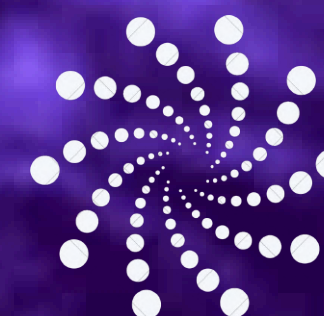




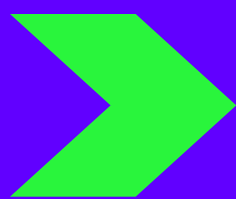
# Build week III

Malware analisi avanzata



TEAM 6

24/06/2024



GIORNO 1

# Traccia

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 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.



TEAM 6





Per analizzare un file eseguibile e visualizzare le sezioni, le funzioni, i parametri e le variabili, possiamo utilizzare due strumenti: **CFF Explorer** e **IDA**. Di seguito viene fornita una guida su come utilizzare ciascuno strumento per raggiungere questi obiettivi.

## CFF Explorer

**CFF Explorer** è uno strumento potente per esplorare i file eseguibili e le loro strutture interne. Segui questi passaggi per visualizzare le informazioni desiderate:

### 1. Apertura del File:

- Avvia CFF Explorer.
- Apri il file eseguibile che desideri analizzare.

### 2. Visualizzazione delle Sezioni:

- Naviga nella sezione "NT Headers".
- Seleziona "File Header" per visualizzare le informazioni di base del file.
- Vai a "Optional Header" per dettagli aggiuntivi.
- Sotto "Section Headers", troverai le varie sezioni del file (es. .text, .data, .rdata).

### 3. Esplorazione delle Funzioni e delle Variabili:

- Nella sezione ".text" puoi visualizzare il codice eseguibile.
- Utilizza il "Disassembler" integrato per vedere il codice assembly e identificare le funzioni.
- Le variabili globali possono essere trovate nella sezione ".data" o ".bss".



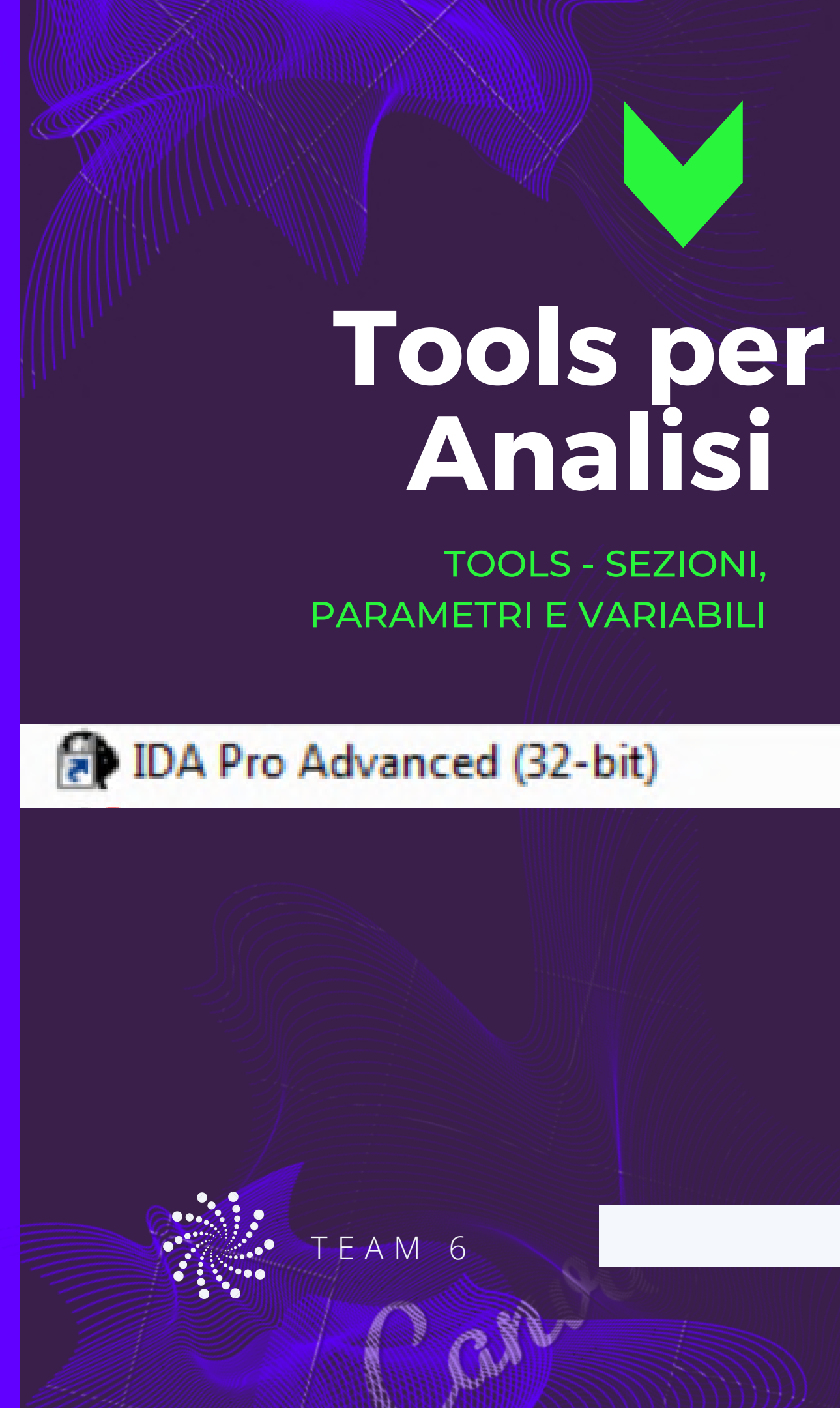
## IDA (Interactive Disassembler)

IDA è uno strumento avanzato di disassemblaggio e debug. Ecco come utilizzarlo:

- Caricamento del File:
  - Avvia IDA.
  - Apri il file eseguibile da analizzare.
- Analisi delle Sezioni:
  - IDA caricherà e disassemblerà automaticamente il file.
  - Nella finestra principale, potrai vedere una panoramica delle sezioni del file.
- Disassemblaggio del Codice:
  - Passa alla vista di disassemblaggio per esplorare il codice assembly.
  - Usa la vista grafica per visualizzare le funzioni come grafici di flusso.
- Identificazione di Funzioni e Parametri:
  - Le funzioni sono visualizzate con il loro indirizzo di memoria.
  - Seleziona una funzione per vedere i dettagli dei suoi parametri e variabili locali.
- Esplorazione delle Variabili:
  - Le variabili globali sono spesso indicate nella sezione dei dati.
  - Puoi usare la funzione di ricerca per trovare specifiche variabili nel codice.

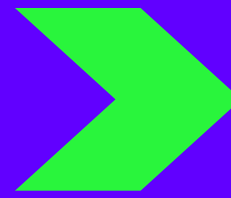
## Conclusione

Utilizzando CFF Explorer e IDA, puoi ottenere una visione dettagliata delle sezioni, funzioni, parametri e variabili di un file eseguibile. CFF Explorer offre una panoramica rapida e accesso alle sezioni, mentre IDA fornisce un'analisi approfondita e dettagliata del codice assembly e della struttura del programma. Questi strumenti sono complementari e insieme possono fornire un quadro completo dell'analisi del file.





1



QUANTI PARAMETRI SONO PASSATI ALLA  
FUNZIONE MAIN()?

The screenshot shows the IDA Pro interface. On the left, the 'Functions window' lists several functions, with `_main` selected. The main window displays the assembly code for `_main`. The function signature is `int __cdecl main(int argc, const char **argv, const char **envp)`. Below the signature, the stack frame parameters are listed: `hModule= dword ptr -11Ch`, `Data= byte ptr -118h`, `var_117= byte ptr -117h`, `var_8= dword ptr -8`, and `var_4= dword ptr -4`. A red box highlights the parameters `argc= dword ptr 8`, `argv= dword ptr 0Ch`, and `envp= dword ptr 10h`. The word 'Parametri' is written in red text next to the highlighted parameters.

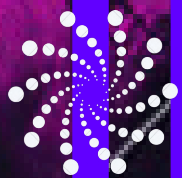
```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

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

push    ebp
mov     ebp, esp
```

I parametri vengono passati dalla funzione chiamante e si trovano sopra l'indirizzo di ritorno nel frame dello stack. Sono accessibili tramite un offset positivo in riferimento al puntatore EBP.

I parametri sono posizionati sopra EBP perché vengono spinti sullo stack prima della chiamata alla funzione.



2

QUANTE VARIABILI SONO DICHIARATE  
ALL'INTERNO DELLA FUNZIONE

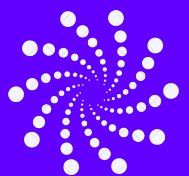
Variabili



The screenshot shows the IDA Pro interface. On the left, the 'Functions window' lists several functions, with 'start' selected. The main window displays the assembly code for the 'start' function. A red box highlights the local variable declarations: 'hModule= dword ptr -11Ch', 'Data= byte ptr -118h', 'var\_117= byte ptr -117h', 'var\_8= dword ptr -8', and 'var\_4= dword ptr -4'. The code also shows argument pointers and stack frame setup instructions.

Le variabili vengono allocate dalla funzione chiamata nello spazio sotto l'EBP corrente. Sono accessibili con offset negativo in riferimento al puntatore EBP.

Le variabili locali sono allocate sotto EBP perché lo spazio viene riservato dalla funzione dopo aver salvato il contesto (EBP e l'indirizzo di ritorno).







## QUALI SEZIONI SONO PRESENTI ALL'INTERNO DEL FILE ESEGUIBILE? DESCRIVETE BREVEMENTE ALMENO 2 DI QUELLE IDENTIFICATE

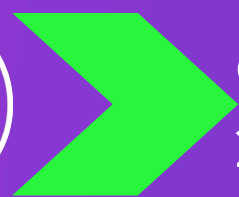
Malware_Build_Week_U3.exe					
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address
Byte[8]	Dword	Dword	Dword	Dword	Dword
.text	00005646	00001000	00006000	00001000	00000000
.rdata	000009AE	00007000	00001000	00007000	00000000
.data	00003EA8	00008000	00003000	00008000	00000000
.rsrc	00001A70	0000C000	00002000	0000B000	00000000

le sezioni presente sono **.text** **.rdata** **.data** e **.rsrc**

### Sezione **.text**

- Contenuto:
  - La sezione **.text** contiene il codice eseguibile dell'applicazione. Questo è il luogo in cui risiede il codice macchina che la CPU eseguirà.
- Caratteristiche:
  - Eseguita: Questa sezione è marcata come eseguibile, il che significa che il contenuto può essere eseguito dalla CPU.
  - Sola Lettura: Tipicamente, questa sezione è impostata come sola lettura per evitare modifiche accidentali al codice eseguibile durante l'esecuzione.
  - Dimensioni: La dimensione della sezione **.text** può variare notevolmente a seconda della complessità e delle dimensioni del programma.
- Utilizzo:
  - Contiene tutte le istruzioni del programma, organizzate in funzioni.
  - È fondamentale per il funzionamento del programma, poiché qualsiasi corruzione in questa sezione può impedire al programma di funzionare correttamente.





QUALI SEZIONI SONO PRESENTI ALL'INTERNO DEL FILE ESEGUIBILE? DESCRIVETE BREVEMENTE ALMENO 2 DI QUELLE IDENTIFICATE

Malware_Build_Week_U3.exe					
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

### Sezione .rdata

- Contenuto:
  - La sezione .rdata (Read-Only Data) contiene dati di sola lettura che il programma utilizza. Questo include variabili costanti, tabelle di stringhe, e altri dati che non devono essere modificati durante l'esecuzione del programma.
- Caratteristiche:
  - Sola Lettura: Come suggerisce il nome, questa sezione è progettata per contenere dati che non cambiano. È marcata come sola lettura per garantire l'integrità dei dati.
  - Non Eseguita: A differenza della sezione .text, .rdata non è eseguibile, il che significa che la CPU non eseguirà il contenuto di questa sezione come codice.
- Utilizzo:
  - Memorizza dati come stringhe di testo che vengono utilizzate in vari punti del programma.
  - Può contenere tabelle di costanti, numeri magici, e altri dati di configurazione che il programma deve leggere ma non modificare.
  - È utilizzata per organizzare dati che devono essere accessibili rapidamente, ma che non devono essere modificati.





QUALI LIBRERIE IMPORTA IL MALWARE ? PER OGNUNA DELLE LIBRERIE IMPORTATE, FATE DELLE 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.

Malware_Build_Week_U3.exe					
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA
0000769E	N/A	000074EC	000074F0	000074F4	000074F8
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0

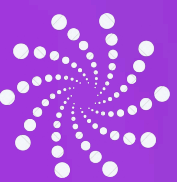
OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA

Le librerie importate dal malware sono “**KERNEL32.dll**” e “**ADVAPI32.dll**”

**KERNEL32.dll** è una delle librerie di sistema più fondamentali nei sistemi operativi Windows. Fornisce un'ampia gamma di funzioni essenziali che supportano le operazioni di base del sistema operativo, inclusa la gestione della memoria, la gestione dei processi e dei thread, la gestione degli input/output e l'interazione con l'hardware. Ecco una descrizione dettagliata delle sue caratteristiche e delle funzioni principali:

#### Caratteristiche di KERNEL32.dll

- **Core System Functionality:** KERNEL32.dll contiene funzioni critiche per il funzionamento del sistema operativo, fornendo una base su cui si costruiscono molte altre funzionalità di Windows.
- **Backward Compatibility:** Questa libreria è progettata per mantenere la compatibilità con le versioni precedenti di Windows, permettendo a software meno recente di funzionare correttamente sulle nuove versioni del sistema operativo.
- **Low-Level Operations:** KERNEL32.dll fornisce l'accesso a operazioni di basso livello necessarie per la gestione delle risorse di sistema.



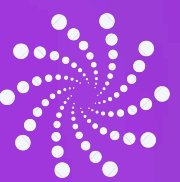


## Funzioni Principali di **KERNEL32.dll**

- Gestione della Memoria:
  - VirtualAlloc / VirtualFree: Funzioni per allocare e liberare memoria virtuale.
  - GlobalAlloc / GlobalFree: Funzioni per allocare e liberare memoria globale.
  - HeapCreate / HeapDestroy: Funzioni per creare e distruggere heap di memoria.
- Gestione dei Processi e dei Thread:
  - CreateProcess: Avvia un nuovo processo.
  - ExitProcess: Termina il processo chiamante.
  - CreateThread: Crea un nuovo thread di esecuzione.
  - ExitThread: Termina il thread chiamante.
  - WaitForSingleObject / WaitForMultipleObjects: Funzioni per sincronizzare l'esecuzione dei thread.
- Gestione degli Input/Output:
  - ReadFile / WriteFile: Funzioni per leggere e scrivere file.
  - SetFilePointer: Funzione per impostare il puntatore di lettura/scrittura in un file.
  - CreateFile: Funzione per creare o aprire un file o un dispositivo.
- Interazione con l'Hardware:
  - GetSystemTime / SetSystemTime: Funzioni per ottenere e impostare l'ora del sistema.
  - GetTickCount: Restituisce il tempo trascorso dall'avvio del sistema.
  - Sleep: Sospende l'esecuzione del thread chiamante per un intervallo di tempo specificato.
- Gestione delle Risorse del Sistema:
  - LoadLibrary / FreeLibrary: Funzioni per caricare e scaricare librerie dinamiche (DLL).
  - GetProcAddress: Ottiene l'indirizzo di una funzione esportata da una DLL.
  - SetHandleCount: Imposta il numero di handle che un processo può avere.

### Utilizzo di KERNEL32.dll

KERNEL32.dll viene utilizzata internamente da quasi tutti i programmi che girano su Windows per svolgere operazioni di sistema. Gli sviluppatori possono chiamare direttamente le sue funzioni tramite l'API di Windows per eseguire operazioni di basso livello necessarie per le loro applicazioni.





4



# ADVAPI32.dll

La libreria **ADVAPI32.dll** è una Dynamic Link Library (DLL) di Windows che contiene funzioni avanzate delle API (Application Programming Interface) per l'amministrazione dei servizi di sistema. Queste funzioni sono utilizzate per gestire e interagire con il registro di sistema, le chiavi di sicurezza, le sessioni, gli utenti e molto altro.

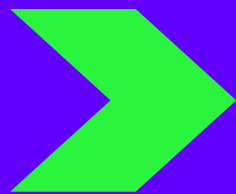
## Funzionalità Principali

1. Gestione del Registro di Sistema: Funzioni per leggere, scrivere, creare ed eliminare chiavi e valori nel registro di sistema.
2. Gestione della Sicurezza: Funzioni per definire e gestire le politiche di sicurezza, compresi i token di accesso, le ACL (Access Control Lists), e i contesti di sicurezza.
3. Gestione dei Servizi: Funzioni per creare, avviare, fermare e gestire i servizi di Windows.
4. Gestione degli Eventi: Funzioni per creare e registrare eventi nel registro eventi di Windows.

## Alcune Funzioni Comuni

1. RegOpenKeyEx: Apre una chiave del registro esistente.
2. RegQueryValueEx: Recupera i dati associati a un valore di una chiave del registro.
3. RegSetValueEx: Imposta i dati di un valore di una chiave del registro.
4. RegCreateKeyEx: Crea una nuova chiave del registro.
5. RegDeleteKey: Elimina una chiave del registro.
6. RegDeleteValue: Elimina un valore di una chiave del registro.





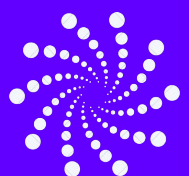
GIORNO 2

# Traccia

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 COMPRESSE TRA GLI INDIRIZZI 00401027 E 00401029 . (SE SERVE, VALUTATE 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.



TEAM 6





1

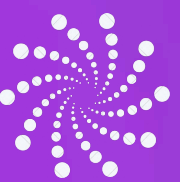


## LO SCOPO DELLA FUNZIONE CHIAMATA ALLA LOCAZIONE DI MEMORIA 00401021

```
.text:00401017 push 0 ; reserved  
.text:00401018 push offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...  
.text:00401021 call ds:RegCreateKeyExA  
.text:00401027 test eax, eax  
.text:00401029 jz short loc_401032
```

La funzione Call alla memoria 00401021 è **RegCreateKeyExA**. Questa funzione è una API di Windows che viene utilizzata per creare o aprire una chiave del registro di sistema.

La funzione **RegCreateKeyExA** viene chiamata per creare o aprire la chiave di registro specificata. Dopo la chiamata, se `eax` è zero, significa che la chiave è stata creata o aperta con successo. Se `eax` è diverso da zero, significa che si è verificato un errore.



```
.text:00401000 1  push    ebp
.text:00401001 2  mov     ebp, esp
.text:00401003 3  push    ecx
.text:00401004 4  push    0                ; lpdwDisposition
.text:00401006 5  lea     eax, [ebp+hObject]
.text:00401009 6  push    eax                ; phkResult
.text:0040100A  push    0                ; lpSecurityAttributes
.text:0040100C  push    0F003Fh           ; sanDesired
.text:00401011  push    0                ; dwOptions
.text:00401013  push    0                ; lpClass
.text:00401015  push    0                ; Reserved
.text:00401017  push    offset SubKey     ; "SOFTWARE\\Microsoft\\V
.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
```

I parametri vengono passati nello stack per la chiamata di funzione **“RegCreateKeyEx”**.

1. L'istruzione **push** mette il registro “ebp” in cima allo stack.
2. L'istruzione **mov** copia il registro “esp” nel registro “ebp”.
3. L'istruzione **push** inserisce il registro ecx nello stack
4. L'istruzione **push** inserisce il parametro 0
5. L'istruzione **lea** copia la variabile locale nel registro eax
6. Le seguenti istruzioni **push** continuano ad inserire tutti i parametri nel codice.



3



## CHE OGGETTO RAPPRESENTA IL PARAMETRO ALLA LOCAZIONE 00401017

```
.text:00401015 push 0 ; Reserved  
.text:00401017 push offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...  
.text:0040101C push 80000002h ; hKey
```

L'oggetto passato come parametro alla locazione 00401017 è l'**offset SubKey**.

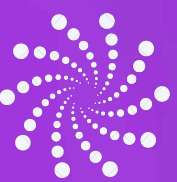
SubKey rappresenta un offset a una stringa in memoria. Questa stringa contiene il percorso di una sottochiave nel Registro di Windows, come suggerisce il commento.

Nella programmazione su Windows una "sottochiave" è una chiave all'interno della gerarchia del registro. Il Registro di Windows è un database gerarchico che memorizza impostazioni di basso livello per il sistema operativo e per le applicazioni. Una sottochiave può essere vista come una cartella all'interno della struttura ad albero del registro.

"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion" è una sottochiave sotto la radice HKEY\_LOCAL\_MACHINE. Questa sottochiave contiene varie impostazioni di configurazione relative alla versione corrente del sistema operativo Windows.

La SubKey è definita nella sezione **.data** del codice assembly come una stringa terminata da null (denotata da , 0 alla fine).

Nella sezione **.text**, l'offset SubKey ottiene l'indirizzo di questa stringa e lo mette nello stack.



4



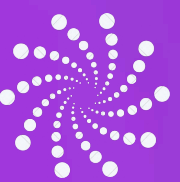
## IL SIGNIFICATO DELLE ISTRUZIONI COMPRESSE TRA GLI INDIRIZZI 00401027 E 00401029 . (SE SERVE, VALUTATE UN'ALTRA O ALTRE DUE RIGHE ASSEMBLY )

```
.text:00401027      test     eax, eax  
.text:00401029      jz       short loc_401032  
.text:0040102B      mov     eax, 1  
.text:00401030      jmp     short loc_40107B  
.text:00401032 ; -----  
.text:00401032      loc_401032:      mov     ecx, [ebp+cbData] ; CODE XREF: sub_401000+29↑j  
.text:00401032
```

Il significato di queste istruzioni è il seguente:

1. **test eax, eax**: Esegue un'operazione logica AND tra il registro eax e sé stesso. Questo viene fatto per impostare i flag del processore in base al valore di eax senza modificarlo.
2. **jz short loc\_40107B**: Salta all'etichetta loc\_40107B se il risultato dell'istruzione test è zero, cioè se eax è zero.

Queste istruzioni sono utilizzate per controllare se la funzione RegCreateKeyExA ha avuto successo. Se eax è zero, significa che la funzione è stata eseguita correttamente





5



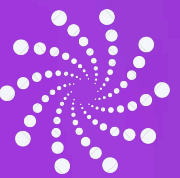
CON RIFERIMENTO ALL'ULTIMO QUESITO, TRADURRE IL CODICE ASSEMBLY NEL CORRISPONDENTE COSTRUTTO C

## Costrutto while

```
▼ while (eax!=0) {  
    //codice da eseguire finché eax diventa 0  
}
```

## Costrutto if

```
▼ if (eax == 0) {  
    loc_401032;  
}  
▼ else {  
}
```



6



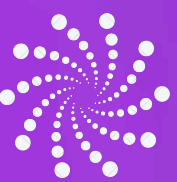
VALUTATE ORA LA CHIAMATA ALLA LOCAZIONE 00401047, QUAL È IL VALORE DEL PARAMETRO «VALUENAME»?

```
.text:00401036      mov     edx, [ebp+lpData]
.text:00401039      push    edx                ; lpData
.text:0040103A      push    1                  ; dwType
.text:0040103C      push    0                  ; Reserved
.text:0040103E      push    offset ValueName ; "GinaDLL"
.text:00401043      mov     eax, [ebp+hObject]
.text:00401046      push    eax                ; hKey
.text:00401047      call    ds:RegSetValueExA
.text:0040104D      test    eax, eax
.text:0040104F      jz      short loc_401062
.text:00401051      mov     ecx, [ebp+hObject]
.text:00401054      push    ecx                ; hObject
.text:00401055      call    ds:CloseHandle
.text:0040105B      mov     eax, 1
.text:00401060      jmp     short loc_40107B
```

Sull'allocazione di memoria 00401047 troviamo la chiamata di funzione "RegSetValue" che imposta il valore di un nome specifico e i dati all'interno di una chiave di registro aperta in Windows. La chiave di registro viene utilizzata per memorizzare impostazioni o dati di cui il malware ha bisogno per funzionare in modo persistente.

Il valore del parametro "valuenam" é GinaDLL.

GinaDLL é una libreria che viene caricata durante il processo di avvio di Windows (Winlogon). La modifica di questa voce può consentire al malware di eseguire il proprio codice durante il processo di autenticazione dell'utente. Inoltre GinaDLL si incarica di gestire la GUI di Windows





# 6



## SPIEGATE QUALE FUNZIONALITÀ STA IMPLEMENTANDO IL MALWARE IN QUESTA SEZIONE.

Questo malware ha l'obiettivo di rimanere attivo sul computer infetto anche dopo che viene riavviato o quando un utente si riconnette. Per farlo, modifica il modo in cui Windows gestisce il processo di login dell'utente.

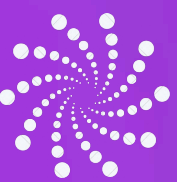
Passaggi Chiave del Malware:

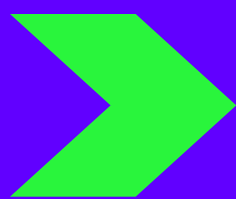
1. **Apertura o Creazione di una Chiave di Registro:** Il malware accede a una specifica sezione del registro di sistema di Windows. Il registro di sistema è come un grande database che Windows usa per tenere traccia di impostazioni e configurazioni. Il malware apre o crea una chiave (un elemento) in questo database sotto il percorso: HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon.
2. **Impostazione di un Valore:** Una volta aperta o creata la chiave di registro, il malware aggiunge o modifica un valore all'interno di questa chiave. In particolare, imposta un valore chiamato "ginaDLL". Questo valore dice a Windows di caricare un file DLL specifico ogni volta che un utente effettua il login. Una DLL (Dynamic Link Library) è un tipo di file che contiene codice eseguibile.
3. **Chiusura della Chiave di Registro:** Dopo aver impostato il valore, il malware chiude la chiave di registro per terminare l'operazione.

Perché il Malware Fa Questo:

Impostando il valore "ginaDLL", il malware istruisce Windows a caricare una DLL personalizzata durante il processo di login. Questa DLL contiene codice malevolo che viene eseguito automaticamente ogni volta che un utente si logga nel sistema. In questo modo, il malware riesce a eseguire azioni dannose ogni volta che il computer viene avviato o l'utente si riconnette, garantendo che rimanga attivo e mantenga il controllo sul sistema infetto.

Questa tecnica è comune nei malware perché assicura che il codice maligno venga eseguito automaticamente, anche dopo un riavvio del sistema o un nuovo login dell'utente, permettendo al malware di rimanere persistente e attivo.





GIORNO 3

# Traccia

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 L'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.



TEAM 6



1



QUAL È IL VALORE DEL PARAMETRO «RESOURCENAME»  
PASSATO ALLA FUNZIONE FINDRESOURCEA ()

```
.text:004010B8 ; -----  
.text:004010B8  
.text:004010B8 loc_4010B8:      ; CODE XREF: sub_4010B8+2F1j  
.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+module]  
.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      jnp     loc_4010BF  
.text:004010DF ; -----  
.text:004010DF
```

Il parametro ResourceName corrisponde al parametro **lpName**, il cui valore viene prima copiato nel registro ecx e successivamente immesso nello stack tramite l'istruzione push per essere passato alla funzione FindResourceA.

**lpName** è un puntatore a una stringa contenente il nome da cercare tramite **FindResourceA**. In notazione ungherese, il prefisso lp indica un "long pointer", solitamente utilizzato per puntare a stringhe. Il valore di questa stringa da cercare è TGAD.



```
.text:004010B8 ; -----
.text:004010B8
.text:004010B8 loc_4010B8:      mov     eax, lpType      ; CODE XREF: sub_401080+2F↑j
.text:004010B8      push    eax           ; lpType
.text:004010BD      mov     ecx, lpName
.text:004010BE      push    ecx           ; lpName
.text:004010C4      mov     edx, [ebp+hModule]
.text:004010C5      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:004010DF ; -----
.text:004010DF

.text:00401116      push    eax           ; hResInfo
.text:00401117      mov     ecx, [ebp+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     short loc_4011A5
.text:0040112C ; -----
.text:0040112C loc_40112C:      ; CODE XREF: sub_401080+A8↑j
.text:0040112C      push    4             ; flProtect
.text:0040112E      push    1000h          ; flAllocationType
.text:00401133      mov     edx, [ebp+Count]
.text:00401136      push    edx           ; dwSize
.text:00401137      push    0             ; lpAddress
.text:00401139      call    ds:VirtualAlloc
```

Sulla base delle funzionalità analizzate, possiamo classificare questo malware come un malware di persistenza e iniezione di codice.  
Funzionalità implementate:

### 1. Persistenza

Il malware modifica il registro di Windows per assicurarsi che venga eseguito ad ogni avvio del sistema o login dell'utente. In particolare, l'impostazione della chiave di registro `ginaDLL` sotto `HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon` garantisce che una DLL specificata venga caricata durante il processo di login, consentendo al malware di mantenere il controllo sul sistema infetto.

### 2. Iniezione di Codice

Il malware cerca di trovare, calcolare la dimensione e allocare memoria per una risorsa specifica nel modulo. Questa sequenza di operazioni è tipica dei malware che mirano a iniettare e eseguire codice arbitrario all'interno di un processo legittimo:

Categoria: Trojan persistente con funzionalità di iniezione di codice

### Tecniche utilizzate:

- Modifica delle chiavi di registro critiche per il processo di login di Windows.
- Utilizzo delle API di Windows per la gestione delle risorse e la memoria.

Il malware mostra un comportamento sofisticato tipico dei Trojan avanzati che cercano di mantenere il controllo sul sistema infetto e eseguire codice arbitrario senza essere rilevati. La combinazione di tecniche di persistenza e iniezione di codice rende questo malware particolarmente pericoloso, in quanto può rimanere nascosto nel sistema e agire in modo dannoso nel momento più opportuno.



3



È POSSIBILE IDENTIFICARE QUESTA FUNZIONALITÀ  
UTILIZZANDO L'ANALISI STATICA BASICA?



L'analisi statica basica di un malware è una tecnica di sicurezza informatica utilizzata per esaminare il codice in superficie di un malware senza eseguirlo.



3

4



## SUSSEGUIRSI DELLE CHIAMATE DI FUNZIONE E FUNZIONALITÀ IMPLEMENTATA

### Chiamata a **FindResourceA**:

```
.text:0040108B mov eax, lpType
.text:0040108D push eax
.text:0040108E mov ecx, lpName
.text:00401090 push ecx
.text:00401091 push ecx ; lpName
.text:00401092 mov edx, [ebp+hModule]
.text:00401095 push edx ; hModule
.text:00401096 call ds:FindResourceA
```

Trova una risorsa specificata dal tipo (lpType) e dal nome (lpName) all'interno del modulo (hModule).

### Chiamata a **SizeofResource**:

```
.text:00401084 push ecx
.text:00401085 call ds:SizeofResource
.text:0040108B mov [ebp+Count], eax
.text:0040108E cmp [ebp+Count], 0
.text:00401092 ja short loc_40112C
.text:00401094 jmp short loc_4010A5
```

Determina la dimensione della risorsa trovata.

### Chiamata a **VirtualAlloc**:

```
.text:0040112C push 4 ; flProtect
.text:0040112E push 1000h ; flAllocationType
.text:00401133 mov edx, [ebp+Count]
.text:00401136 push edx ; dwSize
.text:00401137 push 0 ; lpAddress
.text:00401139 call ds:VirtualAlloc
```

Alloca memoria virtuale per la risorsa.

La funzionalità implementata dal malware è di trovare una risorsa specifica all'interno di un modulo, determinarne la dimensione e allocare memoria per essa. Questa sequenza è tipica per i malware che vogliono caricare ed eseguire codice aggiuntivo.

### Identificazione della Funzionalità con Analisi Statica

**Confermiamo che è possibile identificare questa funzionalità utilizzando l'analisi statica.**

Le evidenze sono:

Chiamate a **FindResourceA** e **SizeofResource**: Indicano chiaramente che il malware sta cercando di localizzare e lavorare con risorse specifiche.  
 Uso di **VirtualAlloc**: Mostra che il malware sta allocando memoria, suggerendo che intende caricare e potenzialmente eseguire la risorsa trovata.





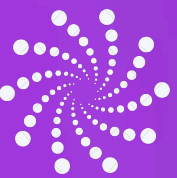


## FUNZIONALITA' IMPLEMENTATE DALL'INTERO CODICE

In questa sezione di codice, il malware sta implementando una funzionalità che riguarda la gestione delle risorse in un ambiente Windows. Vediamo il susseguirsi delle chiamate di funzione e ciò che viene fatto:

- **Inizializzazione:**
  - La funzione inizia con la configurazione del frame dello stack (push ebp; mov ebp, esp) e l'inizializzazione di diverse variabili locali a zero (mov [ebp+var], 0).
  - Verifica se il parametro hModule è zero e, in tal caso, salta alla fine della funzione.
- **Trovare una Risorsa:**
  - Chiama la funzione FindResourceA con i parametri lpType e lpName.
  - Se la risorsa non viene trovata, salta a un blocco di gestione degli errori.
- **Caricare la Risorsa:**
  - Chiama la funzione LoadResource con l'handle del modulo e le informazioni sulla risorsa ottenute da FindResourceA.
  - Se la risorsa non può essere caricata, salta a un blocco di gestione degli errori.
- **Bloccare la Risorsa:**
  - Chiama la funzione LockResource per ottenere un puntatore ai dati della risorsa.
  - Se la risorsa non può essere bloccata, salta a un blocco di gestione degli errori.
- **Ottenere la Dimensione della Risorsa:**
  - Chiama la funzione SizeofResource per ottenere la dimensione della risorsa caricata.
  - Se la dimensione è zero o si verifica un errore, salta a un blocco di gestione degli errori.
- **Allocare Memoria:**
  - Alloca memoria per la risorsa utilizzando la dimensione ottenuta e alcuni flag di allocazione.
  - Se l'allocazione di memoria fallisce, salta a un blocco di gestione degli errori.

**Funzionalità implementata dal malware:** Questo malware sembra implementare una funzionalità che gli permette di trovare, caricare, bloccare e ottenere informazioni su una risorsa incorporata in un modulo eseguibile. Questa è una tecnica comune usata dai malware per accedere ai propri dati nascosti o per caricare componenti aggiuntivi senza scrivere direttamente su disco, rendendo più difficile la rilevazione da parte dei software di sicurezza.





## PSEUDO CODICE IN LINGUAGGIO C

```
int sub_401088(HMODULE hModule)
{ // Inizializzazione delle variabili locali
void* hResInfo = NULL;
void* hResData = NULL;
void* Str = NULL;
int Count = 0;
int var_C = 0;
FILE* File = NULL;
// Verifica se hModule è valido
if (hModule == 0) { return 0;
}
// Trova la risorsa
hResInfo = FindResourceA(hModule, lpType, lpName);
if (hResInfo == NULL) { return 0;
}
// Carica la risorsa hResData =
LoadResource(hModule, hResInfo);
if (hResData == NULL) { return 0;
}
// Blocca la risorsa Str =
LockResource(hResData);
if (Str == NULL) { return 0;
}
// Ottiene la dimensione della risorsa Count =
SizeofResource(hModule, hResInfo);
if (Count == 0) { return 0;
}
// Alloca memoria per la risorsa void*
allocatedMemory = VirtualAlloc(NULL, Count, MEM_COMMIT, PAGE_READWRITE);
if (allocatedMemory == NULL) { return 0;
}
// Ulteriore elaborazione andrebbe qui...
return 1; }
```

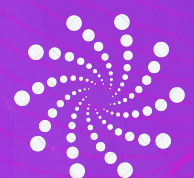
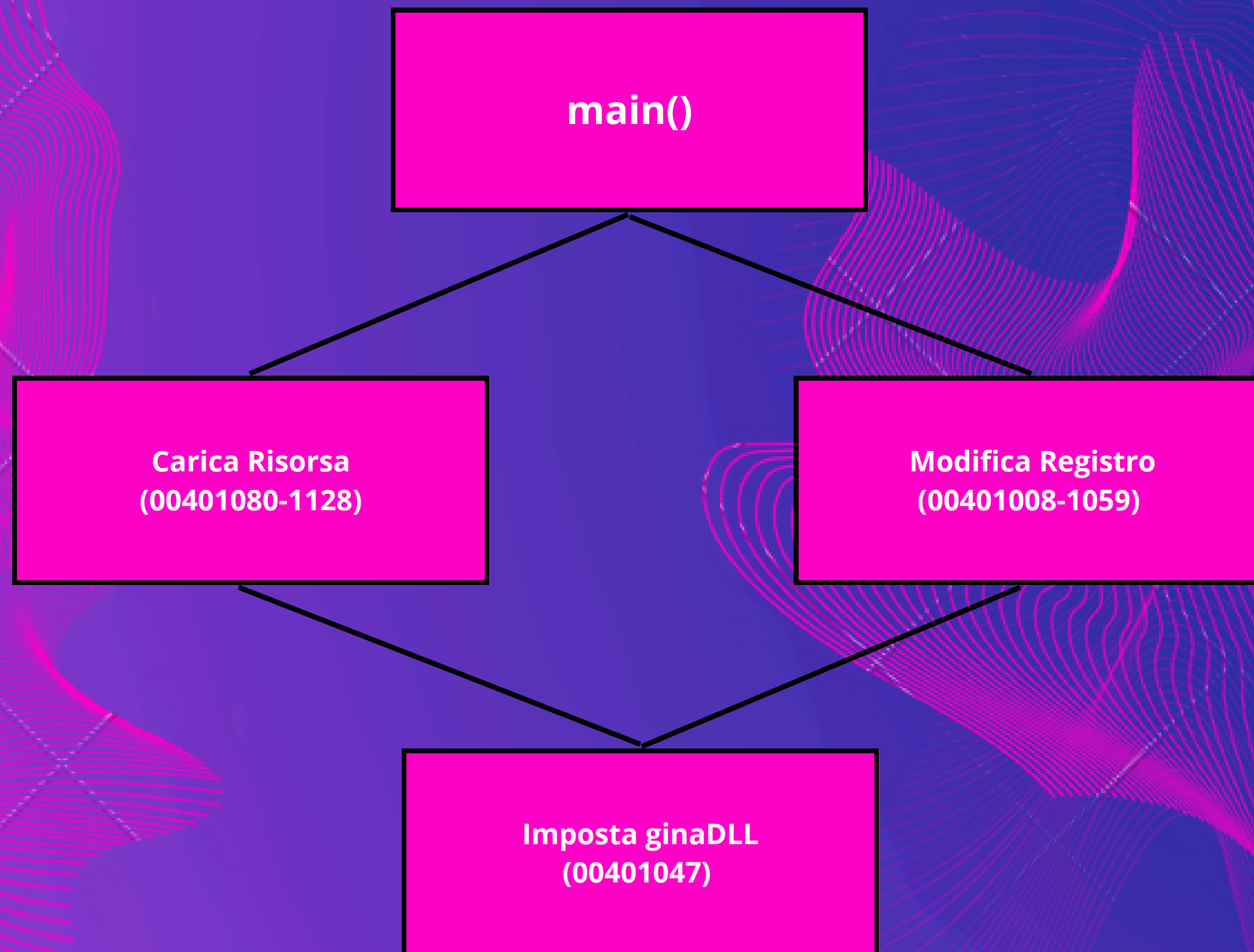


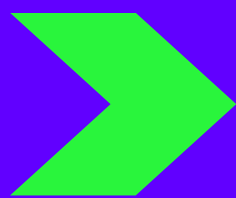


5



DISEGNARE UN DIAGRAMMA DI FLUSSO CHE COMPRENDA LE 3 FUNZIONI.





GIORNO 4

# Traccia

PREPARATE L'AMBIENTE ED I TOOL PER L'ESECUZIONE DEL MALWARE (SUGGERIMENTO: AVVIATE PRINCIPALMENTE PROCESS MONITOR ED ASSICURATE DI ELIMINARE OGNI FILTRO CLICCANDO SUL TASTO «RESET» QUANDO RICHiesto IN FASE DI AVVIO). ESEGUITE IL MALWARE , FACENDO DOPPIO CLICK SULL'ICONA DELL'ESEGUIBILE

1.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

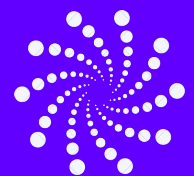
2.QUALE CHIAVE DI REGISTRO VIENE CREATA?

3. QUALE VALORE VIENE ASSOCIATO ALLA CHIAVE DI REGISTRO CREATA?

PASSATE ORA ALLA VISUALIZZAZIONE DELL'ATTIVITÀ SUL FILE SYSTEM

4.QUALE CHIAMATA DI SISTEMA HA MODIFICATO IL CONTENUTO DELLA CARTELLA DOVE È PRESENTE L'ESEGUIBILE DEL MALWARE ?

5.UNITE TUTTE LE INFORMAZIONI RACCOLTE FIN QUI SIA DALL'ANALISI STATICA CHE DALL'ANALISI DINAMICA PER DELINEARE IL FUNZIONAMENTO DEL MALWARE .



TEAM 6



1



## COSA NOTATE ALL'INTERNO DELLA CARTELLA DOVE È SITUATO L'ESEGUIBILE DEL MALWARE?



msgina32.dll

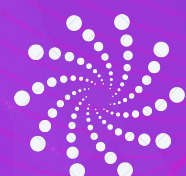
26/06/2024 10:17

Estensione dell'ap...

7 KB

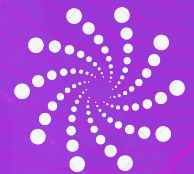
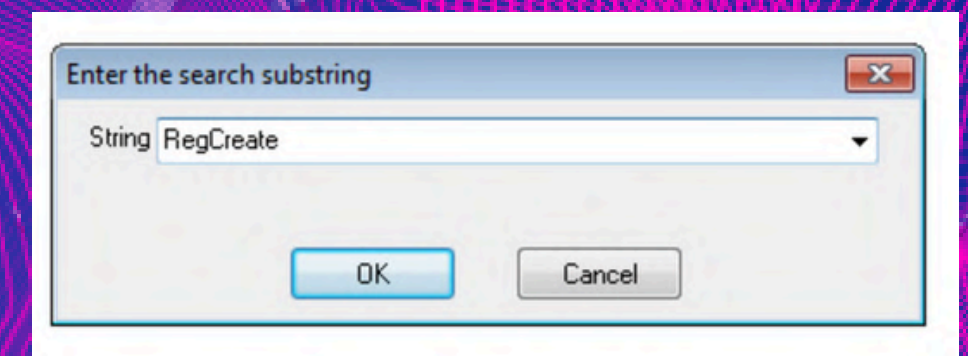
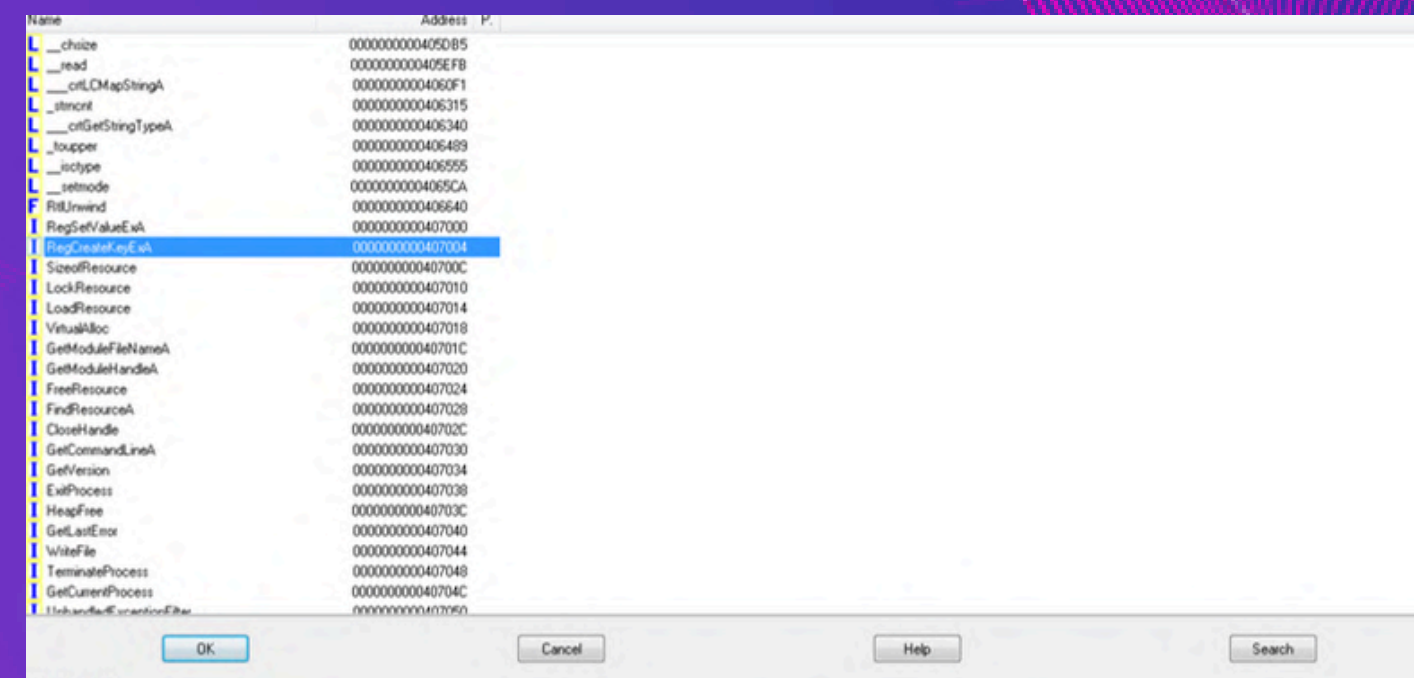
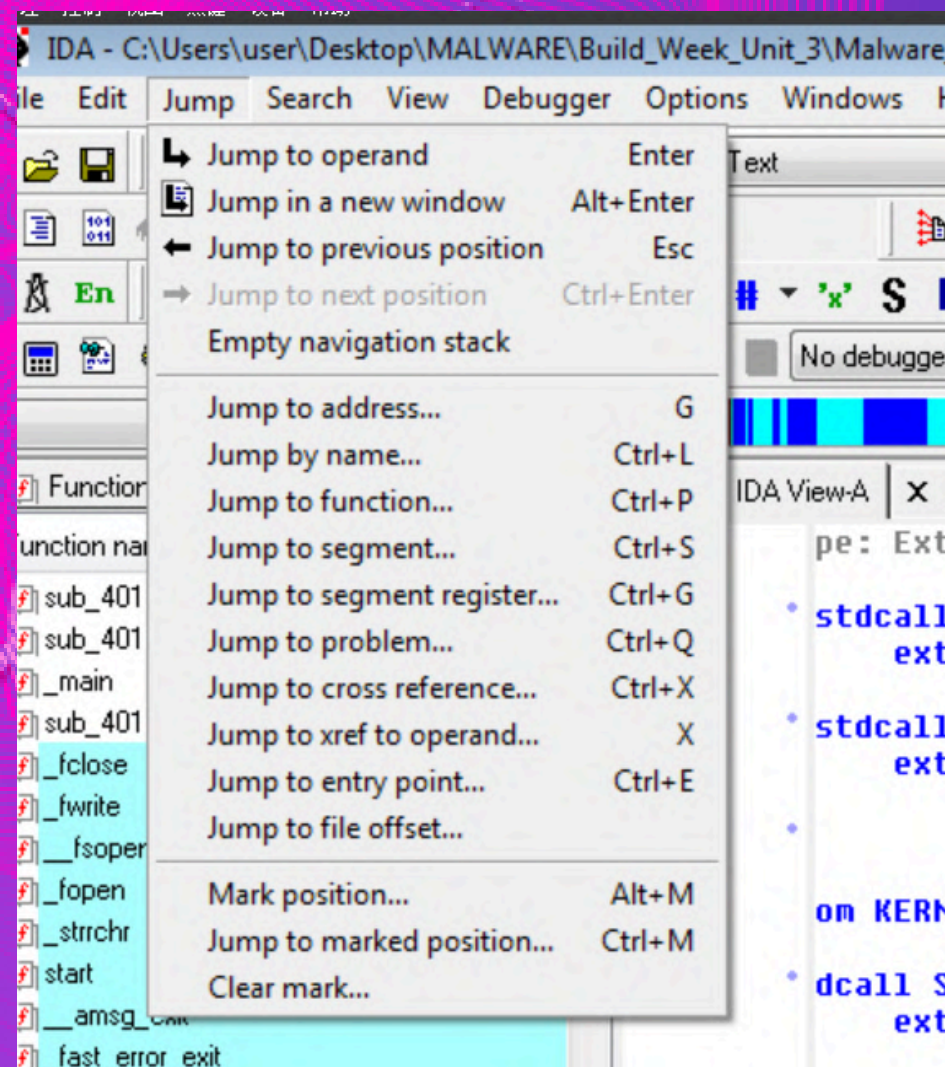
Dopo aver cliccato due volte sull'eseguibile del malware e poi su "esegui" nella finestra a comparsa, notiamo che nella cartella in cui è situato l'eseguibile del malware è stata aggiunta una nuova libreria: **msgina32.dll**

Abbiamo cercato di analizzare i cambiamenti causati dal malware con Process Monitor, ma ci siamo resi conto che il malware non è compatibile con Windows 7 e quindi andrebbe spostato su Windows XP. Prima di fare ciò proviamo a vedere se riusciamo ad analizzarlo staticamente con i tool a nostra conoscenza.



# IDA Pro

Per trovare rapidamente la funzione `RegCreateKeyExA`, utilizziamo il metodo "jump by name". Successivamente, clicchiamo sul pulsante "search" e inseriamo il nome della funzione indicata. In questo modo, troveremo il suo indirizzo velocemente.





## QUALE CHIAVE DI REGISTRO VIENE CREATA?

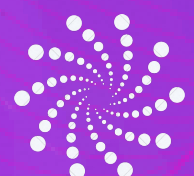
Successivamente, nella finestra principale di IDA, verremo portati immediatamente all'indirizzo ricercato in precedenza. Abbiamo visto che l'indirizzo è  $401000 + 21$ , che è uguale a 401021. Posizionando il mouse sopra il numero 21, possiamo vedere il codice del funzione che abbiamo già spiegato dettagliatamente il giorno 3.

```
.idata:00407000 ; LSTATUS __stdcall RegSetValueExA(HKEY hKey, LPCSTR lpValueName, DWORD Reserved,
; extrn RegSetValueExA:dword ; CODE XREF: sub_401000+47↑p
; DATA XREF: sub_401000+47↑tr ...
; LSTATUS __stdcall RegCreateKeyExA(HKEY hKey, LPCSTR lpSubKey, DWORD Reserved, LPSTR lpClass, DWOR
; extrn RegCreateKeyExA:dword ; CODE XREF: sub_401000+21↑p
; DATA XREF: sub_401000+21↑tr
; Imports from KERNEL32.dll
; DWORD __stdcall SizeofResource(HMODULE hModule, HRSRC hResInfo)
; extrn SizeofResource:dword ; CODE XREF: sub_401000+9B↑p
```

```
; LSTATUS __stdcall RegCreateKeyExA(HKEY hKey, LPCSTR lpSubKey, DWORD Reserved, LPSTR lpClass, DWOR
; extrn RegCreateKeyExA:dword ; CODE XREF: sub_401000+21↑p
; DATA XREF: sub_401000+21↑tr
push    eax                ; phkResult
push    0                  ; lpSecurityAttributes
push    0F003Fh            ; samDesired
push    0                  ; dwOptions
push    0                  ; lpClass
push    0                  ; Reserved
push    offset SubKey      ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h          ; hKey
call    ds:RegCreateKeyExA
test    eax, eax
```

0407004: .idata:RegCreateKeyExA

La chiave di registro creata è **SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlog**



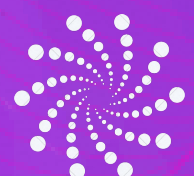


## WINDOWS 7 -> WINDOWS XP

A questo punto dobbiamo spostarci su Windows XP perché possiamo ipotizzare il funzionamento del malware, ma non siamo capaci di vedere le modifiche nell'effettivo. Ad esempio, anche aprendo Regedit su Windows 7 e analizzando i registri di Windows, non riusciamo a trovare la chiave che contiene GinaDLL, cosa possibile invece su Windows XP.

### **Come spostare il malware da Windows 7 a Windows XP**

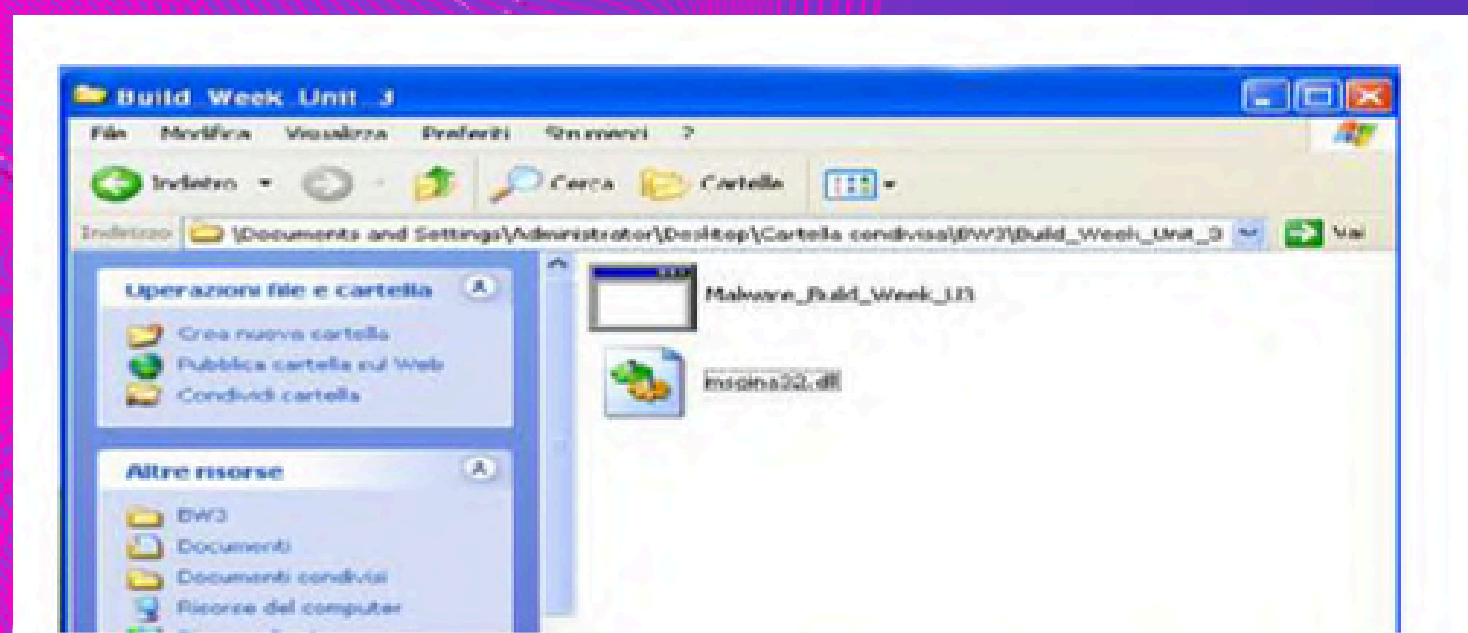
- creare un disco fisso VDI vuoto su VirtualBox;
- montare il disco vuoto su Windows 7;
- da Windows 7 copiare la cartella del malware sul disco vuoto;
- montare lo stesso disco (che a questo punto contiene i dati) su WindowsXP.







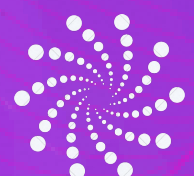
Da Windows XP possiamo avviare Procmon e filtrare la ricerca, come mostrato nelle slide della traccia, solo con i processi effettuati dal malware, in modo da vedere i cambiamenti che questo ha portato alla macchina. Per rispondere alla prima domanda, ci soffermiamo sulla funzione RegCreateKey e notiamo come la chiave sia stata creata con successo nel path indicato.



RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Executio...	NAME NOT FOUND	Desired Access: Read
RegOpenKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Desired Access: Read
RegQueryValue	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
RegCloseKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
RegOpenKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: Read
RegQueryValue	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
RegCloseKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Executio...	NAME NOT FOUND	Desired Access: Read
RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Executio...	NAME NOT FOUND	Desired Access: Read
RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Image File Executio...	NAME NOT FOUND	Desired Access: Read
RegOpenKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Desired Access: Read
RegQueryValue	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
RegCloseKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: Read
RegQueryValue	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTr...	NAME NOT FOUND	Length: 144
RegCloseKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: Maximum Allowed
RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon	NAME NOT FOUND	Desired Access: Read
RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon	NAME NOT FOUND	Desired Access: Read
RegCreateKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All Access
RegSetValue	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\OleDLL	SUCCESS	Type: REG_SZ, Length: 520, Data: C:\Doc...
RegCloseKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	

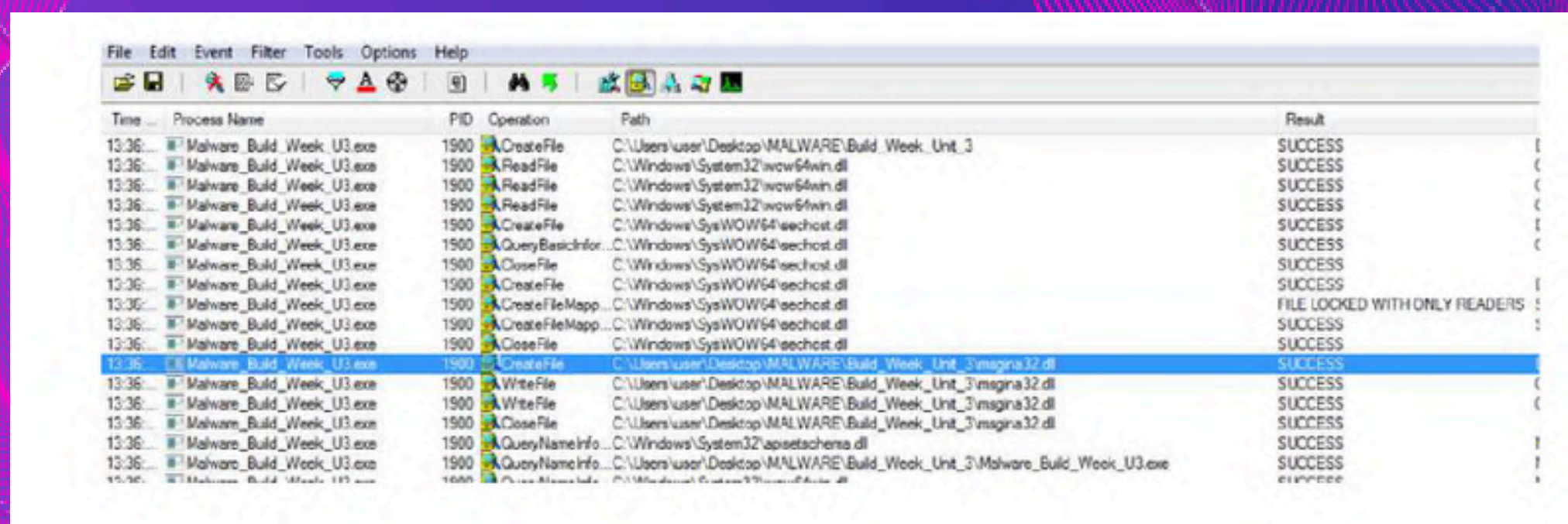


Su Regedit possiamo navigare nel path della chiave, entrare nella cartella Winlog e vedere che è stato aggiunto il valore **GinaDLL**



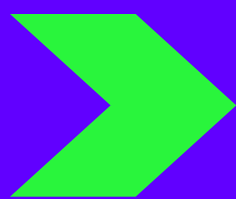


Sempre su Procmon impostando il filtro per le attività del sistema, otteniamo in output tutte le azioni compiute dal malware e tra queste notiamo **CreateFile** che è la funzione responsabile della creazione di **msgina32.dll** nella stessa cartella dell'eseguibile, come conferma il path a destra della funzione.



Time	Process Name	PID	Operation	Path	Result
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Windows\SysWOW64\eechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryBasicInfor...	C:\Windows\SysWOW64\eechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Windows\SysWOW64\eechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Windows\SysWOW64\eechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFileMapp...	C:\Windows\SysWOW64\eechost.dll	FILE LOCKED WITH ONLY READERS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFileMapp...	C:\Windows\SysWOW64\eechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Windows\SysWOW64\eechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryNameInfo...	C:\Windows\System32\apisetachema.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryNameInfo...	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\Malware_Build_Week_U3.exe	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryNameInfo...	C:\Windows\System32\apisetachema.dll	SUCCESS





GIORNO 5

# Traccia

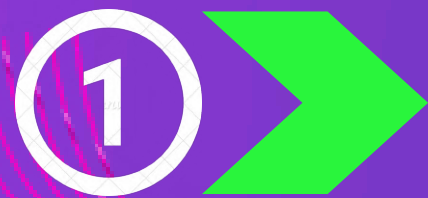
GINA (GRAPHICAL IDENTIFICATION AND AUTHENTICATION ) È UN COMPONENTE LECITO DI WINDOWS CHE PERMETTE L'AUTENTICAZIONE DEGLI UTENTI TRAMITE INTERFACCIA GRAFICA OVVERO PERMETTE AGLI UTENTI DI INSERIRE USERNAME E PASSWORD NEL CLASSICO RIQUADRO WINDOWS, COME QUELLO CHE USATE ANCHE VOI PER ACCEDERE ALLA MACCHINA VIRTUALE.

1.COSA PUÒ SUCCEDERE SE IL FILE . DLL LECITO VIENE SOSTITUITO CON UN FILE . DLL MALEVOLO, CHE INTERCETTA I DATI INSERITI?

2.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



TEAM 6



## COSA PUÒ SUCCEDERE SE IL FILE . DLL LECITO VIENE SOSTITUITO CON UN FILE . DLL MALEVOLO, CHE INTERCETTA I DATI INSERITI?

**GINA (Graphical Identification and Authentication)** è un componente lecito di Windows utilizzato per gestire l'autenticazione degli utenti tramite un'interfaccia grafica. Se il file DLL lecito di GINA viene sostituito con un file DLL malevolo, possono verificarsi le seguenti conseguenze:

- **Intercettazione delle Credenziali:**

La DLL malevola può catturare tutte le informazioni inserite dall'utente, come username e password. Questo permette agli attaccanti di ottenere le credenziali di accesso degli utenti.

- **Persistenza e Controllo Continuo:**

La DLL malevola può garantire la persistenza del malware eseguendo codice malevolo ogni volta che l'utente effettua il login. Questo permette al malware di mantenere il controllo sul sistema infetto.

- **Modifiche al Comportamento del Sistema:**

La DLL malevola può alterare il comportamento del processo di autenticazione, introducendo backdoor o meccanismi di accesso nascosti che gli attaccanti possono utilizzare per accedere al sistema in qualsiasi momento.

- **Escalation dei Privilegi:**

Poiché GINA opera con privilegi elevati, una DLL malevola può sfruttare questa posizione per eseguire operazioni privilegiate sul sistema, come la modifica di altre impostazioni di sicurezza o l'installazione di ulteriori componenti malevoli.







Sulla base delle analisi precedenti, il malware può essere profilato come un **Trojan persistente** con funzionalità di iniezione di codice e intercettazione delle credenziali

### Funzionalità principali:

- **Persistenza:**

Modifica del registro di Windows per garantire l'esecuzione automatica della DLL malevola durante il login.

- **Iniezione di Codice:**

Utilizzo delle API di Windows per trovare, calcolare la dimensione e allocare memoria per una risorsa specifica nel modulo, con l'intento di caricare ed eseguire codice malevolo.

- **Intercettazione delle Credenziali:**

Sostituzione della DLL GINA legittima con una malevola per catturare le credenziali di accesso degli utenti.

- **Escalation dei Privilegi:**

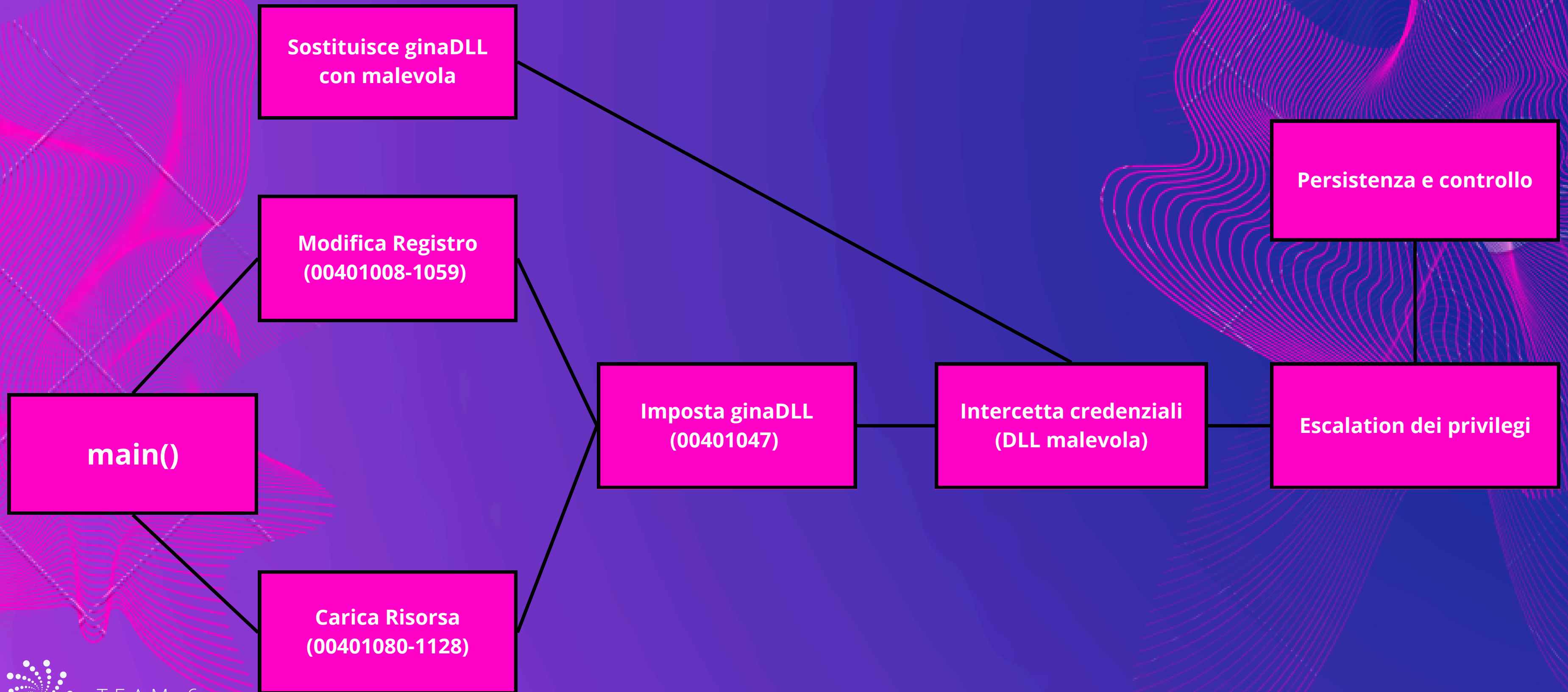
Sfruttamento dei privilegi elevati della DLL GINA per eseguire operazioni privilegiate e garantire il controllo sul sistema.



2



## DIAGRAMMA DI FLUSSO AD ALTO LIVELLO







# TEAM



**Anapaula**



**André**



**Mara**  
TEAM LEADER



**Mario**



**Roberta**



**Zhong**

