

# BUILD WEEK 3

M E M B R I

Marco D'Antoni - Sara Spaccialbelli - Gerardo Carrabs

Davide Andreozzi - Andrea Mandelli - Mattia Bassani

Cristian Calvaruso

MALWARE ANALYSIS

2024

.....

# Giorno 1

## Analisi statica

### Richiesta

Con riferimento al file eseguibile Malware\_Build\_Week\_U3 rispondere ai seguenti quesiti:

1. Parametri della Funzione Main

- Quanti parametri vengono passati alla funzione principale.

3. Sezioni del File Esegibile

- Descrizione di almeno due sezioni chiave identificate.

2. Variabili nella Funzione Main

- Numero di variabili dichiarate internamente.

4. Librerie Importate e Potenziali Funzionalità

- Elenco delle librerie importate.
- Ipotesi sulle funzionalità del malware basate sull'analisi statica e sulle funzioni utilizzate.

# 1. Quanti parametri e quante variabili sono dichiarate all'interno della funzione main()?

Per analizzare il malware, utilizziamo IDA pro, un software utilizzato per il reverse engineering.

```
; 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
```

**Nella funzione main () troviamo:**

- 3 parametri, offset positivo rispetto al registro EBP
- 5 variabili, offset negativo rispetto al registro EBP

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

Per l'analisi delle sezioni, utilizziamo CFF Explorer

Malware_Build_Week_U3.exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00005646	00001000	00006000	00001000	00000000	00000000	0000	0000	60000020
.rdata	000009AE	00007000	00001000	00007000	00000000	00000000	0000	0000	40000040
.data	00003EA8	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040

**.text**: Contiene il codice eseguibile del programma, ovvero la parte del file che viene effettivamente eseguita dal processore quando il programma viene avviato.

**.rdata**: contiene i dati "read-only" del programma. Questi dati sono solo per lettura e solitamente includono costanti, stringhe e altre informazioni che non vengono modificate durante l'esecuzione del programma

**.data**: contiene i dati modificabili del programma. Qui vengono memorizzate variabili globali e altri dati che possono essere modificati durante l'esecuzione del programma

**.rsrc**: contiene risorse del programma, le quali possono essere utilizzate dall'applicazione per fornire elementi grafici e altre funzionalità che non sono parte diretta del codice eseguibile, ma sono utili per l'interfaccia utente e altre funzionalità del programma.

### 3.Quali librerie importa il Malware?

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

KERNEL32.dll → libreria di sistema Windows. Essa fornisce un'ampia varietà di funzioni di basso livello per il funzionamento con il sistema operativo, tra cui gestione della memoria, gestione dei processi, dei file e delle directory, ecc.

ADVAPI32.dll → è una libreria che fornisce molte funzioni per l'accesso e la gestione del registro di sistema, oltre ad altre funzionalità legate alla sicurezza, alla crittografia e ai servizi di Windows.

## 4. Funzionalità implementate (KERNEL32.dll)

**SizeOfResource**: Questa funzione restituisce la dimensione, in byte, di una risorsa specifica all'interno di un modulo eseguibile o di una DLL. Può essere utilizzata per ottenere le dimensioni di risorse come immagini, icone, stringhe.

**LockResource**: Questa funzione restituisce un puntatore al contenuto della risorsa, che può quindi essere utilizzato per leggerne o modificarne i dati.

**LoadResource**: Questa funzione viene utilizzata per caricare una risorsa specifica da un modulo eseguibile o da una DLL in memoria.

Malware_Build_Week_U3.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
0000769E	N/A	000074EC	000074F0	000074F4	000074F8	000074FC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource

## 4. Funzionalità implementate (ADVAPI.dll)

Malware_Build_Week_U3.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
000076D0	N/A	00007500	00007504	00007508	0000750C	00007510
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000076AC	000076AC	0186	RegSetValueExA
000076BE	000076BE	015F	RegCreateKeyExA

- **RegSetValueExA:** Questa funzione viene utilizzata per impostare il valore di una voce specifica nel registro di sistema, per esempio il nome della chiave di registro, il nome del valore, il tipo di dati e i dati stessi.
- **RegCreateKeyExA:** Essa viene utilizzata per creare o aprire una chiave di registro. Un malware in questo caso, potrebbe modificare il registro di sistema assegnando nuovi valori, (es. modifica delle impostazioni di sicurezza).

## IPOTESI

Sulla base delle funzioni analizzate (LoadResource, LockResource, SizeOfResource) potrebbe trattarsi di un dropper. Queste API permettono di localizzare all'interno della sezione «risorse» il malware da estrarre, per la esecuzione futura.

# Giorno 2

## Analisi statica

### Richiesta

**Con riferimento al malware in analisi, spiegare:**

1. Funzione alla Locazione 00401021
  - Scopo della funzione specifica.
2. Passaggio Parametri alla Locazione 00401021
  - Metodo di passaggio dei parametri a questa funzione.
3. Parametro alla Locazione 00401017
  - Oggetto o valore rappresentato dal parametro.
4. Istruzioni da 00401027 a 00401029
  - Significato delle operazioni eseguite in queste istruzioni.
  - Eventuale considerazione di ulteriori linee assembly per la comprensione.
5. Traduzione Codice Assembly in C
  - Conversione del segmento di codice Assembly in un equivalente costrutto in linguaggio C.
6. Chiamata alla Locazione 00401047
  - Valore del parametro "ValueName" utilizzato nella chiamata.

Nel contesto delle funzionalità menzionate,  
identificare quella attualmente implementata dal  
malware in questa sezione.

## 1. Scopo della funzione chiamata alla locazione di memoria 00401021

Lo scopo della funzione chiamata alla locazione di memoria 00401021 è quello di creare o aprire una chiave di registro.

```
.text:00401021          call    ds:RegCreateKeyExA
```

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

I parametri vengono passati alla funzione tramite “push”. Ovvero vengono messi sullo stack prima della chiamata della funzione.

```
.text:00401009          push    eax      ; phkResult
.text:0040100A          push    0         ; lpSecurityAttributes
.text:0040100C          push    0F003Fh   ; samDesired
.text:00401011          push    0         ; dwOptions
.text:00401013          push    0         ; lpClass
.text:00401015          push    0         ; Reserved
.text:00401017          push    offset SubKey  ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Run"
.text:0040101C          push    80000002h   ; hKey
.text:00401021          call    ds:RegCreateKeyExA
```

### 3. Che oggetto rappresenta il parametro alla locazione 00401017

L'oggetto rappresentato dal parametro alla locazione 00401017 rappresenta il nome della chiave che si vuole creare o aprire e il relativo path (percorso).

La chiave principale in questo caso è HKEY\_LOCAL\_MACHINE. L'offset SubKey "Microsoft//Windows NT/CurrentVersion/Winlogon" indica una sottocategoria all'interno della chiave principale.

La chiave "Winlogon" in questione contiene informazioni relative all'avvio del sistema operativo Windows. Questa chiave gestisce vari aspetti dell'interazione dell'utente con il sistema operativo, inclusi i processi di accesso, l'autenticazione e le impostazioni dell'interfaccia utente.

L'offset SubKey "SOFTWARE/Microsoft//Windows NT/CurrentVersion/Winlogon" si riferisce quindi a una parte specifica del Registro di sistema che gestisce l'avvio e l'interazione dell'utente con il sistema operativo Windows.

.text:00401009	push	eax	; phkResult
.text:0040100A	push	0	; lpSecurityAttributes
.text:0040100C	push	0F003Fh	; samDesired
.text:00401011	push	0	; dwOptions
.text:00401013	push	0	; lpClass
.text:00401015	push	0	; Reserved
.text:00401017	push	offset SubKey	; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040101C	push	80000002h	; hKey
.text:00401021	call	ds:RegCreateKeyExA	

#### 4. Spiegare il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.

- La prima istruzione serve a vedere il valore del registro eax è 0.
- La seconda istruzione è un salto condizionale: salta se eax è 0.
- Si tratta quindi di un costrutto "if".

.text:00401027	test	eax, eax
.text:00401029	jz	short loc_401032
.text:0040102B	mov	eax, 1
.text:00401030	jmp	short loc_40107B
.text:00401032 ; -----		

#### 5. Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C

```
if(eax == 0)
{
    goto loc_401032;
}
```

## 6. Valutate ora la chiamata alla locazione 00401047, qual è il valore del parametro «ValueName»?

- Il valore del parametro "ValueName" è "GinaDLL".

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

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

- Le funzionalità appena viste possono creare e modificare le chiavi di registro e permettere al malware di ottenere la persistenza. Ovvero, anche in caso di spegnimento del sistema il malware si riattiverà ad ogni accensione. Inoltre il valore GinaDLL suggerisce il tentativo di manipolare il login al sistema.

# Giorno 3

## Analisi statica/dinamica

### Richiesta

#### **Analisi del Codice: Routine 00401080 - 00401128**

##### 1. Parametro "ResourceName" in FindResourceA()

- Valore del parametro "ResourceName" passato alla funzione.

##### 2. Funzionalità Implementata dal Malware

- Identificazione della funzionalità basata sulle chiamate di funzione osservate.
- Correlazione con gli argomenti trattati durante le lezioni teoriche.

##### 3. Possibilità di Identificazione tramite Analisi Statica

- Valutazione della capacità di identificare questa funzionalità attraverso l'analisi statica basica.
- In caso affermativo, elenco delle evidenze che supportano questa identificazione.

##### 4. Entrambe le funzionalità principali del Malware viste finora sono richiamate all'interno della funzione Main().

- Disegnare un diagramma di flusso che comprenda le 3 funzioni.

# Analisi del codice



```
 ResourceType => "BINARY"  
 Malware_.00408038  
 ResourceName = "TGAD"  
  
 hModule  
 FindResourceA
```

Nel giorno 3 abbiamo trovato il valore del parametro **ResourceName** con l'ausilio di OllyDbg, all'indirizzo 004010C4 possiamo notare il valore del parametro che risulterà essere TGAD.

Successivamente analizzando tra gli indirizzi 00401080 e 00401128 possiamo notare che il malware effettua le seguenti chiamate:

- **FindResourceA**
- **LoadResource**
- **LockResource**
- **SizeofResource**

Queste chiamate fanno riferimento a un comportamento del malware che possiamo classificare come **Dropper**.

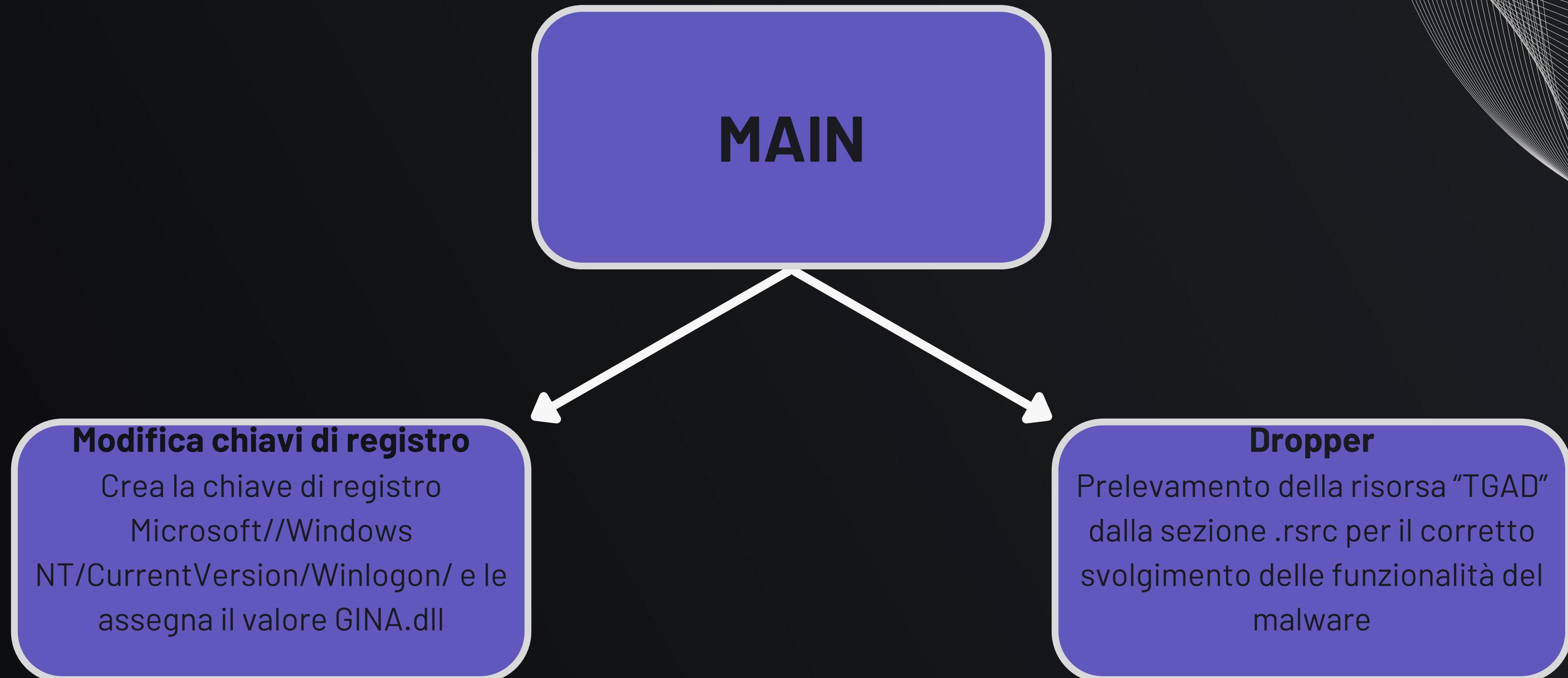
Il *dropper* è un malware che contiene al suo interno un altro malware o componenti addizionali/aggiuntivi utili per il corretto svolgimento delle funzionalità che il malware implementa, in questo caso abbiamo all'interno della sezione **.rsrc** il componente **TGAD** da estrarre.

Questo tipo di funzionalità è anche possibile ipotizzarla con l'utilizzo della sola analisi statica basica, infatti, visualizzando le funzioni importate con CFF Explorer in "Import Directory" nella categoria della libreria KERNEL32.dll possiamo notare la presenza di queste funzioni, la sezione **.rsrc** inoltre rafforzerà l'ipotesi precedentemente detta.

# Dropper

004010B8	> A1 30804000	MOV EDX, DWORD PTR DS:[408030]	
004010BD	. 50	PUSH EAX	
004010BE	. 8B0D 34804000	MOV ECX, DWORD PTR DS:[408034]	
004010C4	. 51	PUSH ECX	
004010C5	. 8B55 08	MOV EDX, DWORD PTR SS:[EBP+8]	
004010C8	. 52	PUSH EDX	
004010C9	. FF15 28704000	CALL DWORD PTR DS:[<&KERNEL32.FindResourceA>]	
004010CF	. 8945 EC	MOV DWORD PTR SS:[EBP-14], EAX	
004010D2	. 837D EC 00	CMP DWORD PTR SS:[EBP-14], 0	
004010D6	. 75 07	JNZ SHORT Malware_.004010DF	
004010D8	. 33C0	XOR EAX, EAX	
004010DA	. E9 E0000000	JMP Malware_.004011BF	
004010DF	> 8B45 EC	MOV EAX, DWORD PTR SS:[EBP-14]	
004010E2	. 50	PUSH EAX	
004010E3	. 8B4D 08	MOV ECX, DWORD PTR SS:[EBP+8]	
004010E6	. 51	PUSH ECX	
004010E7	. FF15 14704000	CALL DWORD PTR DS:[<&KERNEL32.LoadResource>]	
004010ED	. 8945 E8	MOV DWORD PTR SS:[EBP-18], EAX	
004010F0	. 837D E8 00	CMP DWORD PTR SS:[EBP-18], 0	
004010F4	. 75 05	JNZ SHORT Malware_.004010FB	
004010F6	. E9 AA000000	JMP Malware_.004011A5	
004010FB	> 8B55 E8	MOV EDX, DWORD PTR SS:[EBP-18]	
004010FE	. 52	PUSH EDX	
004010FF	. FF15 10704000	CALL DWORD PTR DS:[<&KERNEL32.LockResource>]	
00401105	. 8945 F8	MOV DWORD PTR SS:[EBP-8], EAX	
00401108	. 837D F8 00	CMP DWORD PTR SS:[EBP-8], 0	
0040110C	. 75 05	JNZ SHORT Malware_.00401113	
0040110E	. E9 92000000	JMP Malware_.004011A5	
00401113	> 8B45 EC	MOV EAX, DWORD PTR SS:[EBP-14]	
00401116	. 50	PUSH EAX	
00401117	. 8B4D 08	MOV ECX, DWORD PTR SS:[EBP+8]	
0040111A	. 51	PUSH ECX	
0040111B	. FF15 0C704000	CALL DWORD PTR DS:[<&KERNEL32.SizeofResource>]	
00401121	. 8945 F0	MOV DWORD PTR SS:[EBP-10], EAX	
00401124	. 837D F0 00	CMP DWORD PTR SS:[EBP-10], 0	
00401128	. 77 02	JA SHORT Malware_.0040112C	

# Rappresentazione funzionalità



# Giorno 4

## Analisi dinamica

### Avvio malware

#### Osservazioni Post-Avviamento

##### 1.Effetti nella Cartella dell'Eseguibile

- Descrizione delle modifiche rilevate nella cartella contenente l'eseguibile del malware.
- Sintesi delle evidenze raccolte che spiegano tali cambiamenti.

#### Process monitor analysys

##### 2.Filtrate includendo solamente l'attività sul registro di Windows

- Quale chiave di registro viene creata?
- Quale valore viene associato alla chiave di registro creata?
- Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?

#### Funzionamento Malware

Descrizione del comportamento del malware basato sulle evidenze raccolte.

## Osservazioni Post-Avviamento

All'interno della cartella dove è situato il malware, una volta che esso viene avviato, si va a creare una libreria corrotta (caricata da `Malware_Build_Week_U3`) chiamata “**msgina32.dll**” (come possiamo vedere dalla seguente immagine).

 <b>Malware_Build_Week_U3</b>	<b>17/01/2024 17:48</b>	<b>Applicazione</b>	<b>52 KB</b>
 <b>msgina32.dll</b>	<b>04/03/2024 15:11</b>	<b>Estensione dell'a...</b>	<b>7 KB</b>

## Process monitor analysys

Continuando l'analisi del malware, ci accorgiamo che viene creata una chiave di registro con il seguente valore:

...“`C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll`”, ovvero il percorso della libreria corrotta vista precedentemente.



15:11:...	 Malware_Build_...	368	 RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
15:11:...	 Malware_Build_...	368	 RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
15:11:...	 Malware_Build_...	368	 RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
15:11:...	 Malware_Build_...	368	 RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	ACCESS DENIED
15:11:...	 Malware_Build_...	368	 RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS

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

Prima abbiamo notato che all'interno della cartella dove è situato l'eseguibile del malware, viene creato un'altro file.

Andando ad analizzare nel dettaglio questa fase, possiamo notare che:

1. Viene creato il file msgina32.dll (*libreria di windows*).
2. Viene scritto su di esso.
3. Il file viene chiuso.

15:11:...	Malware_Build_...	368	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\vmsgina32.dll	SUCCESS
15:11:...	Malware_Build_...	368	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\vmsgina32.dll	SUCCESS
15:11:...	Malware_Build_...	368	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\vmsgina32.dll	SUCCESS
15:11:...	Malware_Build_...	368	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\vmsgina32.dll	SUCCESS

**Msgina32.dll**: è una libreria di sistema essenziale che gestisce l'interfaccia di autenticazione e di accesso al sistema operativo Windows

# Giorno 5

## Analisi dinamica

### Gina (Graphical Identification and Authentication)

Componente legato di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica

1. Sostituzione DLL Legita con DLL Malevola
  - Cosa accade se il file .dll, viene sostituito da uno .dll malevolo che intercetta i dati inseriti?



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

```
call    _vsnwprintf  
push   offset Mode      ; Mode  
push   offset Filename ; "msutil32.sys"  
call    _wfopen  
mov    esi, eax
```

Per capire cosa succede andiamo ad aprire la dll malevola con IDAPro

Dentro troviamo un riferimento al file **msutil32.sys** preceduto dalla funzione **\_vsnwprintf** e seguito dalla funzione **\_wfopen**

Queste due funzioni servono rispettivamente a scrivere ed aprire un file, questo ci fa supporre che la dll và a modificare il file **msutil32.sys**

**vsnprintf**, **\_vsnprintf**, **\_vsnwprintf**, **\_vsnwprintf\_l**

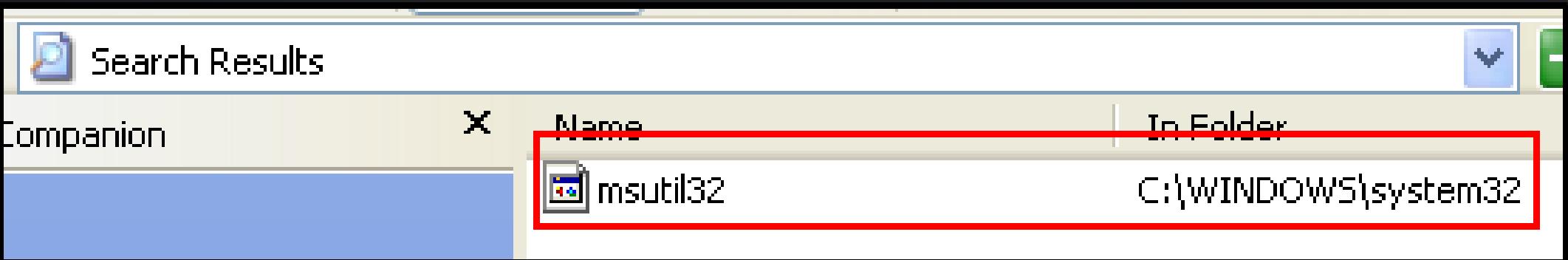
Write formatted output using a pointer to a list of arguments. More secure  
`vsnprintf_s`, `_vsnprintf_s`, `_vsnprintf_s_l`, `_vsnwprintf_s`, `_vsnwprintf_s_l`.

**fopen**, **\_wfopen**

Article • 05/01/2023 • 12 contributors

Opens a file. More-secure versions of these functions are available.

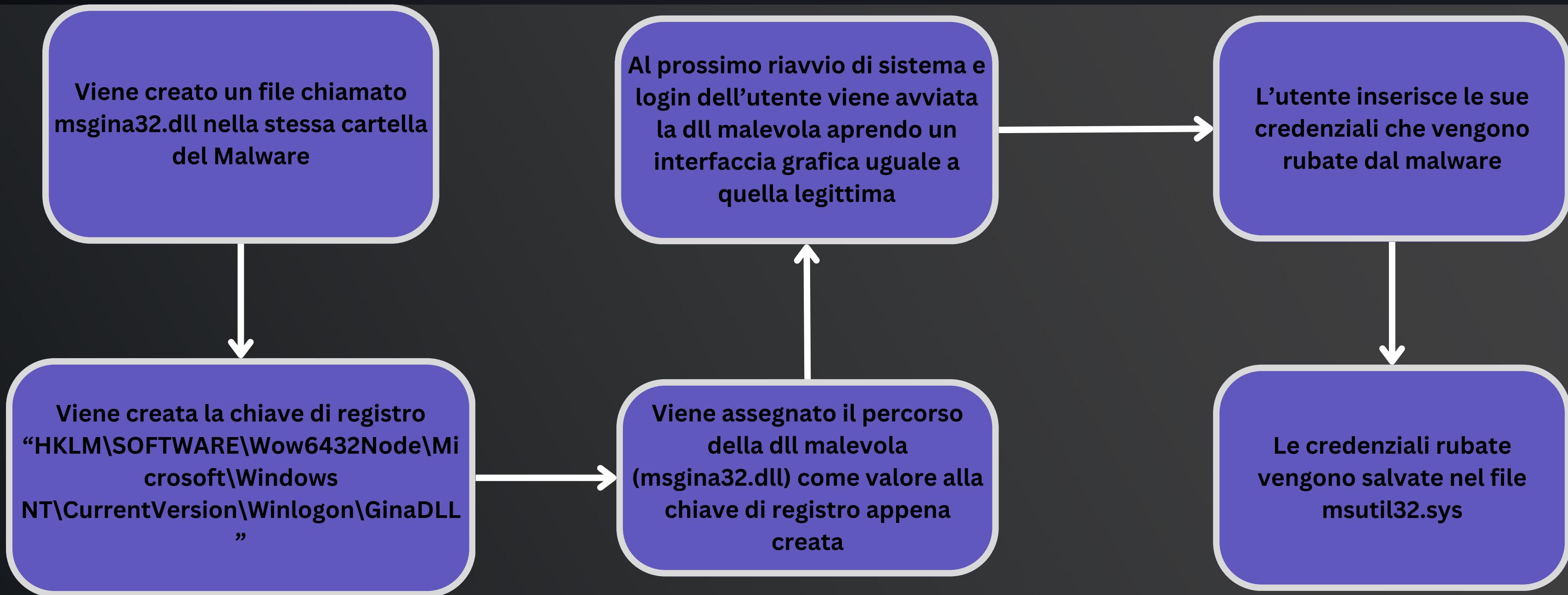
Cercando il file in questione  
nel sistema scopriamo che si  
trova nel percorso  
C:\WINDOWS\system32



Aprendolo con un editor di testo  
scopriamo che all'intero si  
trovano le credenziali rubate  
all'utente.  
Questo conferma che si tratta  
della location in cui vengono  
salvate le credenziali rubate.



# Scopo Malware Ad Alto Livello



# BUILD WEEK 3

# GRAZIE

Marco D'Antoni - Sara Spaccialbelli - Gerardo Carrabs  
Davide Andreozzi - Andrea Mandelli - Mattia Bassani  
Cristian Calvaruso

