



Epicode - Build Week III

CS - 0124

TEAM 6



Leader:

Patrizio Simili

Membri:

Ayman Hani

Francesco Ficetti

Francesco Mineo

Rafael Mango



Indice

- **Giorno 1**

- Traccia
- Parametri e variabili
- Sezioni
- Librerie

- **Giorno 2**

- Traccia
- Locazione 00401021
- Locazione 00401071
- Istruzioni e traduzione in C
- Funzionalità

- **Giorno 3**

- Parametro "ResourceName"
- Funzionalità implementata
- Confronto con analisi statica
- Diagramma di flusso

- **Giorno 4**

- Modifiche alla cartella
- Chiave di registro
- Chiamata di sistema

- **Giorno 5**

- Sostituzione file .dll
- Grafico del profilo

Giorno 1

Traccia

Con riferimento al malware allegato, rispondere ai seguenti quesiti:

- 1) Quanti parametri sono passati alla funzione Main()?
Quante variabili sono dichiarate all'interno della funzione?
- 2) Quali sezioni sono presenti all'interno del file eseguibile?
Descriverle brevemente.
- 3) Quali librerie importa il malware? Per ognuna di esse fare delle ipotesi, sulla base della sola analisi statica, delle funzionalità che il malware potrebbe implementare. Utilizzare le funzioni che sono richiamate all'interno delle librerie per supportare le ipotesi.

1) Parametri e variabili

Il software utilizzato per questa prima parte di analisi è IDA, un software disassembler che permette di tradurre il linguaggio macchina, di cui è composto il file eseguibile, in linguaggio assembly.

Parametri

I parametri hanno un offset positivo rispetto al registro *EBP*.

Variabili

Le variabili hanno un offset negativo rispetto al registro *EBP*.

```
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
```

I parametri individuati sono **3**: *argc*, *argv*, *envp*.

Le variabili individuate sono **5**: *hModule*, *Data*, *var_117*, *var_8*, *var_4*.

2) Sezioni

Il software utilizzato in questa parte di analisi è CFF Explorer, un software che permette di analizzare l'header di un file eseguibile.

Il malware analizzato è composto da 4 sezioni:

- **.text**: contiene le istruzioni di macchina che vengono eseguite dal processore. È la sezione che viene caricata in memoria quando il programma viene avviato.
- **.rdata**: contiene dati di sola lettura, come stringhe costanti, tabelle di costanti e altre informazioni che non vengono modificate durante l'esecuzione del programma.
- **.data**: contiene dati inizializzati, come variabili globali e locali.
- **.rsrc**: contiene le risorse, come icone, stringhe localizzate, bitmap e altri dati associati al programma. Esse possono essere utilizzate per fornire informazioni aggiuntive al programma o per supportare l'interfaccia utente.

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	00003EAB	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040

3) Librerie

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

Fornisce molte funzioni di basso livello, necessarie per gestire le risorse di sistema e fornire servizi di base ai programmi in esecuzione.

Advapi32.dll

Fornisce funzionalità di supporto per la gestione della sicurezza e dei servizi.

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

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

Funzioni kernel32.dll

SizeofResource: il malware potrebbe utilizzare questa funzione per determinare la dimensione delle risorse incorporate all'interno di se stesso, di conseguenza utilizzare queste informazioni per gestire il caricamento o l'estrazione delle risorse in memoria.

VirtualAlloc: questa funzione potrebbe essere utilizzata dal malware per allocare spazio di memoria. Ad esempio, potrebbe essere utilizzata per creare un buffer di memoria in cui inserire il proprio codice dannoso prima di eseguirlo.

GetModuleFileNameA: il malware potrebbe utilizzare questa funzione per ottenere il percorso del proprio eseguibile o di altri file eseguibili presenti sul sistema. Le ragioni sono molteplici, il caricamento di risorse aggiuntive, la modifica o l'infezione di altri file eseguibili, o nascondere la presenza utilizzando nomi di file arbitrari.

Funzioni advapi32.dll

RegSetValueExA: il malware potrebbe utilizzare questa funzione per scrivere dati nel registro di sistema. Potrebbe farlo per creare nuove chiavi di registro, modificare valori di chiavi esistenti o modificare le impostazioni del sistema per favorire la sua operatività o nascondere la propria presenza.

RegCreateKeyExA: questa funzione potrebbe essere utilizzata dal malware per creare nuove chiavi di registro. Queste chiavi potrebbero essere utilizzate per memorizzare configurazioni, impostazioni o informazioni di tracciamento relative al malware stesso.

Giorno 2

Traccia

Con riferimento al malware, spiegare:

- 1) Lo scopo della funzione alla locazione di memoria 00401021 e come vengono passati i parametri alla stessa.
- 2) Spiegare quale oggetto rappresenta il parametro alla locazione 00401017.
- 3) Spiegare il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029 e tradurre il codice assembly nel corrispondente costruito C.
- 4) Valutate la chiamata alla locazione 00401047, qual è il valore del parametro "ValueName"?
Spiegare quale funzionalità sta implementando il malware in questa sezione.



1) Locazione 00401021

La funzione chiamata alla locazione di memoria *00401021* è **RegCreateKeyExA**. Il suo scopo è quello di creare una chiave di registro se non esistente, altrimenti aprirla. I parametri della funzione vengono passati tramite una serie di istruzioni **push**.

2) Locazione 00401017

Il parametro alla locazione 00401017 rappresenta un indirizzo di una chiave di registro, nello specifico: *SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon*.

```
push    0                ; Reserved
push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h        ; hKey
call    ds:RegCreateKeyExA
```

3) Istruzioni e traduzione in C

Agli indirizzi di memoria *00401027* e *00401029* troviamo le seguenti istruzioni:

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

TEST

L'istruzione **test** è utilizzata per eseguire un'operazione logica **and** tra due operandi, impostando il valore di alcuni flag del processore in base al risultato. Modifica il ZF (Zero Flag), che viene settato ad 1 se il risultato dell'and è 0.

JZ

L'istruzione **jz** è utilizzata per effettuare un salto condizionale alla locazione di memoria indicata, solo se il ZF è impostato ad 1.

Di seguito la traduzione, in linguaggio C, del codice assembly analizzato.

```
if (eax == 0) {
    goto loc_00401032;
} else {
    eax = 1;
    goto loc_0040107B;
}
```

Se il valore del registro `eax` è uguale a 0, salta alla locazione di memoria indicata, altrimenti, imposta il valore del registro ad 1 e salta alla locazione di memoria indicata.

4) Funzionalità

Il valore del parametro **ValueName** è **GinaDLL**. Questo induce a pensare che il malware stia cercando di ottenere la persistenza all'interno del sistema. Tale ipotesi sarà confermata o smentita durante delle analisi più approfondite.

Giorno 3

Traccia

Analizzare le routine tra le locazioni di memoria 00401080 e 00401128:

- 1) Qual è il valore del parametro "ResourceName" passato alla funzione FindResourceA()?
- 2) Che funzionalità sta implementando il malware?
- 3) È possibile identificare questa funzionalità utilizzando l'analisi statica basica? In caso di risposta affermativa, elencare le evidenze a supporto.
- 4) Entrambe le funzionalità principali del malware viste finora sono richiamate all'interno della funzione Main(). Disegnare un diagramma di flusso che comprenda le funzioni.

1) Parametro "ResourceName"

Il valore del parametro "ResourceName", passato alla funzione "FindResourceA()", è "TGAD".

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

2) Funzionalità implementata

Il malware sta implementando la funzionalità di dropper, ovvero un programma che contiene un malware al suo interno.

Viene riconosciuto perché, una volta eseguito, effettua una ricerca sulla posizione del malware al suo interno e lo estrae.

Il funzionamento può variare in due modi:

- il malware viene salvato in memoria ed eseguito successivamente;
- il malware viene caricato in memoria ed eseguito al momento.

In questo caso è del secondo tipo, lo si capisce dalle funzioni che vengono richiamate.

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

```
hResource
hModule
LoadResource
```

```
Protect = PAGE_READWRITE
AllocationType = MEM_COMMIT
Size
Address = NULL
VirtualAlloc
```

```
hResource
LookResource
```

```
hResource
hModule
SizeofResource
```

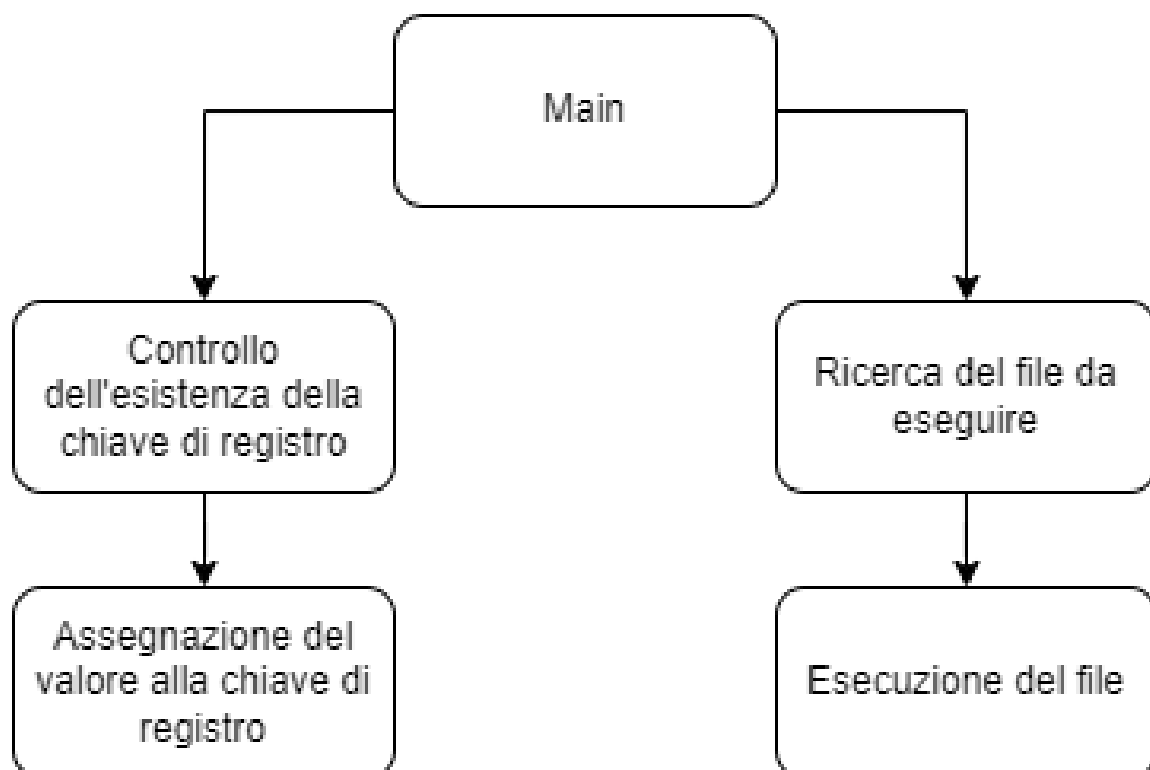
3) Confronto con analisi statica

Durante l'analisi statica basica, si può solamente ipotizzare un comportamento del genere, dato che è limitata alle librerie ed alle funzioni importate dal malware, e non si ha una visione d'insieme sul funzionamento. La certezza la si ha solamente durante l'analisi statica avanzata, quando si analizza il codice.

Analizzando il file con CFF Explorer, si trovano alcuni elementi a supporto della nostra tesi:

- nella sezione **.text**, si può vedere il richiamo alle funzioni citate in precedenza;
- nella sezione **.rsrc**, si può vedere la parte di codice che contiene il malware.

4) Diagramma di flusso



Giorno 4

Traccia

Eseguire il malware:

1) Spiegare che cosa è successo all'interno della cartella del file eseguibile.

Analizzare i processi:

2) Quale chiave di registro viene creata? Quale valore le viene associato?



3) Quale chiamata di sistema ha modificato il contenuto della cartella del file eseguibile?

4) Unire le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del malware.

1) Modifiche alla cartella

Dopo aver eseguito il malware, all'interno della cartella viene creato un file chiamato ***msgina32.dll***.

Quando il file si avvia, esegue una ricerca all'interno della sezione .rsrc, estrae ed esegue il codice, che crea questo file .dll malevolo.





	Malware_Build_Week_U3	17/01/2024 17:48	Applicazione	52 KB
	msgina32.dll	10/04/2024 10:02	Estensione dell'applicazione	7 KB

2) Chiave di registro

La chiave di registro creata è la seguente:

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

Il valore ad essa assegnato è il percorso del file malevolo creato.





	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

3) Chiamata di sistema

Il file viene creato con la funzione **CreateFile**.

Il suo contenuto viene scritto tramite la funzione **WriteFile**.

Infine, il file viene chiuso e salvato con la funzione **CloseFile**.

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

4) Funzionamento del malware

Una volta avviato, il malware, esegue principalmente queste azioni:

1. Effettua una ricerca, all'interno della sezione **.rsrc** del proprio header, del codice malevolo da eseguire.
2. Alloca uno spazio sulla memoria, per poter eseguire il codice.
3. Esegue il codice, che crea una copia malevola del file che gestisce **GINA** (un componente di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica).
4. Crea una chiave di registro all'interno della cartella **Winlogon** (componente di windows che gestisce gli accessi al sistema).
5. Assegna, come valore di questa chiave, il percorso del file creato in precedenza, in questo modo, il malware ottiene la persistenza all'interno del sistema target.

Giorno 5

Traccia

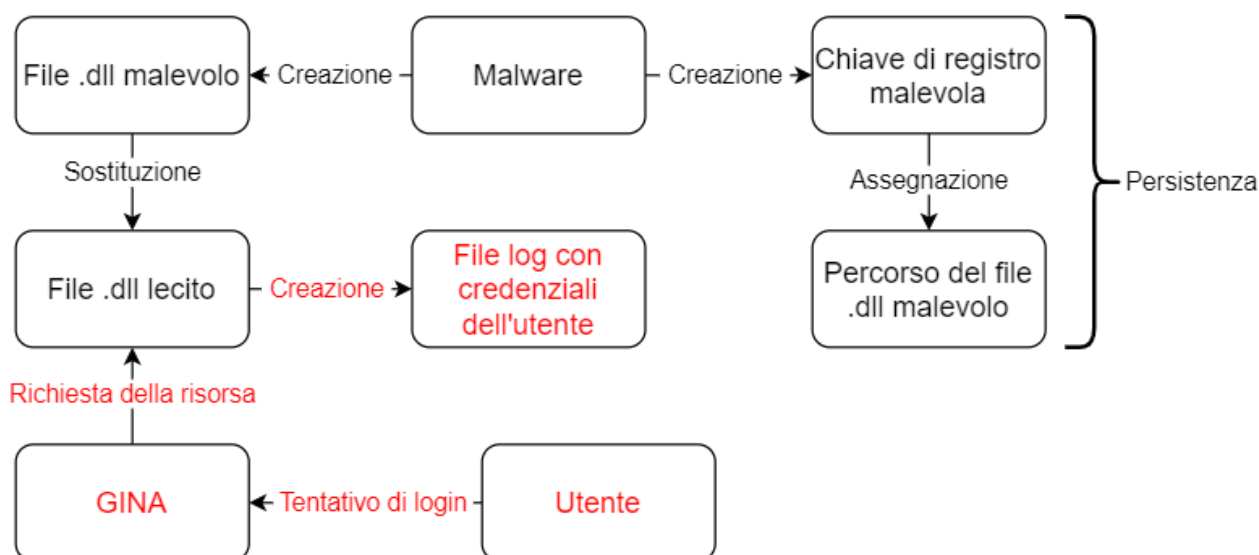
- 1) Cosa può succedere se il file .dll lecito viene sostituito con un file .dll malevolo, che intercetta i dati inseriti?
- 2) Sulla base della risposta precedente, delineare il profilo del malware e delle sue funzionalità. Unire tutti i punti per creare un grafico che ne rappresenti lo scopo ad alto livello.

1) Sostituzione file .dll

Se il malware sostituisce il file .dll lecito che gestisce GIN, con uno malevolo che intercetta i dati inseriti, tutte le password inserite potrebbero essere salvate su un file di log, che potrebbe essere inviato all'attaccante, in modo tale da garantirgli l'accesso alla macchina.

Uno scenario ancora peggiore, si verificherebbe se il computer facesse parte di un dominio. In questo caso, avendo la password di un utente, si ha la possibilità di accedere ad ogni sistema su cui quell'utente ha l'accesso. Inoltre, se sul sistema venissero inserite le credenziali dell'account **administrator**, si otterrebbe l'accesso all'intero dominio.

2) Grafico del profilo



Epicode - Build Week III

**Grazie per
l'attenzione!**