

BUILD WEEK III

MALWARE ANALYSIS



NET.
REBELS.

TRACCIA

GIORNO 1

PARTE 1

Con riferimento al file eseguibile Malware_Build_Week_U3, rispondere ai seguenti quesiti utilizzando i tool e le tecniche apprese nelle lezioni teoriche:

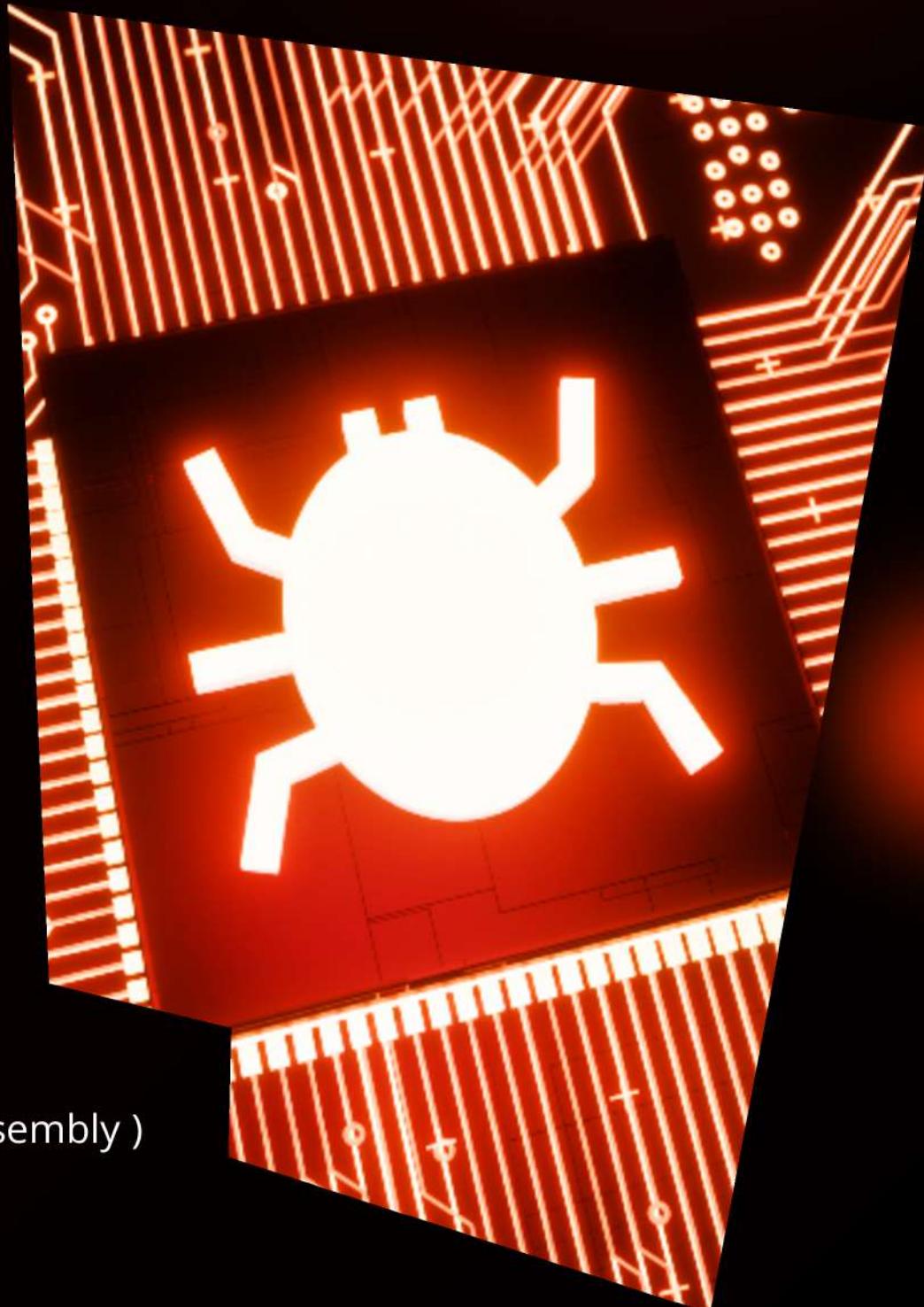
1. Quanti parametri sono passati alla funzione Main()?
2. Quante variabili sono dichiarate all'interno della funzione
3. Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate
4. Quali librerie importa il Malware? Per ognuna delle librerie importate, 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.

PARTE 2

Con riferimento al Malware in analisi, spiegare:

1. Lo scopo della funzione chiamata alla locazione di memoria 00401021
2. Come vengono passati i parametri alla funzione alla locazione 00401021 ;
3. Che oggetto rappresenta il parametro alla locazione 00401017
4. Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029. se serve, valutare un'altra o altre due righe assembly)
5. Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C
6. Valutate ora la chiamata alla locazione 00401047 , qual è il valore del parametro « ValueName»?

Nel complesso delle due funzionalità appena viste, spiegate quale funzionalità sta implementando il Malware in questa sezione.

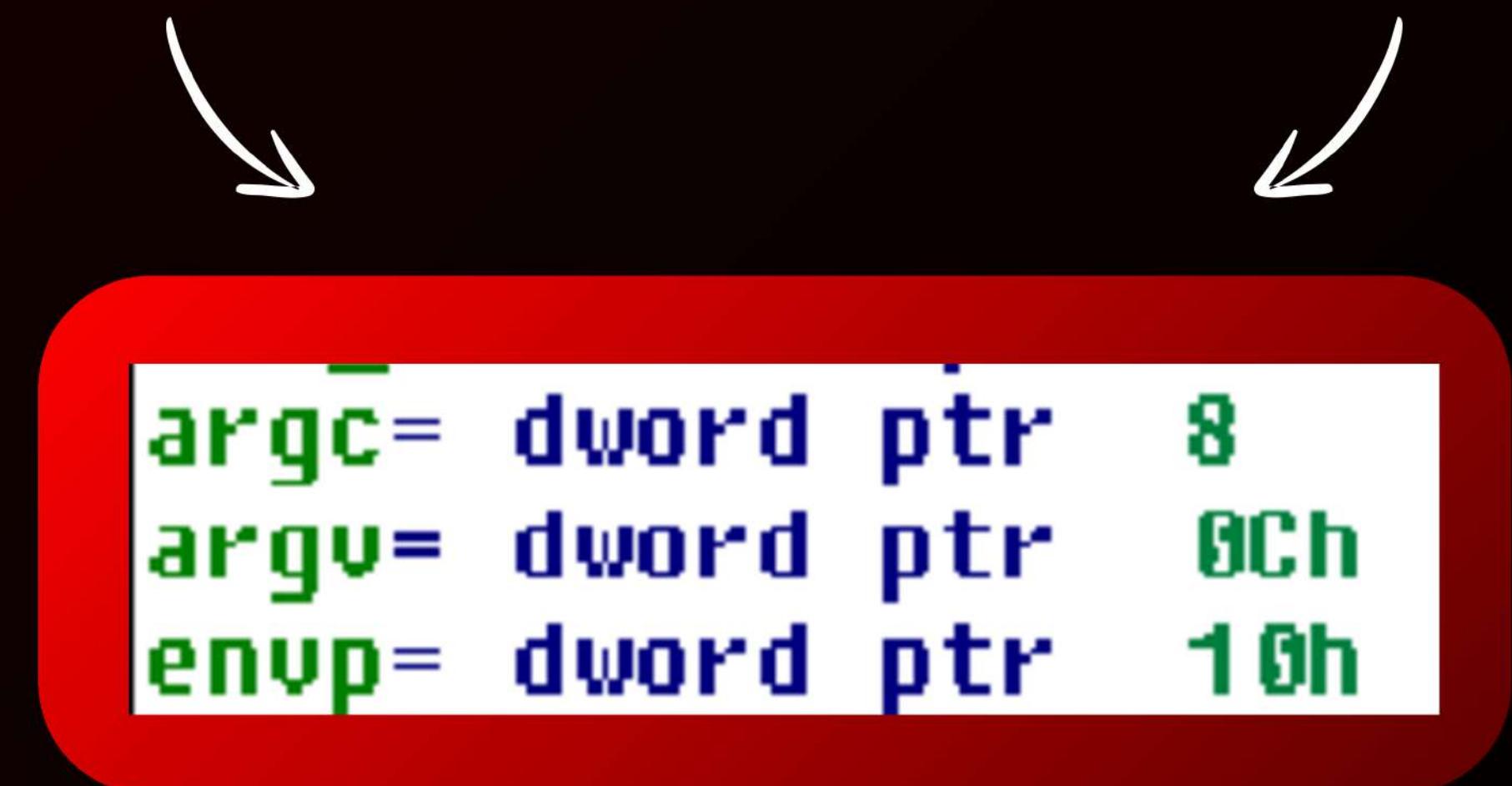


PARTE 1

1. Quanti parametri sono passati alla funzione Main()

I parametri sono i dati forniti a una funzione o a una subroutine al momento della sua chiamata. Essi rappresentano gli input della funzione e influenzano il comportamento della funzione stessa. Una volta che la funzione termina, i parametri vengono generalmente rimossi e non sono più accessibili. I parametri si trovano ad un **offset positivo rispetto ad EBP**.

Come è possibile notare nella figura i parametri sono **3**:



PARTE 1

2. Quante variabili sono dichiarate all'interno della funzione?

Le variabili sono spazi di memoria che possono contenere valori e sono utilizzate per memorizzare dati temporanei durante l'esecuzione del programma.

Esse possono essere utilizzate in vari punti del programma per diverse operazioni.

La durata delle variabili dipende dal loro ambito di visibilità. Le variabili locali a una funzione esistono solo durante l'esecuzione di quella funzione, mentre le variabili globali esistono per tutta la durata del programma. **Le variabili sono ad un offset negativo rispetto al registro EBP.**

Come è possibile notare nella figura le variabili sono **5**:

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

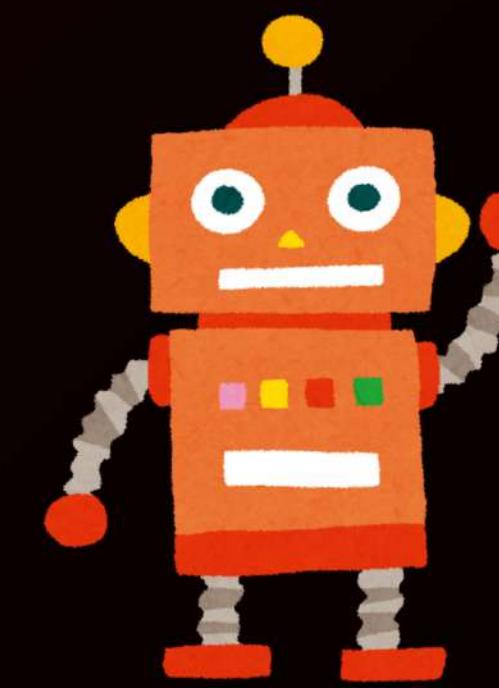
PARTE I

3. Quali sezioni sono presenti all'interno del file eseguibile?

Come prima cosa avviamo la nostra macchina “**Window 7 - malware analysis**”, poi clicchiamo sulla cartella che si trova nel Desktop con il nome di “**Software Malware Analysis**” e da qui apriamo il software di nostro interesse che è **CFF Explorer**.

Una volta aperto è possibile da qui caricare un file per la seguente scansione, quindi clicchiamo sulla cartellina in alto a destra e cerchiamo il file di nostro interesse; che troviamo nella cartella “**MALWARE**” con il nome di «**Malware_Build_Week_U3**».

Ora ci spostiamo nella sezione “**Section Headers [x]**” e qui possiamo trovare le sezioni (blocchi di dati distinti che contengono vari tipi di informazioni necessarie per l'esecuzione del programma).



The screenshot shows the CFF Explorer VIII interface. On the left, there's a tree view of file headers: File, Dos Header, Nt Headers, File Header, Optional Header, Data Directories [x], and Section Headers [x]. The 'Section Headers [x]' item is highlighted with a blue selection bar. On the right, there's a table titled 'Malware_Build_Week_U3.exe' showing section details:

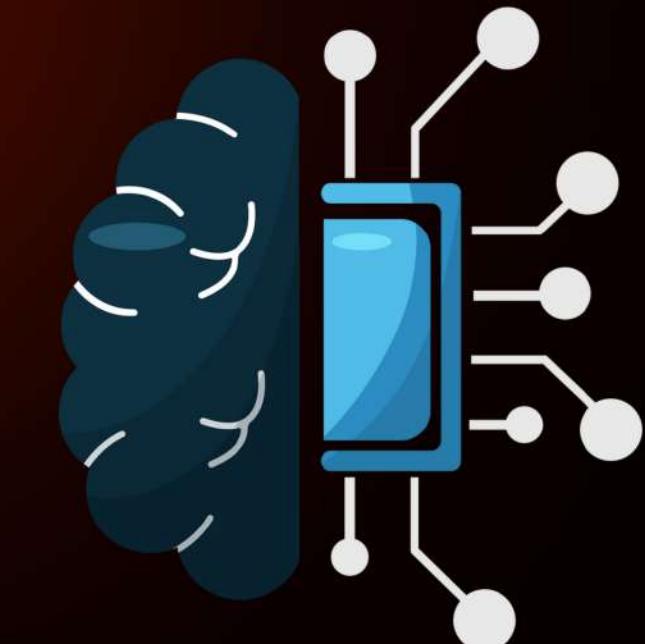
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	00005646	00001000	00006000	00001000	00000000
.rdata	000009AE	00007000	00001000	00007000	00000000
.data	00003EA8	00008000	00003000	00008000	00000000
.rsrc	00001A70	0000C000	00002000	0000B000	00000000

PARTE I

3. Quali sezioni sono presenti all'interno del file eseguibile?

- **.txt** : contiene il codice eseguibile del programma. Questa è la parte del file che contiene le istruzioni macchina che il processore esegue. È marcata come leggibile ed eseguibile, ma non scrivibile, per garantire la protezione del codice. Questa sezione è **cruciale per il funzionamento di qualsiasi programma**, poiché contiene le istruzioni che il processore eseguirà.
- **.rdata** : (**read-only data**) contiene dati costanti e altre informazioni di sola lettura che non devono essere modificate durante l'esecuzione del programma. Garantisce che **i dati non possano essere modificati a runtime**, migliorando la sicurezza e l'integrità del programma.

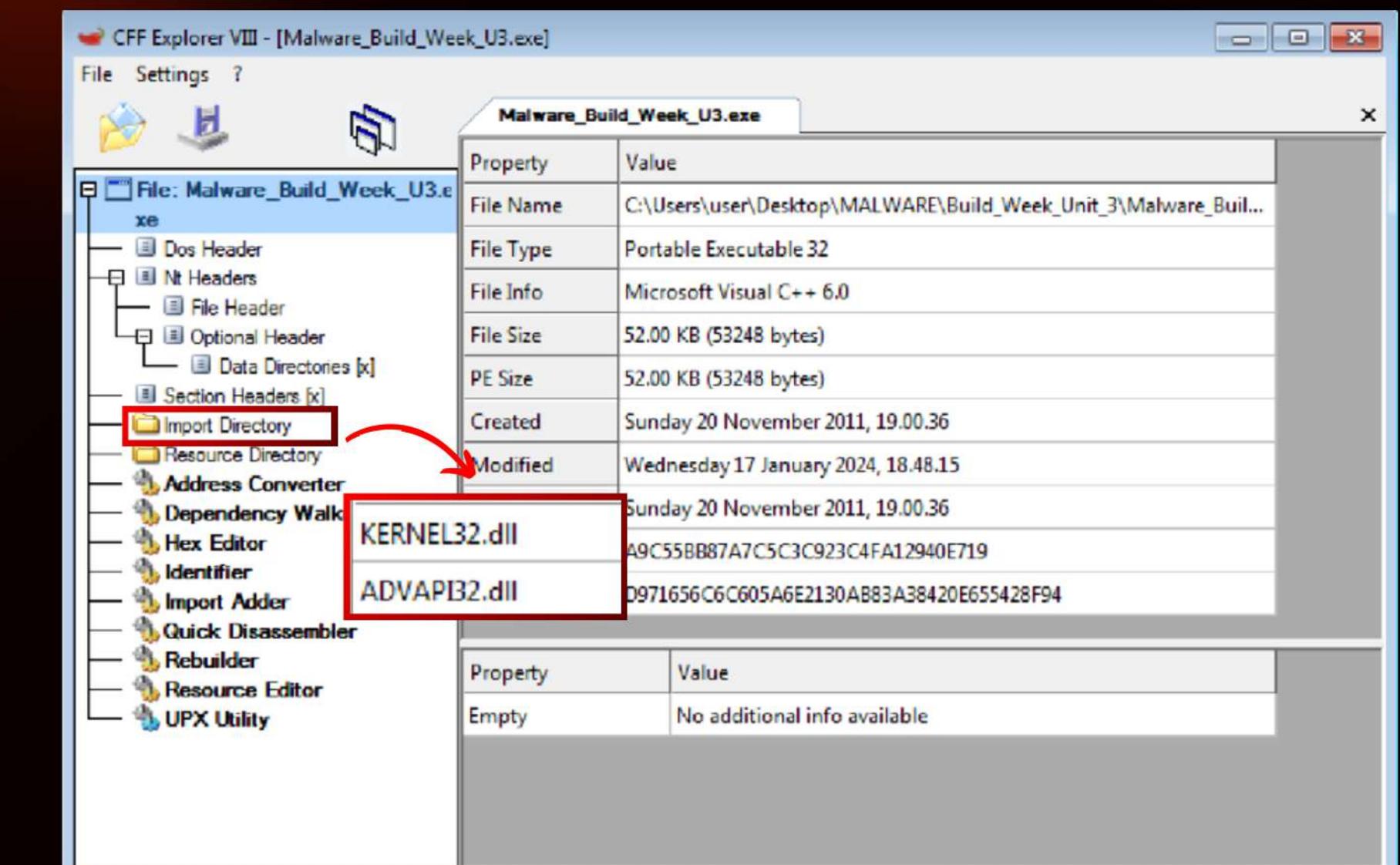
- **.data** : contiene dati globali e statici che il programma può leggere e scrivere durante l'esecuzione. Include variabili inizializzate. **I dati sono accessibili e modificabili durante l'esecuzione del programma**, e l'inizializzazione avviene all'avvio del programma
- **.rsrc** : (**Resources**) è una sezione speciale in file eseguibili Windows che contiene risorse incorporate utilizzate dall'applicazione. Le risorse possono includere una varietà di elementi come icone, cursori, stringhe di testo, menu, dialoghi, e altre informazioni necessarie **per l'interfaccia utente e la funzionalità dell'applicazione**



PARTE I

4. Quali librerie importa il Malware? E quali funzionalità?

- **KERNEL32.DLL:** È essenziale per molte operazioni di base del **sistema operativo**. Necessaria per la gestione delle risorse del sistema, come la memoria, i processi e i thread, l'accesso ai file e la gestione degli input/output. È una delle librerie più fondamentali di Windows e viene utilizzata da quasi tutte le applicazioni per eseguire operazioni comuni come la **gestione della memoria, l'accesso ai file e la comunicazione tra processi** (ES: CreateFile, ReadFile, WriteFile, CreateProcess, Sleep, GetTickCount). La gestione efficiente della memoria è cruciale per la stabilità e le prestazioni del sistema operativo e delle applicazioni.
- **ADVAPI32.DLL:** è una libreria di collegamento dinamico (DLL) fondamentale in Windows che fornisce una serie di API per la gestione avanzata delle **funzionalità di sistema**. Il nome ADVAPI32 sta per "Advanced Windows API 32-bit," indicando che si tratta di una **DLL a 32 bit** con funzioni avanzate per l'interazione con il sistema operativo. Le sue funzionalità sono la **gestione della sicurezza e controllo degli Accessi, gestione dei servizi di Windows, registrazione e manipolazione del registro di sistema, gestione degli eventi di sistema**.



PARTE I

4. Quali librerie importa il Malware? E quali funzionalità?

Se un malware ha accesso alle librerie ADVAPI32.dll e KERNEL32.dll, potrebbe sfruttare una vasta gamma di funzionalità per compromettere un sistema Windows. Queste librerie forniscono API che il malware può utilizzare per eseguire operazioni di sistema avanzate, manipolare la configurazione di sistema, gestire la sicurezza e interagire con il registro di sistema.

1. Controllo dei Permessi e Sicurezza

- **Manipolazione dei Token di Accesso:** Utilizzando ADVAPI32.dll, questo potrebbe permettere al malware di elevare i privilegi o ottenere accesso a risorse protette.
- **Modifica delle ACL:** Il malware può modificare le liste di controllo degli accessi (ACL) sui file e sulle risorse di sistema per ottenere accesso non autorizzato

2. Gestione dei Servizi di Windows

- **Creazione e Manipolazione dei Servizi:** con ADVAPI32.dll, il malware può creare e gestire servizi di Windows. Questo potrebbe essere usato per creare servizi che eseguono il malware all'avvio del sistema, mascherandosi come un servizio legittimo.
- **Controllo dei Servizi:** usate per controllare o rimuovere servizi esistenti, facilitando la persistenza del malware o interferendo con le operazioni di sicurezza del sistema.

PARTE I

4. Quali librerie importa il Malware? E quali funzionalità?

3. Manipolazione del Registro di Sistema

- **Lettura e Scrittura nel Registro:** il malware può leggere e scrivere chiavi e valori nel registro di sistema. Questo può essere usato per persistere sul sistema o per raccogliere informazioni sensibili.
- **Modifica delle Chiavi di Avvio:** Il malware può aggiungere o modificare chiavi nel registro di sistema per assicurarsi che venga eseguito ogni volta che il sistema si avvia

4. Gestione degli Eventi di Sistema

- **Creazione di Eventi e Sincronizzazione:** il malware può gestire eventi di sistema per sincronizzare l'esecuzione del codice o per comunicare tra diversi processi o thread.

5. Interazione con il File System e le API di Sistema

- **Gestione dei File:** Sebbene molte operazioni sui file siano gestite tramite KERNEL32.dll, il malware può utilizzare funzioni per manipolare file, nascondere i propri componenti, o rubare dati.
- **Gestione della Memoria:** utilizzate per manipolare la memoria dei processi, iniettare codice o alterare il comportamento di altre applicazioni.

6. Esecuzione e Iniezione di Codice

- **Iniezione di Codice:** Il malware può utilizzare funzioni di KERNEL32.dll per iniettare codice in altri processi.

PARTE 2

1. Lo scopo della funzione chiamata alla locazione di memoria **00401021**

Per prima cosa avviamo la macchina windows e poi apriamo il malware che dobbiamo analizzare con il tool **OllyDbg**.

Alla locazione di memoria **00401021**, il codice sta effettuando una chiamata alla funzione **RegCreateKeyExA** tramite la libreria **ADVAPI32.dll**. Questa funzione viene utilizzata per creare o aprire una chiave di registro nel Windows Registry.

Scopo della Funzione: La funzione **RegCreateKeyExA** ha lo scopo di creare una nuova chiave di registro o di aprirne una esistente. Nel contesto di un malware, questo potrebbe essere usato per stabilire la persistenza, memorizzare configurazioni, o tenere traccia di informazioni utili per il malware stesso.

The screenshot shows the assembly code for a portion of the malware's memory dump. The code is as follows:

```

00401004: 6A 00          PUSH 0
00401006: 8D45 FC      LEA EAX,DWORD PTR SS:[EBP-4]
00401009: 50             PUSH EAX
0040100A: 6A 00          PUSH 0
0040100C: 68 3F000F00    PUSH 0F003F
00401011: 6A 00          PUSH 0
00401013: 6A 00          PUSH 0
00401015: 6A 00          PUSH 0
00401017: 68 54804000    PUSH Malware_.00408054
0040101C: 68 02000080    PUSH 80000002
00401021: FF15 04704000  CALL DWORD PTR DS:[&ADVAPI32.RegCreateKeyExA]
00401027: 85C0             TEST EAX,EAX
00401029: 74 07             JE SHORT Malware_.00401032
0040102B: B8 01000000    MOU EAX,1

```

The instruction at address **00401021** is highlighted in red and labeled **call ds:RegCreateKeyExA**. To the right of the assembly code, a callout box displays the parameters for the **RegCreateKeyExA** function:

```

pDisposition = NULL
pHandle
pSecurity = NULL
Access = KEY_ALL_ACCESS
Options = REG_OPTION_NON_VOLATILE
Class = NULL
Reserved = 0
Subkey = "SOFTWARE\Microsoft\Windows"
hKey = HKEY_LOCAL_MACHINE
RegCreateKeyExA

```

Below the assembly code, the label **.text:00401021** is shown in blue.

PARTE 2

2. Come vengono passati i parametri alla funzione alla locazione 00401021

I parametri alla funzione alla locazione **00401021** vengono passati tramite lo stack come in figura:

The screenshot shows assembly code in a debugger. The assembly code includes several **PUSH** instructions followed by a **CALL** instruction to **RegCreateKeyExA**. To the right of the assembly, the parameters are listed with their corresponding memory addresses and values.

Parameter	Value	Type
pDisposition	NULL	pointer
pHandle	NULL	pointer
pSecurity	NULL	pointer
Access	KEY_ALL_ACCESS	integer
Options	REG_OPTION_NON_VOLATILE	integer
Class	NULL	pointer
Reserved	0	integer
Subkey	"SOFTWARE\Microsoft\Windows"	string
hKey	HKEY_LOCAL_MACHINE	pointer
RegCreateKeyExA	(highlighted in red)	function

Si possono notare che i parametri vengono passati alla funzione utilizzando lo stack. Prima di effettuare la chiamata alla funzione (**call**), i parametri vengono spinti (**push**) nello stack in ordine inverso rispetto a come la funzione si aspetta di riceverli.

I parametri passati a **RegCreateKeyExA** sono i seguenti:

hKey: Questo è l'handle della chiave del registro di partenza.

- **Parametro**: 80000002
- **Significato**: Questo valore corrisponde a HKEY_LOCAL_MACHINE, uno dei principali "root keys" nel registro di Windows.

PARTE 2

2. Come vengono passati i parametri alla funzione alla locazione 00401021

SubKey: Puntatore a una stringa che specifica il nome della sottochiave da creare o aprire.

- **Parametro:** Malware_.00408054
- **Significato:** Questo è l'indirizzo di memoria che contiene la stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon". Questa sottochiave viene creata sotto HKEY_LOCAL_MACHINE.

Reserved: Questo parametro è riservato e deve essere zero.

- **Parametro:** push 0

Class: Puntatore a una stringa che definisce la classe della chiave (opzionale, può essere NULL).

- **Parametro:** push 0
- **Significato:** In questo caso, è passato come NULL, quindi non viene specificata alcuna classe per la chiave.

Options: Opzioni per la chiave. Questo valore può influenzare il comportamento della chiave.

- **Parametro:** push 0x000003F
- **Significato:** In questo caso, il valore passato è REG_OPTION_NON_VOLATILE, che significa che la chiave è memorizzata nel registro e non andrà persa dopo un riavvio del sistema.

PARTE 2

2. Come vengono passati i parametri alla funzione alla locazione 00401021

Access: Specifica i diritti di accesso desiderati per la chiave.

- **Parametro:** push 0xF003F
- **Significato:** Il valore 0xF003F rappresenta KEY_ALL_ACCESS, che concede tutti i diritti di accesso alla chiave.

pSecurity: Puntatore a una struttura che definisce il descrittore di sicurezza della chiave. Può essere NULL se si utilizza il descrittore predefinito.

- **Parametro:** push 0
- **Significato:** In questo caso, è passato come NULL, quindi si utilizza il descrittore di sicurezza predefinito.

pHandle: Puntatore a una variabile che riceverà l'handle della chiave aperta o creata.

- **Parametro:** push EAX
- **Significato:** Qui, EAX contiene l'indirizzo di una variabile che riceverà l'handle della chiave.

Registers (FPU)	
EAX	0018FE0C
ECX	00000000
EDX	0018FE30 ASCII "C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll"
EBX	0000000E
ESP	0018FDE8
EBP	0018FE10
ESI	004080B2 Malware_.004080B2
EDI	0018FE6D ASCII "_Week_U3.exe"
EIP	00401021 Malware_.00401021

pDisposition: Puntatore a una variabile che riceve un valore che indica se la chiave è stata creata o semplicemente aperta.

- **Parametro:** push 0
- **Significato:** In questo caso, è passato come NULL, quindi l'informazione non viene restituita

PARTE 2

3. Che oggetto rappresenta il parametro alla locazione 00401017

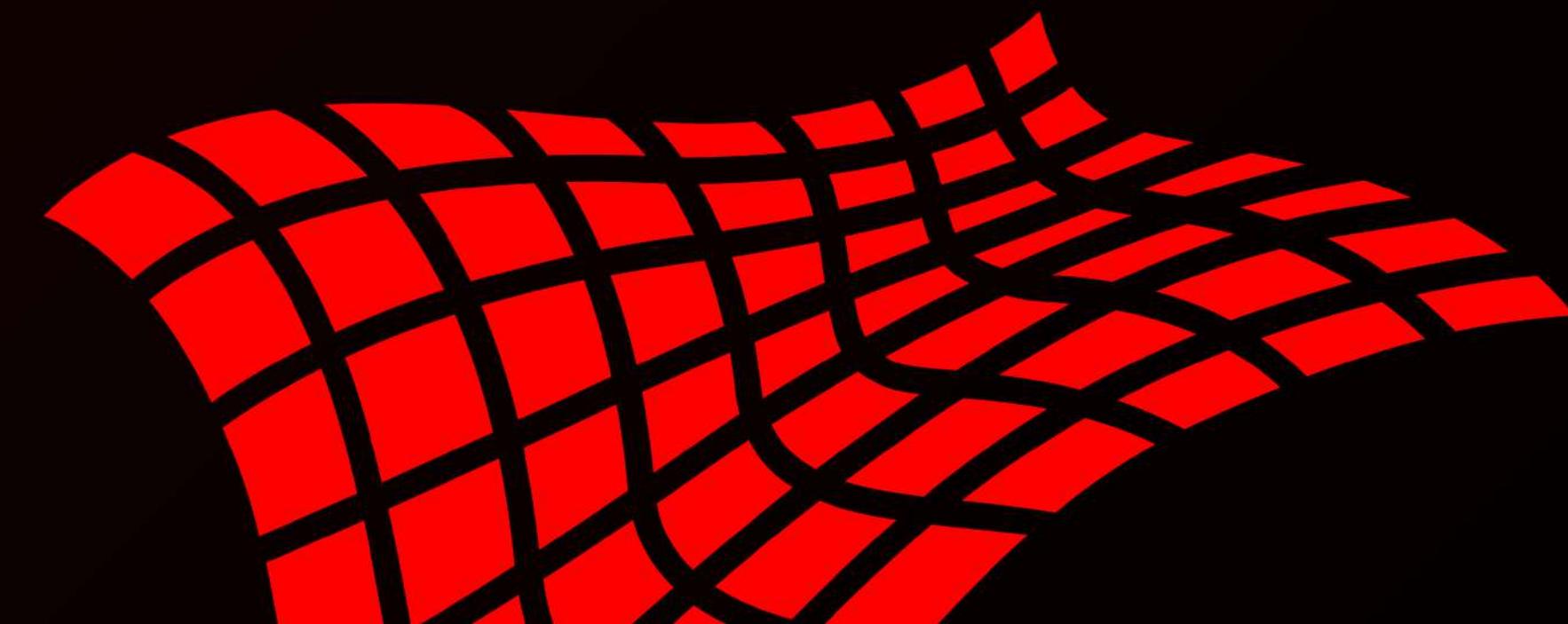
.text:00401017

push offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"

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

l'oggetto alla locazione 00401021 rappresenta il Path della chiave di registro windows (Winlogon) percorso specifico di una chiave utilizzata per la configurazione delle impostazioni di login.

Lo **scopo del Malware** può essere quello di andare a configurare a proprio piacimento la modalità di accesso: come l'impostazione di uno script di accesso automatico o estrarre le credenziali.



PARTE 2

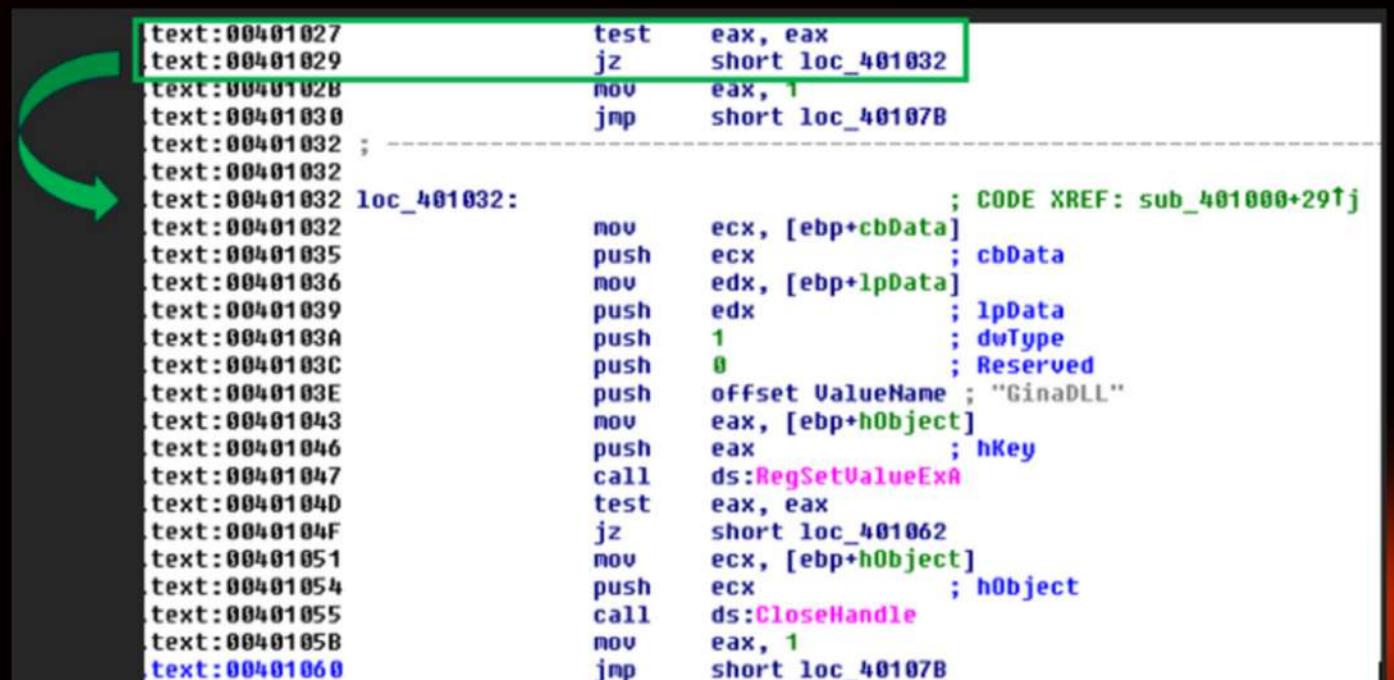
4. Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029

Le istruzioni Assembly presenti agli indirizzi di memoria 00401027 e 00401029 rappresentano un **costrutto IF**.

test eax, eax	jz short loc_401032:
<p>Questa istruzione esegue un'operazione di "AND" bit a bit tra il registro «EAX» e sé stesso. Essenzialmente, non cambia il valore di «EAX», ma aggiorna i flag del processore basati sul risultato. In particolare, il flag Zero (ZF) viene impostato su 1 se il risultato è zero, altrimenti è impostato su 0. Poiché stiamo eseguendo un AND tra lo stesso registro, il risultato sarà zero solo se «EAX» è zero.</p>	<p>Questa istruzione è un'istruzione di salto condizionato. «JZ» (Jump if Zero) fa sì che il salto avvenga solo se il flag Zero (ZF) è impostato a 1. Dopo l'istruzione ««test», il flag ZF è impostato a 1 se il registro «EAX» è zero. Quindi, se «EAX» è zero, l'esecuzione del programma salterà all'indirizzo «loc_401032». Se EAX» non è zero, l'esecuzione continuerà con l'istruzione successiva.</p>

Funzione loc_401032

Se il salto condizionale avviene il programma si sposta all'indirizzo di memoria 401032 che è l'inizio di una funzione.



PARTE 2

5. Tradurre il codice Assembly nel corrispondente costrutto C

Come già detto nella slide precedente ci troviamo dinanzi ad un costrutto IF. Osservando le istruzioni possiamo notare la chiamata ad una funzione RegCreateKeyExA.

Il valore di ritorno di una funzione viene salvato nel registro «EAX». Questo ci fa capire che il costrutto IF va a verificare il risultato della funzione RegCreateKeyExA. Se la funzione è andata a buon fine allora viene effettuato un salto alla posizione 401032.

Scrivendo dello pseudo codice In C avremo qualcosa del genere:

```
if ( esito_funzione == 0 )
{
    //Codice in posizione 401032
}
```

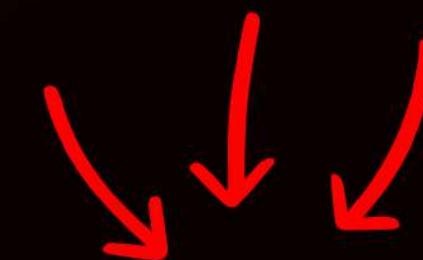


PARTE 2

6. Locazione 00401047 , qual è il valore del parametro « ValueName »?

Il parametro **ValueName** si trova all'indirizzo di memoria 0040103E, dove è presente un'istruzione di push che passa un offset in memoria. L'offset in questione punta a una stringa il cui valore è "GinaDLL".

GinaDLL (Graphical Identification and Authentication DLL) è un componente del sistema operativo Windows utilizzato nelle versioni precedenti a Windows Vista. Si tratta di una DLL (Dynamic-Link Library) che gestisce l'interfaccia grafica per l'autenticazione degli utenti. GinaDLL è responsabile per la presentazione delle schermate di login e logout, la gestione dei dialoghi di cambio password, e altre funzioni di autenticazione e identificazione. Le personalizzazioni di GINA consentono di modificare o estendere il comportamento di autenticazione predefinito del sistema operativo.



.text:0040103E

push offset ValueName ; "GinaDLL"

PARTE 2

7. Funzionalità implementate dal Malware

Il malware utilizza la funzione **RegCreateKeyExA** per aprire la chiave di registro **SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon**. Questa chiave è fondamentale, poiché contiene le configurazioni utilizzate dal sistema operativo Windows per gestire il processo di login degli utenti.

Successivamente, il malware utilizza la funzione **RegSetValueExA** per impostare un valore chiamato **GinaDLL all'interno di questa chiave di registro**. GinaDLL (Graphical Identification and Authentication DLL) è una componente che gestisce l'interfaccia grafica di autenticazione degli utenti, come le schermate di login e logout.

L'azione di modificare il valore GinaDLL è una tecnica utilizzata dai malware per sostituire il file DLL di autenticazione predefinito di Windows con un file dannoso controllato dagli attaccanti. Questo consente al malware di intercettare le credenziali di accesso degli utenti, ottenere privilegi elevati o persino creare backdoor per un accesso non autorizzato al sistema.

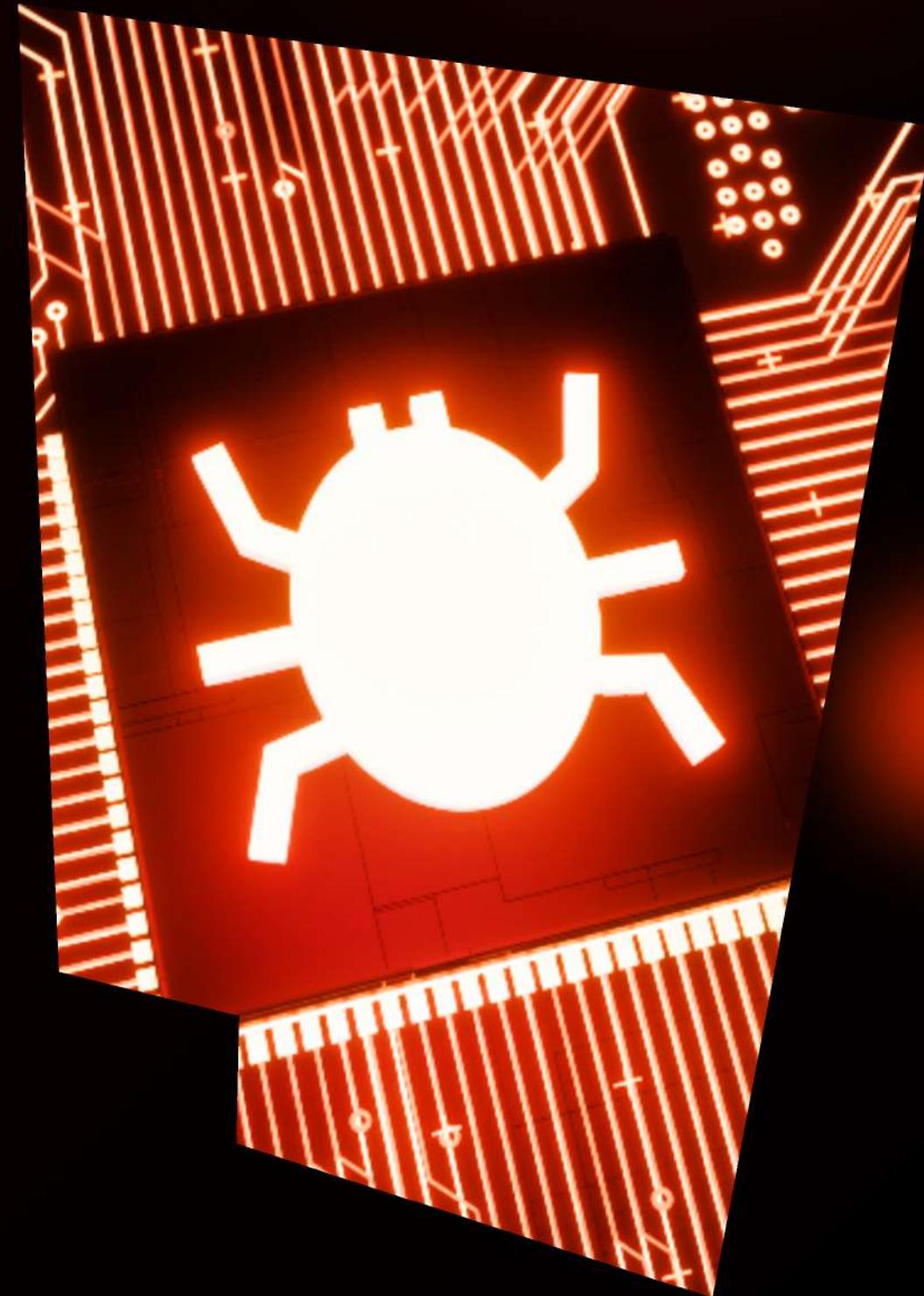
TRACCIA GIORNO 2

PARTE 1

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

1. Qual è il valore del parametro «`ResourceName` » passato alla funzione `FindResourceA()`;
2. Il susseguirsi delle chiamate di funzione che effettua il Malware in questa sezione di codice come abbiamo visto durante le lezioni teoriche. Che funzionalità sta implementando il Malware?
3. È possibile identificare questa funzionalità utilizzando l'analisi statica basica
4. In caso di risposta affermativa, elencare le evidenze a supporto.

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



TRACCIA GIORNO 2

PARTE 2

Preparate l'ambiente ed i tool per l'esecuzione del Malware (suggerimento: avviate principalmente Esercizio Giorno 2 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.

- Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware ? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda

Analizzate ora i risultati di Process Monitor (consiglio: utilizzate il filtro come in figura sotto per estrarre solo le modifiche apportate al sistema da parte del Malware). Fate click su «ADD» poi su « Apply » come abbiamo visto nella lezione teorica.

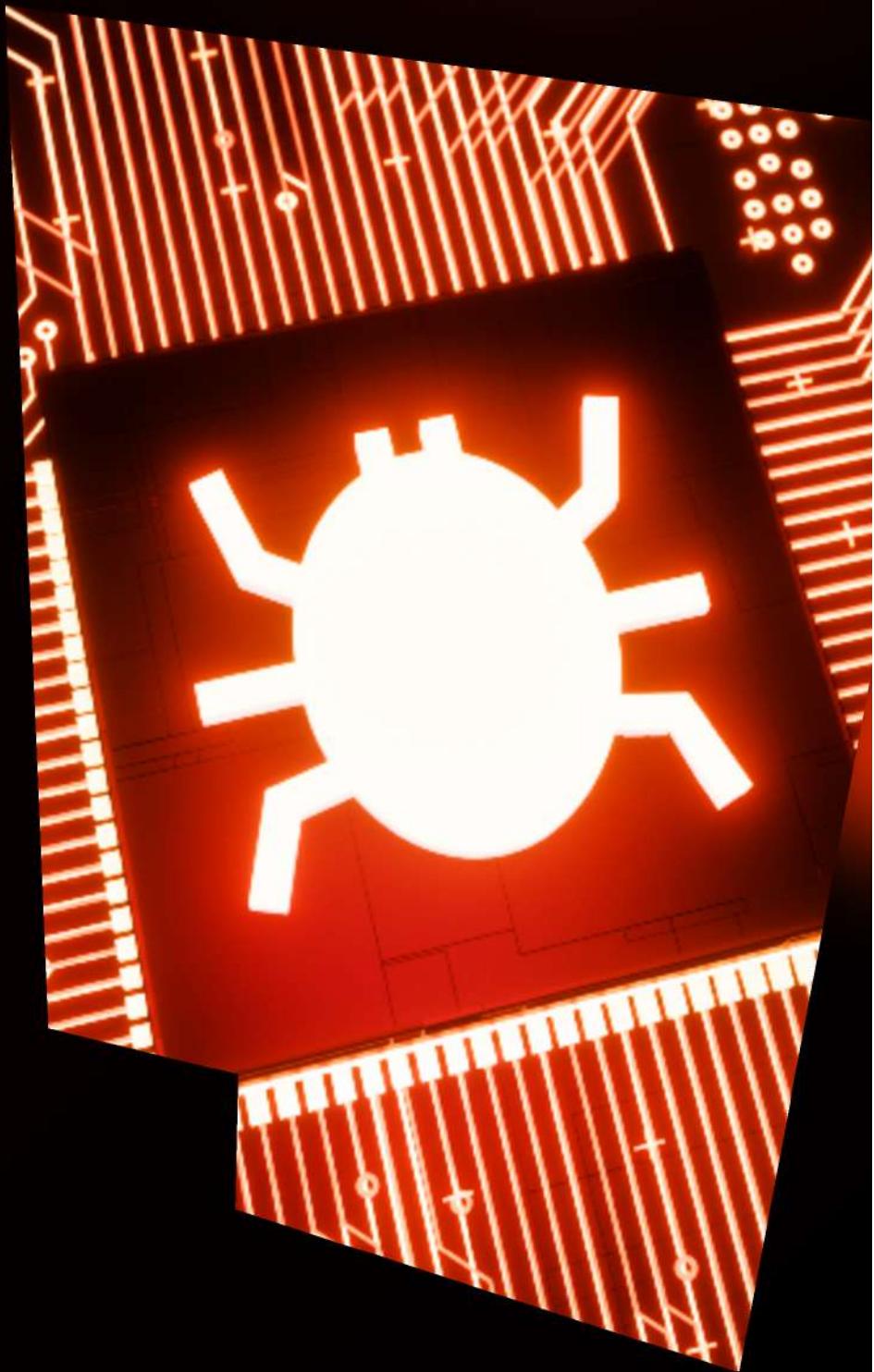
Filtrate includendo solamente l'attività sul **registro di Windows**

- Quale chiave di registro viene creata?
- Quale valore viene associato alla chiave di registro creata?

Passate ora alla visualizzazione dell'attività sul **FileSystem**

- 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 .



PARTE I

1. valore del parametro «**ResourceName** »

E' stata eseguita l'analisi di una porzione di codice malware combinando tecniche di analisi statica e dinamica avanzata, utilizzando due strumenti: **IDA Pro** e **OLLYDBG**. IDA Pro ci ha permesso di esplorare il codice in modo dettagliato senza eseguirlo, mentre OLLYDBG è stato utilizzato per impostare breakpoint e monitorare il comportamento del malware in esecuzione

Prima della chiamata alla prima funzione, il codice esegue una serie di operazioni per preparare l'ambiente di esecuzione. Queste operazioni includono il confronto di valori e salti condizionali per determinare se procedere con la ricerca della risorsa. Vengono impostati i parametri necessari, come **lpType**, **lpName** e **hModule**.

OLLYDBG ci ha permesso di impostare un breakpoint sulla funzione FindResourceA (prima funzione), in questo modo siamo riusciti ad **individuare il contenuto del parametro ResourceName**.

```
- ResourceType => "BINARY"
Malware_.00408038
ResourceName => "TGAD"

hModule
-FindResourceA
```

PARTE I

1. valore del parametro «ResourceName»

- **Registro ecx:** Prima di chiamare FindResourceA, il codice carica il valore di lpName nel registro ecx. In molti contesti, lpName contiene il nome della risorsa (cioè, il ResourceName) che la funzione deve cercare, che in questo caso è TGAD
- **Registro eax:** Il valore di lpType viene caricato nel registro eax. Questo indica il tipo di risorsa che la funzione FindResourceA deve cercare.
- **Registro edx:** Il valore di hModule, che rappresenta il modulo in cui cercare la risorsa, viene caricato nel registro edx.

.text:004010B8	mov	eax, lpType	
.text:004010BD	push	eax	; lpType
.text:004010BE	mov	ecx, lpName	
.text:004010C4	push	ecx	; lpName
.text:004010C5	mov	edx, [ebp+hModule]	
.text:004010C8	push	edx	; hModule

PARTE I

2. Che funzionalità sta implementando il Malware?

L'analisi del codice tra **00401080** e **00401128** mostra che il malware sta implementando una funzionalità legata alla gestione delle risorse di un modulo. Le operazioni principali includono:

- **FindResourceA**: Cerca una risorsa specifica all'interno del modulo.
- **LoadResource**: Carica la risorsa trovata in memoria.
- **LockResource**: Blocca la risorsa in memoria per l'utilizzo.

ANALISI DETTAGLIATA DEL FLUSSO DI CODICE

Cerca la Risorsa: La funzione FindResourceA() viene chiamata per cercare una risorsa specifica (identificata da lpType e lpName) all'interno del modulo specificato (hModule).

```
.text:004010C9          call    ds:FindResourceA
```

Verifica del Risultato: Il risultato di FindResourceA() viene memorizzato in hResInfo. Se hResInfo è diverso da 0 (indicando che la risorsa è stata trovata), il flusso continua.

```
.text:004010CF          mov     [ebp+hResInfo], eax
.text:004010D2          cmp     [ebp+hResInfo], 0
.text:004010D6          inz    short loc 4010DF
```

PARTE I

2. Che funzionalità sta implementando il Malware?

Carica la Risorsa: Se la risorsa è stata trovata, il malware chiama LoadResource() per caricare la risorsa in memoria.

```
* .text:004010E7          call    ds:LoadResource
```

Blocca la Risorsa: Successivamente, il malware chiama LockResource() per bloccare la risorsa in memoria, rendendola pronta per l'uso

```
* .text:004010FF          call    ds:LockResource
```

Verifica Finale: Il codice controlla se la risorsa è stata bloccata correttamente (Str non è 0). Se sì, continua il flusso.

```
* .text:00401105          mov     [ebp+Str], eax
* .text:00401108          cmp     [ebp+Str], 0
* .text:0040110C          jnz     short loc_401113
```

Dimensione della Risorsa: Infine, viene chiamata SizeofResource() per determinare la dimensione della risorsa caricata.

```
* .text:0040111B          call    ds:SizeofResource
```

Conclusione: Il malware sta implementando una funzionalità di gestione delle risorse che include la ricerca, il caricamento e il blocco di una risorsa specifica all'interno di un modulo. Questo tipo di comportamento è tipico di malware che cerca di caricare codice o dati nascosti per eseguire operazioni dannose

PARTE I

3. Identificare questa funzionalità utilizzando l'analisi statica basica

Sì, è possibile identificare questa funzionalità utilizzando l'analisi statica. Non è necessario eseguire il codice per capire che sta cercando di caricare e utilizzare una risorsa specifica. L'analisi delle chiamate a funzioni standard di Windows come **FindResourceA**, **LoadResource**, **LockResource**, e **SizeofResource** indica chiaramente un comportamento volto alla **gestione di risorse in memoria**.

4. Se sì, elencare le evidenze a supporto

Per confermare l'ipotesi che l'analisi statica è sufficiente per identificare le funzionalità del malware, abbiamo **utilizzato il CFF Explorer**, un tool che consente di esplorare i file eseguibili senza eseguirli. Questo strumento permette di analizzare ed visualizzare tutte le funzioni importate da librerie, come KERNEL32.dll e ADVAPI32.dll.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource

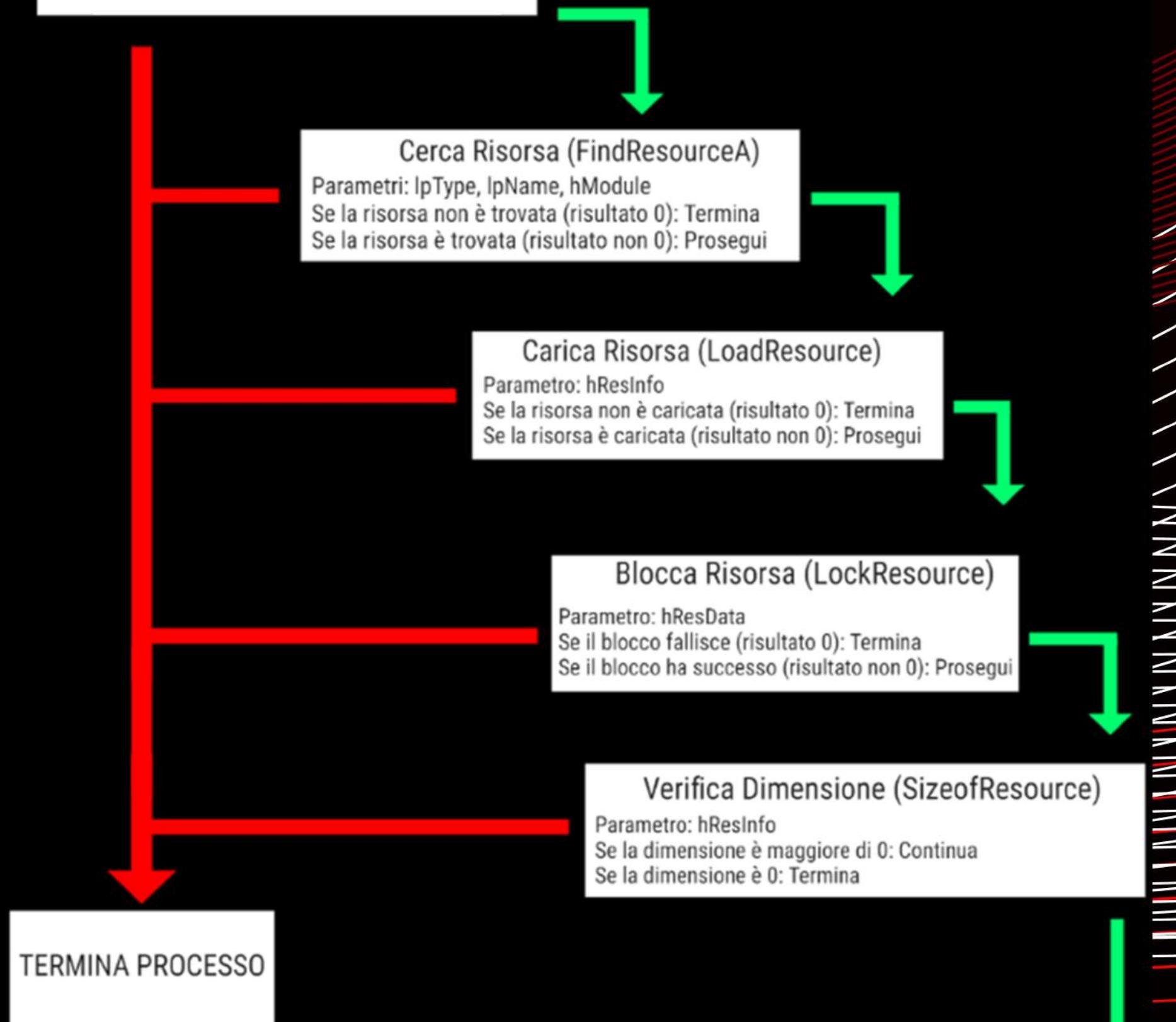
Le immagini mostrano chiaramente che il malware importa una serie di funzioni dalla libreria KERNEL32.dll, tra cui:



PARTE I

5. diagramma di flusso

INIZIALIZZAZIONE DEL PROGRAMMA



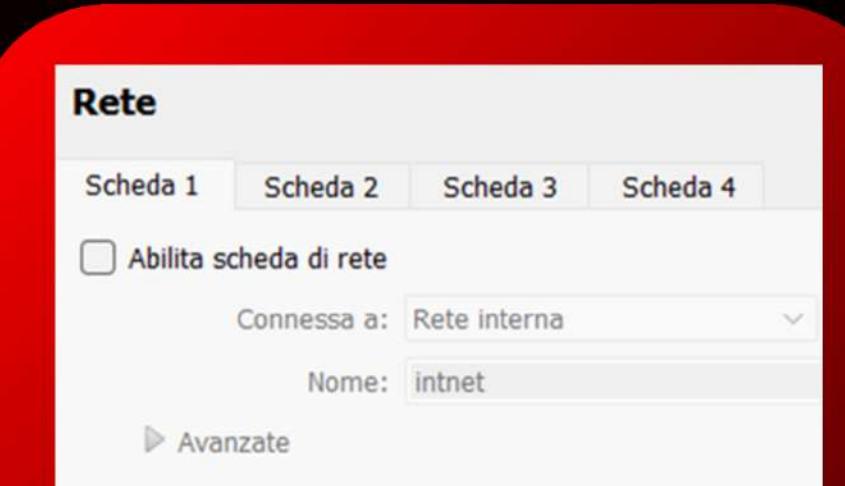
PARTE 2

Preparazione dell'ambiente - Isolamento

Isolare un ambiente di test di malware è fondamentale per garantire che il malware analizzato non si diffonda al di fuori del contesto di test controllato. Ciò è cruciale per prevenire danni accidentali ai sistemi produttivi e per evitare la contaminazione della rete.

1. Isolamento Macchina dalla rete interna e internet

Disabilitare la connessione di rete nell'ambiente di test impedisce al malware di comunicare con server esterni, esfiltrare dati o ricevere comandi da un server di comando e controllo (C&C). Ciò **mantiene il malware confinato**, impedendo interazioni con l'esterno e aumentando la sicurezza.



2. Disabilitare USB

Disabilitare le porte USB e l'accesso a dispositivi di archiviazione esterni **previene il trasferimento accidentale del malware** su altri sistemi. Questa misura evita anche che il malware possa sfruttare periferiche USB per diffondersi o esfiltrare dati.



PARTE 2

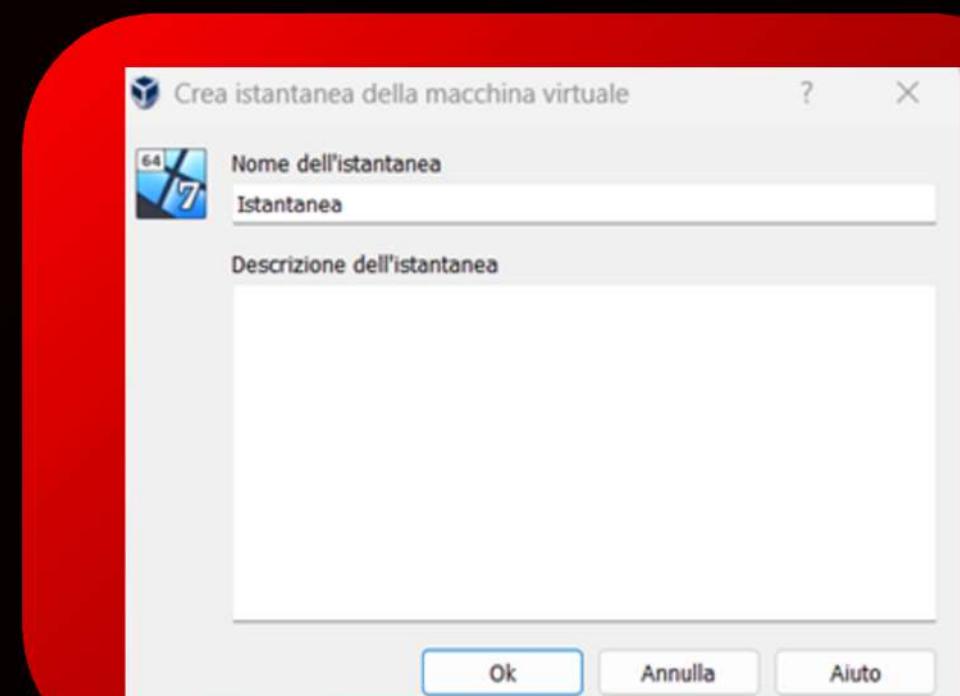
3. Disabilitare cartelle condivise

Impedire l'uso di cartelle condivise tra l'ambiente di test e altri sistemi o la macchina host **evita che il malware possa utilizzare queste risorse per diffondersi o accedere a file non autorizzati**. Questa pratica isola ulteriormente il malware, mantenendolo confinato all'interno dell'ambiente di test.



4. Creazione Istantanee

Le istantanee (snapshots) **consentono di salvare lo stato esatto dell'ambiente di test in un momento specifico**. Se il malware danneggia il sistema o modifica i file in modo indesiderato, l'istantanea permette di ripristinare rapidamente l'ambiente al suo stato originale, senza la necessità di reinstallare o riconfigurare il sistema da zero.

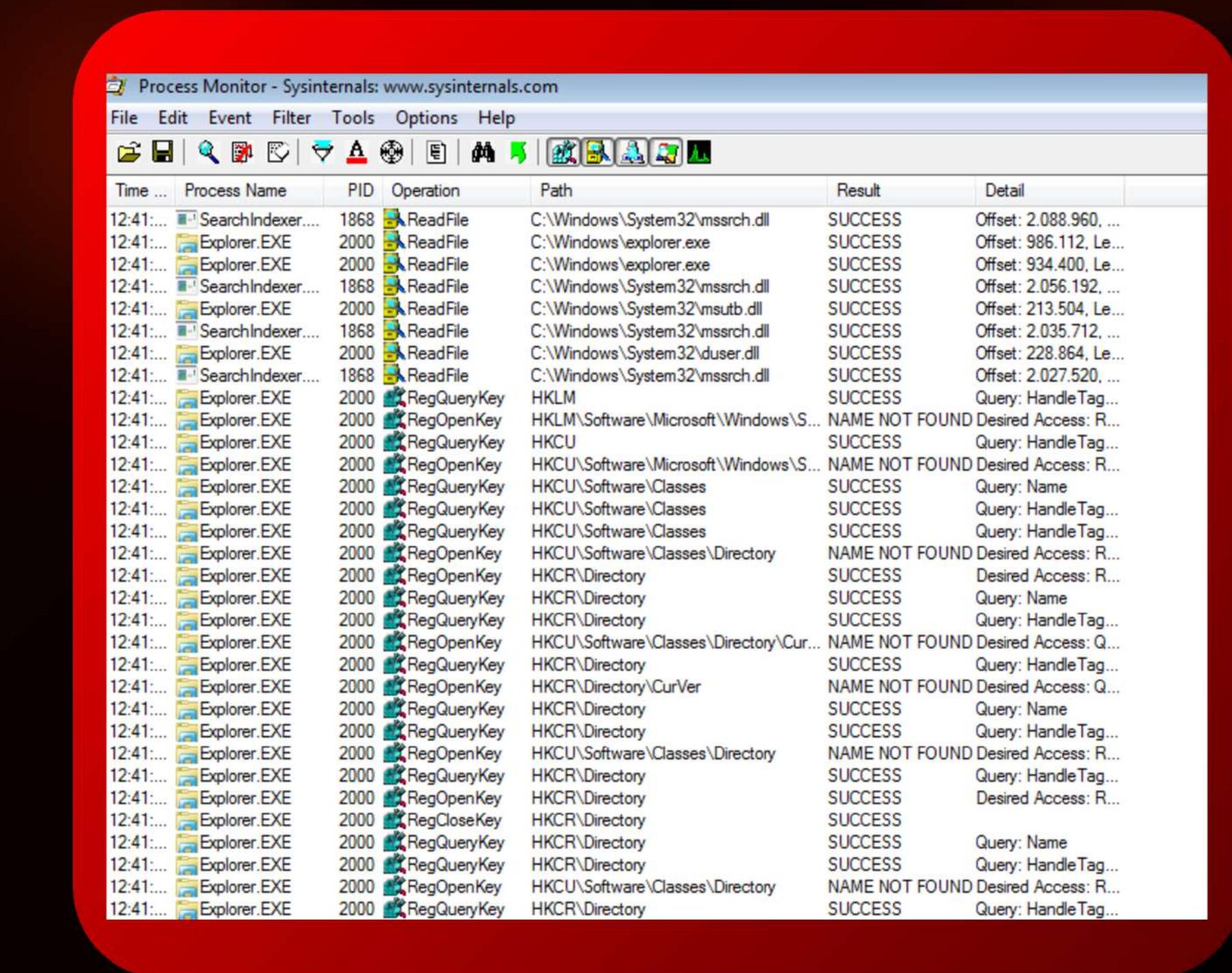


PARTE 2

Preparazione dell'ambiente - Process Monitor

Per analizzare il Malware dobbiamo usare un tool avanzato per il monitoraggio e l'analisi in tempo reale delle attività di sistema su Windows. Per questa esercitazione useremo lo strumento **Process Monitor**.

Il tool deve essere avviato prima dell'esecuzione del malware così da catturarne le attività.



The screenshot shows the Process Monitor application window. The title bar reads "Process Monitor - Sysinternals: www.sysinternals.com". The menu bar includes File, Edit, Event, Filter, Tools, Options, and Help. Below the menu is a toolbar with various icons. The main area is a table with columns: Time, Process Name, PID, Operation, Path, Result, and Detail. The table lists numerous entries from the system, primarily involving the "SearchIndexer" and "Explorer.EXE" processes. Most operations are "ReadFile" or "RegQueryKey" requests, often targeting paths like "C:\Windows\System32\msrch.dll", "HKLM\Software\Microsoft\Windows\S...", or registry keys under "HKCU\Software\Classes". The "Result" column shows mostly "SUCCESS" with some "NAME NOT FOUND Desired Access: R..." entries. The "Detail" column provides specific offsets and handle tags for each operation.

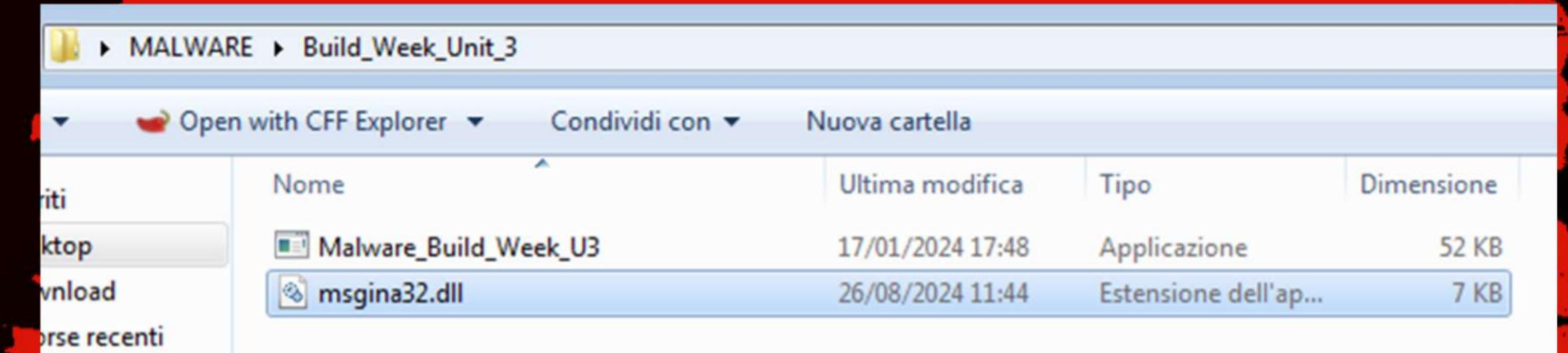
Time	Process Name	PID	Operation	Path	Result	Detail
12:41:...	SearchIndexer....	1868	ReadFile	C:\Windows\System32\mssrch.dll	SUCCESS	Offset: 2.088.960, ...
12:41:...	Explorer.EXE	2000	ReadFile	C:\Windows\explorer.exe	SUCCESS	Offset: 986.112, Le...
12:41:...	Explorer.EXE	2000	ReadFile	C:\Windows\explorer.exe	SUCCESS	Offset: 934.400, Le...
12:41:...	SearchIndexer....	1868	ReadFile	C:\Windows\System32\mssrch.dll	SUCCESS	Offset: 2.056.192, ...
12:41:...	Explorer.EXE	2000	ReadFile	C:\Windows\System32\msutb.dll	SUCCESS	Offset: 213.504, Le...
12:41:...	SearchIndexer....	1868	ReadFile	C:\Windows\System32\mssrch.dll	SUCCESS	Offset: 2.035.712, ...
12:41:...	Explorer.EXE	2000	ReadFile	C:\Windows\System32\duser.dll	SUCCESS	Offset: 228.864, Le...
12:41:...	SearchIndexer....	1868	ReadFile	C:\Windows\System32\mssrch.dll	SUCCESS	Offset: 2.027.520, ...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKLM	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKLM\Software\Microsoft\Windows\S...	NAME NOT FOUND	Desired Access: R...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKCU\Software\Microsoft\Windows\S...	NAME NOT FOUND	Desired Access: R...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKCU\Software\Classes\Directory	NAME NOT FOUND	Desired Access: R...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKCR\Directory	SUCCESS	Desired Access: R...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: Name
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKCU\Software\Classes\Directory\Cur...	NAME NOT FOUND	Desired Access: Q...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKCR\Directory\CurVer	NAME NOT FOUND	Desired Access: Q...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: Name
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKCU\Software\Classes\Directory	NAME NOT FOUND	Desired Access: R...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKCR\Directory	SUCCESS	Desired Access: R...
12:41:...	Explorer.EXE	2000	RegCloseKey	HKCR\Directory	SUCCESS	
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: Name
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: HandleTag...
12:41:...	Explorer.EXE	2000	RegOpenKey	HKCU\Software\Classes\Directory	NAME NOT FOUND	Desired Access: R...
12:41:...	Explorer.EXE	2000	RegQueryKey	HKCR\Directory	SUCCESS	Query: HandleTag...

PARTE 2

1. Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware ?

Completata la fase di preparazione possiamo **spostarci nella cartella che contiene il malware per eseguirlo**

Notiamo che all'interno della cartella dove è situato l'eseguibile del Malware è stato creato il file «**msgina32.dll**». Probabilmente questo file è la versione malevola della libreria **GinaDLL** che il malware carica per ottenere accesso e persistenza nel sistema.



PARTE 2

2. Quale chiave di registro viene creata?

3. Quale valore viene associato alla chiave di registro creata?

Torniamo su **Process Monitor** per analizzare le attività del malware. Visto che il sistema esegue innumerevoli attività andiamo ad applicare un filtro (per nome del processo) per visualizzare solo le operazioni eseguite dal malware.

The screenshot shows the Process Monitor interface. On the left is a large log window displaying numerous file operations (ReadFile, WriteFile, RegQueryKey, RegOpenKey) for processes like Explorer.exe, Searchindexer.exe, and others. On the right is a smaller window titled 'Filter' containing a table with four columns: Column, Relation, Value, and Action. The 'Column' column has a dropdown menu with 'Process Name' selected. The 'Relation' column has a dropdown menu with 'is' selected. The 'Value' column contains the text 'Malware_Build_Week_U3.exe'. The 'Action' column has a dropdown menu with 'Include' selected. A red arrow points from the bottom right towards the log window.

Column	Relation	Value	Action
Process Name	is	Malware_Build_Week_U3.exe	Include

PARTE 2

2. Quale chiave di registro viene creata?

3. Quale valore viene associato alla chiave di registro creata?

Attivando l'opzione «Show Registry Activity» di Process Monitor possiamo visualizzare le solo attività del malware sui registri di sistema. Così facendo notiamo l'uso di due funzioni interessanti:

RegCreateKey e RegSetValue.

 RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
 RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
 RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon
 RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL

RegCreateKey: Crea o apre una chiave di registro. In questo caso, la chiave Winlogon esiste già, quindi viene aperta.

RegSetValue: Qui si vede che il malware tenta di impostare per la voce GinaDLL il seguente valore:

C:\Users\user\Desktop\MALWARE_Build_Week_Unit_3\msgina32.dll

Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll |

PARTE 2

4. Quale chiamata di sistema ha modificato il contenuto della cartella

Per capire quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware dobbiamo attivare il **filtro «Show File System Activity»** di Process Monitor.

Malware_Build_... 1912	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_... 1912	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_... 1912	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
Malware_Build_... 1912	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll

Il malware chiama la funzione CreateFile per creare il file «msgina32.dll», mentre con la funzione WriteFile va a scriverne il contenuto. Finite queste operazioni viene invocata la funzione ColseFile per chiudere il file.

COMPORTAMENTO DEL MALWARE

Attraverso le informazioni raccolte possiamo ipotizzare che si tratta di un malware progettato per ottenere accesso non autorizzato al sistema e mantenere la persistenza modificando i meccanismi di autenticazione di Windows.

Per ottenere questo risultato, il malware tenta di modificare la voce di registro GinaDLL situata sotto la chiave di registro:

HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon

La modifica consiste nel puntare a una DLL personalizzata chiamata msgina32.dll. La DLL malevola viene creata tramite funzioni di File System nel path del malware, che si trova nel percorso: C:\Users\user\Desktop\MALWARE_Build_Week_Unit_3\msgina32.dll.

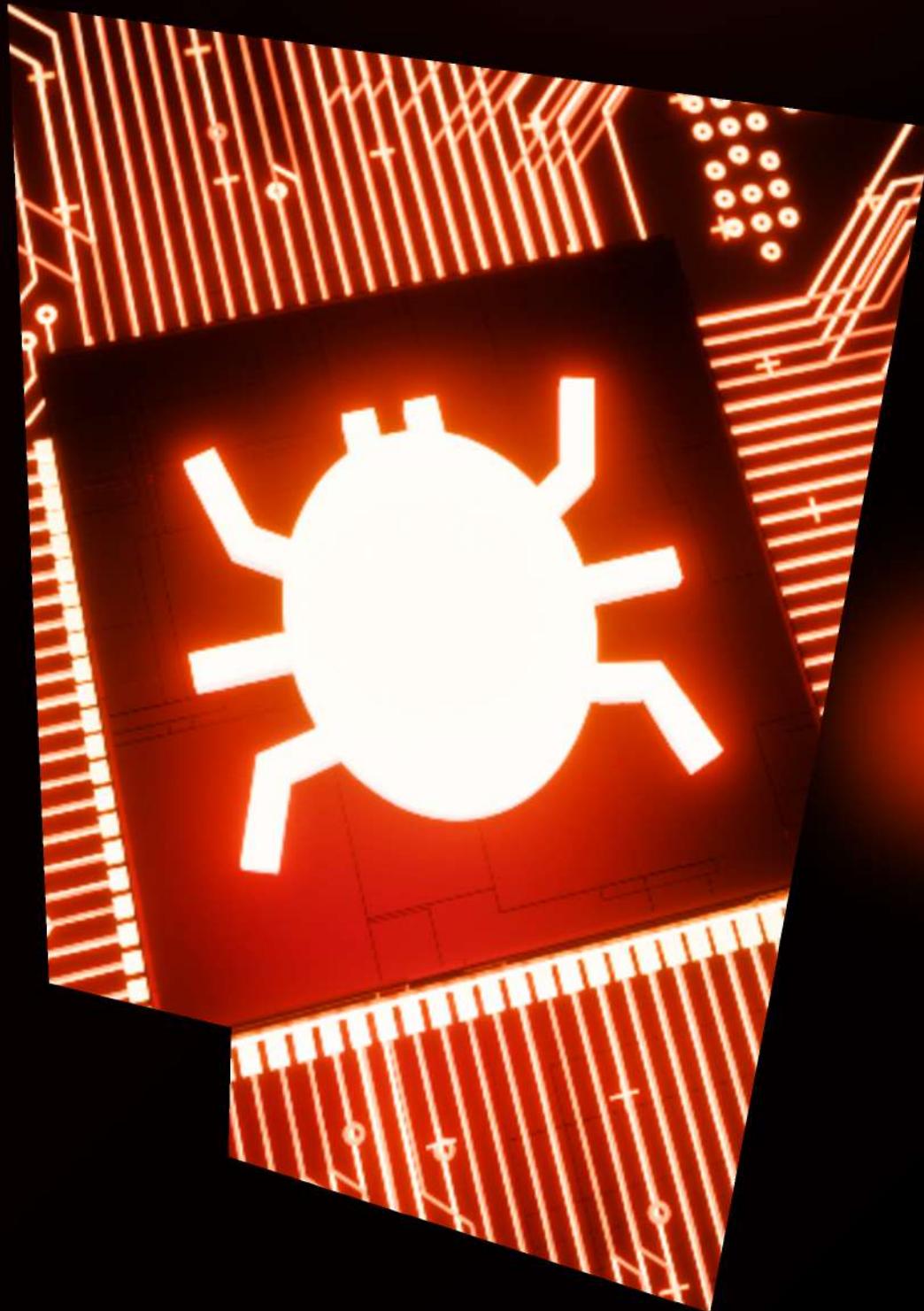
TRACCIA

GIORNO 3

GINA (Graphical identification and authentication) è un componente lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica - ovvero permette agli utenti di inserire **username e password** nel classico riquadro Windows, che usate anche voi per accedere alla macchina virtuale.

- Cosa può succedere se il file . dll lecito 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.



PARTE I

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

Effetti della Sostituzione della DLL GINA con una DLL Malevola

GINA (Graphical Identification and Authentication) è una componente essenziale di Windows, responsabile della gestione dell'autenticazione degli utenti tramite un'interfaccia grafica. Quando viene sostituita con una DLL malevola, il malware potrebbe catturare tutti i dati che l'utente inserisce durante la fase di login, incluso username e password.

Quando un malware sostituisce la DLL GINA con una versione malevola, il comportamento può includere:

- **Cattura battiture della tastiera:** Il malware potrebbe catturare ogni tasto premuto dall'utente, compresi username, password, e altre informazioni sensibili inserite durante l'autenticazione.
- **Memorizzazione e Trasmissione dei Dati:** Le informazioni raccolte possono essere memorizzate localmente per un successivo recupero da parte dell'attaccante e inviate a un server remoto per ulteriori utilizzi malevoli.
- **Mantenimento dell'Autenticazione Legittima:** Dopo aver catturato i dati, la DLL malevola può comunque passare le credenziali al sistema per permettere l'accesso legittimo, evitando così di allertare l'utente che qualcosa di sospetto stia accadendo.

PARTE I

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



Profilo del Malware Tipologia: Keylogger

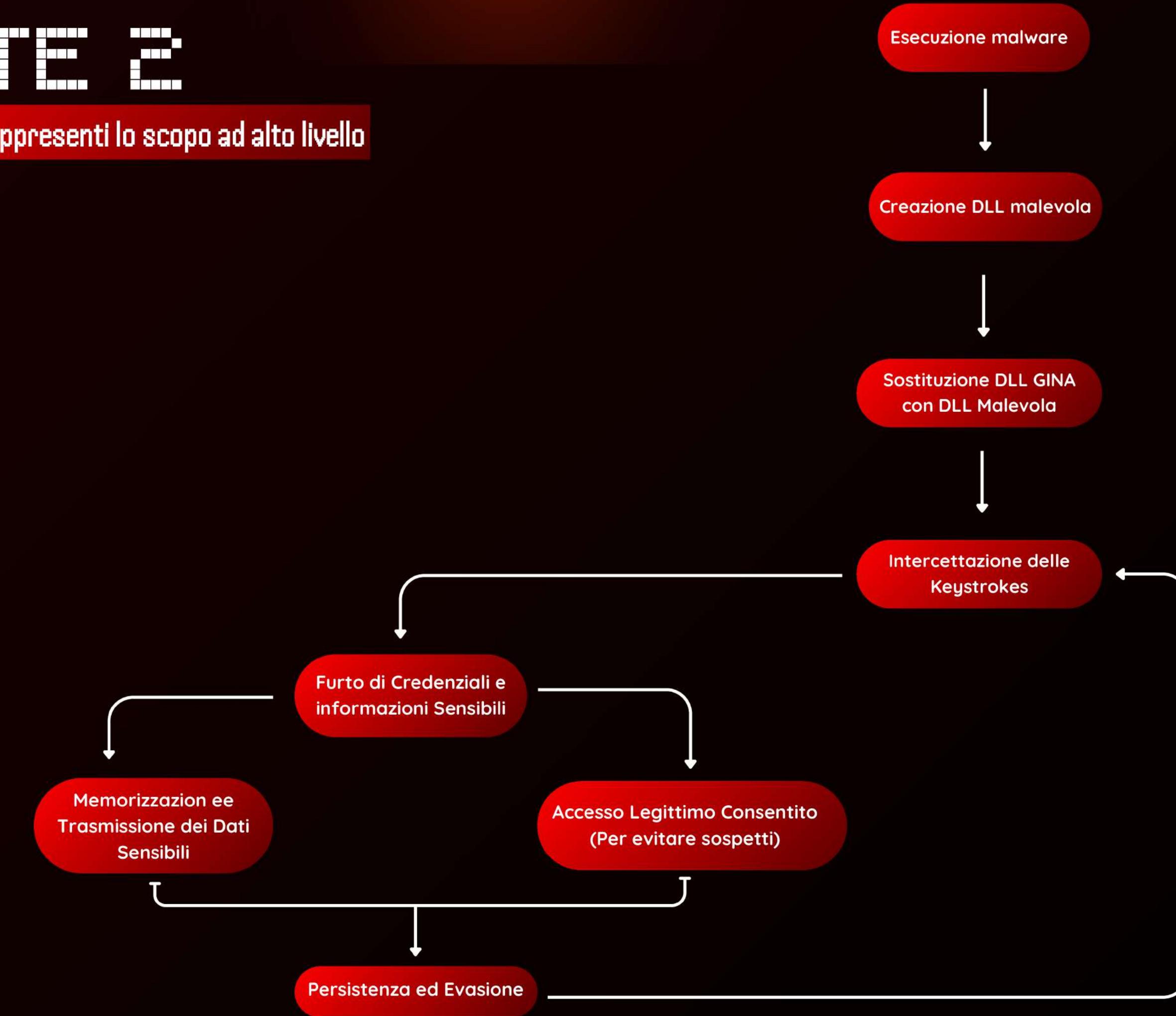
Funzionalità Principali



- **Intercettazione delle Keystrokes:** cattura tutte le battiture della tastiera, specialmente durante la fase di login.
- **Memorizzazione e trasmissione:** Memorizza localmente o trasmette a un server remoto le credenziali e altre informazioni sensibili.
- **Integrazione Trasparente:** Il malware si comporta come un componente legittimo, permettendo all'utente di accedere al sistema senza sospetti.
- **Persistenza:** Rimane attivo e nascosto nel sistema, anche dopo riavvii, per continuare a intercettare dati nel tempo.

PARTE 2

grafico che ne rappresenti lo scopo ad alto livello



PARTE 2

Descrizione dei Passaggi

- **Esecuzione dell'Malware:** Il malware viene inavvertitamente scaricato e si introduce nel sistema target, il cui obiettivo è sostituire la DLL GINA.
- **Creazione DLL Malevola:** Il malware crea una DLL Malevola da sostituire all' DLL vera
- **Sostituzione DLL GINA:** La DLL legittima viene sostituita con una versione malevola e caricata all'avvio del sistema, integrandosi nel processo di autenticazione.
- **Intercettazione delle Keystrokes:** Ogni battitura, inclusa username e password, viene catturata dalla DLL malevola.
- **Furto di Credenziali e Informazioni Sensibili:** L'attaccante può ora utilizzare le credenziali rubate per accessi non autorizzati o per altri scopi malevoli.
- **Memorizzazione o Trasmissione dei Dati:** Le informazioni raccolte vengono salvate localmente e inviate ad un server remoto.
- **Accesso Legittimo Consentito:** Dopo aver catturato le credenziali, il malware consente l'accesso legittimo per evitare sospetti.
- **Persistenza ed Evasione:** Il malware rimane attivo e nascosto, continuando a raccogliere dati

THANK YOU