dell'handle:

L'istruzione CMP DWORD PTR SS:[EBP+8], 0 confronta il valore presente alla posizione di memoria [EBP+8] con zero, che probabilmente rappresenta un handle.

Se l'handle non è zero (indicando che è stato fornito un handle valido), il flusso del programma salta alla sezione "Trovato\_Handle", altrimenti viene eseguita l'istruzione XOR EAX, EAX per azzerare il registro EAX, seguita da un salto alla fine del programma.

# ANALISI STATICÀ BASICA

## Ricerca della risorsa:

Se l'handle è valido, il programma utilizza la funzione FindResourceA per cercare una risorsa all'interno del modulo specificato.

I valori dei tipi di risorsa e dei nomi sembrano essere predefiniti.

Se la ricerca della risorsa ha successo, il flusso del programma salta alla sezione

"Risorsa\_Trovata";

altrimenti, viene eseguita l'istruzione XOR EAX, EAX per azzerare il registro EAX, seguita da un salto alla fine del programma

Come possiamo vedere è possibile avere un'idea di ciò che va ad eseguire il programma con una semplice analisi basica.

i nostri campanelli d'allarme scatteranno sulle diciture che vanno a modificare e implementare le chiavi di registro alterandone i valori.

# ANALISI STATICÀ BASICA

## Caricamento della risorsa:

Se la risorsa è stata trovata, il programma utilizza la funzione LoadResource per caricare la risorsa in memoria.

Se il caricamento ha successo, il flusso del programma salta alla sezione "Risorsa\_Caricata"; altrimenti, viene eseguita l'istruzione JMP per saltare alla fine del programma.

## Bloccaggio della risorsa:

Se la risorsa è stata caricata correttamente, il programma utilizza la funzione LockResource per ottenere un puntatore alla risorsa in memoria.

Se il bloccaggio ha successo, il flusso del programma salta alla sezione "Risorsa\_Lockata"; altrimenti, viene eseguita l'istruzione JMP per saltare alla fine del programma.

## Misurazione della risorsa:

Se la risorsa è stata bloccata con successo, il programma utilizza la funzione SizeofResource per ottenere la dimensione della risorsa.

Se la dimensione della risorsa è maggiore di zero, il flusso del programma continua verso la fine del programma; altrimenti, il flusso del programma salta direttamente alla fine.

# ANALISI DELLA FUNZIONE MAIN

La funzione **main** inizia dichiarando diverse variabili per gestire il processo di caricamento e l'elaborazione della risorsa. Queste variabili includono `moduleName`, `resourceType`, `resourceName`, `dwSize`, e `lpData`. `moduleName` è un puntatore a carattere inizializzato a `nullptr`, indicando che viene utilizzato il modulo corrente. `resourceType` e `resourceName` sono puntatori a carattere ottenuti attraverso la macro `MAKEINTRESOURCE`, che rappresentano rispettivamente il tipo e il nome della risorsa da caricare. `dwSize` è una variabile di tipo `DWORD` inizializzata a 0, utilizzata per memorizzare la dimensione della risorsa, mentre `lpData` è un puntatore a `void` utilizzato per contenere l'indirizzo di memoria della risorsa caricata.

## Caricamento della Risorsa:

Dopo la dichiarazione delle variabili, la funzione `loadResource` viene chiamata per caricare la risorsa specificata utilizzando i parametri `moduleName`, `resourceType`, `resourceName`, e `dwSize`. Se il caricamento fallisce, ossia se `lpData` è `nullptr`, il programma emette un messaggio di errore e termina con un codice di errore 1.

# ANALISI DELLA FUNZIONE MAIN

## Operazioni Aggiuntive:

Nel caso in cui il caricamento della risorsa abbia successo, vengono eseguite operazioni aggiuntive sulla risorsa caricata. Tuttavia, i dettagli specifici di queste operazioni non sono inclusi nel codice fornito e richiederebbero un'analisi più approfondita.

## Gestione del Modulo Corrente:

La funzione `GetModuleHandle(NULL)` viene utilizzata per ottenere l'handle del modulo corrente (l'applicazione stessa) e memorizzarlo nella variabile `hModule`. Tuttavia, al momento non sembra essere utilizzato nel contesto della funzione `main`.

## Altre Variabili Locali:

Infine, vengono dichiarate altre variabili locali, come `Data`, `cbData`, `lpData2`, `msgina32_dll`, e `p`, che sembrano essere utilizzate per ulteriori operazioni all'interno della funzione, come la gestione delle stringhe e delle operazioni di file system.

La funzione `main` costituisce un punto di partenza fondamentale per il processo di caricamento e gestione delle risorse all'interno dell'applicazione.

```
44 int main(int argc, char** argv) {
45     // Nome del modulo corrente (l'applicazione stessa)
46     const char* moduleName = nullptr; // NULL per il modulo corrente
47
48     // Tipo e nome della risorsa da cercare
49     const char* resourceType = MAKEINTRESOURCE(408030); // "BINARY"
50     const char* resourceName = MAKEINTRESOURCE(408034); // "TGAD"
51
52     // Variabile per memorizzare la dimensione della risorsa
53     DWORD dwSize = 0;
54
55     // Carica la risorsa
56     LPVOID lpData = loadResource(moduleName, resourceType, resourceName, dwSize);
57     if (!lpData) {
58         std::cerr << "Errore durante il caricamento della risorsa." << std::endl;
59         return 1;
60     }
61
62     // Operazioni aggiuntive con la risorsa...
63     std::cout << "Risorsa caricata correttamente. Dimensione: " << dwSize << " bytes." << std::endl;
64
65     // Ottiene l'handle del modulo corrente (l'applicazione stessa)
66     HMODULE hModule = GetModuleHandle(NULL);
67
68     // Dichiarazione delle variabili locali
69     char Data[MAX_PATH] = {0};           // Buffer per contenere il percorso del modulo corrente
70     DWORD cbData;                      // Dimensione dei dati da passare a una funzione
71     LPSTR lpData2;                     // Puntatore ai dati da passare a una funzione
72     const char *msgina32_dll = "\\\msgina32.dll"; // Stringa per cercare il percorso di msgina32.dll
73     char *p;                           // Puntatore per le operazioni di stringa
74
75     // Ciclo infinito per garantire che l'operazione sia completata con successo
76     while (1) {
77         // Ottiene l'handle del modulo corrente (l'applicazione)
```

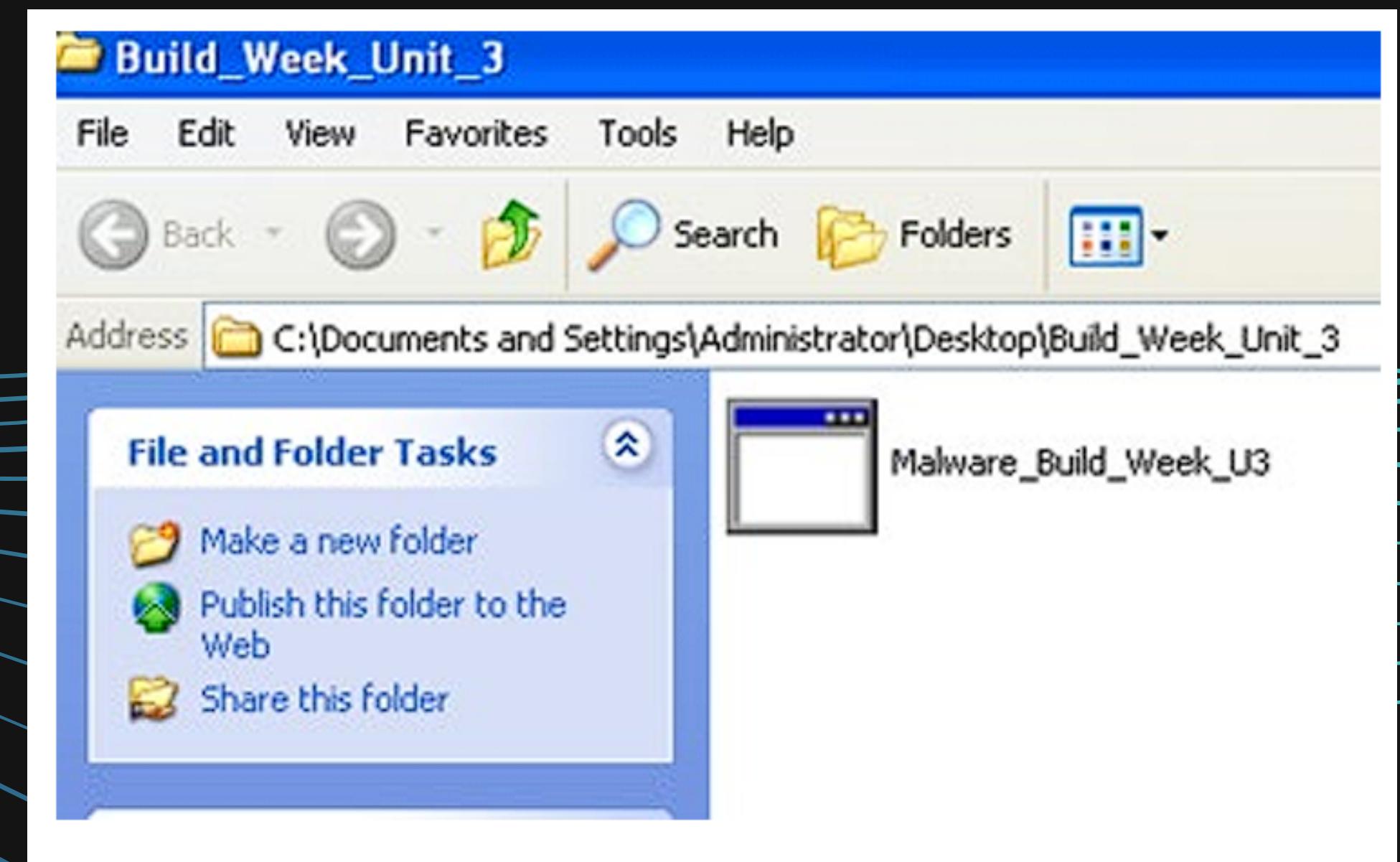
# INDICE

## GIORNO 4:

- TRACCIA pag.38
- CONTENUTO CARTELLA pag.41
- CREAZIONE CHIAVI DI REGISTRO E DEL FILE pag.42

# TRACCIA

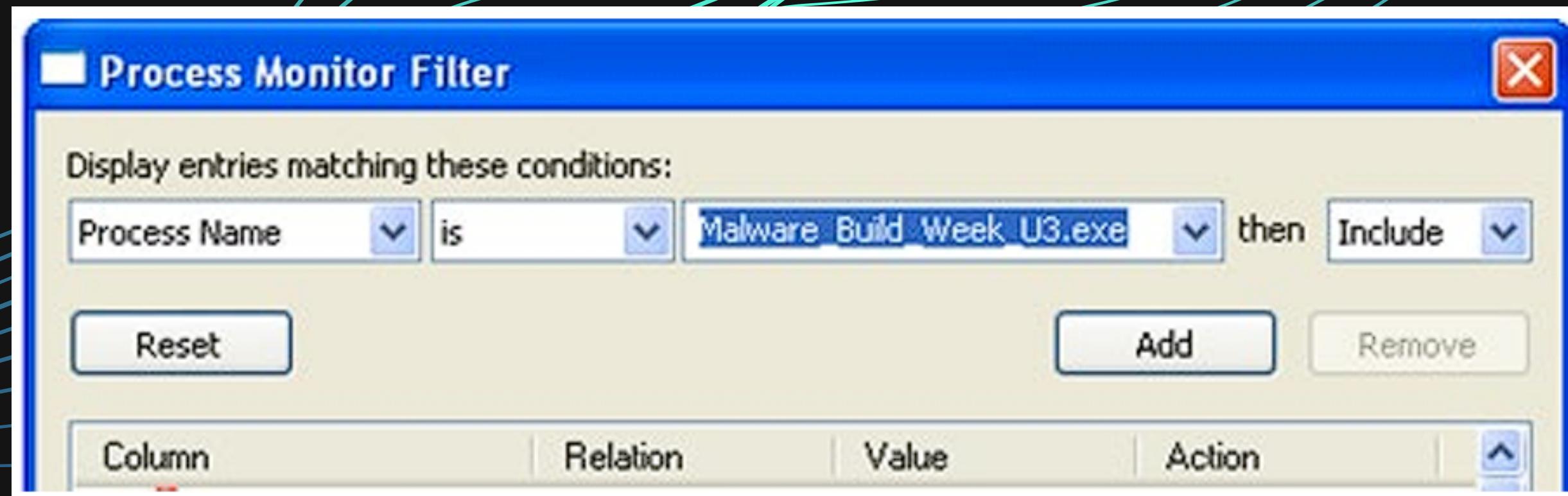
Preparate l'ambiente ed i tool per l'esecuzione del Malware (suggerimento: avviate principalmente Process Monitor ed assicurate di eliminare ogni filtro cliccando sul tasto «reset» quando richiesto in fase di avvio). Eseguite il Malware, facendo doppio click sull'icona dell'eseguibile



# TRACCIA

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.



# TRACCIA

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.

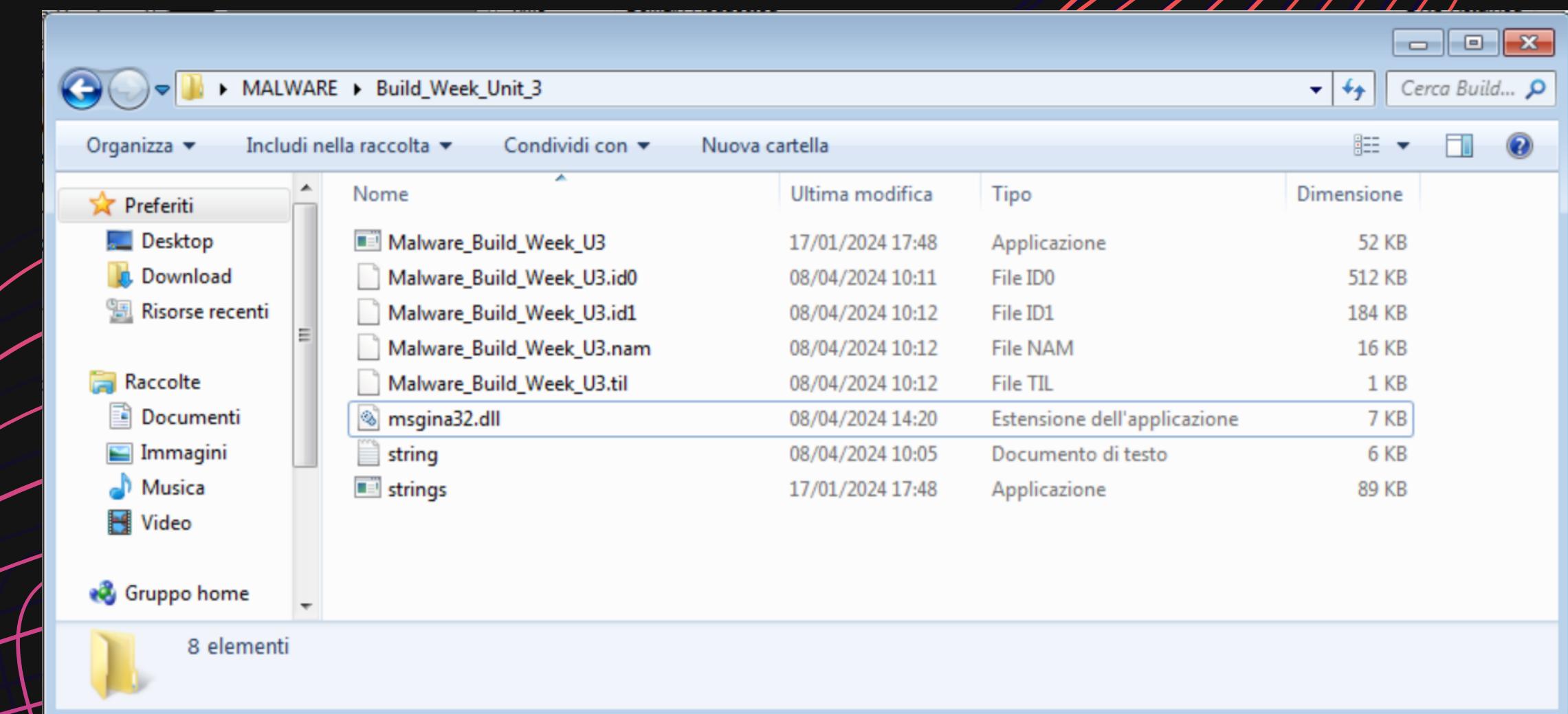
Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?

Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

# CONTENUTO CARTELLA

All'interno della cartella del malware possiamo notare la creazione del file “**msgina32.dll**”, il file che abbiamo trovato nelle stringhe.

Date le precedenti analisi deduciamo che questo sia il file che viene immesso nel computer dal malware che stiamo analizzando, possiamo presupporre che questo file sia la versione “corrotta” del processo che all'avvio di windows permette il log-in dell'utente



# CREAZIONE CHIAVI DI REGISTRO

14:41:...	Malware_Build_Week_U3.exe	1516	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All...
14:41:...	Malware_Build_Week_U3.exe	1516	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	KeySetInformation...
14:41:...	Malware_Build_Week_U3.exe	1516	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Query: Handle Tag...
14:41:...	Malware_Build_Week_U3.exe	1516	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	ACCESS DENIED	Type: REG_SZ, Le...
14:41:...	Malware_Build_Week_U3.exe	1516	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	

Possiamo vedere il processo di creazione della chiave di registro, azione con la quale il virus prende persistenza all'interno del computer.

La persistenza è la capacità del malware di insinuarsi nel sistema operativo e avviarsi all'avvio del computer.

## PROCESSO DI CREAZIONE DEL FILE

Malware_Build_...	1032	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3
Malware_Build_...	1032	CreateFile	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	1032	QueryBasicInfor...	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	1032	CloseFile	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	1032	CreateFile	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	1032	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	1032	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	1032	CloseFile	C:\Windows\SysWOW64\sechost.dll
Malware_Build_...	1032	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_...	1032	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_...	1032	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_...	1032	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll

Osserviamo la creazione del file "msgina32.dll" nella cartella del malware.

# INDICE

## GIORNO 5:

- TRACCIA pag.44
- PROFILO E FUNZIONALITÀ pag.45
- DIAGRAMMA DI FLUSSO pag.46

## TRACCIA

GINA (Graphical identification and authentication) è un componente legale 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 a destra che usate anche voi per accedere alla macchina virtuale.

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

Sulla base della risposta sopra, delineate il profilo del Malware e delle sue funzionalità. Unite tutti i punti per creare un grafico che ne rappresenti lo scopo ad alto livello.



# PROFILO E FUNZIONALITÀ

Il profilo del malware che sostituisce il file .dll legittimo potrebbe includere le seguenti funzionalità:

1. **Interception:** Il malware è in grado di intercettare e registrare le informazioni inserite dagli utenti durante il processo di autenticazione.
2. **Persistenza:** Il malware è in grado di mantenere la sua presenza nel sistema, garantendo che rimanga attivo anche dopo il riavvio del computer.
3. **Criptaggio delle Informazioni Rubate:** Per garantire la sicurezza delle informazioni rubate, il malware potrebbe criptare i dati prima di trasmetterli al suo server di comando e controllo.
4. **Stealth:** Il malware potrebbe tentare di eludere la rilevazione da parte degli antivirus e delle soluzioni di sicurezza, adottando tecniche di occultamento e mimetizzazione.
5. **Remote Command Execution:** Il malware potrebbe essere in grado di eseguire comandi remoti inviati dal suo server di comando e controllo, consentendo agli attaccanti di prendere il controllo del sistema infetto.

## ESECUZIONE DEL MALWARE

Gina.dll “malevolo” già  
presente all'interno della  
macchina virtuale

Autenticazione dell'utente  
(username e password) nella  
pagina di autenticazione  
alterata

Credenziali rubate e ipotizziamo  
salvate o inviate all'attaccante

Creazione di  
Gina.dll  
malevolo

Creazione e successiva  
modifica della chiave di  
registro winlogon

**GRAZIE**