

Aprile 2024



**InfiniTech  
S.p.A.**

# **BUILD** **WEEK III** **MALWARE** **ANALYSIS** **EPICODE**

**TEAM LEADER:**  
Francesco Perticaroli

**TEAM MEMBERS:**  
Manuel Buonanno  
Bruno Falconi  
Flaviano Sedici  
Jacopo Trovato

# Indice

03

**Giorno 1**

11

**Giorno 2**

**EPICODE**

# Giorno 1

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

- **Traccia:**

Con riferimento al file eseguibile “**Malware\_Build\_Week\_U3**”, rispondere ai seguenti quesiti utilizzando i tool e le tecniche apprese nelle lezioni teoriche:

1. Quanti parametri sono passati alla funzione Main()?
2. Quante variabili sono dichiarate all'interno della funzione Main ()?
3. Quali sezioni sono presenti all'interno del file eseguibile?  
Descrivete brevemente almeno 2 di quelle identificate
4. Quali librerie importa il Malware? Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

# Svolgimento

Per l'analisi del malware in questione, abbiamo impiegato **IDA Pro**, un software che esamina il contenuto del malware e restituisce il suo codice in linguaggio assembly. Il codice disassemblato viene visualizzato tramite una barra colorata; ciò che ci interessa è il colore blu, che rappresenta il codice del malware. Una volta completate le operazioni da parte di IDA, sullo schermo comparirà il punto di ingresso dell'applicazione, cioè il punto in cui il programma inizia ad eseguire il suo codice.

Quanti parametri sono passati alla funzione Main()? 3

Nel main della funzione le variabili dichiarate sono tutte quelle che hanno i numeri negativi in verde.

Invece, nella parentesi nera sono presenti i parametri, che hanno valore positivo.

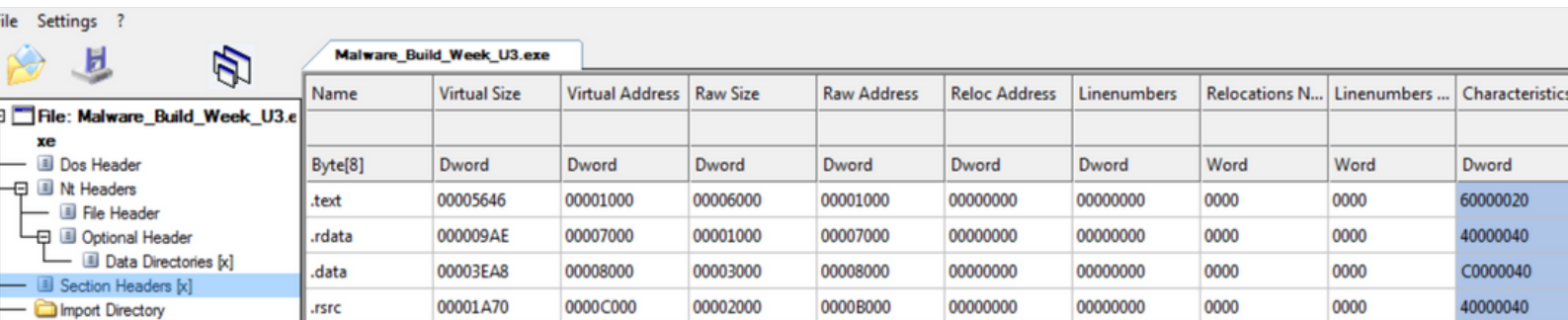
```
.text:004011D0 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:004011D0 _main          proc near          ; CODE XREF: start+AF↓p
.text:004011D0
.text:004011D0 hModule          = dword ptr -11Ch
.text:004011D0 Data            = byte ptr -118h
.text:004011D0 var_117         = byte ptr -117h
.text:004011D0 var_8           = dword ptr -8
.text:004011D0 var_4           = dword ptr -4
.text:004011D0 argc            = dword ptr 8
.text:004011D0 argv            = dword ptr 0Ch
.text:004011D0 envp            = dword ptr 10h
.text:004011D0
```

Quante variabili sono dichiarate all'interno della funzione Main()? 5

Quali sezioni sono presenti all'interno del file eseguibile?

Descrivete brevemente almeno 2 di quelle identificate

Per controllare le sezioni di un file eseguibile spostiamoci nel pannello a sinistra nella sezione «section headers». Il pannello principale a destra mostrerà le informazioni circa le sezioni di cui si compone l'eseguibile.



| 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        |

La figura riporta non solo il nome delle sezioni ma anche altre importanti informazioni, come ad esempio:

- **Virtual size:** indica lo spazio allocato per la sezione durante il processo di caricamento dell'eseguibile in memoria.
- **Rawsize:** indica lo spazio occupato dalla sezione quando è sul disco.

L'header del formato PE fornisce molte altre informazioni importanti oltre alle funzioni/librerie importate ed esportate. Ogni sezione ha un preciso scopo, e conoscerle è una preziosa informazione per le analisi. In questo caso le sezioni del formato PE sono:

- **.text:** contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.
- **.rdata:** include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer.
- **.data:** contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile.
- **.rsrc:** include le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell'eseguibile stesso.

L'**analisi statica basica** consiste nell'esaminare un eseguibile senza vedere le istruzioni che lo compongono. Lo scopo è quello di confermare se un dato file è malevolo e fornire informazioni generiche circa le sue funzionalità. Questa metodologia è sicuramente la più intuitiva e semplice da mettere in pratica, ma risulta anche essere la più inefficiente soprattutto contro malware sofisticati.

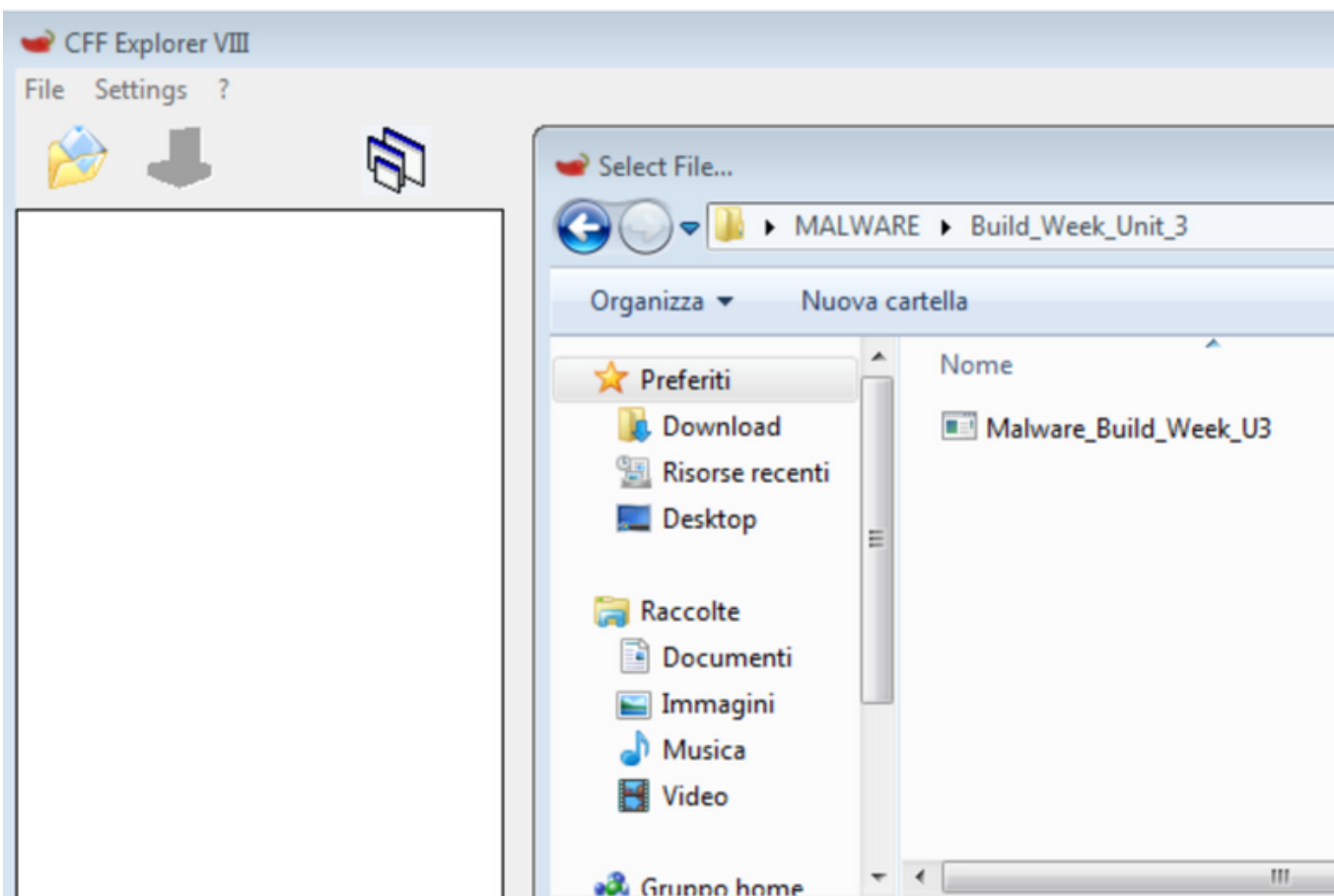
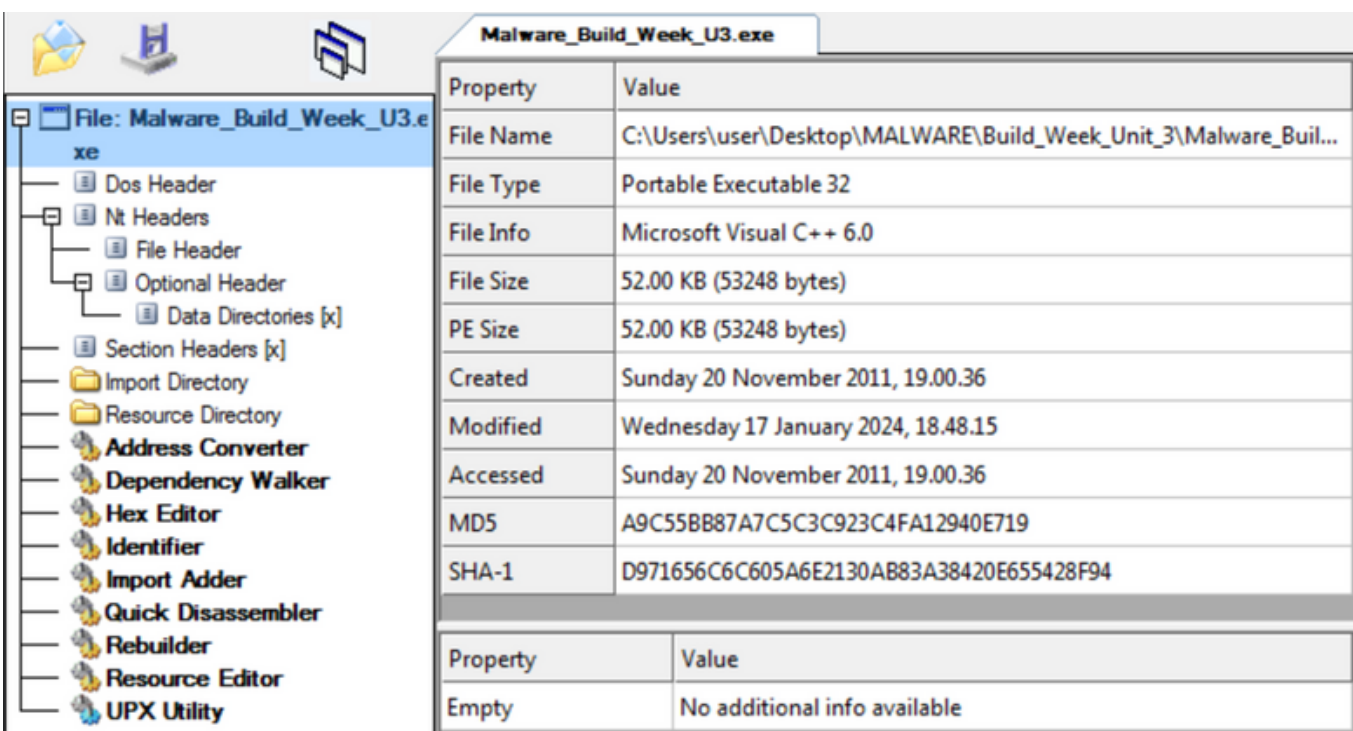
Windows utilizza per la maggior parte dei file eseguibili il formato **PE**, "**PortableExecutable**". Il formato PE al suo interno contiene delle informazioni necessarie al sistema operativo per capire come gestire il codice del file, come ad esempio le librerie e funzioni.

Quando un programma ha bisogno di una funzione "chiama" una libreria al cui interno è definita la funzione necessaria. Oltre alle funzioni importate, un file eseguibile può esportare funzioni. Ovvero, può mettere a disposizione di altri programmi o dell'utente delle funzioni da "chiamare".

L'header del formato PE contiene anche un elenco delle funzioni esportate da un eseguibile. Per controllare le funzioni importate ed esportate da un malware, possiamo utilizzare il tool **CFF Explorer**.

CFF Explorer è uno strumento software progettato per analizzare e modificare i file eseguibili di Windows, come i file EXE, DLL, OCX e molti altri. Il suo nome "**CFF**" sta per "**Cavalry File Format**", che è un formato di file proprietario introdotto da lui stesso. Le principali funzionalità di CFF Explorer includono l'analisi dei file, la modifica dei file, il supporto per plugin ed estensioni, l'esportazione di dati e il controllo della sicurezza.

Questo software è utilizzato principalmente dagli sviluppatori software, dagli analisti di sicurezza e dagli esperti di reverse engineering per analizzare, modificare e comprendere i file eseguibili di Windows.

Interfaccia  
DLL explorerInterfaccia  
del Malware

Possiamo notare come CFF Explorer analizza tutto il file che abbiamo caricato restituendoci molte informazioni su esso.

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.

Per controllare le librerie e le funzioni importate, ci spostiamo su «import directory» nel menù a sinistra. Il pannello darà informazioni sulle librerie importate dall'eseguibile.

In questo caso il malware che abbiamo analizzato importa 2 librerie:

- **Kernel32.dll:** contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria;
- **Advapi32.dll:** contiene le funzioni per interagire con i servizi ed i registri del sistema operativo;

| 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       |
| 00007664 | 00007664  | 00A3 | FindResourceA      |
| 00007604 | 00007604  | 001B | CloseHandle        |
| 000076DE | 000076DE  | 00CA | GetCommandLineA    |
| 000076F0 | 000076F0  | 0174 | GetVersion         |



Andando poi a selezionare queste librerie ci verrà mostrato nel pannello inferiore una lista delle funzioni richieste all'interno della libreria selezionata.

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

Queste sono comunemente utilizzate in un malware per eseguire per ottenere il controllo del sistema. Ecco cosa si può dedurre da ciascuna di esse:

- **szAnsi**: Nome personalizzato assegnato dall'autore del malware.
- **SizeOfResource**: Restituisce la dimensione, in byte, di una risorsa specificata all'interno di un file eseguibile.
- **LockResource**: Blocca una risorsa specificata in memoria e restituisce un puntatore a essa.
- **LoadResource**: Carica una risorsa specificata in memoria.
- **VirtualAlloc**: Alloca memoria in un processo specificato.
- **GetModuleFileNameA**: Restituisce il percorso completo del file eseguibile per il modulo specificato.
- **GetModuleHandleA**: Restituisce un handle per il modulo caricato nel processo chiamante.
- **FreeResource**: Rilascia una risorsa precedentemente caricata in memoria.
- **FindResourceA**: Ricerca una risorsa specificata all'interno di un file eseguibile.
- **CloseHandle**: Chiude un handle aperto.
- **GetCommandLineA**: Restituisce una stringa contenente gli argomenti della riga di comando del programma.
- **GetVersion**: Restituisce informazioni sulla versione del sistema operativo in esecuzione.
- **RegSetValueExA**: Imposta il valore di un'entry nel Registro di sistema di Windows.
- **RegCreateKeyExA**: Crea o apre una chiave nel Registro di sistema.

52  
/ 71

Community Score

52/71 security vendors and no sandboxes flagged this file as malicious

Reanalyze Similar More

57d8d248a8741176348b5d12dcf29f34c8f48ede0ca13c30d12e5ba0384056d7

Size  
52.00 KB

Last Modification Date  
23 hours ago

EXE

Lab11-01.exe

peexe armadillo spreader checks-user-input

DETECTION

DETAILS

RELATIONS

BEHAVIOR

TELEMETRY

COMMUNITY 10

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.doina/totbrick

Threat categories trojan

Family labels doina totbrick genericrcxq

Security vendors' analysis ⓘ

Do you want to automate checks?

|                  |                                   |                    |                                     |
|------------------|-----------------------------------|--------------------|-------------------------------------|
| AhnLab-V3        | ⓘ Trojan/Win32.Agent.C39204       | Alibaba            | ⓘ Trojan:Win32/Totbrick.db39e83f    |
| AliCloud         | ⓘ Backdoor                        | ALYac              | ⓘ Gen:Variant.Doina.65814           |
| Antiy-AVL        | ⓘ Trojan/Win32.Agent              | Arcabit            | ⓘ Trojan.Doina.D10116               |
| Avast            | ⓘ Win32:Trojan-gen                | AVG                | ⓘ Win32:Trojan-gen                  |
| Avira (no cloud) | ⓘ TR/Agent.53248.465              | BitDefender        | ⓘ Gen:Variant.Doina.65814           |
| BitDefenderTheta | ⓘ Gen:NN.ZedlaF.36802.aq4@a0c1rOb | Bkav Pro           | ⓘ W32.AIDetectMalware               |
| ClamAV           | ⓘ Win.Trojan.Agent-595082         | CrowdStrike Falcon | ⓘ Win/malicious_confidence_100% (W) |

- Riferimento:  
VirusTotal

# Giorno 2

- **Traccia**

Con riferimento al Malwarein 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, valutate anche un'altra o altre due righe assembly)
5. Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito 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 Malwarein questa sezione.

# Svolgimento

Lo scopo della funzione chiamata alla locazione di memoria 00401021 è il seguente:

## RegCreateKeyExA

La funzione "RegCreateKeyExA" è una chiamata di API di Windows utilizzata per creare una chiave di registro specificata nel Registro di sistema di Windows.

Il suffisso Ex indica la chiamata alla versione Extended di una libreria, mentre il suffisso "A" indica che questa è una versione della funzione che accetta parametri in formato ANSI (American National Standards Institute), comunemente utilizzato per supportare la codifica dei caratteri ASCII tramite l'utilizzo di un solo byte.

Come vengono passati i parametri alla funzione alla locazione 00401021:

I parametri vengono passati alla funzione tramite i comandi push che si trovano nelle righe precedenti al comando call.

In particolare per eseguire la funzione RegCreateKeyExA sono necessari i seguenti parametri:

C++

```
LSTATUS RegCreateKeyEx(  
    [in] HKEY hKey,  
    [in] LPCSTR lpSubKey,  
    DWORD Reserved,  
    [in, optional] LPSTR lpClass,  
    [in] DWORD dwOptions,  
    [in] REGSAM samDesired,  
    [in, optional] const LPSECURITY_ATTRIBUTES lpSecurityAttributes,  
    [out] PHKEY phkResult,  
    [out, optional] LPDWORD lpdwDisposition  
);
```

```

.text:00401000      push    ebp
.text:00401001      mov     ebp, esp
.text:00401003      push    ecx
.text:00401004      push    0                ; lpdwDisposition
.text:00401006      lea     eax, [ebp+hObject]
.text:00401009      push    eax                ; phkResult
.text:0040100A      push    0                | ; lpSecurityAttributes
.text:0040100C      push    0F003Fh          ; samDesired
.text:00401011      push    0                ; dwOptions
.text:00401013      push    0                ; lpClass
.text:00401015      push    0                ; Reserved
.text:00401017      push    offset SubKey     ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040101C      push    80000002h        ; hKey
.text:00401021      call    ds:RegCreateKeyExA

```

Che oggetto rappresenta il parametro alla locazione 00401017:

### [in] lpSubKey

Nome di una sottochiave aperta o creata da questa funzione. La sottochiave specificata deve essere una sottochiave della chiave identificata dal parametro hKey.

Se **lpSubKey** è un puntatore a una stringa vuota, **phkResult** riceve un nuovo handle alla chiave specificata da hKey.

Questo parametro non può essere NULL.

Il paramentro alla locazione 00401017 rappresenta la sottochiave che verrà modificata dalla chiamata di funzione.

*HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\*

```

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

```

Il significato delle istruzioni comprese tra gli indirizzi 0041027 e 00401029:

Per quanto riguarda le istruzioni comprese tra gli indirizzi 00401027 e 00401029 troviamo nella locazione di memoria 00401027 l'istruzione "test eax, eax". L'istruzione in linguaggio assembly esegue un'operazione AND tra il contenuto del registro eax e sé stesso. Quando viene eseguita l'istruzione avviene un'operazione AND bit per bit tra il contenuto del registro e sé stesso. Se il registro eax contiene zero, l'istruzione imposta il bit **Zero Flag** (ZF) nel registro dei flag a 1. Al contrario, se il registro eax contiene un valore diverso da zero, il bit Zero Flag (ZF) viene impostato a 0. In eax viene impostato il puntatore di memoria (tramite lea) che indica la cella in cui dovrà essere memorizzato il valore di output phkResult della funzione RegCreateKeyExA, nel caso la funzione effettui correttamente l'operazione, tale valore risulterà pari a 0, altrimenti sarà diverso da 0. Subito dopo troviamo un salto condizionale, nello specifico un "jz" ovvero un "Jump if Zero" quindi se eax è pari a 0 allora effettuerà il jz al registro di memoria specificato, se invece eax sarà diverso da zero il salto non verrà effettuato, proseguendo con l'esecuzione del codice.



Ricorrendo alla visualizzazione grafica di IDA Pro possiamo vedere che effettivamente il salto viene effettuato andando poi alla locazione di memoria “loc\_401032”.

Nel caso l'esecuzione del codice prosegua senza effettuare il salto, il sistema effettuerà lo spostamento del valore 1 nel registro eax e un jump non condizionale alla “loc\_40107B”.

Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C:

```
if (RegCreateKeyEx([.....]) != 0) {  
    goto loc_40107B;  
} else {  
    if (RegSetValueEx([.....]) != 0) {  
        goto loc_40107B;  
    } else {  
        goto loc_401062;  
    }  
}
```

```
loc_40107B:  
CloseHandle(hObject);  
hObject = 0;
```

```
loc_401062:  
CloseHandle(hObject);  
hObject = 0;
```

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

Valutate ora la chiamata alla locazione 00401047:

Il valore del parametro passato alla funzione RegSetValueExA come “ValueName” è GinaDLL.

Quindi la modifica che il malware effettua al registro è inerente all'interfaccia grafica di login di Windows.



**GINA** (Graphical Identification and Authentication) è una tecnologia utilizzata nei sistemi operativi Windows per la gestione dell'identificazione e dell'autenticazione degli utenti. GINA fornisce un'interfaccia grafica per l'accesso e può essere personalizzata per supportare metodi di autenticazione alternativi, come ad esempio i lettori di impronte digitali o le smart card. È stata ampiamente utilizzata nelle versioni precedenti di Windows, ma è stata sostituita dal componente Credential Provider nelle versioni più recenti del sistema operativo.

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

la modifica del valore del registro GinaDLL può avere diverse implicazioni a seconda del contesto e di come viene utilizzato il registro:

- **Autenticazione personalizzata:** Il valore GinaDLL nel registro di sistema di Windows è utilizzato per specificare il componente DLL responsabile dell'autenticazione dell'utente. Modificare questo valore potrebbe consentire a un attaccante di sostituire il componente di autenticazione predefinito con uno personalizzato, ad esempio per catturare le credenziali degli utenti.
- **Persistenza del malware:** Un malware può modificare il valore GinaDLL per ottenere persistenza nel sistema, cioè per essere eseguito automaticamente ogni volta che viene effettuato il processo di login.
- **Accesso non autorizzato:** Un attaccante potrebbe modificare il valore GinaDLL per ottenere accesso non autorizzato al sistema, bypassando i controlli di autenticazione o introducendo backdoor nel sistema.



Aprile 2024



**InfiniTech  
S.p.A.**

# **BUILD** **WEEK III**

**GRAZIE PER L'ATTENZIONE**

**TEAM LEADER:**  
Francesco Perticaroli

**TEAM MEMBERS:**  
Manuel Buonanno  
Bruno Falconi  
Flaviano Sedici  
Jacopo Trovato