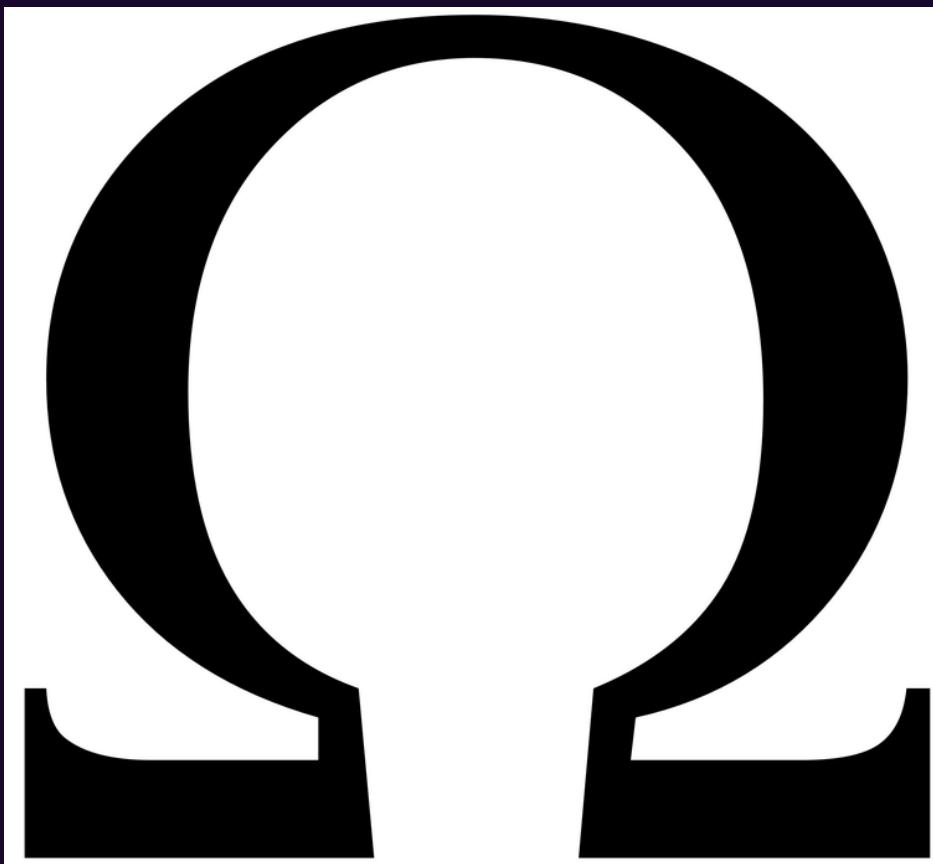




# BUILDING WEEK

## TEAM OMEGA



Omar Fougani  
Leonardo di Federico  
Leonardo Guglielminini  
Drago Picari  
Marta Aramu  
Saverio Giangiuli





---

MalwareAnalysis Il Malware da analizzare è nella cartella Build\_Week\_Unit\_3 presente sul desktop della macchina virtuale dedicata.

Giorno 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 all'interno della funzione Main()?
- Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate
- Quali librerie importa il Malware? Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.



---

# GIORNO 1

---



## PARAMETRI - VARIABILI

Nella funzione main sono passati 3 parametri, in quanto averti offset positivo e 5 variabili in quanto averti offset negativo.

```
; 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_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
```



# SEZIONI

- **.text**: contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato.
- **.rdata**: include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile.
- **.data**: contiene tipicamente i dati / le 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.

Nome	Offset	Size
<b>.text</b>	<b>00005646</b>	<b>00001000</b>
<b>.rdata</b>	<b>0000009AE</b>	<b>00007000</b>
<b>.data</b>	<b>00003EA8</b>	<b>00008000</b>
<b>.rsrc</b>	<b>00001A70</b>	<b>0000C000</b>



# LIBRERIE

- Kernel32.dll: contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria.

- Advapi32.dll: contiene le funzioni per interagire con i servizi ed i registri del sistema operativo

Malware_Build_Week_U3.exe						
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

	Hint	Name			
	Word	szAnsi	Dword	Word	szAnsi
			00007632	0295	SizeofResource
			00007644	01D5	LockResource
			00007654	01C7	LoadResource
C	0186	RegSetValueExA	00007622	02BB	VirtualAlloc
E	015F	RegCreateKeyExA	00007674	0124	GetModuleFileNameA
			0000768A	0126	GetModuleHandleA

# FUNZIONI RICHIAMATE

Alcune operazioni coinvolgono la libreria di sistema di Windows, Advapi32.dll. In particolare, sembra interagire con GetModuleHandleA e GetModuleFileNameA, che sono funzioni utilizzate per ottenere il handle del modulo corrente e il percorso del file eseguibile, rispettivamente.

La funzione RegSetValueExA è una funzione dell'API di Windows utilizzata per impostare il valore di una chiave di registro specifica o creare una nuova chiave di registro e impostare il suo valore. Se un malware interagisce con RegSetValueExA, potrebbe essere coinvolto in attività malevole come la modifica delle impostazioni di sistema o la persistenza nel sistema.

Manipola inoltre la stringa contenente il percorso del file eseguibile. Viene cercata la stringa "\msgina32.dll" nel percorso del modulo corrente. I risultati della ricerca vengono elaborati e copiati in una nuova stringa.

**ADVAPI32.dll**

**KERNEL32.dll**

**GetModuleFileNameA**

**GetModuleHandleA**

**RegSetValueExA**

**"\msgina32.dll"**

# FUNZIONI RICHIAMATE

---

## FreeResource:

Una funzione utilizzata per liberare le risorse di sistema o di memoria che sono state precedentemente allocate da un'applicazione. Questo può essere parte di una gestione efficiente delle risorse per prevenire perdite di memoria.

**FreeResource**  
**FindResourceA**

**KERNEL32**  
**KERNEL32**

## FindResource:

Associato alle API di Windows e può essere utilizzato per individuare e ottenere un handle a una risorsa all'interno di un file eseguibile o di una libreria dinamica (DLL). Questo può includere icone, cursori, bitmap, stringhe e altri dati incorporati nell'eseguibile.

---



L'interazione con la chiave di registro HKEY\_LOCAL\_MACHINE\SYSTEM è particolarmente significativa. Modificare o manipolare questa chiave può avere gravi conseguenze per il funzionamento del sistema.

Process Hacker [user-PC\user]

Hacker View Tools Users Help

Refresh Options Find handles or DLLs System

Processes Services Network Disk

Name PID CPU I/O total

Name	PID	CPU	I/O total
svchost.exe	888		
svchost.exe	324		
svchost.exe	928		
spoolsv.exe	1196		
taskhost.exe	1224		
svchost.exe	1284		
svchost.exe	1412	0,03	88
sppsvc.exe	1984		
SearchIndexer.exe	1732		
wmpnetwk.exe	1120		
svchost.exe	2272		
svchost.exe	1940		
lsass.exe	476		
lsm.exe	484		
csrss.exe	376	0,04	
conhost.exe	2736		
winlogon.exe	404		
explorer.exe	1804	1,56	
VBoxTray.exe	1824	0,02	56
CFF Explorer.exe	2608		
OLLYDBG.EXE	720	0,13	
Malware_Build_Week...	1132		
Process Hacker.exe	1848	1,20	

Proprietà - Malware\_Build\_Week\_U3.exe (1132)

General Statistics Performance Threads Token

Modules Memory Environment Handles Comment

Hide unnamed handles

Type	Name	Handle
Directory	\KnownDlls	0x8
Directory	\KnownDlls32	0xc
Directory	\KnownDlls32	0x18
File	C:\Windows	0x10
File	C:\Users\user\Desktop\MALWARE\B...	0x1c
Key	HKLM\SOFTWARE\MICROSOFT\WIN...	0x4
Key	HKLM\SOFTWARE\MICROSOFT\WIN...	0x14
Key	HKLM\SYSTEM\ControlSet001\Contr...	0x24
Key	HKLM	0x3c

File C:\Users\user\Desktop\MALWARE\B... 0x1c

Key HKLM\SOFTWARE\MICROSOFT\WIN... 0x4

Key HKLM\SOFTWARE\MICROSOFT\WIN... 0x14

Key HKLM\SYSTEM\ControlSet001\Contr... 0x24

Key HKLM 0x3c



## Modifica chiavi di registro in HKEY\_LOCAL\_MACHINE\SYSTEM

---

### **Persistenza del malware**

HKEY\_LOCAL\_MACHINE\SYSTEM può essere utilizzato per garantire la persistenza del malware nel sistema, assicurandosi che venga eseguito automaticamente all'avvio

### **Nascondere tracce**

Modificare chiavi di registro in questa area può essere utilizzato per nascondere tracce del malware, rendendo più difficile la sua rilevazione.

---



# Principali manipolazioni delle stringhe

## Principali manipolazioni delle stringhe nel codice:

Viene chiamata la funzione GetModuleFileNameA per ottenere il percorso completo del file eseguibile. Il risultato viene salvato nella variabile Data.

Viene cercata l'ultima occorrenza del carattere '\' nella stringa Data utilizzando la funzione \_strrchr. La posizione di questa occorrenza viene salvata nella variabile var\_8.

Viene impostato il byte successivo alla posizione trovata da var\_8 a 0 (mov byte ptr [eax], 0). Questa operazione termina la stringa a quella posizione, troncando il percorso del file.

Viene caricata la stringa "\msgina32.dll" in edi.

Viene eseguita una ricerca nella stringa Data per la posizione di "\msgina32.dll". Una volta trovata la posizione, vengono copiati i byte successivi nella stringa, sovrascrivendo "\msgina32.dll" con il resto del percorso del file eseguibile

```
call  ds:GetModuleFileNameA
push  5Ch          ; Ch
lea   edx, [ebp+Data]
push  edx          ; Str
call  _strrchr
add   esp, 8
mov   [ebp+var_8], eax
mov   eax, [ebp+var_8]
mov   byte ptr [eax], 0
mov   edi, offset aMsgina32_dll ; "\\msgina32.dll"
lea   edx, [ebp+Data]
or    ecx, 0xFFFFFFFF
xor   eax, eax
repne scasb
```



---

# GIORNO 2

---





00401021 -> call ds:RegCreateKeyExA

00401017 -> push offset SubKey "Software\Microsoft\Windows NT\CurrentVersion\..."

00401027 -> test eax, eax

00401029 -> jz short loc\_401032

00401047 -> call ds:RegSetValueExA

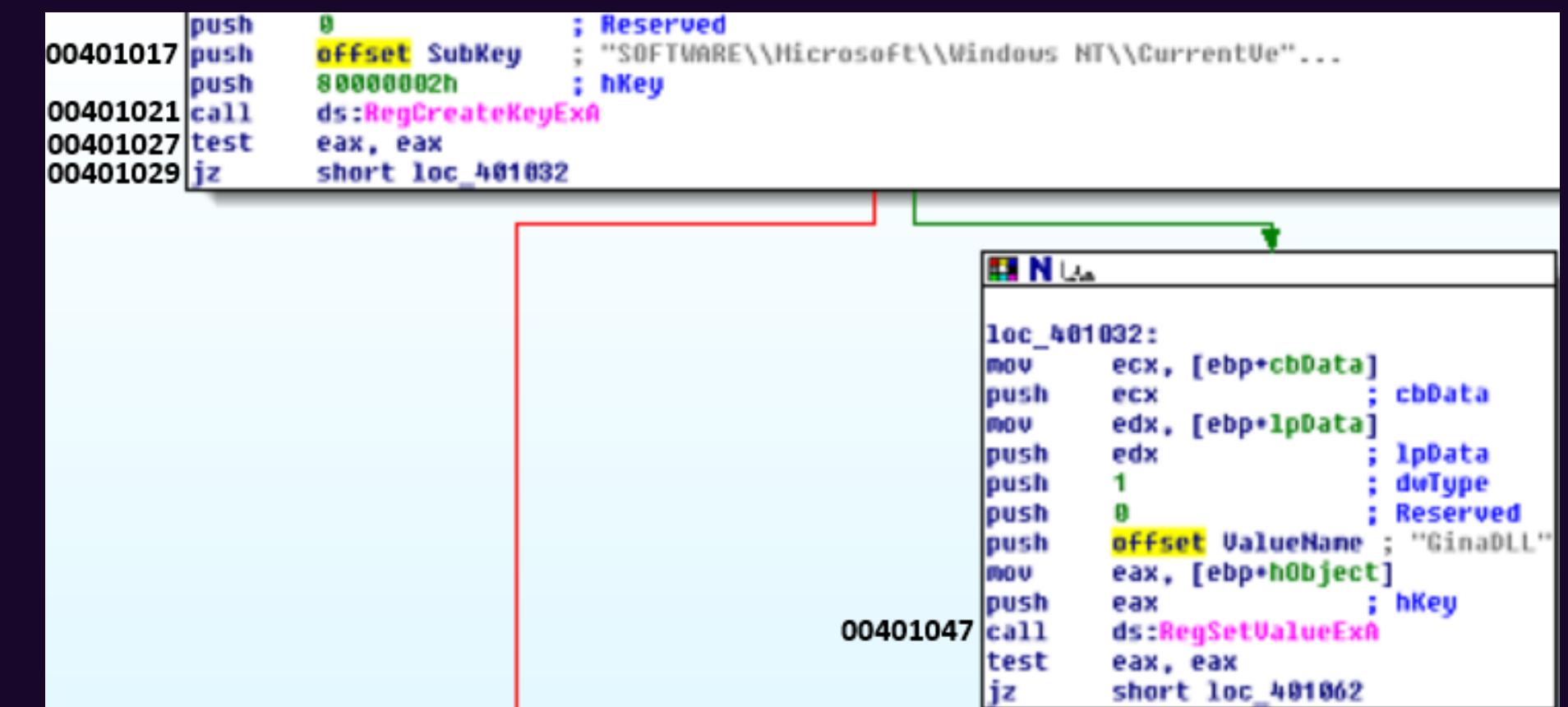
```
00401017 push 0 ; Reserved
00401017 push offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\...
00401021 push 80000002h ; hKey
00401021 call ds:RegCreateKeyExA
00401027 test eax, eax
00401029 jz short loc_401032
```

```
loc_401032:
00401047 mov ecx, [ebp+cbData]
00401047 push ecx ; cbData
00401047 mov edx, [ebp+lpData]
00401047 push edx ; lpData
00401047 push 1 ; dwType
00401047 push 0 ; Reserved
00401047 push offset ValueName ; "GinaDLL"
00401047 mov eax, [ebp+hObject]
00401047 push eax ; hKey
00401047 call ds:RegSetValueExA
00401047 test eax, eax
00401047 jz short loc_401062
```

# Scopo della funzione chiamata alla locazione di memoria 00401021

La funzione RegCreateKeyExA è una funzione dell'API di Windows che viene utilizzata per creare una nuova chiave nel registro di sistema di Windows o per aprire una chiave esistente.

Quando viene chiamata, questa funzione restituisce un handle alla chiave che è stata aperta o creata. I parametri che vengono passati alla funzione determinano l'azione esatta che verrà eseguita



```
00401017 push 0 ; Reserved
00401018 push offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion"
00401019 push 80000002h ; hKey
00401020 call ds:RegCreateKeyExA
00401021 test eax, eax
00401022 jz short loc_401032

loc_401032:
00401047 mov    ecx, [ebp+cbData]
00401048 push   ecx ; cbData
00401049 mov    edx, [ebp+lpData]
0040104a push   edx ; lpData
0040104b push   1 ; dwType
0040104c push   0 ; Reserved
0040104d push   offset ValueName ; "GinaDLL"
0040104e mov    eax, [ebp+hObject]
0040104f push   eax ; hKey
00401050 call   ds:RegSetValueExA
00401051 test   eax, eax
00401052 jz    short loc_401062
```



# Come vengono passati i parametri alla funzione alla locazione 00401021?

Nel codice assembly mostrato nell'immagine, i parametri vengono passati alla funzione RegCreateKeyExA utilizzando il meccanismo dello stack.

Ecco come vengono passati i parametri in questo caso:

- push 0 - viene utilizzata per inserire il valore zero nello stack. Inizializzare i registri o variabili prima di utilizzarle in operazioni successive
- push offset SubKey - Questo comando inserisce nello stack l'indirizzo della stringa che rappresenta il nome della sottocchiave da aprire o creare nel registro di sistema.
- push 80000002h - Questo valore esadecimale è probabilmente l'handle della chiave radice, che in questo caso sembra essere HKEY\_LOCAL\_MACHINE (dato che il valore 0x80000002 è l'handle standard per HKEY\_LOCAL\_MACHINE).
- call ds:RegCreateKeyExA - Questa istruzione è il punto in cui la funzione viene effettivamente chiamata. L'indirizzo della funzione RegCreateKeyExA è memorizzato in un registro o in un punto nella memoria indicato da ds (il segmento dei dati).

```
00401017 push 0 ; Reserved
00401018 push offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\GinaDLL"
00401019 push 80000002h ; hKey
0040101A call ds:RegCreateKeyExA
0040101B test eax, eax
0040101C jz short loc_401032

loc_401032:
00401047 mov ecx, [ebp+cbData]
00401048 push ecx ; cbData
00401049 mov edx, [ebp+lpData]
0040104A push edx ; lpData
0040104B mov eax, 1 ; dwType
0040104C push eax ; dwType
0040104D mov ebx, 0 ; Reserved
0040104E push ebx ; Reserved
0040104F push offset ValueName ; "GinaDLL"
00401050 mov eax, [ebp+hObject]
00401051 push eax ; hKey
00401052 call ds:RegSetValueExA
00401053 test eax, eax
00401054 jz short loc_401062
```

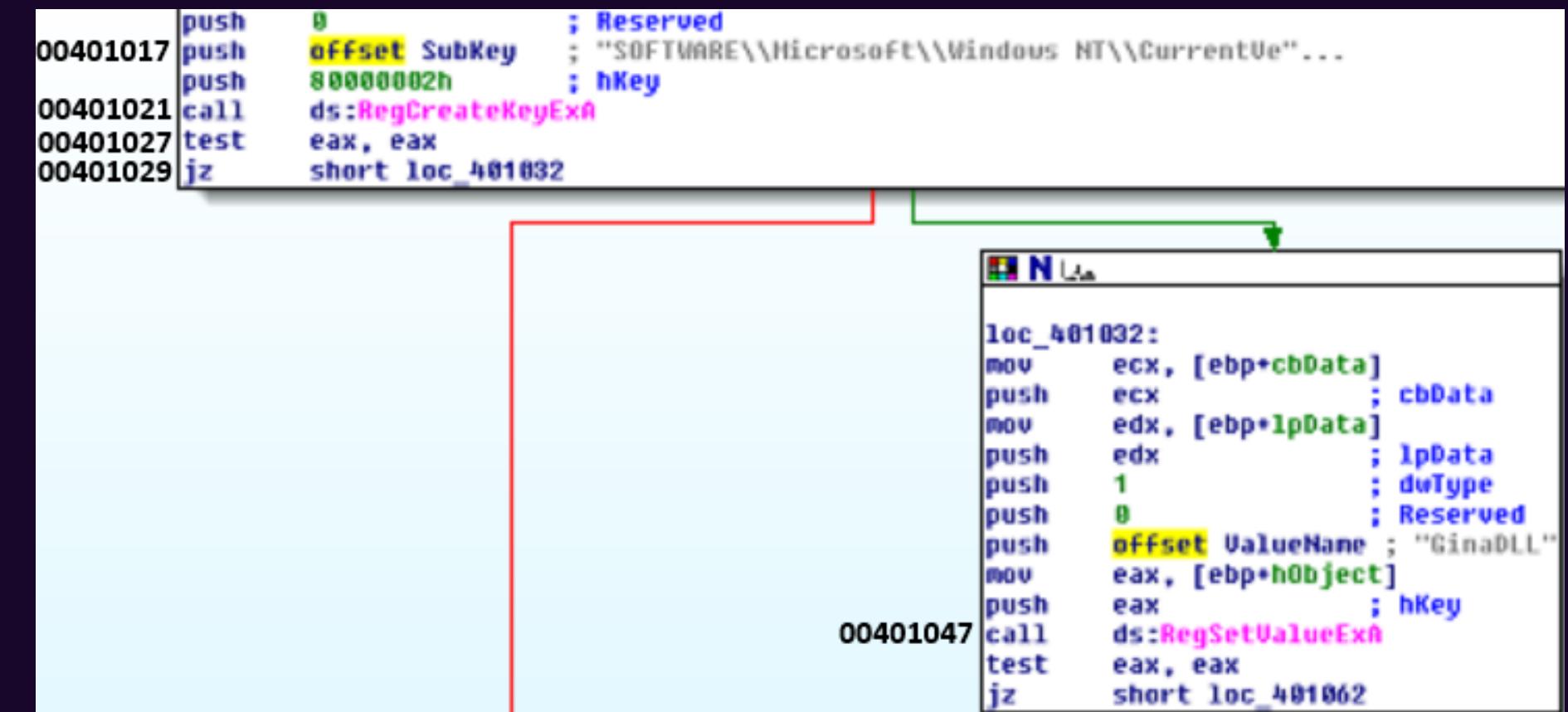
Le istruzioni push sopra elencate posizionano i parametri in ordine inverso nello stack perché, nell'architettura x86, lo stack cresce verso il basso.



## Oggetto rappresentato dal parametro alla locazione 00401017

All'indirizzo 00401017 nel codice assembly, vediamo l'istruzione push offset SubKey. Questo parametro è un puntatore (o offset) che rappresenta l'indirizzo di memoria dove è situata la stringa per lpSubKey. La stringa a cui si riferisce è il nome della sottocchiave che la funzione RegCreateKeyExA cercherà di aprire o creare nel registro di sistema.

Quindi, il parametro alla locazione 00401017 rappresenta il nome della sottocchiave di registro sotto forma di indirizzo di memoria della stringa che identifica quella sottocchiave all'interno del registro di sistema di Windows.



```
00401017 push 0 ; Reserved
00401017 push offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\GinaDLL"
00401017 push 80000002h ; hKey
00401021 call ds:RegCreateKeyExA
00401027 test eax, eax
00401029 jz short loc_401032

loc_401032:
00401047 mov ecx, [ebp+cbData]
00401047 push ecx ; cbData
00401047 mov edx, [ebp+lpData]
00401047 push edx ; lpData
00401047 push 1 ; dwType
00401047 push 0 ; Reserved
00401047 push offset ValueName ; "GinaDLL"
00401047 mov eax, [ebp+hObject]
00401047 push eax ; hKey
00401047 call ds:RegSetValueExA
00401047 test eax, eax
00401047 jz short loc_401062
```



# Significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029

Tra gli indirizzi di memoria 00401027 e 00401029, ci sono due istruzioni assembly:

```
test eax, eax  
jz short loc_401032
```

Ecco il loro significato:

- **test eax, eax:** Questa istruzione esegue un'operazione AND bitwise tra il registro eax e se stesso. L'operazione AND non modifica il valore di eax, ma imposta i flag di stato in base al risultato dell'operazione. In particolare, il flag zero (ZF) viene impostato se il risultato è zero, cioè se eax contiene zero. Questo è spesso utilizzato per verificare se il risultato di una precedente operazione (come una chiamata di funzione il cui risultato viene spesso memorizzato in eax) è zero o non zero.
- **jz short loc\_401032:** Questa istruzione è un'istruzione di salto condizionato che dipende dallo stato del flag zero (ZF). Il programma salterà all'indirizzo etichettato come loc\_401032 se il flag zero è impostato. In altre parole, se eax è zero (che potrebbe indicare, ad esempio, il successo di una funzione appena chiamata se la convenzione è che zero indica successo), allora esegue il salto a loc\_401032. Se eax non è zero, il flusso del programma continua linearmente senza saltare.

```
00401017 push 0 ; Reserved  
00401017 push offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Gina\GinaDLL"  
00401017 push 80000002h ; hKey  
00401021 call ds:RegCreateKeyExA  
00401027 test eax, eax  
00401029 jz short loc_401032
```

```
loc_401032:  
00401047 mov ecx, [ebp+cbData]  
00401047 push ecx ; cbData  
00401047 mov edx, [ebp+lpData]  
00401047 push edx ; lpData  
00401047 push 1 ; dwType  
00401047 push 0 ; Reserved  
00401047 push offset ValueName ; "Gina\GinaDLL"  
00401047 mov eax, [ebp+hObject]  
00401047 push eax ; hKey  
00401047 call ds:RegSetValueExA  
00401047 test eax, eax  
00401047 jz short loc_401062
```



# Traduzione del codice Assembly in C

Codice assembly indici di memoria  
00401027 test eax, eax  
00401029 jz short loc\_401032

test eax, eax  
jz short loc\_401032

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  RegCreateKeyExA(key, subkey)
5  {
6
7  }
8
9  RegSetValueExA(key)
10 {
11
12 }
13
14 int main()
15 {
16     int eax;
17     int hKey = 80000002h;
18     char subKey[] = "Software\Microsoft\WindowsNT\CurrentVersion";
19
20     RegCreateKeyExA(hkey, subKey);
21
22     if(eax == 0)
23     {
24         goto loc_401032;
25     }
26
27     //
28     //
29
30     loc_401032:
31     hKey= eax;
32     RegSetValueExA(hkey);
33
34 }
```

## Valore del parametro «ValueName»

Il valore del parametro ValueName risulta “GinaDLL”



```
00401017 push 0 ; Reserved
push offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\GinaDLL"
push 80000002h ; hKey
call ds:RegCreateKeyExA
test eax, eax
jz short loc_401032

00401047 loc_401032:
mov ecx, [ebp+cbData]
push ecx ; cbData
mov edx, [ebp+lpData]
push edx
push 1 ; dwType
push 0 ; Reserved
push offset ValueName ; "GinaDLL"
mov eax, [ebp+hObject]
push eax ; hKey
call ds:RegSetValueExA
test eax, eax
jz short loc_401062

push 1 ; dwType
push 0 ; Reserved
push offset ValueName ; "GinaDLL"
mov eax, [ebp+hObject]
push eax ; hKey
call ds:RegSetValueExA
```



# Funzionalità implementate dal Malware in questa sezione

Il codice fornito manipola la stringa contenente il percorso del file eseguibile tramite diverse istruzioni di assembly. Ecco dove avvengono le principali manipolazioni delle stringhe nel codice:

Viene chiamata la funzione GetModuleFileNameA per ottenere il percorso completo del file eseguibile. Il risultato viene salvato nella variabile Data.

Viene cercata l'ultima occorrenza del carattere '\' nella stringa Data utilizzando la funzione \_strrchr. La posizione di questa occorrenza viene salvata nella variabile var\_8.

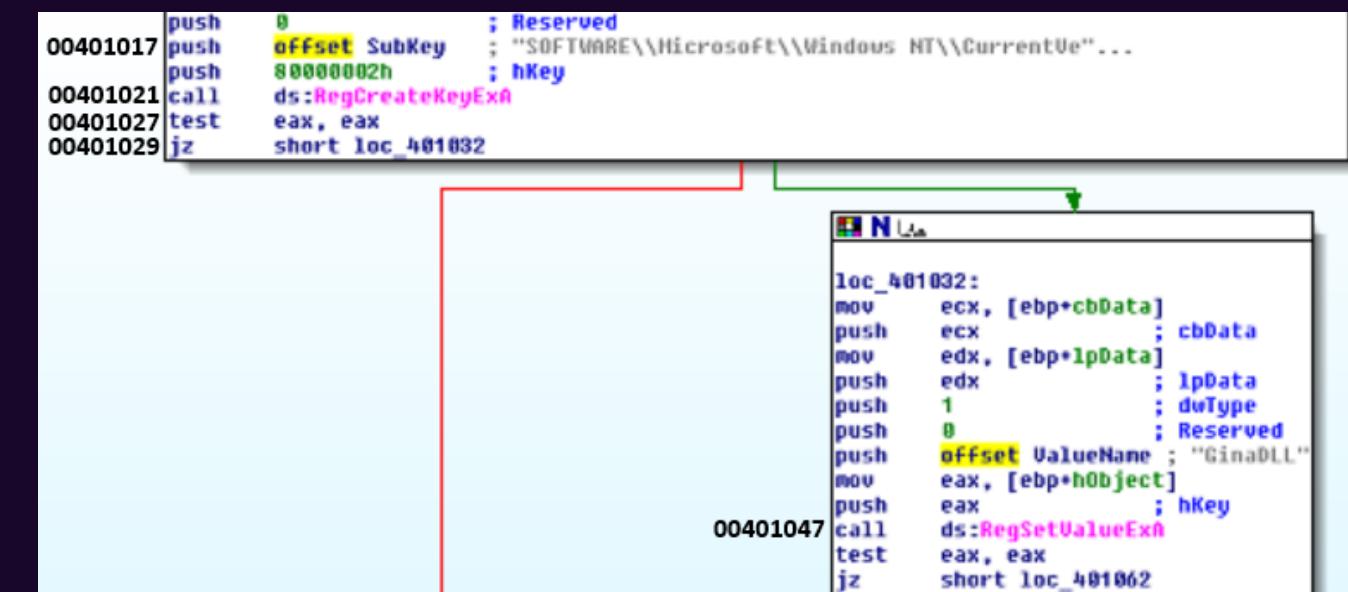
Viene impostato il byte successivo alla posizione trovata da var\_8 a 0 (mov byte ptr [eax], 0). Questa operazione termina la stringa a quella posizione, troncando il percorso del file.

Viene caricata la stringa "\msgina32.dll" in edi.

Viene eseguita una ricerca nella stringa Data per la posizione di "\msgina32.dll". Una volta trovata la posizione, vengono copiati i byte successivi nella stringa, sovrascrivendo "\msgina32.dll" con il resto del percorso del file eseguibile.

Manipola inoltre la stringa contenente il percorso del file eseguibile

Viene cercata la stringa "\msgina32.dll" nel percorso del modulo corrente. I risultati della ricerca vengono elaborati e copiati in una nuova stringa.



```
00401017 push 0 ; Reserved
00401017 push offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion"
00401017 push 8000002h ; hKey
00401021 call ds:RegCreateKeyExA
00401027 test eax, eax
00401029 jz short loc_401032

loc_401032:
00401032 mov ecx, [ebp+cbData]
00401032 push ecx ; cbData
00401032 mov edx, [ebp+lpData]
00401032 push edx ; lpData
00401032 push 1 ; dwType
00401032 push 0 ; Reserved
00401032 push offset ValueName ; "GinaDLL"
00401032 mov eax, [ebp+hObject]
00401032 push eax ; hKey
00401032 call ds:RegSetValueExA
00401032 test eax, eax
00401032 jz short loc_401062
```



```
00401047 push 1 ; dwType
00401047 push 0 ; Reserved
00401047 push offset ValueName ; "GinaDLL"
00401047 mov eax, [ebp+hObject]
00401047 push eax ; hKey
00401047 call ds:RegSetValueExA
```



---

# GIORNO 3

---



# Analisi avanzata del malware tramite IDA e OllyDBG

Per poter analizzare più approfonditamente il malware possiamo utilizzare sia il software IDA che OllyDBG.

Prendiamo in esame il codice tra gli indirizzi di memoria 00401080 e 00401128.

- Qual è il valore del parametro «ResourceName» passato alla funzione FindResourceA()?

```
004010BD: . 50      PUSH EAX
004010BE: . 8B0D 34804000 MOV ECX, DWORD PTR DS:[408034]
004010C4: . 51      PUSH ECX
004010C5: . 8B55 08  MOV EDX, DWORD PTR SS:[EBP+8]
004010C8: . 52      PUSH EDX
004010C9: . FF15 28704000 CALL DWORD PTR DS:[<&KERNEL32.FindResou
```

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

Il valore del parametro ResourceName passato alla funzione FindResourceA(), analizzando il malware tramite OllyDBG, risulta essere "TGAD".

Per poter passare il valore di ResourceName alla funzione viene utilizzata l'istruzione PUSH del registro ECX.

```
00401080: ; Attributes: bp-based Frame
; int __cdecl sub_401080(HMODULE hModule)
sub_401080 proc near
hResData= dword ptr -18h
hResInfo= dword ptr -14h
Count= dword ptr -10h
var_C= dword ptr -8Ch
Str= dword ptr -8
File= dword ptr -4
hModule= dword ptr 8
push  ebp
mov   ebp, esp
sub   esp, 18h
push  esi
push  edi
mov   [ebp+hResInfo], 0
mov   [ebp+hResData], 0
mov   [ebp+Str], 0
mov   [ebp+Count], 0
mov   [ebp+var_C], 0
cmp   [ebp+hModule], 0
jnz   short loc_401088
```

```
loc_401088:
mov   eax, lpType
push  eax          ; lpType
mov   ecx, lpName
push  ecx          ; lpName
mov   edx, [ebp+hModule]
push  edx          ; hModule
call  ds:FindResourceA
mov   [ebp+hResInfo], eax
cmp   [ebp+hResInfo], 0
inz   short loc_4010DF
```

```
loc_4010DF:
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
```

# Analisi avanzata del malware tramite IDA e OLLYDBG

- 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?

Possiamo notare inoltre il susseguirsi di 4 funzioni diverse:

- FindResource()
- LoadResource()
- LockResource()
- SizeofResource()

Partendo dai presupposti dei giorni precedenti e con le informazioni che abbiamo ora possiamo ipotizzare che il malware stia agendo come dropper, ovvero che sia in grado di installare un secondo file malevolo, probabilmente una versione alterata della libreria Gina.dll, durante la sua esecuzione.

Precisamente il malware cerca all'interno della sezione TGAD il file da installare chiamato GINA.dll, lo carica nel sistema per poi poterlo usare per recuperare le credenziali dell'utente durante l'autenticazione.

```
004010BE . 8B0D 34804000 MOV ECX,DWORD PTR DS:[408034]
004010C4 . 51 PUSH ECX
004010C5 . 8B55 08 MOV EDX,DWORD PTR SS:[EBP+8]
004010C8 . 52 PUSH EDX
004010C9 . FF15 28704000 CALL DWORD PTR DS:[<&KERNEL32.FindResourceA]
004010CF . 8945 EC MOV 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
004010DF > 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14]
004010E2 . 50 PUSH EAX
004010E3 . 8B4D 08 MOV ECX,DWORD PTR SS:[EBP+8]
004010E6 . 51 PUSH ECX
004010E7 . FF15 14704000 CALL DWORD PTR DS:[<&KERNEL32.LoadResource]
004010ED . 8945 E8 MOV DWORD PTR SS:[EBP-18],EAX
004010F0 . 837D E8 00 CMP DWORD PTR SS:[EBP-18],0
004010F4 . v75 05 JNZ SHORT Malware_.004010FB
004010F6 . vE9 AA000000 JMP Malware_.004011A5
004010FB > 8B55 E8 MOV EDX,DWORD PTR SS:[EBP-18]
004010FE . 52 PUSH EDX
004010FF . FF15 10704000 CALL DWORD PTR DS:[<&KERNEL32.LockResource]
00401105 . 8945 F8 MOV DWORD PTR SS:[EBP-8],EAX
00401108 . 837D F8 00 CMP DWORD PTR SS:[EBP-8],0
0040110C . v75 05 JNZ SHORT Malware_.00401113
0040110E . vE9 92000000 JMP Malware_.004011A5
00401113 > 8B45 EC MOV EAX,DWORD PTR SS:[EBP-14]
00401116 . 50 PUSH EAX
00401117 . 8B4D 08 MOV ECX,DWORD PTR SS:[EBP+8]
0040111A . 51 PUSH ECX
0040111B . FF15 0C704000 CALL DWORD PTR DS:[<&KERNEL32.SizeofResource]
00401121 . 8945 F0 MOV DWORD PTR SS:[EBP-10],EAX
00401124 . 832D F0 00 CMP DWORD PTR SS:[EBP-10],0
```

# Analisi avanzata del malware tramite IDA e OllyDBG

- È possibile identificare questa funzionalità utilizzando l'analisi statica basica?

Si, è possibile identificare il malware come dropper dalla sola analisi statica basica, per esempio utilizzando il software CFF explorer. La sola analisi statica basica può darci indicazioni su come il malware agisce tramite le librerie importate, in questo caso la libreria ADVAPI e la libreria KERNEL32.

- In caso di risposta affermativa, elencare le evidenze a supporto.

Tramite la sola analisi statica basica tramite CFF explorer possiamo individuare le funzioni ritrovate dall'analisi eseguita con OllyDBG. Questo potrebbe essere un chiaro punto di riferimento da cui poter immaginare e delineare il tipo di malware a grandi linee.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA
0000769E	N/A	000074EC	000074F0	000074F4	000074F8
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword
<b>KERNEL32.dll</b>	51	00007534	00000000	00000000	0000769E
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0

OFTs	FTs (IAT)	Hint	Name
00007540	00007018	00007622	00007624
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA

52/72 security vendors and no sandboxes flagged this file as malicious

57d8d248a8741176348b5d12dcf29f34c8f48ede0ca13c30d12e5ba0384056d7

Lab11-01.exe

Community Score: 52 / 72

peexe spreader armadillo checks-user-input

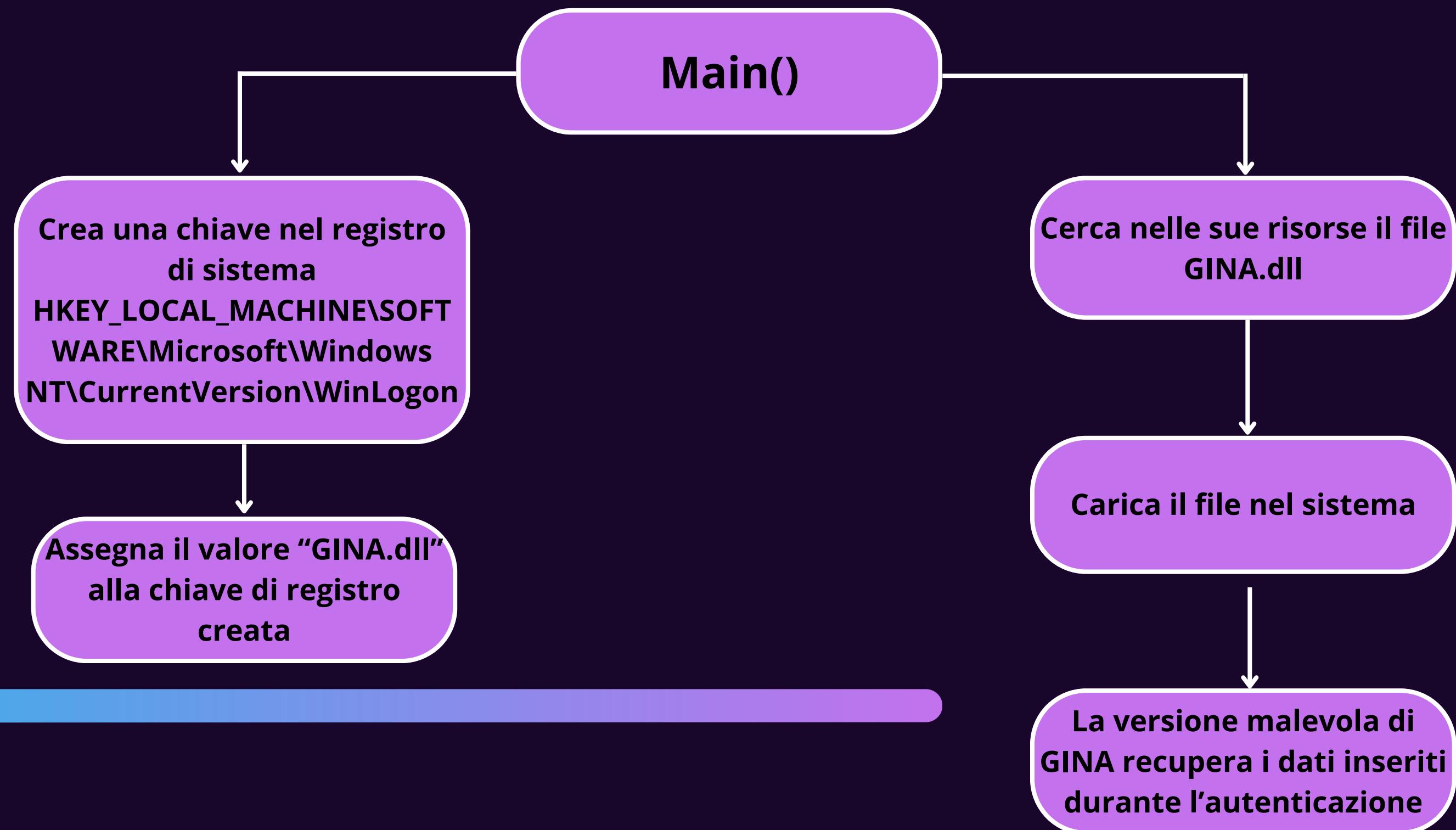
Size: 52.00 KB | Last Modification Date: 15 hours ago | EXE

## Analisi su VirusTotal

Possiamo inoltre controllare tramite i report di VirusTotal che in questo caso ci conferma che 52 provider di sicurezza identificano il file come un malware.

## Analisi avanzata del malware tramite IDA e OLLYDBG

Entrambe le funzionalità principali del Malware viste finora sono richiamate all'interno della funzione Main(). Disegnare un diagramma di flusso che comprenda le 3 funzioni.





# GIORNO 4

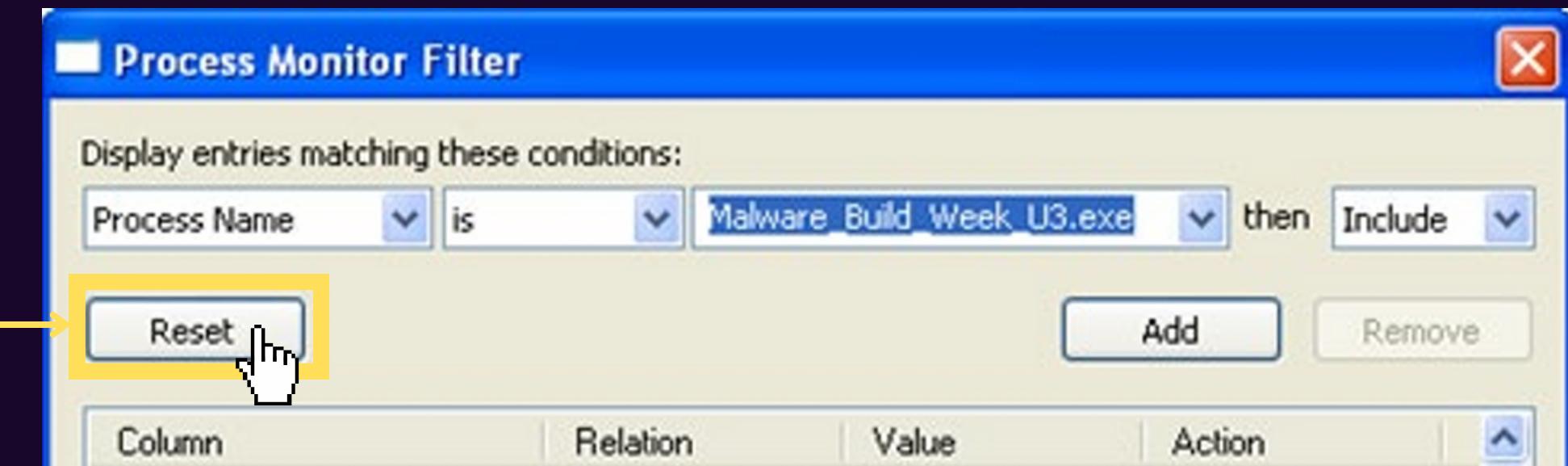




# MalwareAnalysis

"" 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 ""

Come da suggerimento come prima operazione , ci assicuriamo di eliminare ogni filtro presente su **Process Monitor**, cliccando sul pulsante reset.



Eseguiamo il **Malware**  
facendo *doppio click*

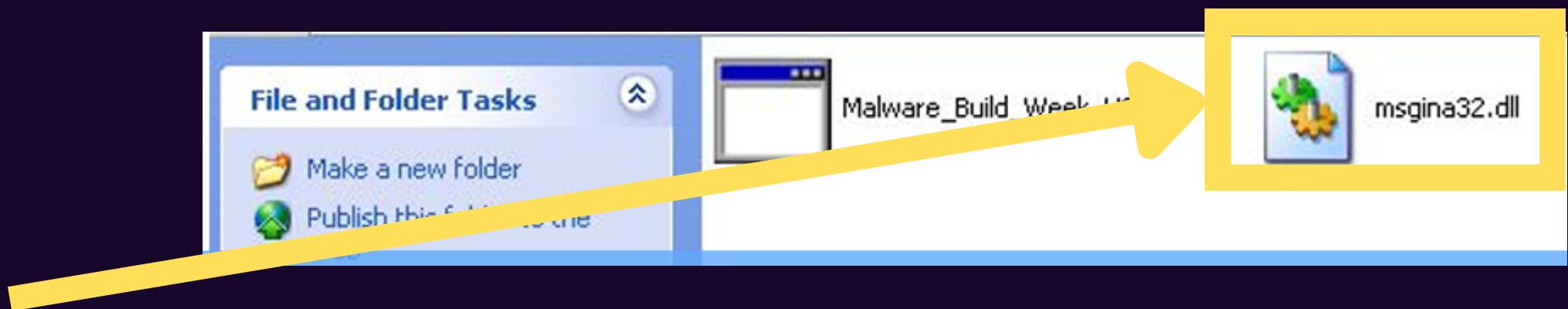




# MalwareAnalysis

-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

Dopo aver fatto doppio click ed avviato il Malware abbiamo subito notato che viene creato un secondo file **msgina32.dll**. Raccolte le informazioni e analizzato il tutto deduciamo che questo tipo di Malware è un **DROPPER**. (Questo perché appena avviato ha tirato fuori il file msgina32.dll ( dalla sezione risorse malware - **TGAD**))





# MalwareAnalysis

Filtrate includendo solamente l'attività sul registro di Windows .

(1) Quale chiave di registro viene creata?

(2) Quale valore viene associato alla chiave di registro creata?

Passate ora alla visualizzazione dell'attività sul File System.

(3) 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

(1) - (2) Dopo aver applicato il filtro su **ProcMon** per concentrarci esclusivamente sugli eventi generati dal malware in esame, esaminiamo attentamente la sezione delle attività relative alle chiavi di registro. Come evidenziato nella figura sottostante, l'analisi dinamica conferma che **il malware genera una nuova chiave di registro e associa il valore "GinaDLL"** a questa chiave appena creata.

8	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
8	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
8	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

82...	Malware_Build_Week_U3...	1016	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
94...	Malware_Build_Week_U3...	1016	FileSystemControl	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
104...	Malware_Build_Week_U3...	1016	QueryOpen	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.e
153...	Malware_Build_Week_U3...	1016	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
161...	Malware_Build_Week_U3...	1016	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
167...	Malware_Build_Week_U3...	1016	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
123...	Malware_Build_Week_U3...	1016	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
180...	Malware_Build_Week_U3...	1016	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
199...	Malware_Build_Week_U3...	1016	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
164...	Malware_Build_Week_U3...	1016	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3

## FUNZIONAMENTO DEL MALWARE

Il malware individua il file **msgina32.dll** nella sezione delle risorse denominata "**TGAD**". Dopo di che, effettua la copia e la creazione del file msgina32.dll nella stessa directory in cui risiede l'eseguibile del malware originale. Questa operazione viene eseguita mediante l'utilizzo di funzioni standard per l'interazione con il file system, quali "**CreateFile()**" e "**WriteFile()**".

Successivamente, il malware procede alla creazione della chiave di registro **HKEY LOCAL MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon**, assegnando il valore "**Gina.dll**" a questa chiave.

# GIORNO 5



## GINA (Graphical identification and authentication)

GINA (Graphical identification and authentication) è un componente legato di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica.

Permette agli utenti di inserire username e password nel classico riquadro Windows, come quello in figura a destra. E' molto usato, per esempio, per accedere alla macchina virtuale.



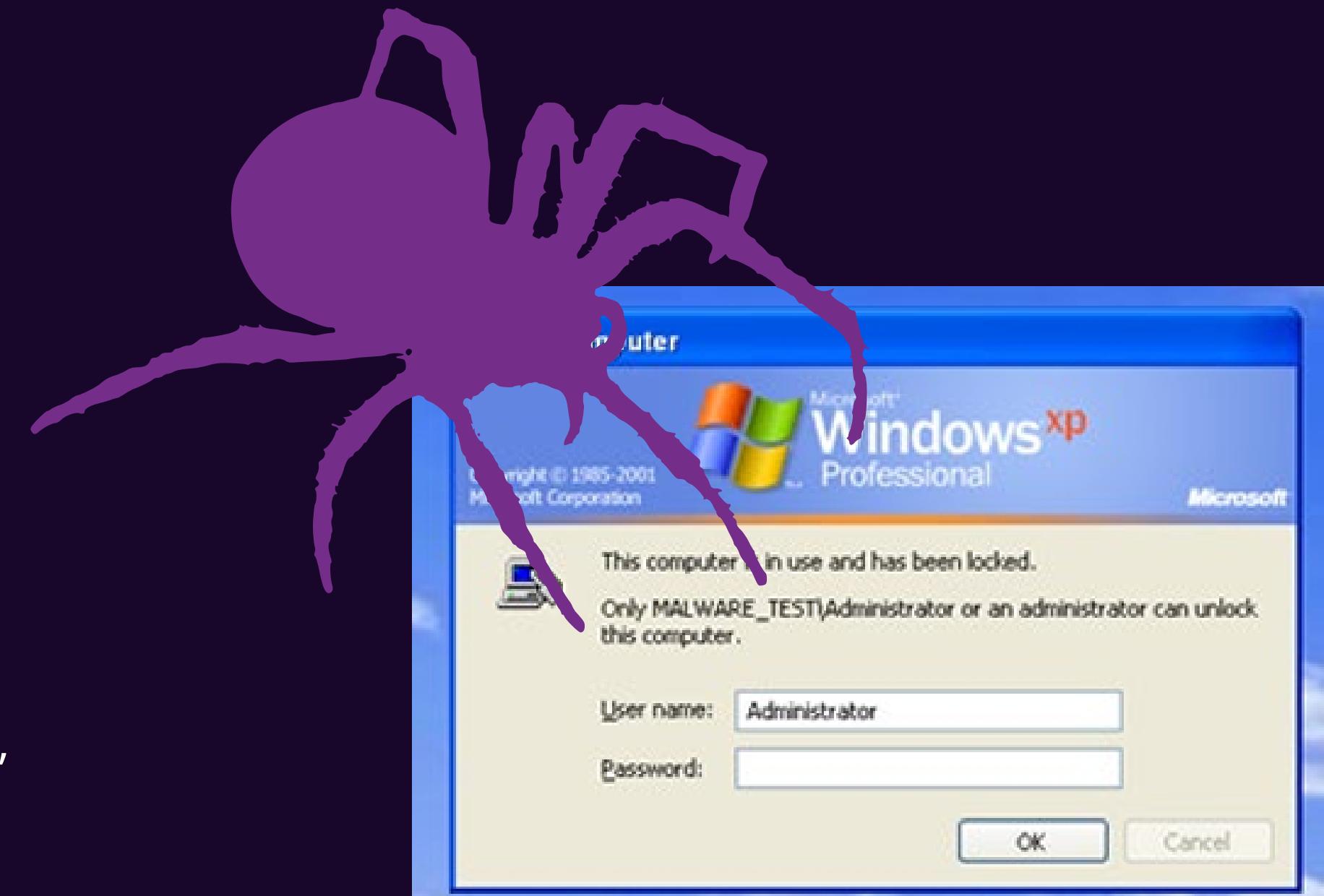


# Sostituzione del file lecito con quello malevolo

Cosa può succedere se il file GINA lecito viene sostituito con un GINA file malevolo?

Questa particolare situazione ci mette di fronte al rischio altissimo che le nostre credenziali vengano usate con intenti malevoli da parte di criminali informatici. Con le nostre credenziali possono accedere alla macchina virtuale e di conseguenza ai file contenenti.

Un particolare tipo di malware, programmato per intercettare tutto ciò che l'utente digita sulla tastiera, è il **Keylogger**. Nel nostro caso può intercettare username e password appunto.





# Keylogger

Le funzionalità di un keylogger sono sfruttate dai criminali informatici per rubare informazioni confidenziali quali ad esempio:

- Password dei sistemi operativi
- Credenziali di amministrazione
- Numeri di conto e informazioni circa carte di credito
- Credenziali di accesso a siti

Su un campione di 100 keylogger, la grande maggioranza può essere divisa in due grosse macro categorie, elencate di seguito:

- I keylogger che utilizzano `GetAsyncKeyState()`;
- I keylogger che utilizzano `SetWindowsHookEx()`;





# Keylogger che utilizza SetWindowsHookEx();

I keylogger che includono tutti quei malware utilizzati per catturare la digitazione utente fanno leva sulla funzione «SetWindowsHookEX».

Questa funzione non fa altro che installare un metodo (una funzione) chiamato «hook» dedicato al monitoraggio degli eventi di una data periferica, come ad esempio la tastiera o il mouse.

Il metodo «hook» verrà allertato ogni qualvolta l'utente digiterà un tasto sulla tastiera e salverà le informazioni su un file di log.

```
push    0          ; dwThreadId
push    eax        ; hmod
push    offset hook_proc ; lpfn
push    WH_KEYBOARD_LL ; idHook
call    SetWindowsHookExA
```



# Profilo grafico del Malware e funzionalità





## programmi/TOOL utilizzati

IDA

Process monitor

OllyDBG

Process hacker

CFF Explorer