



**SECURE**  
SENTINELS

# BUILD WEEK 3

## LAB REPORT

Team Secure Sentinels  
CS0424IT

# Table of contents

**01**

## Day 1

- 1.1 Parametri passati
- 1.2 Variabili dichiarate
- 1.3 Sezioni presenti
- 1.4 Librerie importate
- 1.5 Funzione chiamata
- 1.6 Parametri passati
- 1.7 Oggetto rappresentato
- 1.8 Istruzioni comprese
- 1.9 Traduzione in costrutto C
- 1.10 Valore del parametro
- 1.11 Funzionalità implementata

**02**

## Day 2

- 2.1 Analisi con IDA Pro
- 2.2 Analisi con OllyDBG
- 2.3 Analisi con CFF Explorer
- 2.4 Analisi delle funzioni del Main()
- 2.5 Analisi dinamica basica
- 2.6 Process Monitor
- 2.7 Creazione chiave malevola
- 2.8 Funzione che crea il file
- 2.9 Conclusioni

**03**

## Day 3

- 3.1 Traccia
- 3.2 Overview GINA.dll
- 3.3 Conseguenze
- 3.4 Diagramma e analisi
- 3.5 Conclusioni

# DAY 1

# TRACCIA

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

Con riferimento al file eseguibile Malware\_Build\_Week\_U3, rispondere 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 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.

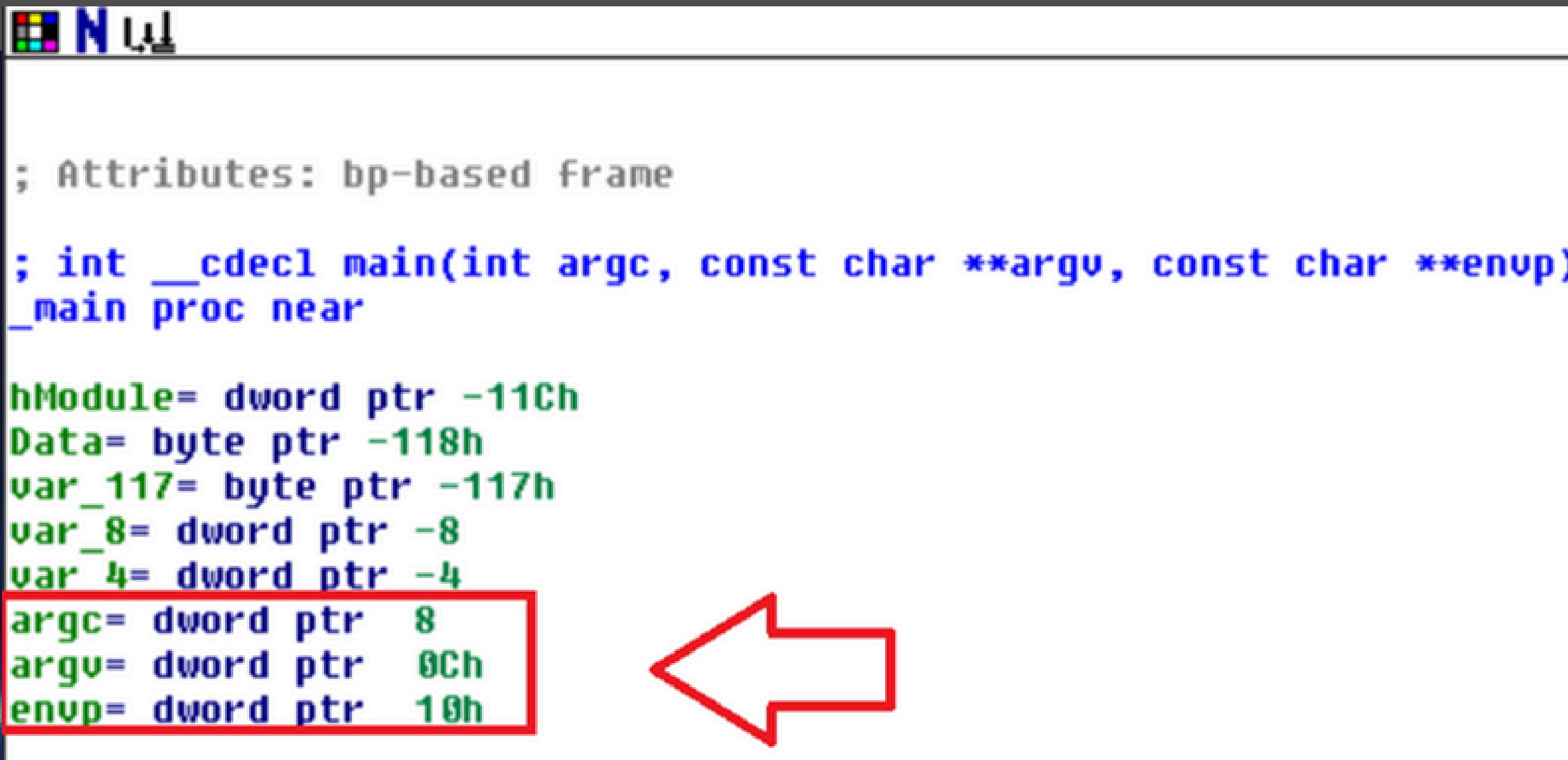
## **Con riferimento al Malware in analisi, spiegare:**

- 5) Lo scopo della funzione chiamata alla locazione di memoria 00401021.
- 6) Come vengono passati i parametri alla funzione alla locazione 00401021.
- 7) Che oggetto rappresenta il parametro alla locazione 00401017.
- 8) Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029 .  
(se serve, valutate anche un'altra o altre due righe assembly ).
- 9) Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C.
- 10) Valutate ora la chiamata alla locazione 00401047 , qual è il valore del parametro « ValueName»?
- 11) Nel complesso delle due funzionalità appena viste, spiegate quale funzionalità sta implementando il Malware in questa sezione.

## 1.1. Parametri Passati

Utilizzando il tool IDA Pro possiamo identificare i parametri considerando che essi hanno un offset positivo.

Como abbiamo evidenziato nello screen di seguito, determiniamo che nella funzione Main() sono presenti 3 parametri.



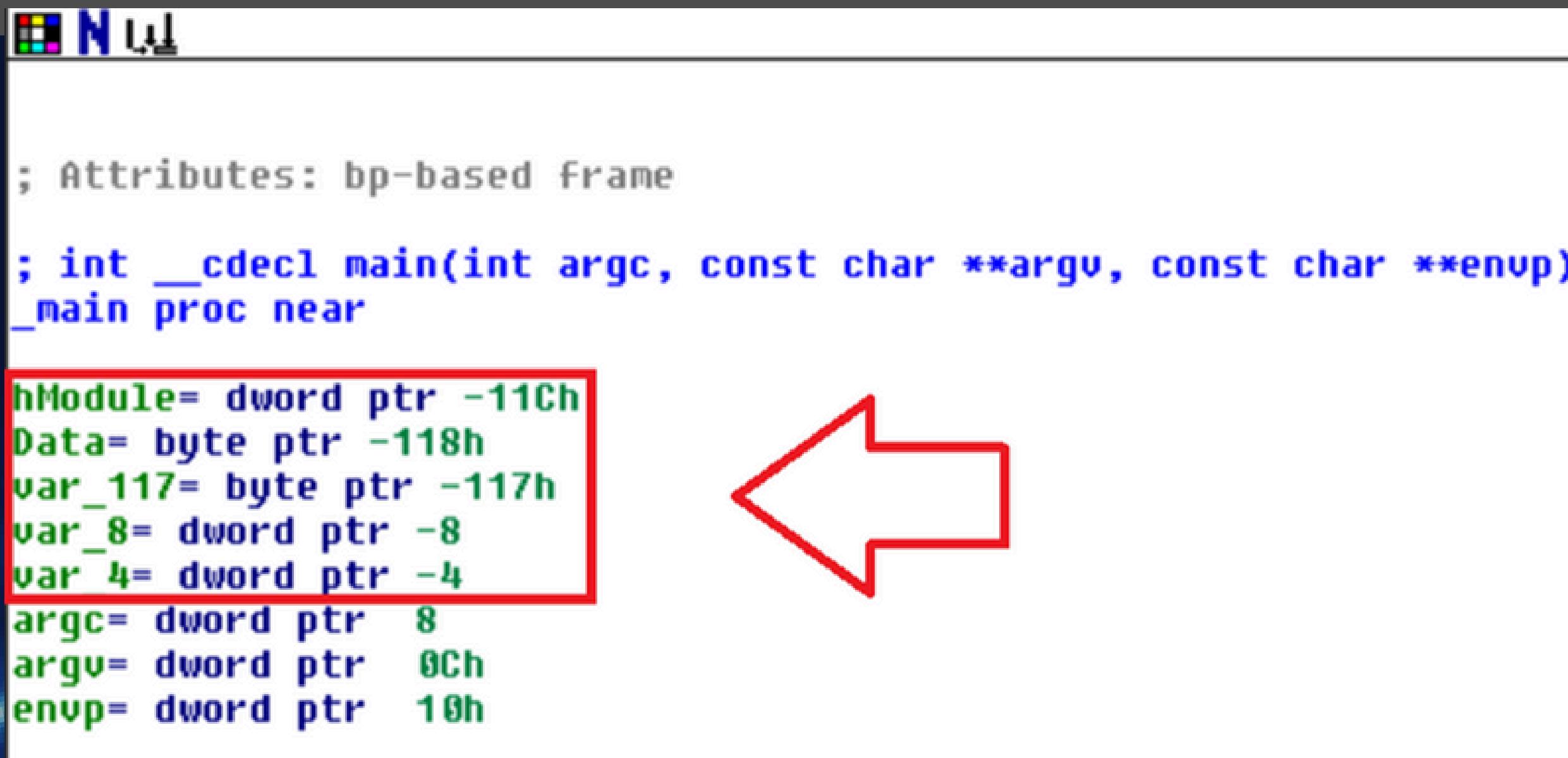
```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

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
```

## 1.2. Variabili Dichiarate

Utilizzando il tool IDA Pro possiamo identificare le variabili considerando che esse hanno un offset negativo.

Como abbiamo evidenziato nello screen di seguito, determiniamo che nella funzione Main() sono 5 variabili.



```
; 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
```

## 1.3. Sezioni Presenti

Utilizzando il tool CFF Explorer possiamo identificare le sezioni del file eseguibile come evidenziamo nello screen di seguito.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
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	000021A8	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040

## 1.3. Sezioni Presenti

### Descrizione delle sezioni identificate:

**.text:** Questa sezione contiene il codice eseguibile del programma, ossia le istruzioni macchina che verranno eseguite dal processore. È generalmente marcata come di sola lettura ed eseguibile, per prevenire modifiche al codice durante l'esecuzione.

**.rdata:** Questa sezione .rdata (read-only data) contiene dati di sola lettura, come stringhe costanti, indirizzi di funzione, tabelle di salto, o altre informazioni che il programma utilizza ma non deve modificare durante l'esecuzione.

**.data:** Questa sezione .data contiene dati modificabili dal programma durante l'esecuzione. In essa si trovano variabili globali e statiche che hanno un valore iniziale definito nel codice. A differenza della sezione .rdata, questa sezione è scrivibile.

**.rsrc:** Questa sezione .rsrc contiene le risorse del programma, come icone, immagini, stringhe di testo, dialoghi, menu e altre risorse che possono essere utilizzate dal programma durante l'esecuzione. Queste risorse sono generalmente non eseguibili ma leggibili.

## 1.4. Librerie Importate

Utilizzando il tool CFF Explorer possiamo identificare le librerie importate dal file eseguibile come evidenziamo nello screen di seguito.

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

## 1.4. Librerie Importate

### Descrizione delle librerie importate:

**KERNEL32.dll:** Questa libreria è una delle più essenziali del sistema operativo Windows, responsabile della gestione di molte delle operazioni di base che consentono il funzionamento delle applicazioni. Questa libreria fornisce una serie di funzioni fondamentali come la gestione della memoria, dei processi e dei thread, nonché l'accesso ai file e alle directory. Inoltre, include funzioni per la gestione della console e della sincronizzazione, rendendola cruciale per l'interazione tra il software e l'hardware a basso livello.

**ADVAPI32.dll:** Questa libreria si occupa di fornire funzioni avanzate per la programmazione delle applicazioni in ambiente Windows, specialmente in ambiti legati alla sicurezza e alla configurazione del sistema. Questa libreria è utilizzata per la gestione delle autorizzazioni e delle politiche di sicurezza, l'accesso e la manipolazione del registro di sistema, la gestione dei servizi di Windows e la criptografia. In sintesi, ADVAPI32.dll è fondamentale per le operazioni che richiedono un controllo più approfondito e sicuro delle risorse del sistema operativo.

## 1.4. Funzioni Importate

**Descrizione delle funzioni importate:**

**VirtualAlloc / VirtualFree:** Queste funzioni sono utilizzate per allocare e liberare memoria in modo dinamico. Un malware potrebbe utilizzare queste funzioni per allocare memoria per il codice dannoso o per nascondere attività malevole.

**LoadLibraryA / GetProcAddress:** Queste funzioni permettono di caricare dinamicamente altre librerie e ottenere l'indirizzo di funzioni specifiche in esse. Un malware potrebbe caricare librerie aggiuntive per eseguire codice dannoso o ottenere funzioni di sistema critiche.

**CreateFileA / WriteFile / ReadFile / SetFilePointer / FlushFileBuffers / SetEndOfFile:** Queste funzioni permettono di creare, scrivere, leggere e manipolare file. Un malware potrebbe utilizzarle per modificare file di sistema, rubare informazioni, o installare altri componenti dannosi.

**ExitProcess / TerminateProcess:** Queste funzioni terminano il processo corrente o un altro processo specificato. Un malware potrebbe utilizzarle per chiudere processi di sicurezza, come antivirus o firewall.

**CloseHandle:** Questa funzione chiude un handle aperto, che potrebbe essere associato a un file, una risorsa o un processo. Un uso improprio potrebbe portare alla perdita di risorse o all'interruzione di processi critici.

## 1.4. Funzioni Importate

**GetModuleHandleA / GetModuleFileNameA / LoadResource / LockResource / FindResourceA / SizeofResource / FreeResource:** Queste funzioni sono utilizzate per gestire moduli e risorse del programma. Un malware potrebbe usarle per manipolare o iniettare codice dannoso in moduli esistenti.

**UnhandledExceptionFilter:** Questa funzione imposta un gestore per le eccezioni non gestite. Un malware potrebbe utilizzarla per nascondere crash o errori, rendendo più difficile il rilevamento.

**HeapCreate / HeapAlloc / HeapReAlloc / HeapFree / HeapDestroy:** Queste funzioni gestiscono la memoria heap. Un malware potrebbe usarle per allocare memoria per il proprio codice o per alterare il comportamento del programma.

**RegSetValueExA:** Questa funzione viene utilizzata per impostare il valore di una chiave di registro. Un malware potrebbe utilizzarla per modificare il registro di sistema in modo da alterare il comportamento del sistema operativo, ad esempio, impostando il malware per eseguire automaticamente all'avvio del sistema, disabilitare funzionalità di sicurezza, o modificare configurazioni critiche.

**RegCreateKeyExA:** Questa funzione consente di creare una nuova chiave di registro o di aprirne una esistente. Un malware potrebbe usarla per creare nuove chiavi di registro che puntano a codice dannoso, per ottenere persistenza nel sistema, o per modificare impostazioni di sistema in modo dannoso.

# 1.4. Scansione Virus Total

58 / 75

Community Score

58/75 security vendors flagged this file as malicious

57d8d248a8741176348b5d12dcf29f34c8f48ede0ca13c30d12e5ba0384056d7  
Malware\_Build\_Week\_U3.exe

Size 52.00 KB Last Analysis Date 24 days ago

peaxe checks-user-input spreader armadillo

Detection Details Relations Behavior

Join our Community and enjoy additional community insights & features!

Popular threat label: trojan.doina/totbrick

AhnLab-V3 Trojan/Win32.Agent.C39204 Alibaba Trojan:Win32/Totbrick.48594dcf

AliCloud Trojan:Win/Totbrick.Gen ALYac Gen:Variant.Doina.65814

Anti-AVL Trojan/Win32.Agent Arcabit Trojan.Doina.D10116

Avast Win32:Trojan-gen AVG Win32:Trojan-gen

Avira (no cloud) TR/Agent.53248.465 BitDefender Gen:Variant.Doina.65814

BitDefenderTheta Gen:NN.ZediaF.36810.aq4@a0clrOb Bkav Pro W32.AI Detect Malware

ClamAV Win.Trojan.Agent-595082 CrowdStrike Falcon Win/malicious\_confidence\_100% (W)

Cybereason Malicious.87a7c5 Cylance Unsafe

Cynet Malicious (score: 99) DeepInstinct MALICIOUS

DrWeb BackDoor.Siggen2.1689 Elastic Malicious (high Confidence)

Emsisoft Gen:Variant.Doina.65814 (B) eScan Gen:Variant.Doina.65814

ESET-NOD32 Win32/Agent.WOU Fortinet W32/Agent.WOU!tr

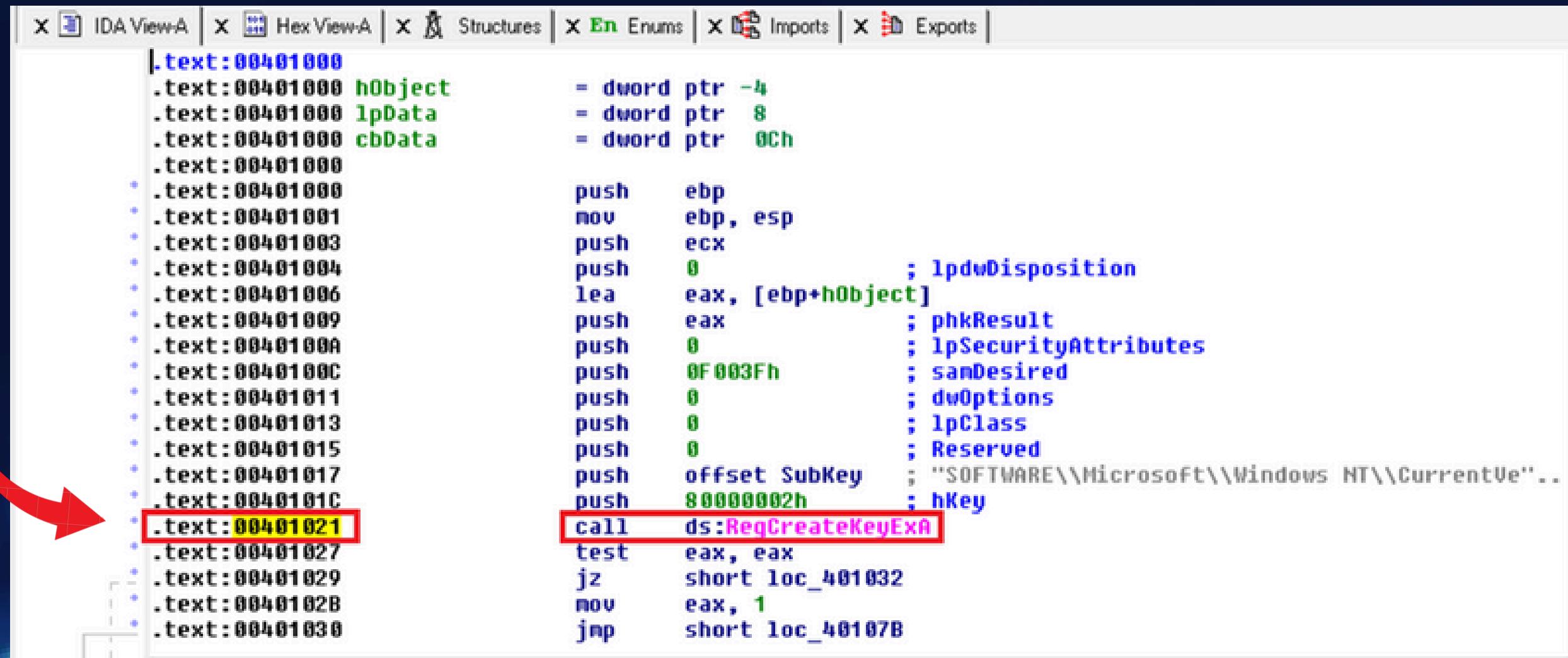


## **1.4. Ipotesi (in base all'analisi statica):**

A seguito di un'analisi delle funzioni importate ed utilizzando Virus Total possiamo ipotizzare che questo malware sembra progettato per iniettarsi in processi o servizi di sistema, assicurandosi di essere sempre attivo (persistente) e difficilmente rilevabile (chiudendo i processi di eventuali FireWall e Antivirus), caricare altri componenti dannosi e manipolare file e processi critici.

## 1.5. Funzione Chiamata alla Locazione 00401021

La funzione chiamata alla locazione 00401021 è RegCreateKeyExA, un'API di Windows utilizzata per creare o aprire una chiave nel registro di sistema.



The screenshot shows the assembly view of the IDA Pro debugger. The assembly code is as follows:

```
.text:00401000 hObject      = dword ptr -4
.text:00401000 lpData       = dword ptr  8
.text:00401000 cbData       = dword ptr  0Ch
.text:00401000
.text:00401000 push    ebp
.text:00401001 mov     ebp, esp
.text:00401002 push    ecx
.text:00401003 push    0          ; lpdwDisposition
.text:00401004 lea     eax, [ebp+hObject]
.text:00401005 push    eax          ; phkResult
.text:00401006 push    0          ; lpSecurityAttributes
.text:00401007 push    0F003Fh      ; dwDesired
.text:00401008 push    0          ; dwOptions
.text:00401009 push    0          ; lpClass
.text:0040100A push    0          ; Reserved
.text:0040100B push    offset SubKey   ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\"
.text:0040100C push    80000002h      ; hKey
.text:00401021 call   ds:RegCreateKeyExA
.text:00401022 test   eax, eax
.text:00401023 jz    short loc_401032
.text:00401024 mov    eax, 1
.text:00401025 jnp   short loc_40107B
```

A red arrow points to the instruction at address 00401021, which is a call to the function RegCreateKeyExA.

## 1.5. Funzione Chiamata alla Locazione 00401021

**Scopo della Funzione:** Lo scopo principale di RegCreateKeyExA è consentire al malware di accedere o creare una chiave di registro specifica all'interno del percorso "SOFTWARE\Microsoft\Windows NT\CurrentVersion". Questa operazione è fondamentale per manipolare le impostazioni del sistema operativo, come ad esempio:

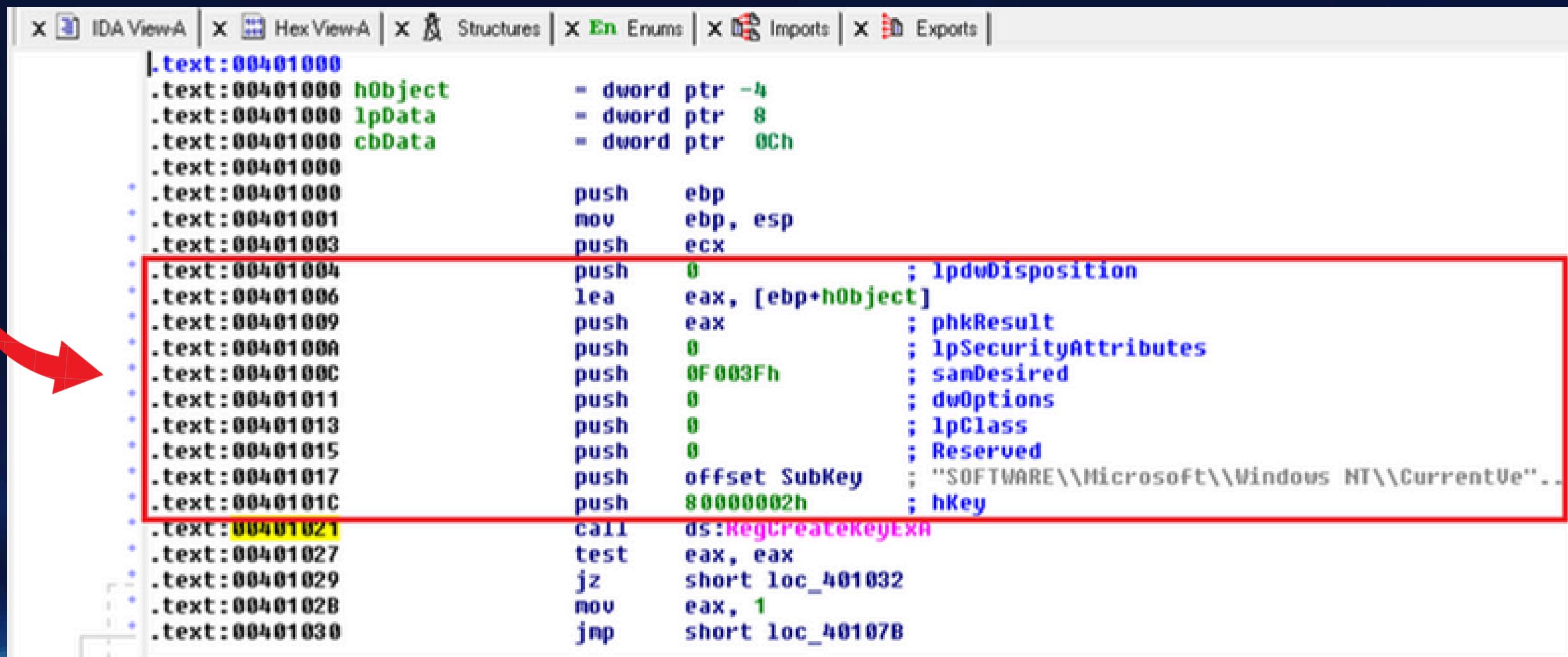
**Modificare Configurazioni di Sistema:** Creando o aprendo una chiave di registro, il malware può modificare configurazioni critiche del sistema, come il comportamento del processo di autenticazione.

**Ottenerе Persistenza:** L'accesso a determinate chiavi di registro permette al malware di garantirsi la possibilità di essere eseguito automaticamente ad ogni avvio del sistema, mantenendo così un accesso persistente.

**Alterare Comportamenti di Windows:** La manipolazione delle chiavi di registro attraverso questa funzione può permettere al malware di alterare il normale funzionamento di Windows, ad esempio iniettando codice malevolo che si attiva durante il login dell'utente.

## 1.6. Parametri Passati alla Locazione 00401021

Dopo che tutti i parametri sono stati posizionati sullo stack, viene eseguita la chiamata alla funzione RegCreateKeyExA con l'istruzione call ds:RegCreateKeyExA.



```
x IDA ViewA | x HexViewA | x Structures | x Enums | x Imports | x Exports | .text:00401000 .text:00401000 hObject = dword ptr -4 .text:00401000 lpData = dword ptr 8 .text:00401000 cbData = dword ptr 0Ch .text:00401000 .text:00401000 push ebp .text:00401001 mov ebp, esp .text:00401003 push ecx .text:00401004 push 0 ; lpdwDisposition .text:00401006 lea eax, [ebp+hObject] .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\\CurrentVer... .text:0040101C push 80000002h ; hKey .text:00401021 call ds:RegCreateKeyExA .text:00401027 test eax, eax .text:00401029 jz short loc_401032 .text:0040102B mov eax, 1 .text:00401030 jnp short loc_40107B
```

## 1.6. Parametri Passati alla Locazione 00401021

**push 80000002h:** Passa hKey con il valore HKEY\_LOCAL\_MACHINE

**push offset SubKey:** Passa il puntatore lpSubKey alla stringa "SOFTWARE\Microsoft\Windows NT\CurrentV...",

**push 0:** Questo valore zero è usato per Reserved, lpClass, e dwOptions, indicando nessuna classe, nessuna opzione speciale.

**push 0F003Fh:** Questo valore rappresenta il parametro samDesired, che specifica le autorizzazioni richieste sulla chiave di registro. In questo caso, 0F003Fh combina i diritti KEY\_READ e KEY\_WRITE, che consentono rispettivamente la lettura e la scrittura sulla chiave di registro.

**push 0:** Questo valore rappresenta il parametro lpSecurityAttributes, che è utilizzato per specificare le attribuzioni di sicurezza della chiave di registro. Essendo impostato a 0, si utilizza il valore predefinito.

**lea eax, [ebp+hObject]:** L'istruzione lea carica l'indirizzo della variabile hObject nel registro EAX. Questo è il puntatore dove verrà restituito l'handle della chiave di registro creata o aperta.

**push eax:** Viene passato il valore di EAX come parametro phkResult, che è un puntatore alla variabile dove verrà restituito l'handle della chiave di registro.

**push 0:** Questo valore rappresenta il parametro lpdwDisposition, che riceverà un valore che indica se la chiave è stata creata o aperta.

# 1.7. Oggetto rappresentato dal parametro alla locazione 00401017

"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" rappresenta il percorso di una chiave di registro che il malware sta tentando di modificare. Questo parametro viene passato alla funzione RegCreateKeyExA, che è utilizzata per creare o aprire chiavi di registro.

.text:00401017

push offset SubKey

; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"



## 1.8. Istruzioni comprese tra gli indirizzi 00401027 e 00401029

In assembly x86, il comando TEST esegue un'operazione di AND bit a bit tra due operandi e aggiorna i flag di stato senza memorizzare il risultato. L'istruzione JZ (Jump if Zero) salta a una determinata etichetta se il flag ZF (Zero Flag) è impostato, indicando che il risultato dell'operazione precedente era zero. Quindi, TEST è usato per verificare condizioni e JZ per dirigere il flusso del programma basandosi su quei risultati.

In questo caso il comando TEST è eseguito tra due registri identici ovvero EAX, EAX; quindi nel caso  $EAX \neq 0$  à  $ZF = 0$ , nel caso  $EAX = 0$  à  $ZF = 1$  (è impostato).

```
.text:00401027  
.text:00401029  
.text:0040102B  
.text:00401030
```



```
test    eax, eax  
jz      short loc_401032  
mov    eax, 1  
jmp    short loc_40107B
```

## 1.9. Traduzione Assembly in costrutto C

```
1 if (a == 0) {  
2 // Corrisponde a Loc_401032  
3 } else {  
4     a = 1; // Imposta a 1 se non lo era già  
5     //Corrisponde a Loc_40107B  
6 }
```

## 1.10. Valore del parametro « ValueName» alla locazione 00401047

Utilizzando il tool OLLY DBG abbiamo identificato che il valore del parametro “ValueName” è “GinaDLL”

The screenshot shows the assembly view of the OLLY debugger. The assembly code is as follows:

Address	OpCode	Instruction
00401032	> 8B40 0C	MOV ECX, DWORD PTR SS:[EBP+C]
00401035	. 51	PUSH ECX
00401036	. 8B55 08	MOV EDX, DWORD PTR SS:[EBP+8]
00401039	. 52	PUSH EDX
0040103A	. 6A 01	PUSH 1
0040103C	. 6A 00	PUSH 0
0040103E	. 68 4C804000	PUSH Malware_.0040804C
00401043	. 8B45 FC	MOV EAX, DWORD PTR SS:[EBP-4]
00401046	. 50	PUSH EAX
00401047	. FF15 00704000	CALL DWORD PTR DS:[<&AUDIOPCI32.RegSetValueExA]

A red arrow points from the assembly instruction at address 00401047 to the parameter list of the `RegSetValueExA` call. The parameter list is shown in a separate window:

Parameter	Value
BufSize	
Buffer	
ValueType	= REG_SZ
Reserved	= 0
ValueName	= "GinaDLL"
hKey	
ReaSetValueExA	

The parameter `ValueName` is highlighted with a red box.

## 1.11. Funzionalità che sta implementando il Malware in questa sezione

Il malware, nella sezione di codice analizzata, sta implementando una funzionalità di manipolazione dell'autenticazione di Windows. Utilizzando le API RegCreateKeyExA e RegSetValueExA, il malware crea o modifica una chiave di registro (GinaDLL) che è cruciale per il processo di autenticazione del sistema. Questa funzionalità consente al malware di:

**Interferire con il processo di login:** Installando una propria DLL, il malware può eseguire codice malevolo durante l'autenticazione.

**Ottenerne persistenza:** Assicurarsi che questa DLL venga caricata ogni volta che l'utente si autentica, garantendo l'esecuzione automatica e il mantenimento dell'accesso al sistema.

Quindi, la funzionalità principale implementata dal malware è la modifica e il controllo del processo di autenticazione di Windows per compromettere la sicurezza del sistema.

# DAY 2

# Traccia

- Analizzare le routine tra le locazioni di memoria 00401080 e 00401128;
  - Valore del parametro «ResourceName » della funzione FindResourceA ();
  - Che funzionalità sta implementando il malware?
  - Si ottiene lo stesso risultato con l'analisi statica basica ?
  - Se si, elencare le evidenze a supporto.
  - Disegnare un diagramma di flusso che comprenda le 3 funzioni principali del Main().
- 
- Preparate l'ambiente ed i tool per l'esecuzione del Malware utilizzando Process Monitor
  - Eseguite il Malware
  - Cosa notate all'interno della cartella dove è situato l'eseguibile? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda
  - Analizzate ora i risultati di Process Monitor
  - Filtrare includendo solamente l'attività sul Registro di Windows
  - Visualizzare attività sul File System
  - Unire tutte le informazioni e trarre le conclusioni

# Analisi con IDA Pro

Abbiamo preso in esame la porzione di codice compresa tra l'indirizzo di memoria 00401080 e l'indirizzo 00401128, come indicato nella traccia. Con IDA Pro si riescono a vedere bene le chiamate a funzione che effettua il programma. Compaiono le seguenti funzioni: FindResourceA, LoadResource, LockResource, SizeOfResource.

Con la visualizzazione a diagramma di flusso (premendo barra spaziatrice) si ottiene anche la visione dei salti che effettua il programma durante l'esecuzione.

The image shows the IDA Pro interface with three main windows. The left window displays assembly code for the range 00401080 to 00401128. The middle window shows a flowchart of the program's execution. Red arrows point from specific assembly instructions in the left window to their corresponding nodes in the flowchart. The right window shows the assembly code for each node in the flowchart.

**Left Window (Assembly View):**

```
.text:00401080 sub_401080 proc near ; CODE XREF: _main+3F↑p
.text:00401080     = dword ptr -18h
.text:00401080     = dword ptr -14h
.text:00401080     = dword ptr -10h
.text:00401080     = dword ptr -8Ch
.text:00401080     = dword ptr -8
.text:00401080     = dword ptr -4
.text:00401080     = dword ptr 8
.text:00401080
.text:00401080     push    ebp
.text:00401080     mov     ebp, esp
.text:00401080     sub    esp, 18h
.text:00401080     push    esi
.text:00401080     push    edi
.text:00401080     mov    [ebp+hResInfo], 0
.text:00401080     mov    [ebp+hResData], 0
.text:00401080     mov    [ebp+Str], 0
.text:00401080     mov    [ebp+Count], 0
.text:00401080     mov    [ebp+var_C], 0
.text:00401080     cmp    [ebp+hModule], 0
.text:00401080     jnz    short loc_4010B8
.text:00401080     xor    eax, eax
.text:00401080     jmp    loc_4011BF
.text:00401080
.text:00401080 loc_4010B8:
.text:004010B8     mov    eax, lpType
.text:004010B8     push   eax
.text:004010B8     mov    ecx, lpName ; lpName
.text:004010B8     push   ecx
.text:004010C4     mov    edx, [ebp+hModule]
.text:004010C8     push   edx
.text:004010C9     call   ds:FindResourceA
.text:004010CF     mov    [ebp+hResInfo], eax
.text:004010D2     cmp    [ebp+hResInfo], 0
.text:004010D6     jnz    short loc_4010DF
.text:004010D8     xor    eax, eax
.text:004010DA     jmp    loc_4011BF
.text:004010DA
```

**Middle Window (Flowchart View):**

```
.text:004010DF ; CODE XREF: sub_401080+56↑j
.text:004010DF loc_4010DF:
.text:004010DF     mov    eax, [ebp+hResInfo]
.text:004010E2     push   eax
.text:004010E3     mov    ecx, [ebp+hModule]
.text:004010E6     push   ecx
.text:004010E7     call   ds:LoadResource
.text:004010ED     mov    [ebp+hResData], eax
.text:004010F0     cmp    [ebp+hResData], 0
.text:004010F4     jnz    short loc_4010FB
.text:004010F6     jmp    loc_4011A5
.text:004010FB ; CODE XREF: sub_401080+74↑j
.text:004010FB loc_4010FB:
.text:004010FB     mov    edx, [ebp+hResData]
.text:004010FE     push   edx
.text:004010FF     call   ds:LockResource
.text:00401105     mov    [ebp+Str], eax
.text:00401108     cmp    [ebp+Str], 0
.text:0040110C     jnz    short loc_401113
.text:0040110E     jmp    loc_4011A5
.text:00401113 ; CODE XREF: sub_401080+2F↑j
.text:00401113 loc_401113:
.text:00401113     mov    eax, [ebp+hResInfo]
.text:00401116     push   eax
.text:00401117     mov    ecx, [ebp+hModule]
.text:0040111A     push   ecx
.text:0040111B     call   ds:SizeofResource
.text:00401121     mov    [ebp+Count], eax
.text:00401124     cmp    [ebp+Count], 0
.text:00401128     ja     short loc_40112C
.text:0040112A     jmp    short loc_4011A5
```

**Right Window (Function View):**

```
loc_4010B8:
mov    eax, lpType
push   eax
mov    ecx, lpName ; lpName
push   ecx
mov    edx, [ebp+hModule]
push   edx
call   ds:FindResourceA
mov    [ebp+hResInfo], eax
cmp    [ebp+hResInfo], 0
short loc_4010DF
```

```
loc_4010DF:
mov    eax, [ebp+hResInfo]
push   eax
mov    ecx, [ebp+hModule]
push   ecx
call   ds:LoadResource
mov    [ebp+hResData], eax
cmp    [ebp+hResData], 0
short loc_4010FB
```

```
loc_4010FB:
mov    edx, [ebp+hResData]
push   edx
call   ds:LockResource
mov    [ebp+Str], eax
cmp    [ebp+Str], 0
short loc_401113
```

```
loc_401113:
mov    eax, [ebp+hResInfo]
push   eax
mov    ecx, [ebp+hModule]
push   ecx
call   ds:SizeofResource
mov    [ebp+Count], eax
cmp    [ebp+Count], 0
ja     short loc_40112C
```

# Analisi con OllyDBG

```
0040103C > A1 34804000 MOV EAX,DWORD PTR DS:[400030]
004010BD . 50 PUSH EAX
004010BE . 8B0D 34804000 MOV ECX,DWORD PTR DS:[400034]
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
004010CF . 8945 EC MOV DWORD PTR SS:[EBP-14],EAX
004010D2 . 837D EC 00 CMP DWORD PTR SS:[EBP-14],0
004010D6 .~75 07 JNZ SHORT Malware_.004010DF
004010D8 . 33C0 XOR EAX,EAX
004010DA .~E9 E0000000 JMP Malware_.004011BF
004010DF > 8B45 FC MOV EBX,DWORD PTR SS:[EBP-14]
```

ResourceType => "BINARY"  
Malware\_.00400000  
ResourceName => "TGAD"  
hModule  
FindResourceA

Successivamente, possiamo notare che il malware va a chiamare altre funzioni: **LoadResource**, **LockResource** e **SizeOfResource**.

Per svolgere il primo punto della traccia, ci siamo serviti degli strumenti di analisi dinamica avanzata. In questo caso il tool utilizzato è **OllyDBG**. Abbiamo impostato un **breakpoint** all'indirizzo di memoria corrispondente alla chiamata della funzione **FindResourceA**, ovvero l'indirizzo di memoria **004010C9**, ed avviando l'esecuzione del programma da **OllyDBG**, riusciamo ad ottenere il valore del parametro che stavamo cercando. Dopo l'esecuzione della funzione, il parametro **ResourceName** avrà al suo interno il valore “**TGAD**”.

```
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.SizeofResou
00401121 . 8945 F0 MOV DWORD PTR SS:[EBP-10],EHX
00401124 . 837D F0 00 CMP DWORD PTR SS:[EBP-10],0
00401128 .~77 02 JA SHORT Malware_.0040112C
00401129 .~EB 79 JMP SHORT Malware_.004011A5
```

hResource  
hModule  
SizeofResource

```
004010E2 . 50 PUSH EHX
004010E3 . 8B4D 08 MOV ECX,DWORD PTR SS:[EBP+8]
004010E6 . 51 PUSH ECX
004010E7 . FF15 14704000 CALL DWORD PTR DS:[<&KERNEL32.LoadResou
004010ED . 8945 E8 MOV DWORD PTR SS:[EBP-18],EHX
004010F0 . 837D E8 00 CMP DWORD PTR SS:[EBP-18],0
004010F4 .~75 05 JNZ SHORT Malware_.004010FB
004010F6 .~E9 AA000000 JMP Malware_.004011A5
004010FB > 8B55 E8 MOV EDX,DWORD PTR SS:[EBP-18]
```

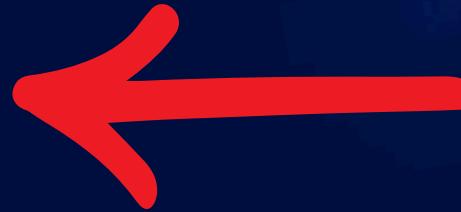
hResource  
hModule  
LoadResource

```
004010FE . 52 PUSH EDX
004010FF . FF15 10704000 CALL DWORD PTR DS:[<&KERNEL32.LockResou
00401105 . 8945 F8 MOV DWORD PTR SS:[EBP-8],EHX
00401108 . 837D F8 00 CMP DWORD PTR SS:[EBP-8],0
0040110C .~75 05 JNZ SHORT Malware_.00401113
0040110E .~E9 92000000 JMP Malware_.004011A5
00401113 > 8B45 FC MOV EBX,DWORD PTR SS:[EBP-14]
```

hResource  
LockResource

# Analisi con IDA Pro

```
-text:004010B8 ;-----  
-text:004010B8 loc_4010B8:          ; CODE XREF: sub_401080+2F↑j  
-text:004010B8     mov    eax, lpType      ; lpType  
-text:004010BD     push   eax           ; lpType  
-text:004010BE     mov    ecx, lpName      ; lpName  
-text:004010C4     push   ecx           ; lpName  
-text:004010C5     mov    edx, [ebp+hModule] ; hModule  
-text:004010C8     push   edx           ; hModule  
-text:004010C9     call   ds:FindResourceA  
-text:004010CF     mov    [ebp+hResInfo], eax  
-text:004010D2     cmp    [ebp+hResInfo], 0  
-text:004010D6     jnz    short loc_4010DF  
-text:004010D8     xor    eax, eax  
-text:004010DA     jmp    loc_4011BF  
text:004010DE
```



**FindResourceA** è una funzione appartente alla libreria KERNEL32 di Microsoft che si occupa di trovare la locazione di una determinata risorsa accettando come parametro il nome ed il tipo della stessa.

```
text:004010DF ;-----  
text:004010DF loc_4010DF:          ; CODE XREF: sub_401080+56↑j  
text:004010DF     mov    eax, [ebp+hResInfo] ; hResInfo  
text:004010E2     push   eax           ; hResInfo  
text:004010E3     mov    ecx, [ebp+hModule] ; hModule  
text:004010E6     push   ecx           ; hModule  
text:004010E7     call   ds:LoadResource  
text:004010ED     mov    [ebp+hResData], eax  
text:004010F0     cmp    [ebp+hResData], 0  
text:004010F4     jnz    short loc_4010FB  
text:004010F6     jmp    loc_4011A5  
text:004010FB
```



**LoadResource** è una funzione che recupera un handle (passato come parametro) che serve per ottenere il puntatore al primo byte di memoria dove risiede la risorsa desiderata.

```
text:004010FB ;-----  
text:004010FB loc_4010FB:          ; CODE XREF: sub_401080+74↑j  
text:004010FB     mov    edx, [ebp+hResData] ; hResData  
text:004010FE     push   edx           ; hResData  
text:004010FF     call   ds:LockResource  
text:00401105     mov    [ebp+Str], eax  
text:00401108     cmp    [ebp+Str], 0  
text:0040110C     jnz    short loc_401113  
text:0040110E     jmp    loc_4011A5
```



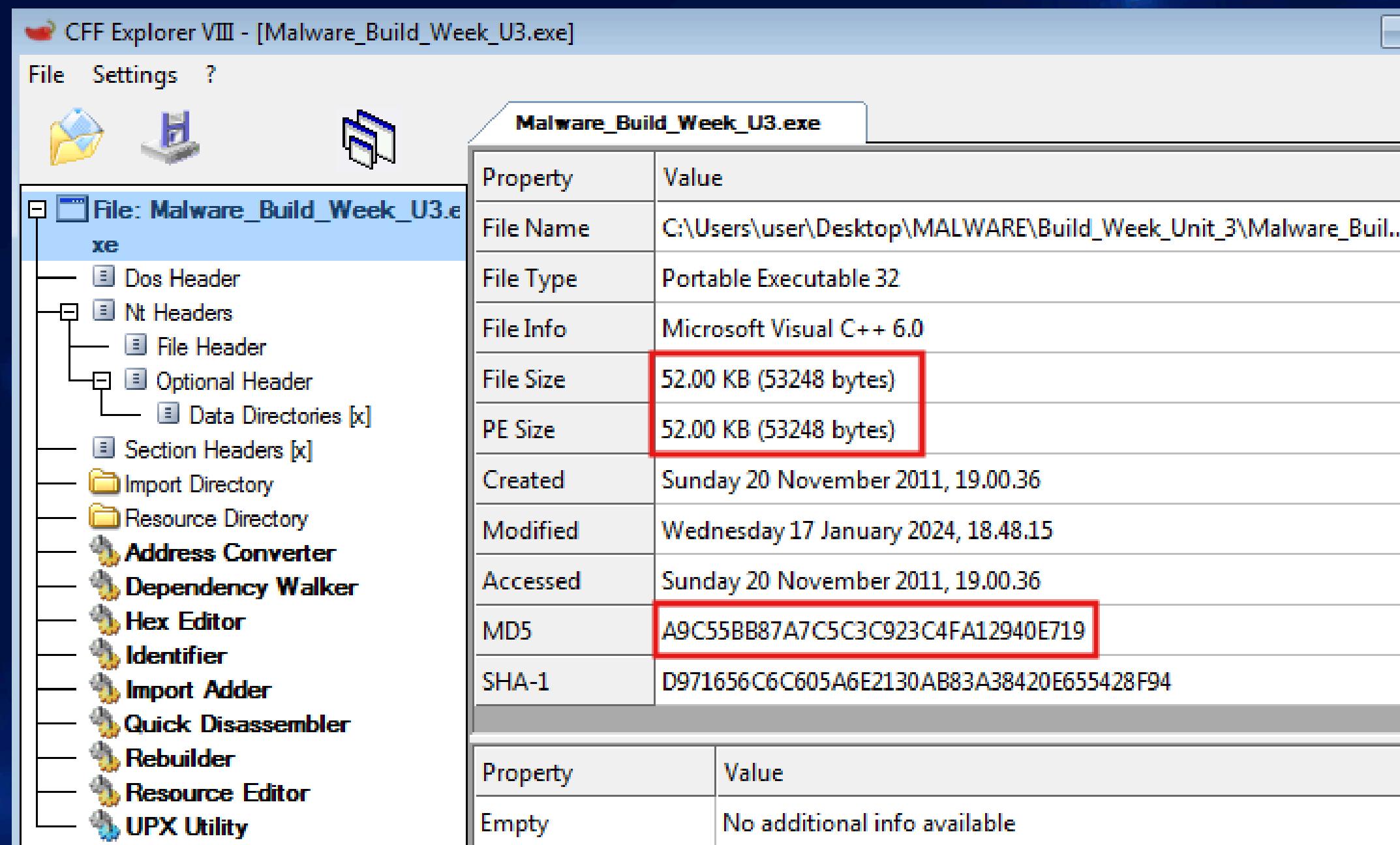
**LockResource** si occupa di recuperare un puntatore a un indirizzo di memoria specifico.

```
text:00401113 ;-----  
text:00401113 loc_401113:          ; CODE XREF: sub_401080+8C↑j  
text:00401113     mov    eax, [ebp+hResInfo] ; hResInfo  
text:00401116     push   eax           ; hResInfo  
text:00401117     mov    ecx, [ebp+hModule] ; hModule  
text:0040111A     push   ecx           ; hModule  
text:0040111B     call   ds:SizeOfResource  
text:00401121     mov    [ebp+Count], eax  
text:00401124     cmp    [ebp+Count], 0  
text:00401128     ja    short loc_40112C  
text:0040112A     jmp    loc_4011A5
```



**SizeOfResource** ottiene la dimensione in byte della risorsa specificata.

# Analisi con CFF Explorer



CFF Explorer

# Analisi con CFF Explorer

Malware_Build_Week_U3.exe				
Module Name	Imports	OFTs	TimeStamp	ForwarderCh
0000769E	N/A	000074EC	000074F0	000074F4
szAnsi	(nFunctions)	Dword	Dword	Dword
<b>KERNEL32.dll</b>	51	00007534	00000000	00000000
ADVAPI32.dll	2	00007528	00000000	00000000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion
000076FE	000076FE	007D	ExitProcess



Nella scheda **Import Directory** di **CFF Explorer** si possono trovare tutte le librerie importate dal malware e le funzioni chiamate.

In questo caso si nota che le librerie importate sono **KERNEL32.dll** e **ADVAPI.dll**. Tra le funzioni chiamate troviamo le stesse della porzione di codice appena analizzata con **IDA Pro** e **OllyDBG** in analisi statica avanzata, ovvero: **FindResourceA**, **LoadResource**, **LockResource**, **SizeOfResource**.

Quindi si sarebbe potuto ottenere il medesimo risultato tramite l'analisi statica basica del malware.

L'unica cosa che non si può ricavare con questo tipo di analisi è il valore del parametro **ResourceName** appartenente alla funzione **FindResourceA**.

# Analisi delle funzioni nel Main()

```
.text:004011D0 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:004011D0 _main          proc near             ; CODE XREF: start+AF↓p
.text:004011D0
.text:004011D0     hModule        = dword ptr -11Ch
.text:004011D0     Data          = byte ptr -118h
.text:004011D0     var_117       = byte ptr -117h
.text:004011D0     var_8         = dword ptr -8
.text:004011D0     var_4         = dword ptr -4
.text:004011D0     argc          = dword ptr 8
.text:004011D0     argv          = dword ptr 0Ch
.text:004011D0     envp          = dword ptr 10h
.text:004011D0
.text:004011D0     push    ebp    |
.text:004011D1     mov     ebp, esp
.text:004011D3     sub     esp, 11Ch
.text:004011D9     push    ebx
.text:004011DA     push    esi
.text:004011DB     push    edi
.text:004011DC     mov     [ebp+var_4], 0
.text:004011E3     push    0           ; lpModuleName
.text:004011E5     call    ds:GetModuleHandleA
.text:004011EB     mov     [ebp+hModule], eax
.text:004011F1     mov     [ebp+Data], 0
.text:004011F8     mov     ecx, 43h
.text:004011FD     xor     eax, eax
.text:004011FF     lea     edi, [ebp+var_117]
.text:00401205     rep    stosd
.text:00401207     stosb
.text:00401208     mov     eax, [ebp+hModule]
.text:0040120E     push   eax           ; hModule
.text:0040120F     call    sub_401080
.text:00401214     add    esp, 4
.text:00401217     mov     [ebp+var_4], eax
.text:0040121A     push   10Eh          ; nSize
.text:0040121F     lea     ecx, [ebp+Data]
.text:00401225     push   ecx           ; lpFilename
.text:00401226     push   0             ; hModule
.text:00401228     call    ds:GetModuleFileNameA
.text:0040122E     push   5Ch           ; Ch
```

Dichiarazione delle variabili locali e dei parametri del Main()

**GetModuleHandleA:** recupera l'handle del modulo che deve essere caricato dal processo chiamante.

**GetModuleFileNameA:** recupera il percorso del modulo specificato che deve esser stato caricato dal processo corrente.

# Analisi delle funzioni nel Main()

C++

```
HMODULE GetModuleHandleA(  
    [in, optional] LPCSTR lpModuleName  
)
```



**GetModuleHandleA**

accetta un parametro di tipo **LPCSTR** nominato **lpModuleName**.

C++

```
DWORD GetModuleFileNameA(  
    [in, optional] HMODULE hModule,  
    [out]          LPSTR   lpFilename,  
    [in]           DWORD    nSize  
)
```



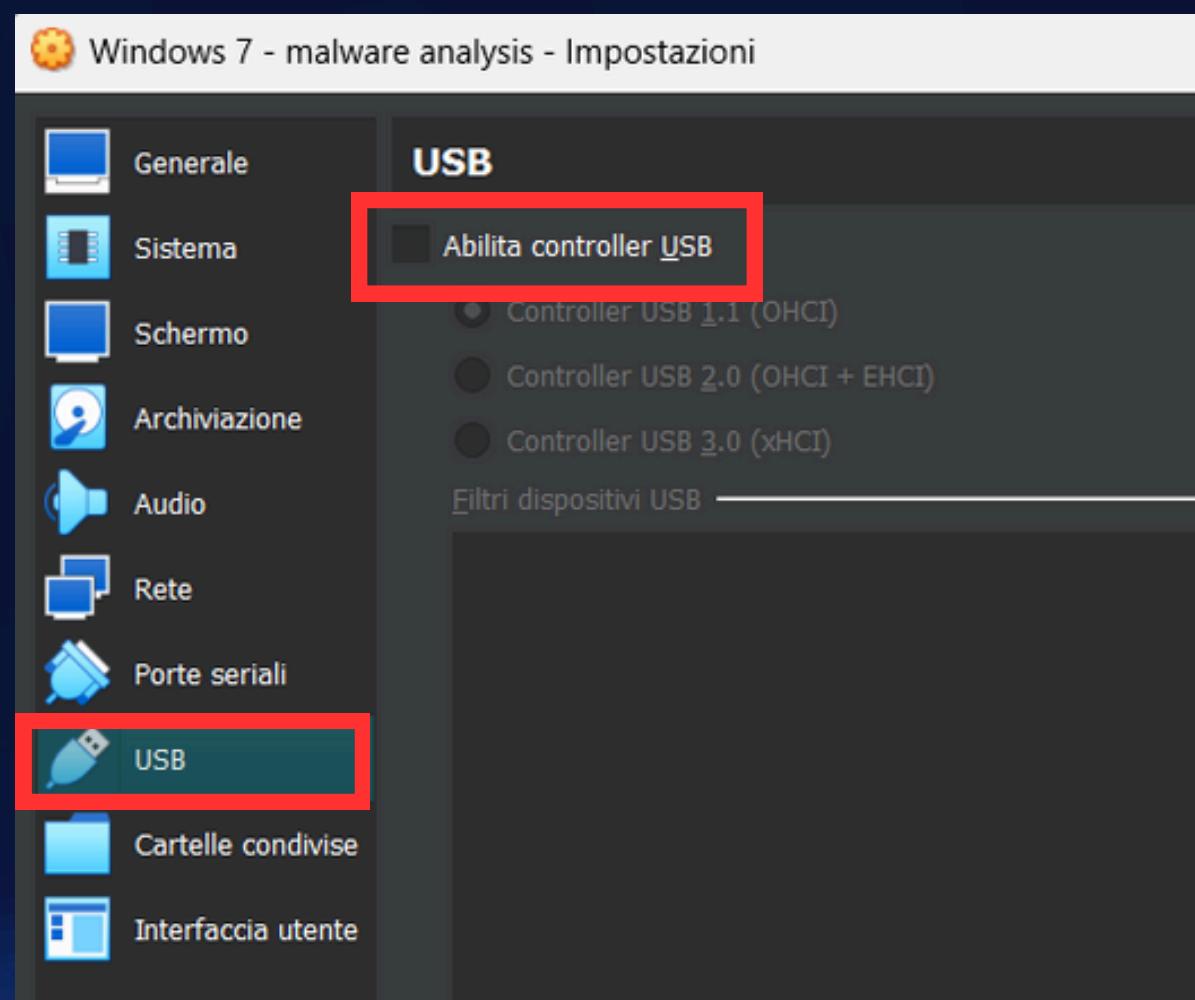
**GetModuleFileNameA**

accetta 3 parametri:

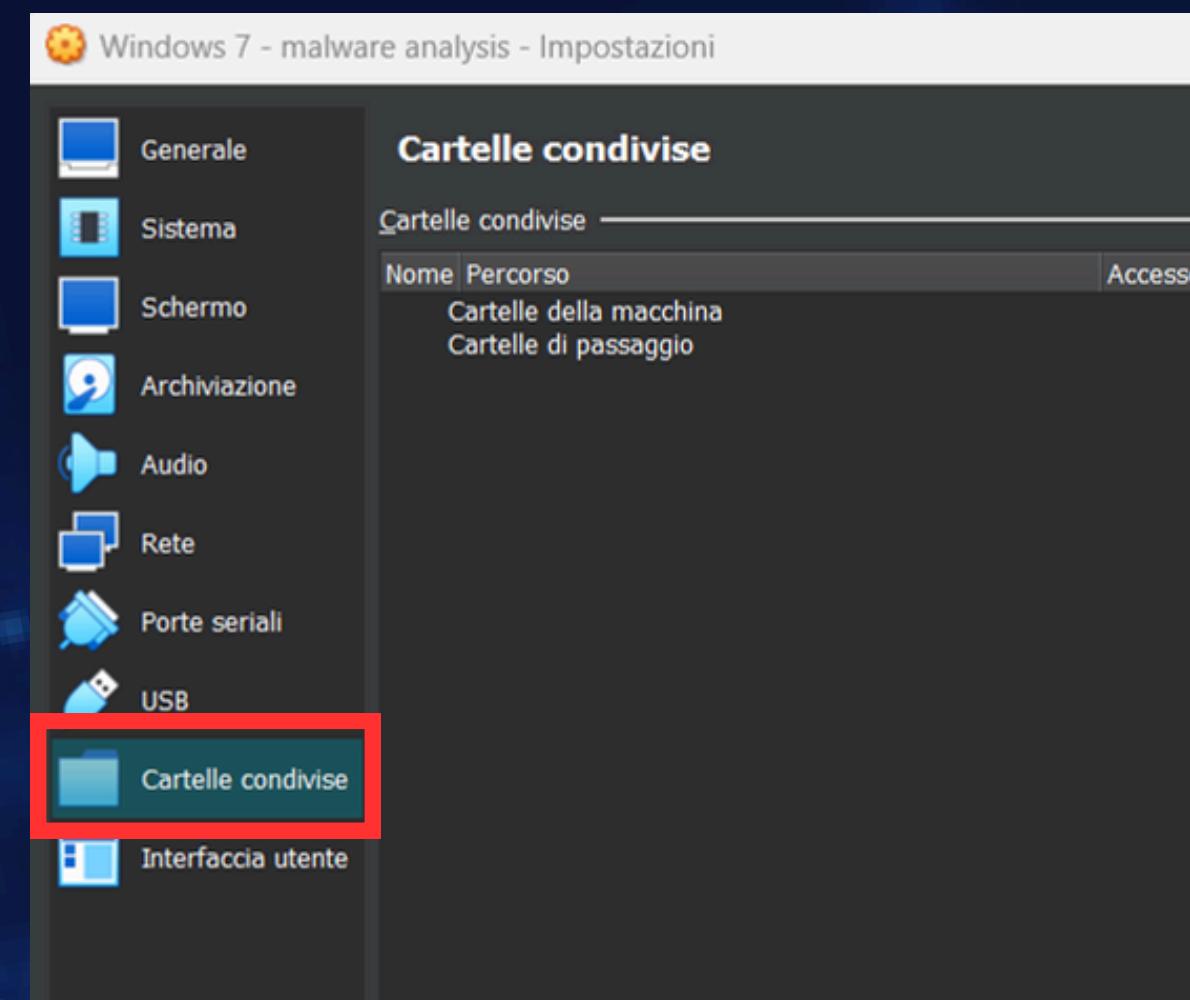
- **hModule** di tipo **HMODULE** (handle)
- **lpFileName** di tipo **LPSTR**
- **nSize** di tipo **DWORD** (intero 32bit)

# Analisi dinamica basica

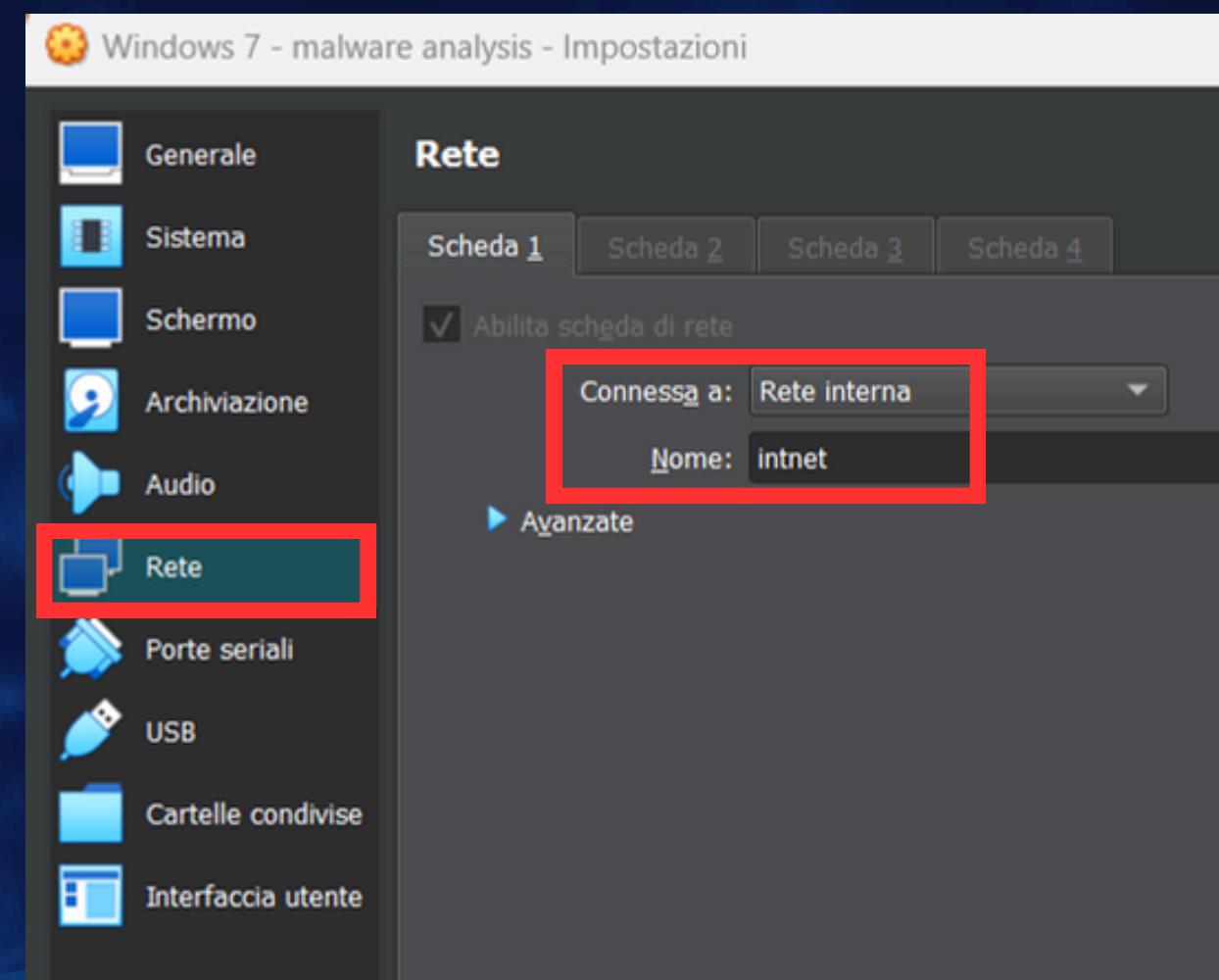
Prima di procedere con l'analisi dinamica basica dobbiamo assicurarci di mettere in sicurezza la macchina virtuale assicurandoci di disabilitare il controller delle porte USB, disattivare le cartelle condivise e metterci in una rete interna.



1-Disabilito controller USB



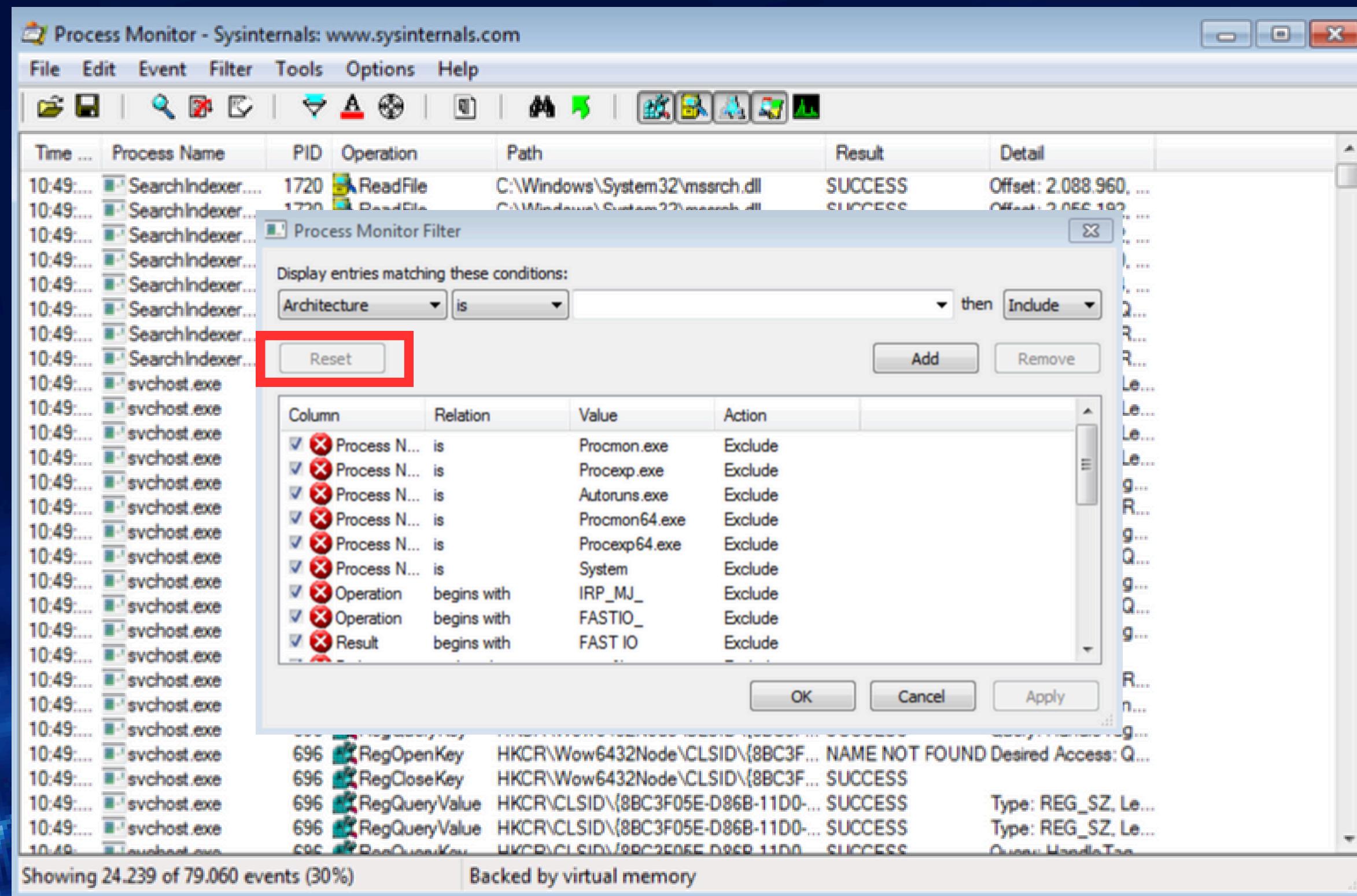
2-Rimuovo cartelle condivise



3- Rete interna

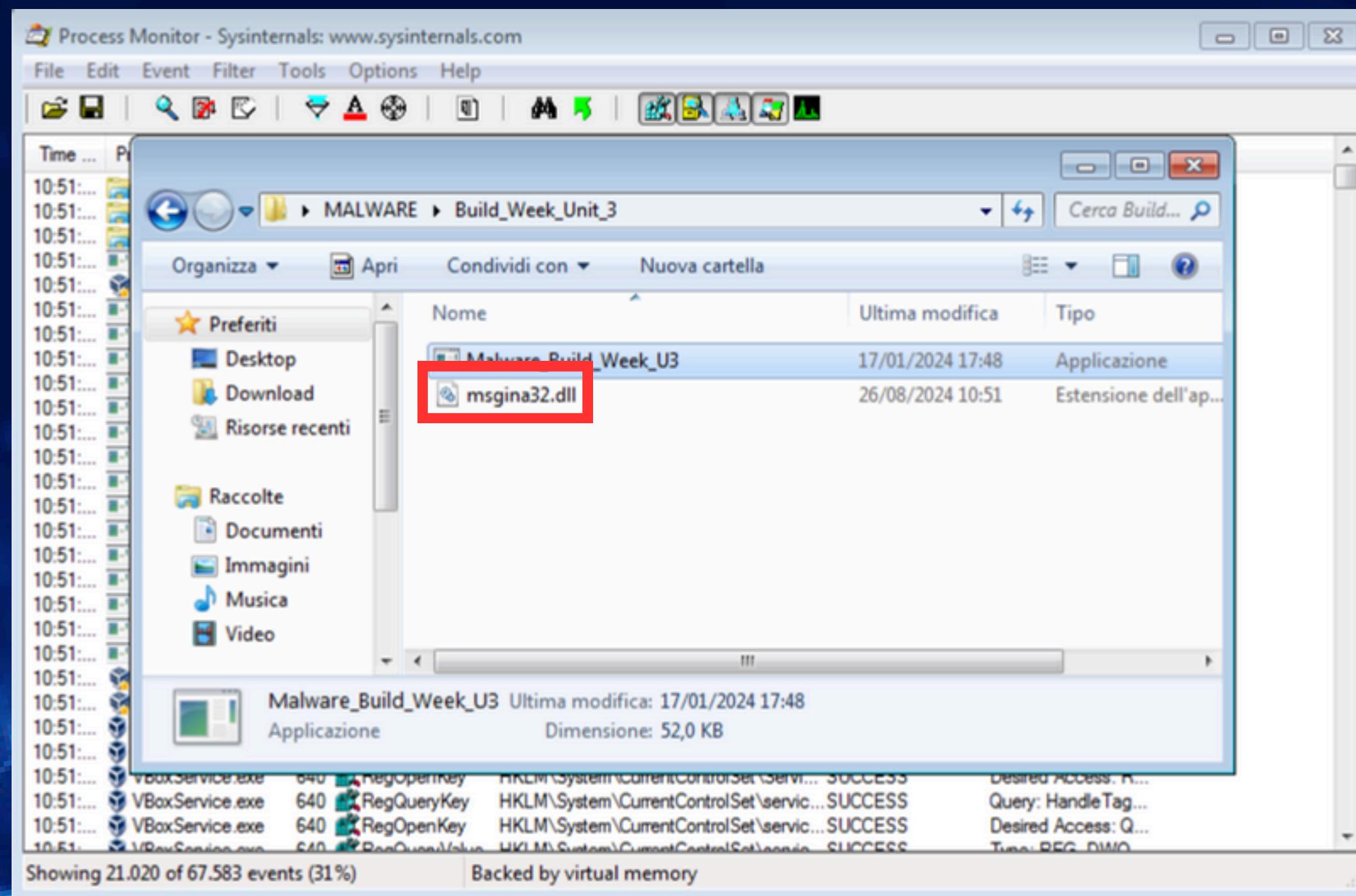
# Process Monitor

Appena avviato Process Monitor ci verrà chiesto di resettare tutti i filtri usati in precedenza così andremo a crearne di nuovi per studiare questo malware.



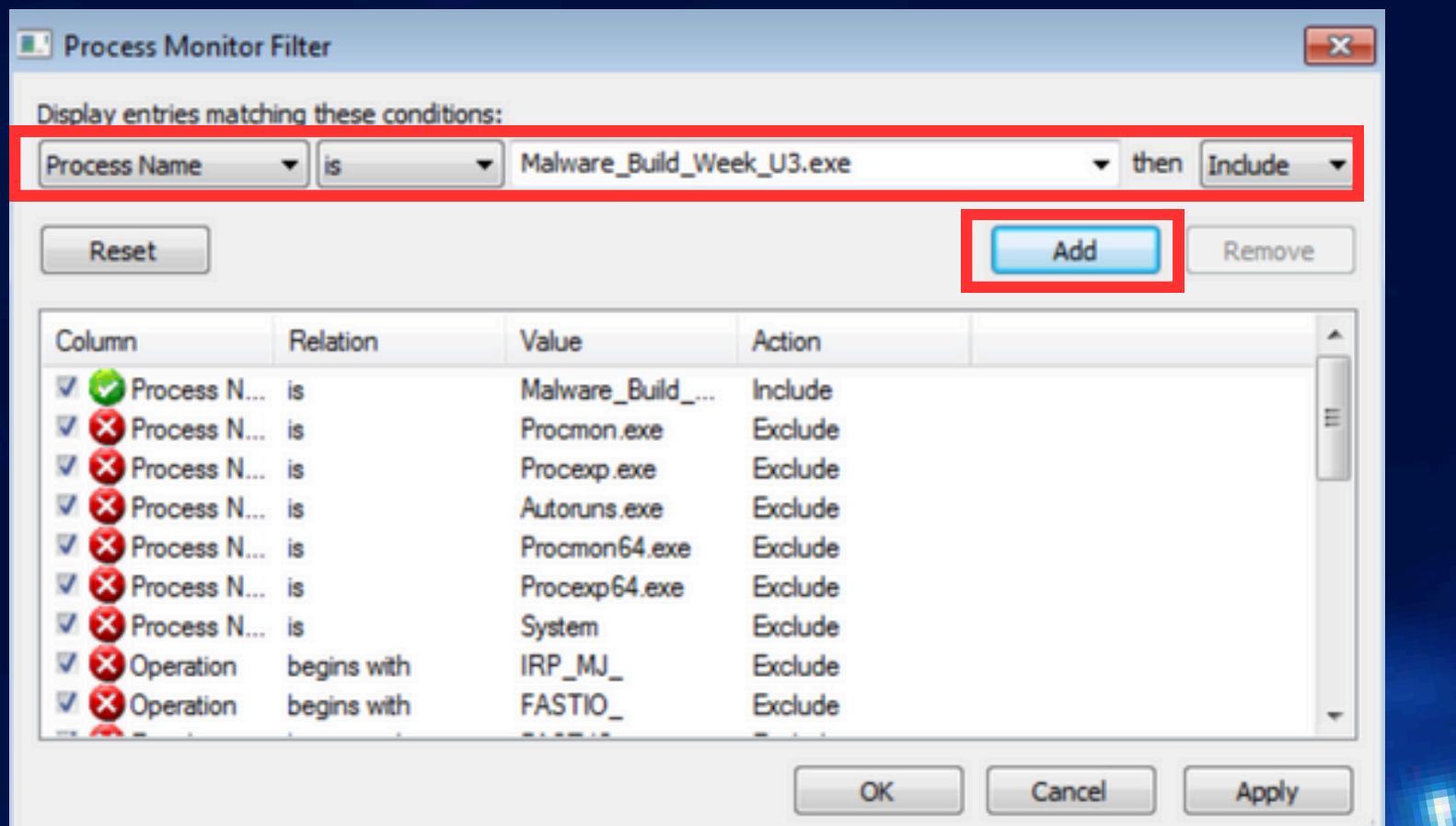
# Avvio del malware

Andiamo ora ad avviare il malware, noteremo che si aprirà e chiuderà in pochi millisecondi il prompt dei comandi, dopo di che comparirà un nuovo file con estensione .dll nella stessa cartella dove risiede il malware come ci aspettavamo sulla base delle funzioni viste in precedenza.



# Analisi con Process Monitor

Andiamo ora ad analizzare nel dettaglio cosa va modificare il malware andando ad inserire dei filtri su Process Monitor, in particolare per vedere nuovi processi eseguiti e i cambiamenti sul registro di sistema e sul File System



Con il filtro mostrato in figura sopra possiamo andare ad evidenziare tutti i processi che riguardano il malware

Time ...	Process Name	PID	Operation	Path	Result	Detail
10:51:	Malware_Build_...	2508	Process Start		SUCCESS	Parent PID: 1972, ...
10:51:	Malware_Build_...	2508	Thread Create		SUCCESS	Thread ID: 2972
10:51:	Malware_Build_...	2508	Load Image	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Image Base: 0x400...
10:51:	Malware_Build_...	2508	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x770...
10:51:	Malware_Build_...	2508	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x772...
10:51:	Malware_Build_...	2508	CreateFile	C:\Windows\Prefetch\MALWARE_BUI...	NAME NOT FOUND	Desired Access: G...
10:51:	Malware_Build_...	2508	RegOpenKey	HKLM\Software\Microsoft\Windows N...	SUCCESS	Desired Access: Q...
10:51:	Malware_Build_...	2508	RegQueryValue	HKLM\SOFTWARE\MICROSOFT\WIN...	NAME NOT FOUND	Length: 1.024
10:51:	Malware_Build_...	2508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
10:51:	Malware_Build_...	2508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
10:51:	Malware_Build_...	2508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 1.024
10:51:	Malware_Build_...	2508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
10:51:	Malware_Build_...	2508	CreateFile	C:\Windows	SUCCESS	Desired Access: E...
10:51:	Malware_Build_...	2508	CreateFile	C:\Windows\System32\wow64.dll	SUCCESS	Desired Access: R...
10:51:	Malware_Build_...	2508	QueryBasicInfor...	C:\Windows\System32\wow64.dll	SUCCESS	CreationTime: 30/0...
10:51:	Malware_Build_...	2508	CloseFile	C:\Windows\System32\wow64.dll	SUCCESS	
10:51:	Malware_Build_...	2508	CreateFile	C:\Windows\System32\wow64.dll	SUCCESS	Desired Access: R...
10:51:	Malware_Build_...	2508	CreateFileMapp...	C:\Windows\System32\wow64.dll	FILE LOCKED WI...	SyncType: SyncTy...
10:51:	Malware_Build_...	2508	CreateFileMapp...	C:\Windows\System32\wow64.dll	SUCCESS	SyncType: SyncTy...
10:51:	Malware_Build_...	2508	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x73f...
10:51:	Malware_Build_...	2508	CloseFile	C:\Windows\System32\wow64.dll	SUCCESS	
10:51:	Malware_Build_...	2508	CreateFile	C:\Windows\System32\wow64win.dll	SUCCESS	Desired Access: R...
10:51:	Malware_Build_...	2508	QueryBasicInfor...	C:\Windows\System32\wow64win.dll	SUCCESS	CreationTime: 30/0...
10:51:	Malware_Build_...	2508	CloseFile	C:\Windows\System32\wow64win.dll	SUCCESS	
10:51:	Malware_Build_...	2508	CreateFile	C:\Windows\System32\wow64win.dll	SUCCESS	Desired Access: R...
10:51:	Malware_Build_...	2508	CreateFileMapp...	C:\Windows\System32\wow64win.dll	FILE LOCKED WI...	SyncType: SyncTy...
10:51:	Malware_Build_...	2508	CreateFileMapp...	C:\Windows\System32\wow64win.dll	SUCCESS	SyncType: SyncTy...
10:51:	Malware_Build_...	2508	Load Image	C:\Windows\System32\wow64win.dll	SUCCESS	Image Base: 0x72e...

Showing 137 of 69.905 events (0.1%)

Backed by virtual memory

Come possiamo vedere nell'immagine sottostante, una volta avviato il malware vengono fatte molte operazioni di creazione file in una cartella critica per Windows come può essere System32 oppure modifiche alle chiavi di registro.

Operazioni sul registro di Windows

Creazione di diversi files

The screenshot shows the Process Monitor application interface. A red arrow points from the 'Operazioni sul registro di Windows' callout to a group of registry-related events in the list. Another red arrow points from the 'Creazione di diversi files' callout to a group of file creation events. The table lists 137 events out of 69,905 total, showing details like time, process name (Malware\_Build\_...), PID, operation type (e.g., CreateFile, RegOpenKey, Load Image), path, result, and detail.

Time ...	Process Name	PID	Operation	Path	Result	Detail
10:51:...	Malware_Build_...	2508	Process Start		SUCCESS	Parent PID: 1972, ...
10:51:...	Malware_Build_...	2508	Thread Create		SUCCESS	Thread ID: 2972
10:51:...	Malware_Build_...	2508	Load Image	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Image Base: 0x400...
10:51:...	Malware_Build_...	2508	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x770...
10:51:...	Malware_Build_...	2508	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x772...
10:51:...	Malware_Build_...	2508	CreateFile	C:\Windows\Prefetch\MALWARE_BUI...	NAME NOT FOUND	Desired Access: G...
10:51:...	Malware_Build_...	2508	RegOpenKey	HKLM\Software\Microsoft\Windows N...	SUCCESS	Desired Access: Q...
10:51:...	Malware_Build_...	2508	RegQueryValue	HKLM\Software\Microsoft\Windows N...	NAME NOT FOUND	Length: 1.024
10:51:...	Malware_Build_...	2508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
10:51:...	Malware_Build_...	2508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
10:51:...	Malware_Build_...	2508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 1.024
10:51:...	Malware_Build_...	2508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
10:51:...	Malware_Build_...	2508	CreateFile	C:\Windows	SUCCESS	Desired Access: E...
10:51:...	Malware_Build_...	2508	CreateFile	C:\Windows\System32\wow64.dll	SUCCESS	Desired Access: R...
10:51:...	Malware_Build_...	2508	QueryBasicInfor...	C:\Windows\System32\wow64.dll	SUCCESS	CreationTime: 30/0...
10:51:...	Malware_Build_...	2508	CloseFile	C:\Windows\System32\wow64.dll	SUCCESS	
10:51:...	Malware_Build_...	2508	CreateFile	C:\Windows\System32\wow64.dll	SUCCESS	Desired Access: R...
10:51:...	Malware_Build_...	2508	CreateFileMapp...	C:\Windows\System32\wow64.dll	FILE LOCKED WI...	SyncType: SyncTy...
10:51:...	Malware_Build_...	2508	CreateFileMapp...	C:\Windows\System32\wow64.dll	SUCCESS	SyncType: SyncTy...
10:51:...	Malware_Build_...	2508	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x73f...
10:51:...	Malware_Build_...	2508	CloseFile	C:\Windows\System32\wow64.dll	SUCCESS	
10:51:...	Malware_Build_...	2508	CreateFile	C:\Windows\System32\wow64win.dll	SUCCESS	Desired Access: R...
10:51:...	Malware_Build_...	2508	QueryBasicInfor...	C:\Windows\System32\wow64win.dll	SUCCESS	CreationTime: 30/0...
10:51:...	Malware_Build_...	2508	CloseFile	C:\Windows\System32\wow64win.dll	SUCCESS	
10:51:...	Malware_Build_...	2508	CreateFile	C:\Windows\System32\wow64win.dll	SUCCESS	Desired Access: R...
10:51:...	Malware_Build_...	2508	CreateFileMapp...	C:\Windows\System32\wow64win.dll	FILE LOCKED WI...	SyncType: SyncTy...
10:51:...	Malware_Build_...	2508	CreateFileMapp...	C:\Windows\System32\wow64win.dll	SUCCESS	SyncType: SyncTy...
10:51:...	Malware_Build_...	2508	Load Image	C:\Windows\System32\wow64win.dll	SUCCESS	Image Base: 0x72e...

# Creazione chiave malevola

Inserendo ora un filtro per evidenziare solo le operazioni fatte sulle chiavi di registro possiamo notare che viene creata con successo una nuova chiave al percorso:  
“HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon”

The screenshot shows the Process Monitor application interface. On the left, a 'Process Monitor Filter' dialog is open, with its search bar set to 'Operation contains Reg'. This filter is highlighted with a red box. On the right, the main window displays a list of registry events. One specific event is highlighted with a red box: '10:51: Malware\_Build\_... 2508 RegCreateKey HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon SUCCESS KeySetInformation...'. This event represents the successful creation of a new key at the specified path.

Time	Process Name	PID	Operation	Path	Result	Detail
10:51...	Malware_Build_...	2508	RegOpenKey	HKLM\Software\Wow6432Node\Policy...	REPARSE	Desired Access: Q...
10:51...	Malware_Build_...	2508	RegOpenKey	HKLM\SOFTWARE\Policies\Microsoft\...	SUCCESS	Desired Access: Q...
10:51...	Malware_Build_...	2508	RegSetInfoKey	HKLM\SOFTWARE\Policies\Microsoft\...	SUCCESS	KeySetInformation...
10:51...	Malware_Build_...	2508	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\...	NAME NOT FOUND Length: 80	
10:51...	Malware_Build_...	2508	RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\...	SUCCESS	
10:51...	Malware_Build_...	2508	RegOpenKey	HKCU\Software\Microsoft\Win...	NAME NOT FOUND Desired Access: Q...	
10:51...	Malware_Build_...	2508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
10:51...	Malware_Build_...	2508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
10:51...	Malware_Build_...	2508	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	KeySetInformation...
10:51...	Malware_Build_...	2508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_SZ, Le...
10:51...	Malware_Build_...	2508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND Length: 260	
10:51...	Malware_Build_...	2508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_SZ, Le...
10:51...	Malware_Build_...	2508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
10:51...	Malware_Build_...	2508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
10:51...	Malware_Build_...	2508	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	KeySetInformation...
10:51...	Malware_Build_...	2508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_DWO...
10:51...	Malware_Build_...	2508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
10:51...	Malware_Build_...	2508	RegOpenKey	HKLM	SUCCESS	Desired Access: M...
10:51...	Malware_Build_...	2508	RegQueryKey	HKLM	SUCCESS	Query: Handle Tag...
10:51...	Malware_Build_...	2508	RegQueryKey	HKLM	SUCCESS	Query: Name
10:51...	Malware_Build_...	2508	RegOpenKey	HKLM\Software\Wow6432Node\Micro...	NAME NOT FOUND Desired Access: R...	
10:51...	Malware_Build_...	2508	RegQueryKey	HKLM\Software\Wow6432Node\Micro...	SUCCESS	Query: Handle Tag...
10:51...	Malware_Build_...	2508	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
10:51...	Malware_Build_...	2508	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	KeySetInformation...
10:51...	Malware_Build_...	2508	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\VM...	SUCCESS	Query: Handle Tag...
10:51...	Malware_Build_...	2508	RegSetValue	HKLM\SOFTWARE\Wow6432Node\VM...	ACCESS DENIED	Type: REG_SZ, Le...
10:51...	Malware_Build_...	2508	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\VM...	SUCCESS	
10:51...	Malware_Build_...	2508	RegCloseKey	HKLM\SOFTWARE\MICROSOFT\WIN...	SUCCESS	
10:51...	Malware_Build_...	2508	RegCloseKey	HKLM\SOFTWARE\MICROSOFT\WIN...	SUCCESS	
10:51...	Malware_Build_...	2508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
10:51...	Malware_Build_...	2508	RegCloseKey	HKLM	SUCCESS	

Subito dopo aver creato la chiave nel registro di sistema il malware prova a modificarne il valore in una sotto-chiave, non riuscendoci però per un ACCESS DENIED. A questo punto il malware chiude la chiave appena creata.

Time	Process Name	PID	Operation	Path	Result	Detail
11:32...	Malware_Build...	2688	RegSetInfoKey	HKLM\SOFTWARE\Policies\Microsoft\...	SUCCESS	KeySetInformation...
11:32...	Malware_Build...	2688	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\...	NAME NOT FOUND	Length: 80
11:32...	Malware_Build...	2688	RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\...	SUCCESS	
11:32...	Malware_Build...	2688	RegOpenKey	HKCU\Software\Policies\Microsoft\Win...	NAME NOT FOUND	Desired Access: Q...
11:32...	Malware_Build...	2688	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
11:32...	Malware_Build...	2688	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
11:32...	Malware_Build...	2688	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	KeySetInformation...
11:32...	Malware_Build...	2688	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_SZ, Le...
11:32...	Malware_Build...	2688	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 260
11:32...	Malware_Build...	2688	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_SZ, Le...
11:32...	Malware_Build...	2688	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
11:32...	Malware_Build...	2688	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
11:32...	Malware_Build...	2688	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	KeySetInformation...
11:32...	Malware_Build...	2688	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 548
11:32...	Malware_Build...	2688	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_DWO...
11:32...	Malware_Build...	2688	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
11:32...	Malware_Build...	2688	RegOpenKey	HKLM	SUCCESS	Desired Access: M...
11:32...	Malware_Build...	2688	RegQueryKey	HKLM	SUCCESS	Query: HandleTag...
11:32...	Malware_Build...	2688	RegQueryKey	HKLM	SUCCESS	Query: Name
11:32...	Malware_Build...	2688	RegOpenKey	HKLM\Software\Wow6432Node\Micro...	NAME NOT FOUND	Desired Access: R...
11:32...	Malware_Build...	2688	RegQueryKey	HKLM	SUCCESS	Query: HandleTag...
11:32...	Malware_Build...	2688	RegQueryKey	HKLM	SUCCESS	Query: Name
11:32...	Malware_Build...	2688	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\M...	SUCCESS	Desired Access: All...
11:32...	Malware_Build...	2688	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\M...	SUCCESS	KeySetInformation...
11:32...	Malware_Build...	2688	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\M...	SUCCESS	Query: HandleTag...
11:32...	Malware_Build...	2688	RegSetValue	HKLM\SOFTWARE\Wow6432Node\M...	ACCESS DENIED	Type: REG_SZ, Le...
11:32...	Malware_Build...	2688	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\M...	SUCCESS	
11:32...	Malware_Build...	2688	RegCloseKey	HKLM\SOFTWARE\Microsoft\WIN...	SUCCESS	
11:32...	Malware_Build...	2688	RegCloseKey	HKLM\SOFTWARE\Microsoft\WIN...	SUCCESS	
11:32...	Malware_Build...	2688	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
11:32...	Malware_Build...	2688	RegCloseKey	HKLM	SUCCESS	

# Funzione che crea il file

Analizzando ora l'attività sul File System possiamo andare a verificare quale chiamata a sistema ha creato il nuovo file msgina32.dll nella cartella del malware. Per farlo abbiamo inserito un nuovo filtro come mostrato in figura sotto e notiamo subito questa operazione CreateFile seguita subito dopo da due WriteFile proprio sul file di nostro interesse msgina32.dll.

The screenshot shows the Process Monitor application interface. On the left, a 'Process Monitor Filter' dialog is open, with its 'Event Class' dropdown set to 'contains' and 'File System' selected. A red box highlights this configuration. Below the dropdown are several filter rules, also highlighted by a red box. The main window displays a log of system events. A red box highlights a specific sequence of events: a 'CreateFile' operation at 11:32... with PID 2688, followed by two 'WriteFile' operations at the same time. The 'CreateFile' event has a detailed tooltip showing it was successful with generic write/read attributes and overwrite disposition. The 'WriteFile' events show offsets 0 and 4.096 respectively. The log continues with other system calls like 'CloseFile', 'QueryNameInfo', and various 'QueryNameInfo' calls for system DLLs.

Time	Process Name	PID	Operation	Path	Result	Detail
11:32...	Malware_Build_...	2688	CreateFile	C:\Windows	SUCCESS	Desired Access: R...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows	SUCCESS	Name: \Windows
11:32...	Malware_Build_...	2688	CloseFile	C:\Windows	SUCCESS	
11:32...	Malware_Build_...	2688	CreateFile	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Desired Access: E...
11:32...	Malware_Build_...	2688	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS	Offset: 338.944, Le...
11:32...	Malware_Build_...	2688	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS	Offset: 330.752, Le...
11:32...	Malware_Build_...	2688	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS	Offset: 62.464, Len...
11:32...	Malware_Build_...	2688	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Desired Access: R...
11:32...	Malware_Build_...	2688	QueryBasicInfor...	C:\Windows\SysWOW64\sechost.dll	SUCCESS	CreationTime: 14/0...
11:32...	Malware_Build_...	2688	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	
11:32...	Malware_Build_...	2688	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Desired Access: R...
11:32...	Malware_Build_...	2688	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll	FILE LOCKED WI...	SyncType: SyncTy...
11:32...	Malware_Build_...	2688	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll	SUCCESS	SyncType: SyncTy...
11:32...	Malware_Build_...	2688	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS	
11:32...	Malware_Build_...	2688	CreateFile	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Desired Access: G...
11:32...	Malware_Build_...	2688	WriteFile	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Offset: 0, Length: 4...
11:32...	Malware_Build_...	2688	WriteFile	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Offset: 4.096, Len...
11:32...	Malware_Build_...	2688	CloseFile	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\System32\apisetschema.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Name: \Users\user...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\System32\wow64win.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\System32\wow64.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\System32\wow64cpu.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\SysWOW64\cryptbase.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\SysWOW64\sspicli.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\SysWOW64\advapi32.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\SysWOW64\KernelBase.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\SysWOW64\msvcr.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\SysWOW64\pcre4.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\SysWOW64\kernel32.dll	SUCCESS	Name: \Windows\...
11:32...	Malware_Build_...	2688	QueryNameInfo	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Name: \Windows\...

# Conclusioni

Sulla base di tutte le informazioni raccolte dall'analisi statica e dinamica sia basica che avanzata possiamo trarre le seguenti conclusioni:

Il malware importa due librerie che contengono funzioni critiche su Windows ovvero KERNEL32.dll per la creazione, rimozione o modifica dei file (CreateFile, WriteFile, LoadLibrary, LoadResource...), e ADVAPI32.dll per interagire con il registro (RegCreateKeyExA e RegSetValueExA).  
Da questa prima analisi statica ci aspettiamo che il malware faccia delle modifiche alle chiavi del registro e crei nuovi file come abbiamo poi dimostrato nell'analisi dinamica con l'esecuzione del malware.

Nel nostro caso il malware si bloccava nella fase di scrittura della chiave Winlogon con codice di errore ACCESS DENIED quindi non riusciva a completare la sua funzione.

Il malware risulta però essere molto pericoloso nelle versioni di Windows precedenti al 7 dal momento che msgina32.dll non era stata deprecata, quindi modificabile e sostituibile con file malevolo per ottenere l'accesso alle credenziali. Con queste informazioni possiamo classificare il malware come un dropper (crea msgina32.dll e se avesse i permessi TGAD0.exe come confermato da VirusTotal).

Dropped Files (5) 			
Scanned	Detections	File type	Name
2022-04-02	44 / 69	Win32 DLL	msgina32.dll
2024-08-16	55 / 74	Win32 DLL	TGAD0.exe

# Conclusioni

Da VirusTotal possiamo notare che il malware effettua anche delle connessioni probabilmente malevole, forse per inviare le credenziali di accesso rubate con il file malevolo. Dall'analisi delle funzioni però non abbiamo notato librerie importate che permettono la connessione ad internet come ad esempio WININET.dll o WS2\_32.dll (Winsock). Possiamo supporre che, dal momento che il malware non termina la sua attività di conseguenza non riesca ad arrivare al punto dove crea il file TGAD0.exe che si occupa poi di effettuare le connessioni ed inviare le credenziali rubate all'attaccante.

Contacted URLs (9) ⓘ			
Scanned	Detections	Status	URL
2024-08-23	0 / 96	200	<a href="http://crl4.digicert.com/DigiCertGlobalRootCA.crl">http://crl4.digicert.com/DigiCertGlobalRootCA.crl</a>
2024-02-12	0 / 91	200	<a href="http://ocsp.digicert.com/MFEwTzBNMEswSTAJBgUrDgMCGgUABBSPwi+rBFUJh18C0nqAFI8du2+GkP0xGIOCEAEF0almGak0">http://ocsp.digicert.com/MFEwTzBNMEswSTAJBgUrDgMCGgUABBSPwi+rBFUJh18C0nqAFI8du2+GkP0xGIOCEAEF0almGak0</a>

Contacted IP addresses (18) ⓘ			
IP	Detections	Autonomous System	Country
104.85.240.187	0 / 94	16625	US
114.114.114.114	2 / 94	21859	CN
117.18.237.29	0 / 94	15133	US
192.168.0.25	0 / 94	-	-
192.168.0.63	0 / 94	-	-
192.229.211.108	0 / 94	15133	US
20.80.129.13	0 / 94	8075	US

# DAY 3

# Traccia

## GINA

GINA (Graphical identification and authentication ) è un componente legato di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica. Esso permette agli utenti di inserire username e password nel classico riquadro Windows.

- Cosa può succedere se il file .dll legato 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.

# Overview GINA

- Componente chiave del processo di autenticazione in Windows 2000, XP (sistemi pre-Vista).
  - Gestisce l'interfaccia grafica per l'inserimento di credenziali.
  - Responsabile della schermata di login degli utenti.
- Funzionalità di GINA.dll:
  - Personalizzazione del processo di login e autenticazione.
  - Supporto per metodi di autenticazione alternativi (smart card, biometria, rete).
  - Ampio utilizzo in ambienti aziendali con esigenze di sicurezza avanzata.
- Rischi associati a GINA.dll:
  - Obiettivo privilegiato per attacchi mirati alla sostituzione della libreria.
  - Potenziale compromissione del processo di autenticazione.
  - Intercettazione delle credenziali degli utenti da parte di malware.



# Conseguenze sostituzione con .dll malevolo

La sostituzione di un file legittimo come GINA.dll con una versione malevola può avere conseguenze devastanti per la sicurezza di un sistema Windows, in particolare nelle versioni precedenti a Windows Vista.

Di seguito vengono esaminati in dettaglio i principali impatti che questo tipo di attacco può avere:

- Intercettazione delle credenziali
- Accesso non autorizzato
- Compromissione della sicurezza complessiva del sistema

In sintesi, la sostituzione del file GINA legito con una versione malevola rappresenta una grave minaccia alla sicurezza, poiché permette agli attaccanti di intercettare informazioni sensibili e ottenere accesso non autorizzato al sistema.

# Intercettazione credenziali

GINA.dll è un componente critico che gestisce l'autenticazione degli utenti attraverso un'interfaccia grafica. Quando un utente tenta di accedere al sistema, GINA.dll presenta la schermata di login e gestisce l'inserimento di username e password.

- Se GINA.dll viene sostituita con una versione malevola, il malware può intercettare tutte le credenziali inserite dagli utenti. Ogni volta che un utente digita il proprio nome utente e password, queste informazioni vengono registrate dal malware.
- L'intercettazione delle credenziali permette all'attaccante di acquisire informazioni sensibili senza che l'utente o l'amministratore di sistema se ne accorgano. Questa capacità rappresenta una grave violazione della privacy e della sicurezza del sistema.

# Accesso non autorizzato

Le credenziali intercettate possono essere utilizzate dagli attaccanti per ottenere accesso non autorizzato al sistema. Questo accesso può avvenire in diverse forme, ognuna con conseguenze potenzialmente gravi:

- **Accesso ai privilegi dell'utente:**

- Con le credenziali rubate, un attaccante può accedere al sistema con i privilegi dell'utente legittimo. Se l'utente compromesso ha diritti amministrativi, l'attaccante può eseguire operazioni di amministrazione, come l'installazione di software, la modifica delle configurazioni di sistema e la creazione di nuovi account utente.

- **Esecuzione di ulteriori attacchi:**

- Una volta ottenuto l'accesso, l'attaccante può eseguire ulteriori attacchi, come l'escalation dei privilegi per ottenere accesso root o amministratore, l'esecuzione di ransomware o il lancio di attacchi DDoS (Distributed Denial of Service) dal sistema compromesso.

- **Modifica delle configurazioni critiche:**

- Con l'accesso non autorizzato, un attaccante può modificare le configurazioni critiche del sistema, compromettendo la stabilità e la sicurezza del sistema stesso. Questo potrebbe includere la disabilitazione delle difese di sicurezza o l'apertura di backdoor per un accesso futuro.

# Compromissione della sicurezza

L'accesso non autorizzato e l'intercettazione delle credenziali sono solo il primo passo. Il malware può sfruttare le credenziali rubate per compiere ulteriori azioni dannose che compromettono gravemente la sicurezza del sistema:

- Installazione di software malevolo:
  - Utilizzando le credenziali acquisite, il malware può installare ulteriori software dannosi, come spyware, keylogger o botnet, ampliando il controllo dell'attaccante sul sistema.
- Esecuzione di comandi dannosi:
  - Il malware può eseguire comandi specifici per disabilitare i meccanismi di sicurezza del sistema, cancellare file critici, o criptare dati sensibili in un attacco di ransomware.
- Compromissione della macchina virtuale e dei dati:
  - Se il sistema compromesso è parte di una macchina virtuale o di una rete più ampia, il malware può propagarsi ad altre macchine virtuali, aumentando l'estensione del danno. I dati sensibili presenti sul sistema possono essere rubati, distrutti o resi inaccessibili.

# Diagramma di flusso



# Analisi ad alto livello

## Infezione del sistema:

- Fase iniziale dell'attacco.
- Vettori comuni: email di phishing, download da siti infetti, vulnerabilità del sistema o delle applicazioni.

## Esecuzione del malware:

- Il malware viene eseguito nel sistema.
- Modalità di esecuzione: automatica (script/programmi avviati all'accensione) o manuale (utente clicca su file eseguibile apparentemente innocuo).

## Creazione di GINA.dll malevolo:

- Il malware sostituisce GINA.dll con una versione malevola.
- Scopo: Intercettare le credenziali di accesso degli utenti.

# Analisi ad alto livello

## Ottenimento della persistenza:

- Il malware implementa meccanismi per mantenere l'accesso al sistema.
- Modifica delle chiavi di registro per avviare il malware all'accensione del sistema (chiave: "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon").

## Furto delle credenziali di accesso:

- Il malware utilizza la versione malevola di GINA.dll per intercettare le credenziali degli utenti.
- Le credenziali rubate vengono inviate agli attaccanti per ottenere accesso non autorizzato al sistema e ad altre risorse protette.

# Conclusioni

## Analisi iniziale:

- Analisi del file eseguibile Malware\_Build\_Week\_U3.
- Identificazione di parametri e variabili nella funzione Main().
- Esplorazione delle sezioni principali del file e delle librerie importate.
- Formulazione di ipotesi preliminari sulle funzionalità del malware.

## Approfondimento tecnico:

- Analisi della funzione alla locazione 00401021:
  - Studio del passaggio dei parametri.
  - Traduzione delle istruzioni (00401027-00401029) in costrutti C.
- Valutazione del parametro “ValueName” alla locazione 00401047:
- Analisi delle routine tra 00401080 e 00401128:
  - Determinazione del valore del parametro "ResourceName".
  - Comprensione delle funzionalità implementate dal malware.

# Conclusioni

## Esecuzione in ambiente controllato:

- Utilizzo di Process Monitor per osservare le modifiche al sistema:
  - Attività sul registro di Windows.
  - Modifiche al file system (creazione di chiavi di registro e modifiche delle cartelle).

## Esplorazione dell'uso di GINA.dll:

- Analisi dell'impatto della sostituzione di GINA.dll con una versione malevola.
- Identificazione del rischio di intercettazione delle credenziali di accesso.
- Osservazione dei meccanismi per ottenere persistenza nel sistema.

## Conclusioni generali:

- Il malware utilizza vari meccanismi per:
  - Infettare il sistema.
  - Eseguire codice malevolo.
  - Ottenere persistenza.
  - Rubare informazioni sensibili.
- Le informazioni ottenute costituiscono una base solida per sviluppare strategie di difesa e mitigazione.
- Miglioramento della sicurezza complessiva del sistema contro tali minacce.



**Thank you!**



# Our Team

