



Team Di Turo

Malware Analysis

Build Week 3

08-12 Aprile 2024

Indice

- 01 Giorno 1
- 02 Giorno 2
- 03 Presentazione Team
- 04 Giorno 3
- 05 Giorno 4
- Giorno 5
- Conclusioni



Team Di Turo

Presentazione Team



Team leader – Davide Di Turo

Team:

- Alessio D'Ottavio
- Manuel Di Gangi
- Verdiana Germani
- Francesco Valcavi
- Michael Robert Antonio Di Liegghio



Traccia e Requisiti

Giorno 1

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

- Quanti parametri sono passati alla funzione Main()?
- Quante variabili sono dichiarate all'interno della funzione Main()?
- Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate
- 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.

Giorno 3

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

- Qual è il valore del parametro «ResourceName» passato alla funzione FindResourceA();
- 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?

- È possibile identificare questa funzionalità utilizzando l'analisi statica basica ? (dal giorno 1 in pratica)
- 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.

Giorno 2

Con riferimento al Malware in analisi, spiegare:

- Lo scopo della funzione chiamata alla locazione di memoria 00401021
- Come vengono passati i parametri alla funzione alla locazione 00401021;
- Che oggetto rappresenta il parametro alla locazione 00401017
- Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029. (se serve, valutate anche un'altra o altre due righe assembly)
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C .
- 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.

Traccia e Requisiti

Giorno 4

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

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

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

Filtrate includendo solamente l'attività sul registro di Windows

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

Passate ora alla visualizzazione dell'attività sul File System.

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

Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

Giorno 5

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 usernamee password nel classico riquadro Windows, come quello in figura a destra che usate anche voi per accedere alla macchina virtuale.

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

Sulla base della risposta sopra, delineate il profilo del Malwaree delle sue funzionalità.

Giorno 1

IDA (Interactive Disassembler) è un software ampiamente utilizzato per il disassemblaggio e il debugging.

Le principali caratteristiche di IDA includono:

- **Disassemblaggio:** IDA può disassemblare eseguibili binari in codice assembly, fornendo una rappresentazione leggibile dall'uomo del codice macchina.
- **Rappresentazione Grafica:** Offre un'interfaccia grafica per navigare attraverso il codice disassemblato, che aiuta a comprendere le strutture di programma complesse.
- **Analisi:** IDA esegue un'analisi statica del binario, identificando funzioni, variabili e strutture di flusso di controllo. Questo aiuta a comprendere la funzionalità del software.



The screenshot shows the IDA Pro interface with the assembly view open. The code displayed is:

```
; 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

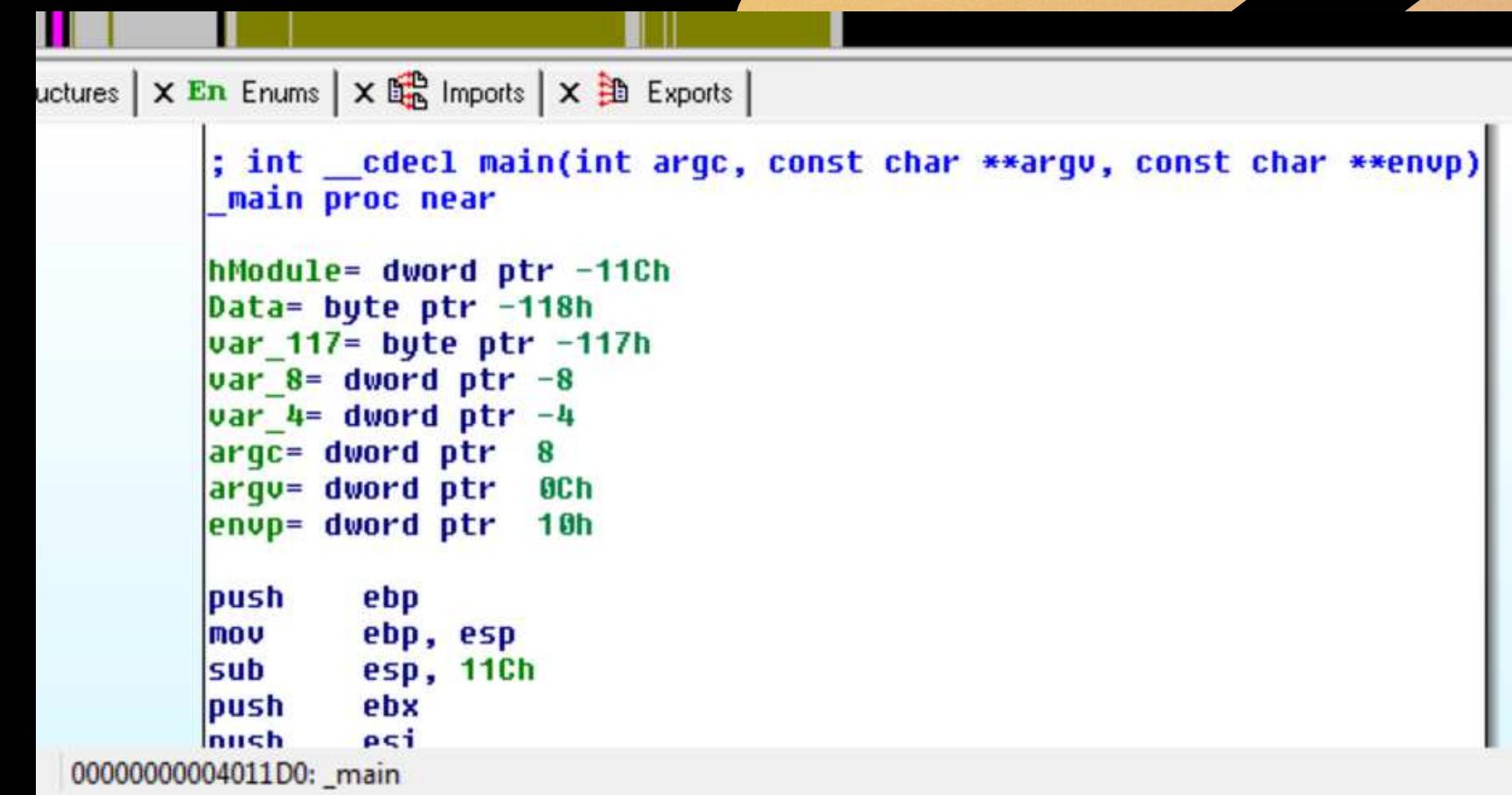
push    ebp
mov     ebp, esp
sub    esp, 11Ch
push    ebx
push    esi
00000000004011D0: _main
```

- **Debugging:** Fornisce funzionalità di debugging, consentendo agli utenti di passare attraverso il codice disassemblato, impostare punti di interruzione, ispezionare la memoria e analizzare il comportamento in fase di esecuzione.
- **Supporto per lo Scripting:** IDA supporta lo scripting utilizzando vari linguaggi di programmazione come IDC (IDA C), Python e IDC Script. Ciò consente di automatizzare compiti e personalizzare il processo di analisi.
- **Ecosistema dei Plugin:** IDA può essere esteso attraverso plugin, che forniscono funzionalità aggiuntive adattate a specifici casi d'uso o compiti di analisi.

Parametri e Variabili

Argomenti nella funzione main:

argc	Indica il numero di entrate nell'array argv
argv	Un array di puntatori ad una stringa che contengono gli argomenti del programma
envp	Un array di puntatori ad una stringa che definisce l'ambiente del programma



The screenshot shows the assembly code for the `_main` function. The code is color-coded to highlight different components: `hModule`, `Data`, `var_117`, `var_8`, `var_4`, `argc`, `argv`, and `envp` are shown in green; `push`, `mov`, `sub`, and `push` are shown in blue; and `ebp`, `esp`, `ebx`, and `esi` are shown in black. The assembly code includes the declaration of the function, local variable declarations, and the main execution flow.

```
; 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
sub    esp, 11Ch
push    ebx
push    esi
000000000004011D0: _main
```

- La funzione main è fornita dall'user ed è dove l'esecuzione del programma ha inizio. La linea di comando al programma è suddivisa in sequenze di tokens o separata da blanks; e sono passate al main come un array di puntatori a stringhe/strings in argv. Il numero di argomenti trovati passa al parametro argc.
- L'argomento argv è un puntatore ad un carattere string contenente il nome del programma. L'ultimo elemento dell'array puntato ad argv è NULL. Gli argomenti contenenti blanks possono essere passati al main racchiudendoli in una citazione (che viene rimossa da quel elemento nel vettore argv).
- L'argomento envp punta ad un array di puntatori a caratteri strings che sono l'ambiente delle stringhe per il corrente processo. Questo valore è identico alla variabile environ definita nel file header <stdlib.h>.

Parametri e Variabili

Nella funzione main troviamo 5 variabili:

hModule	Questa è una variabile usata nel SO Windows per rappresentare un identificatore (handle) associato ad un modulo
Data	Si tratta di una variabile flessibile adatta per memorizzare dati di qualsiasi tipo senza specificare un formato particolare
var_117	Questa potrebbe rappresentare un'area di memoria usata per memorizzare dati temporanei o risultati non definitivi durante l'esecuzione di un programma
var_8	Questa è una variabile che può contenere diversi tipi di dati
var_4	Questa è una variabile che può contenere diversi tipi di dati

```
push    ebx  
push    esi  
push    edi
```

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

Parametri e Variabili

```
push    ebx  
push    esi  
push    edi
```

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

hModule

- Tipo di dato: dword ptr (puntatore a dword, un dword è una parola di 32 bit)
- Offset: -11Ch rispetto al punto di riferimento nella memoria

Data

- Tipo di dato: byte ptr (puntatore a byte)
- Offset: -118h rispetto al punto di riferimento nella memoria

var_117

- Tipo di dato: byte ptr (puntatore a byte)
- Offset: -117h rispetto al punto di riferimento nella memoria

var_8

- Tipo di dato: dword ptr (puntatore a dword)
- Offset: -8 rispetto al punto di riferimento nella memoria

var_4

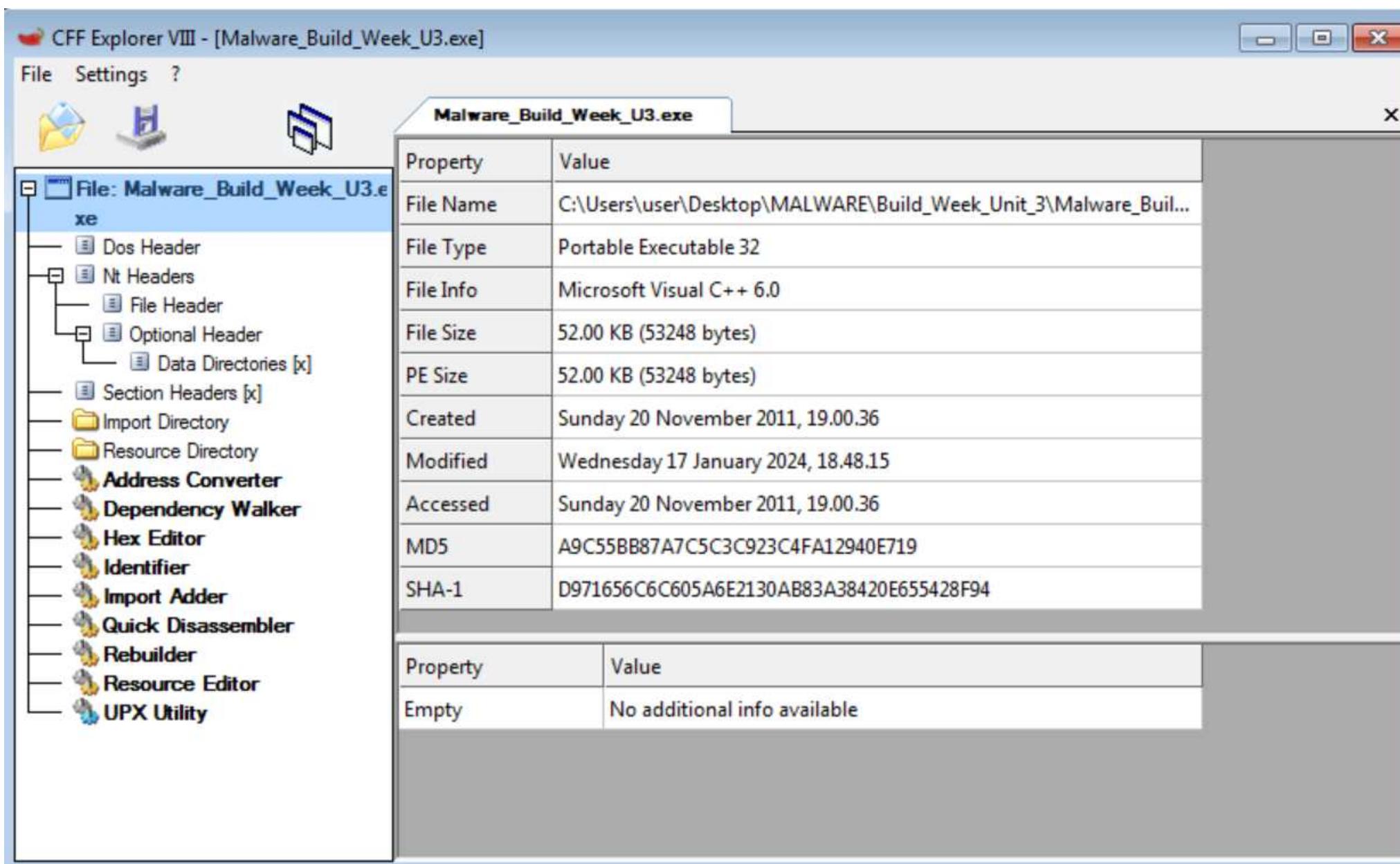
- Tipo di dato: dword ptr (puntatore a dword)
- Offset: -4 rispetto al punto di riferimento nella memoria

Nella descrizione "ptr" sta per puntatore, e dword e byte sono tipi di dato che rappresentano rispettivamente un doppio word (una parola di 32 bit) e un byte (8 bit). Gli offset negativi indicano che le variabili sono posizionate in memoria a una distanza negativa rispetto al punto di riferimento. Questo è un formato comune per la rappresentazione delle variabili in linguaggi di programmazione a basso livello o nell'assembly.

Sezioni

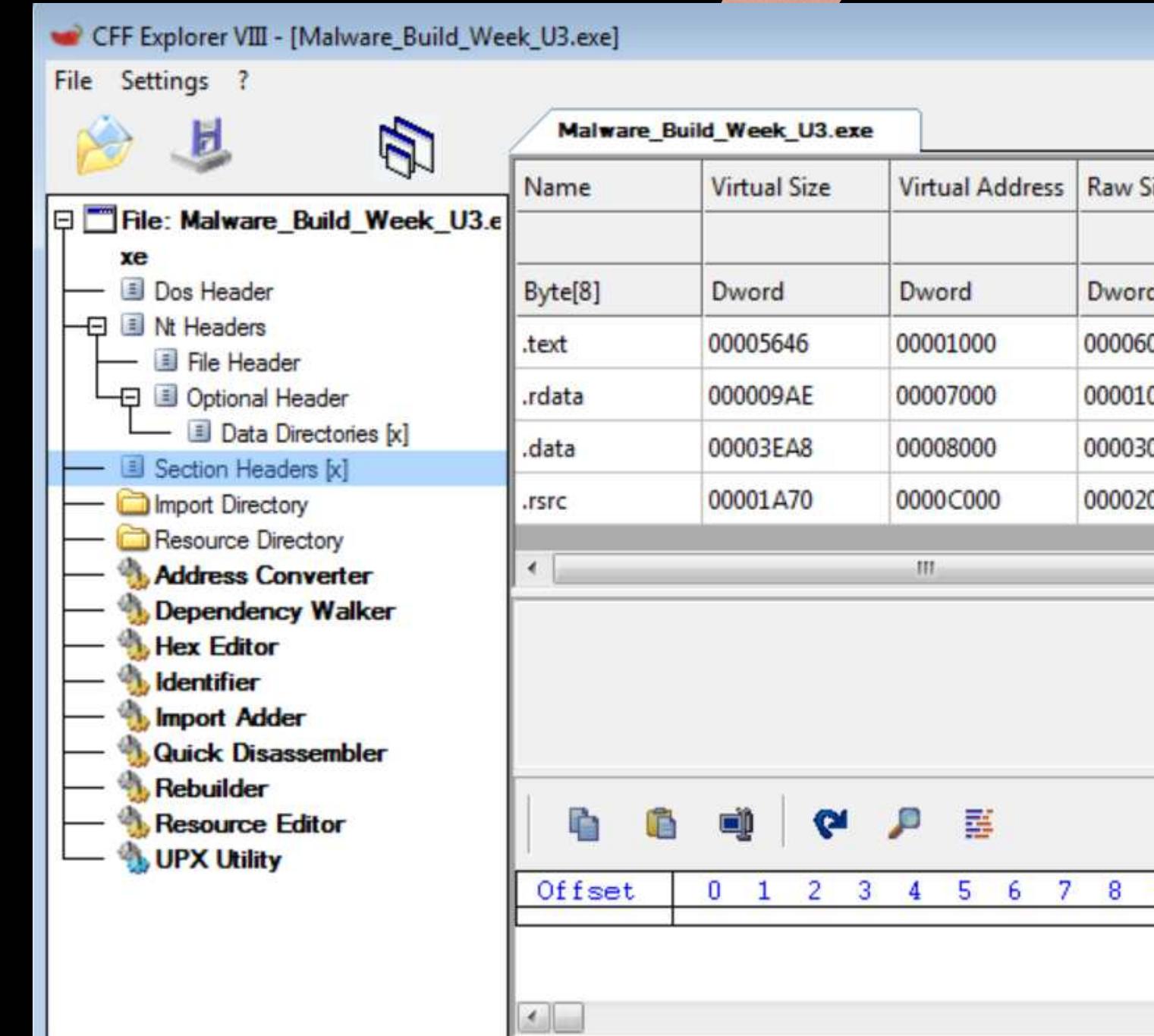


Utilizziamo CFF Explorer per l'analisi dei file eseguibili su Windows; possiamo trovare al suo interno varie funzionalità che ci permettono di analizzare le funzioni, le risorse, le librerie etc.



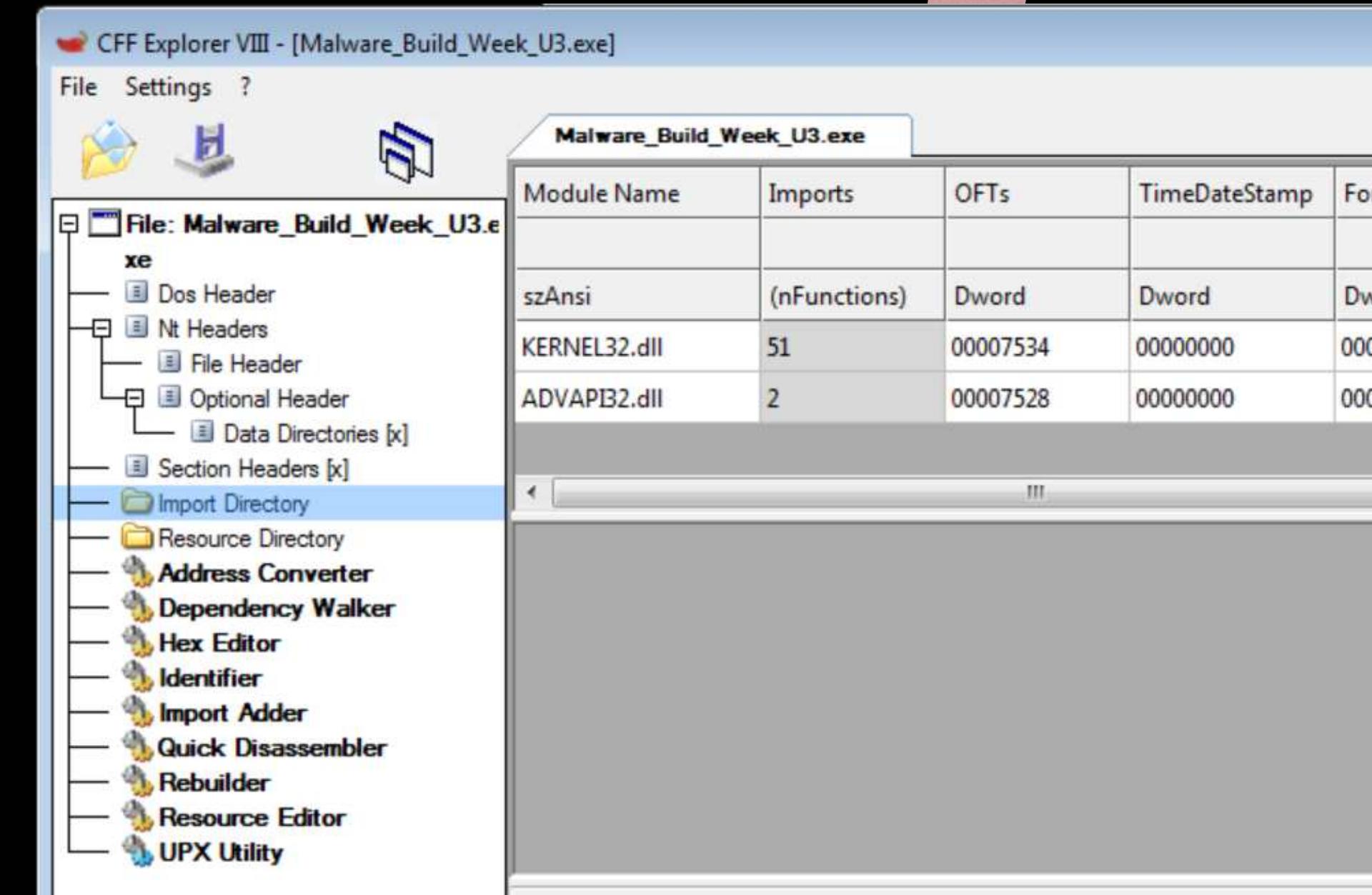
Sezioni

.text	Contiene le istruzioni che il processore eseguirà una volta che il software verrà avviato. Questa è, generalmente, 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 le informazioni circa le librerie e le funzioni importate ed esportate dal file eseguibile
.data	Contiene le variabili globali del programma eseguibile; si tratta di variabili non definite all'interno di un contesto di una funzione, ma bensì globalmente dichiarate e di conseguenza accessibili 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



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



[Back to Agenda Page](#)

Funzioni

Tra le funzioni importate dal malware possiamo vederne alcune del Kernel32.dll:

- **SizeofResource**: essa riporta la dimensione di una risorsa all'interno di un eseguibile o dll.
- **LockResource**: essa viene usata per acquisire un puntatore al cui interno sono presenti i dati di una risorsa bloccata.
- **LoadResource**: carica una risorsa definita all'interno di un modulo.
- **CreateFileA**: viene usata per creare file

Possiamo ora analizzare le funzioni di Advapi32.dll:

- **RegSetValueExA**: permette di aggiungere un nuovo valore all'interno del registro e di settare i rispettivi dati. Accetta come parametri la chiave, la sottochiave e il dato da inserire. Ciò implica che il malware tenta di modificare o aggiungere un valore all'interno del Registro di sistema per poter persistere nel sistema o modificare le configurazioni
- **RegCreateKeyExA**: viene usata per creare una nuova chiave o aprire una chiave esistente nel Registro di sistema. Potrebbe voler dire che il malware tenta di stabilire una persistenza all'interno del sistema o tenta di modificare le impostazioni di sistema.

000078F4	000078F4	00BF	GetCPIInfo
00007900	00007900	00B9	GetACP
0000790A	0000790A	0131	GetOEMCP
00007916	00007916	013E	GetProcAddress
00007928	00007928	01C2	LoadLibraryA

- **LoadLibraryA**: viene usata per caricare un dll
- **GetProcAddress**: permette al programma di ricavare l'indirizzo di una funzione da un dll.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	
			szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource

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

Funzioni

Attraverso queste funzioni possiamo ipotizzare che il comportamento del malware riguardi il caricamento e l'uso di risorse all'interno del sistema, questo può farci pensare al caricamento di dati, asset e alla manipolazione delle risorse di sistema per l'esecuzione di codice malevolo o l'installazione di una backdoor. Inoltre, le funzioni ci permettono anche di comprendere che il malware interagisce con il registro di sistema e usa le funzioni di sistema per acquisire informazioni o probabilmente per modificare le configurazioni di sistema. Dunque, l'analisi effettuata ci porta ad ipotizzare un comportamento tipico di un malware che tenta di ottenere l'accesso e il controllo del sistema attraverso la manipolazione del registro di sistema e l'uso delle risorse di sistema; infatti il registro di Windows è utilizzato per salvare informazioni sul sistema operativo come ad esempio configurazioni del sistema stesso o delle applicazioni che girano sul sistema e i malware lo usano spesso per ottenere la «persistenza».

000078F4	000078F4	00BF	GetCPIInfo
00007900	00007900	00B9	GetACP
0000790A	0000790A	0131	GetOEMCP
00007916	00007916	013E	GetProcAddress
00007928	00007928	01C2	LoadLibraryA

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
.....

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

Giorno 2

Lo scopo della funzione chiamata alla locazione di memoria 00401021

```
IDA View-A | Hex View-A | Structures | Enums | Imports | Exports
.xtext:00401001 mov    ebp, esp
.xtext:00401003 push   ecx
.xtext:00401004 push   0          ; lpdwDisposition
.xtext:00401006 lea    eax, [ebp+hObject]
.xtext:00401008 push   eax          ; phkResult
.xtext:00401009 push   0          ; lpSecurityAttributes
.xtext:0040100A push   0F003Fh    ; samDesired
.xtext:0040100C push   0          ; dwOptions
.xtext:00401011 push   0          ; lpClass
.xtext:00401013 push   0          ; Reserved
.xtext:00401015 push   offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\"
.xtext:00401017 push   80000002h    ; hKey
.xtext:0040101C push   call ds:RegCreateKeyExA
.xtext:00401021 test   eax, eax
.xtext:00401022 jz    short loc_401032
.xtext:00401023 mov    eax, 1

00001001 0000000000401001: sub_401000+1
```

lea eax, [ebp+hObject]	Carica l'indirizzo effettivo di [ebp+hObject] nel registro eax. Questo probabilmente sta configurando un puntatore in cui la funzione memorizzerà un handle alla chiave di registro appena creata o aperta.
push eax	Inserisce il valore di eax nello stack, probabilmente passando il puntatore a phkResult.
push 0	Inserisce un valore nullo nello stack.
push 0F003Fh	Inserisce il valore 0F003Fh nello stack. Questo rappresenta il parametro samDesired, che specifica il mascheramento di accesso alla sicurezza desiderato per la chiave. 0F003Fh di solito concede l'accesso completo.
push 0	Inserisce un valore nullo nello stack.

In generale, questo frammento di codice sta cercando di creare o aprire una chiave di registro sotto HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion.

```

IDA View-A | Hex View-A | Structures | Enums | Imports | Exports
.x .text:00401001        mov    ebp, esp
.x .text:00401003        push   ecx
.x .text:00401004        push   0          ; lpdwDisposition
.x .text:00401006        lea    eax, [ebp+hObject]
.x .text:00401009        push   eax         ; phkResult
.x .text:0040100A        push   0          ; lpSecurityAttributes
.x .text:0040100C        push   0F003Fh   ; samDesired
.x .text:00401011        push   0          ; dwOptions
.x .text:00401013        push   0          ; lpClass
.x .text:00401015        push   0          ; Reserved
.x .text:00401017        push   offset SubKey ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon"
.x .text:0040101C        push   80000002h   ; hKey
.x .text:00401021        call   ds:RegCreateKeyExA
.x .text:00401027        test   eax, eax
.x .text:00401029        jz    short loc_401032
.x .text:0040102B        mov    eax, 1

```

push 0	Inserisce un valore nullo nello stack.
push offset SubKey	Inserisce l'offset della stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" nello stack. Questo è il nome della chiave di registro da creare o aprire.
push 80000002h	Inserisce il valore 80000002h nello stack. Questo rappresenta il parametro hKey, che specifica l'handle di una chiave aperta attualmente o uno dei valori di handle riservati predefiniti. 80000002h rappresenta HKEY_LOCAL_MACHINE.
call ds:RegCreateKeyExA	Chiama la funzione RegCreateKeyExA. Questa funzione tenterà di creare o aprire la chiave di registro specificata con i parametri forniti.

Come vengono passati i parametri?

Attraverso le istruzioni push, i parametri vengono inseriti nello stack di memoria, che la funzione successivamente utilizzerà.

```
.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\Software\Microsoft\Windows\CurrentVersion\Run"
.text:0040101C          push    80000002h ; hKey
```

Cosa rappresenta l'oggetto all'indirizzo 00401017?

```
* | .text:00401017          push    offset SubKey    ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon"
```

L'elemento alla locazione di memoria 00401017 rappresenta la **chiave di registro** utilizzata dal software dannoso per la sua persistenza sul computer della vittima. Questa chiave, con il percorso "Software\Microsoft\Windows NT\CurrentVersion\WinLogon", viene creata tramite la funzione "RegCreateKeyExA".

Nel codice assembly fornito, il parametro alla locazione 00401017 è l'indirizzo dell'inizio della stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon". Questa stringa rappresenta il percorso della chiave di registro che l'applicazione sta cercando di creare o aprire.

Quando si passa un indirizzo di una stringa come parametro a una funzione, in linguaggio assembly si fa riferimento all'offset dell'inizio della stringa rispetto all'inizio del segmento di dati o di testo del programma. Questo offset viene calcolato durante il tempo di compilazione.

Quindi, offset SubKey indica l'offset dell'inizio della stringa "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" rispetto all'inizio del segmento di dati o di testo dell'applicazione.

Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029

Questa parte di codice assembly esegue un controllo condizionale e alcune istruzioni di salto.

```
* .text:00401027  
* .text:00401029  
* .text:0040102B  
* .text:00401030
```

test	eax, eax
jz	short loc_401032
mov	eax, 1
jmp	short loc_40107B

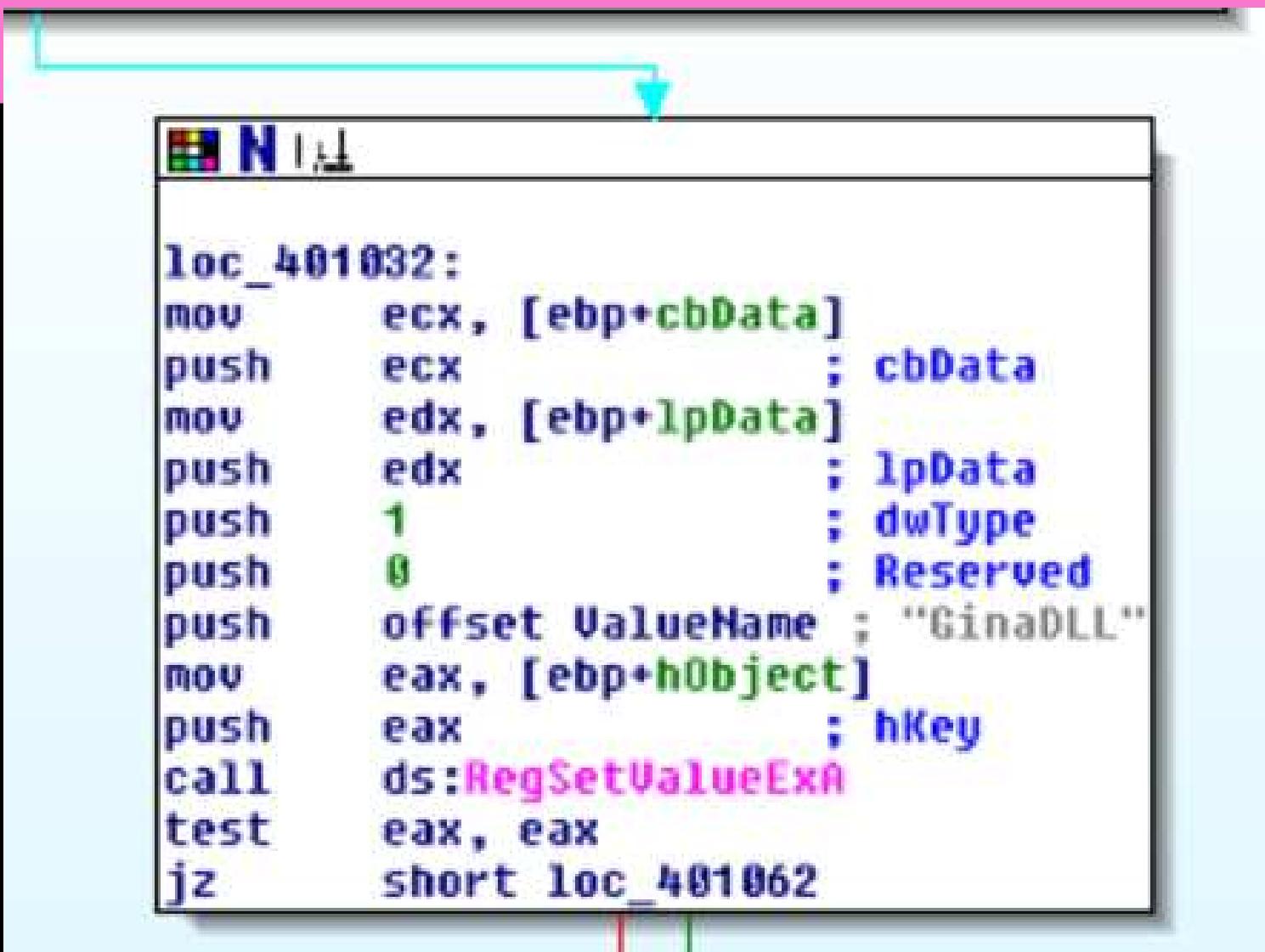
Questo segmento di codice esegue un test su eax, imposta eax a 1 se eax non è zero; quindi, salta all'indirizzo loc_40107B indipendentemente dal risultato del test.

test eax, eax	Questa istruzione effettua un'operazione logica AND tra il registro eax e se stesso. Questo è spesso usato per impostare i flag del processore in base al valore di eax, ma in questo caso, sta semplicemente verificando se il valore di eax è zero.
jz short loc_401032	Questa è un'istruzione di salto condizionato. Se il flag zero (ZF) è impostato a seguito dell'istruzione di test precedente, questo salto porterà l'esecuzione del programma all'indirizzo loc_401032. Altrimenti, l'esecuzione procederà all'istruzione successiva.
mov eax, 1	Se il test precedente ha mostrato che eax non è zero, questo comando imposta il valore di eax a 1.
jmp short loc_40107B	Questo è un salto incondizionato, indipendentemente dal risultato del test. Se l'istruzione mov ha eseguito l'assegnazione a eax, l'esecuzione del programma proseguirà all'indirizzo loc_40107B. Se il test ha portato l'esecuzione a saltare a loc_401032, questa istruzione verrà ignorata.

[Back to Agenda Page](#)

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

Salto visualizzato dal diagramma di flusso:



The screenshot shows a debugger window with assembly code. The code is as follows:

```
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
```

A red arrow points from the 'jz' instruction to the label 'loc_401062', indicating the target of the jump.

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

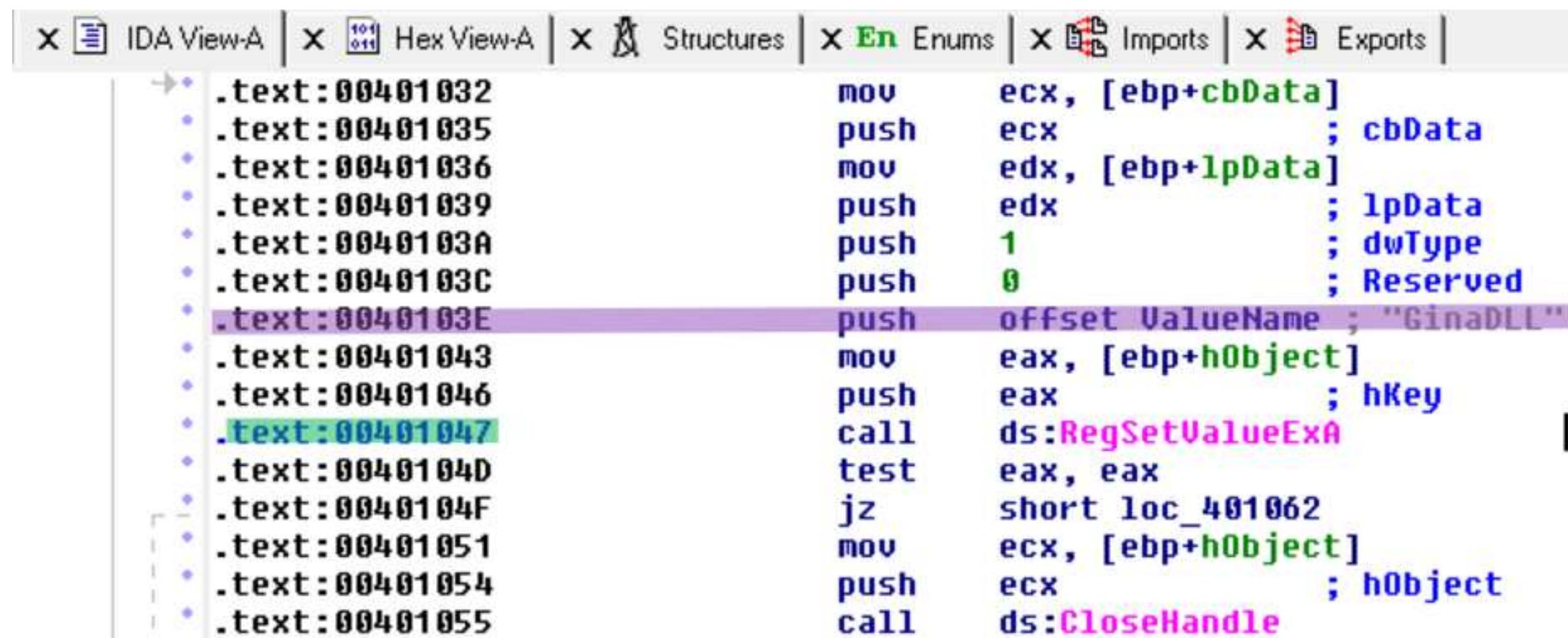
Con questo codice, abbiamo emulato il comportamento dell'istruzione IF nel nostro codice assembly. Abbiamo definito le variabili per eseguire il test tra i due operandi, e successivamente abbiamo impostato delle condizioni basate sul risultato del test. Se il risultato del test fosse 0, il flag ZF (zero flag) verrà impostata a 1 e il salto si verificherà; altrimenti, il flusso del codice continuerà.

```
1 #include <stdio.h>
2
3 int main()
4 {
5
6     int zf; //questa variabile simula la ZF
7     int eax = 12; //questa variabile simula il registro eax
8     int test; //questa simula l'operatore test
9     test = eax - eax; // AND logico
10    if(test == 0){ // inizio del ciclo IF
11        zf = 1;
12    }
13
14    if(zf == 1){ // se la ZF sarà uguale a 1 farà il salto alla locazione "401032"
15        goto loc_401032;
16
17    }else{
18        goto loc_40107B;} // altrimenti continuerà con il codice
19
20    loc_401032: printf("Salto avvenuto");
21    loc_40107B: printf("Salto non avvenuto");
22    return 0;
23 }
```



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

Con riferimento alla chiamata di funzione alla locazione di memoria 00401047, il valore passato al parametro "ValueName" è "GinaDLL".

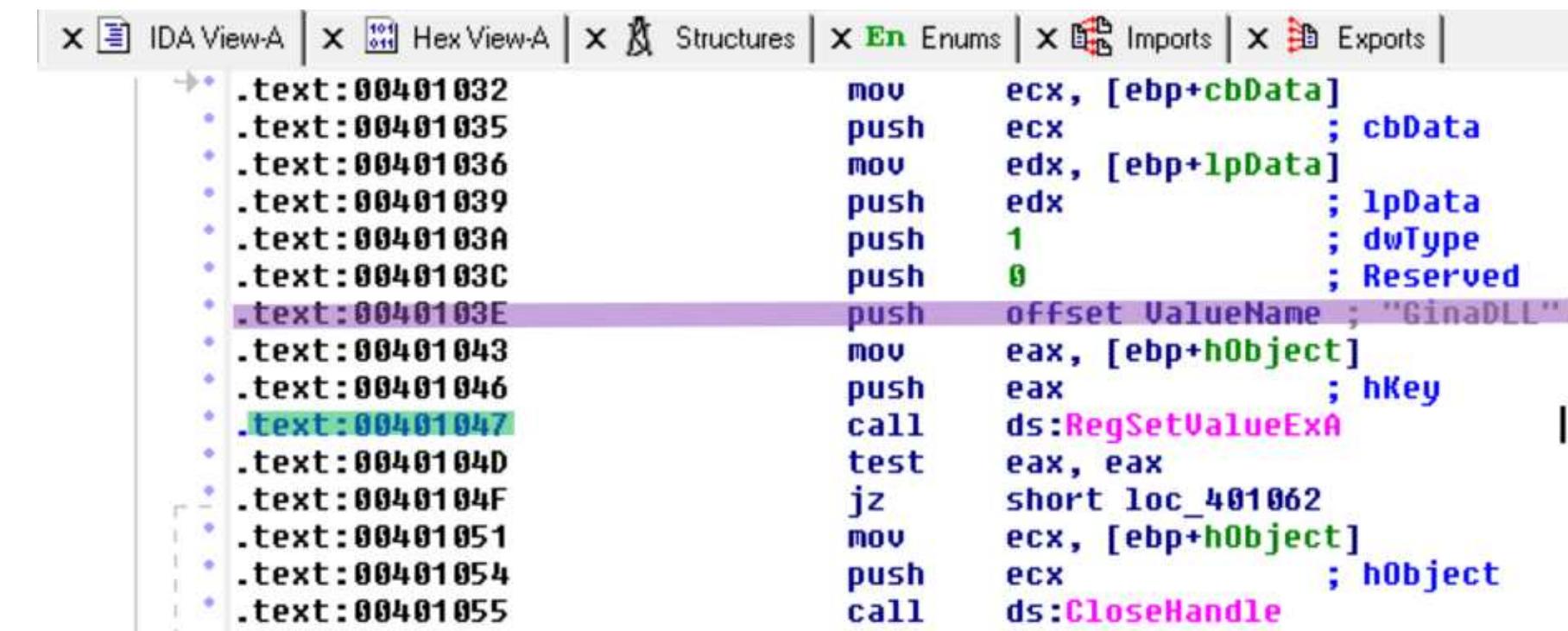


```
IDA View-A | Hex View-A | Structures | Enums | Imports | Exports |  
→ .text:00401032      mov    ecx, [ebp+cbData]  
→ .text:00401035      push   ecx              ; cbData  
→ .text:00401036      mov    edx, [ebp+lpData]  
→ .text:00401039      push   edx              ; lpData  
→ .text:0040103A      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
```



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

Questa sezione del codice assembly implementa una funzionalità di manipolazione del registro di sistema di Windows, probabilmente come parte di una attività dannosa, come l'iniezione di codice malevolo o l'esecuzione di componenti dannosi all'avvio del sistema.



The screenshot shows the assembly view of the malware's code in IDA Pro. The assembly code is as follows:

```
.text:00401032    mov    ecx, [ebp+cbData]
.text:00401035    push   ecx
.text:00401036    mov    edx, [ebp+lpData]
.text:00401039    push   edx
.text:0040103A    push   1
.text:0040103C    push   0
.text:0040103E    push   offset ValueName ; "GinaDLL"
.text:00401043    mov    eax, [ebp+hObject]
.text:00401046    push   eax
.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
.text:00401055    call   ds:CloseHandle
```

In questa sezione del codice assembly, il malware sta manipolando il registro di sistema di Windows. Utilizza la funzione RegCreateKeyExA per creare o aprire una chiave di registro sotto HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion. Successivamente, imposta un valore di registro chiamato "GinaDLL" utilizzando RegSetValueExA. Infine, chiude la handle della chiave di registro se l'operazione ha successo. Questa azione potrebbe essere parte di un'attività dannosa, come l'iniezione di codice malevolo o la modifica delle impostazioni di avvio del sistema.



Giorno 3

Qual è il valore del parametro “**ResourceName**” passato alla funzione **FindResourceA()**

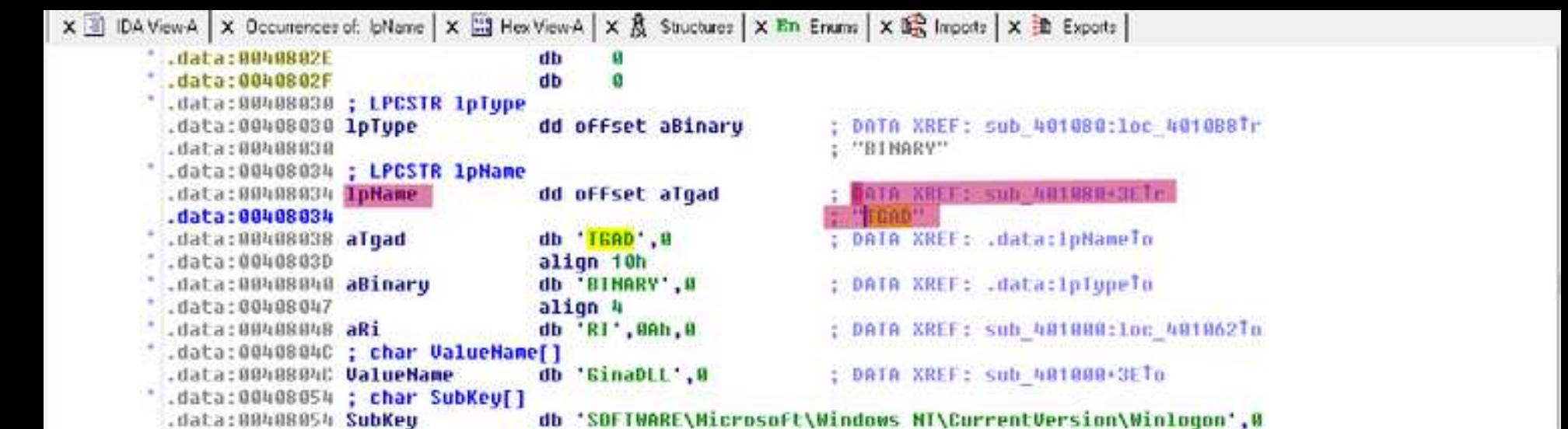


```
.text:004010B8 ; 
.text:004010B8 .text:004010B8 loc_4010B8: 
.text:004010B8     mov    eax, lpType      ; CODE XREF: sub_4010B8+2F1j
.text:004010B8     push   eax          ; lpType
.text:004010BD     mov    ecx, lpName      ; lpName
.text:004010BE     push   ecx          ; lpName
.text:004010C4     mov    edx, [ebp+hModule] ; hModule
.text:004010C5     push   edx          ; hModule
.text:004010C8     push   ds:[FindResourceA]
.text:004010C9     call   [ebp+hResInfo], eax
.text:004010CF     mov    [ebp+hResInfo], 0
.text:004010D2     cmp    [ebp+hResInfo], 0
.text:004010D6     jnz    short loc_4010DF
.text:004010D8     xor    eax, eax
.text:004010DA     jmp    loc_4011BF
.text:004010DF     ; 
```

L'immagine sottostante mostra che il valore "TGAD" viene usato come parametro chiamato **lpName** nella funzione **FindResourceA()**. La funzione XREF viene usata per vedere dove sono chiamate o utilizzate determinate funzioni e oggetti da altre parti del codice.

In questo caso, XREF mostra che il parametro **lpName** è utilizzato nella subroutine che inizia alla locazione di memoria 00401080.

Il valore del parametro "ResourceName" passato alla funzione **FindResourceA()** è contenuto nella variabile **lpName**.



Location	Value	XREFs
.data:0040882E	db 0	
.data:0040802F	db 0	
.data:00408830 ; LPCSTR lpType	dd offset aBinary	; DATA XREF: sub_401080:loc_4010B8Tr ;"BINARY"
.data:00408830 lpType		
.data:00408830		
.data:00408034 ; LPCSTR lpName	dd offset aTgad	; DATA XREF: sub_401080+3E1r ;"TGAD"
.data:00408034 lpName		
.data:00408034		
.data:00408838 aTgad	db 'TGAD',0	; DATA XREF: .data:lpNameTo
.data:0040803D	align 10h	
.data:00408840 aBinary	db 'BINARY',0	; DATA XREF: .data:lpTypeTo
.data:00408047	align 4	
.data:00408848 aRi	db 'RI',0Ah,0	; DATA XREF: sub_401080:loc_4010B62To
.data:0040804C ; char ValueName[]		
.data:0040884C ValueName	db 'BinaDLL',0	; DATA XREF: sub_401080+3ETo
.data:00408054 ; char SubKey[]		
.data:00408054 SubKey	db 'SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon',0	

Qual è il valore del parametro "ResourceName" passato alla funzione FindResourceA()

Per verificare la correttezza della nostra ricerca, abbiamo utilizzato il tool OllyDBG per analizzare il malware. Questo strumento fornisce informazioni più dettagliate e immediate. Quando ci siamo spostati alla locazione di memoria 004010C9, dove è situata la funzione in esame, abbiamo visualizzato questa schermata:

```
004010BD : 50          PUSH EAX
004010BE : 8B0D 34804000 MOU ECX, DWORD PTR DS:[408034]
004010C4 : 51          PUSH ECX
004010C5 : 8B55 08      MOU EDX, DWORD PTR SS:[EBP+8]
004010C8 : 52          PUSH EDX
004010C9 : FF15 28704000 CALL DWORD PTR DS:[<&KERNEL32.FindResourceA>]
```

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

Dall'immagine emerge, evidenziato in verde, che il parametro "ResourceName" passato alla funzione FindResourceA() ha il valore "TGAD".

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?

La sequenza di chiamate di funzione in questa sezione di codice sembra coinvolgere la ricerca e il caricamento di una risorsa da un modulo.

La funzionalità implementata sembra essere legata al recupero e all'utilizzo di risorse da un modulo, che potrebbe essere utilizzato per scopi come il caricamento di file o la manipolazione di risorse.

```
.text:004010DF .text:004010DF loc_4010DF:  
.text:004010DF mov    eax, [ebp+hResInfo]  
.text:004010E2 push   eax      ; hResInfo  
.text:004010E3 mov    ecx, [ebp+hModule]  
.text:004010E6 push   ecx      ; hModule  
.text:004010E7 call   ds:LoadResource  
.text:004010ED mov    [ebp+hResData], eax  
.text:004010F0 cmp    [ebp+hResData], 0  
.text:004010F4 jnz    short loc_4010FB  
.text:004010F6 jmp    loc_4011A5  
  
; CODE XREF: sub_401080+56↑j  
  
.text:004010FB .text:004010FB loc_4010FB:  
.text:004010FB mov    edx, [ebp+hResData]  
.text:004010FE push   edx      ; hResData  
.text:004010FF call   ds:LockResource  
.text:00401105 mov    [ebp+Str], eax  
.text:00401108 cmp    [ebp+Str], 0  
.text:0040110C jnz    short loc_401113  
.text:0040110E jmp    loc_4011A5  
  
; CODE XREF: sub_401080+74↑j
```

La funzione LockResource() è un'API di Windows utilizzata per bloccare l'accesso in memoria ad una risorsa specifica. In particolare, essa prende come argomento un handle di risorsa restituito da LoadResource() e restituisce un puntatore alla risorsa bloccata in memoria.

Nel contesto del frammento di codice fornito, la funzione LockResource() viene utilizzata dopo che una risorsa è stata caricata in memoria con LoadResource(). Questo comportamento suggerisce che il malware stia cercando di ottenere accesso alla risorsa per eseguire operazioni specifiche su di essa.

```
.text:00401113
.text:00401113 loc_401113:
.text:00401113
.text:00401116         mov    eax, [ebp+hResInfo]          ; CODE XREF: sub_401080+8C↑j
.text:00401117         push   eax                           ; hResInfo
.text:00401118         mov    ecx, [ebp+hModule]           ; hModule
.text:00401119         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
```

La funzione SizeofResource() è un'API di Windows utilizzata per ottenere la dimensione, espressa in byte, di una risorsa specifica. Questa funzione prende come argomenti un handle della risorsa (ottenuto solitamente tramite la funzione LoadResource()) e restituisce la dimensione della risorsa.

Nel contesto del frammento di codice fornito, la funzione SizeofResource() viene chiamata dopo che una risorsa è stata bloccata in memoria con la funzione LockResource(). Questo suggerisce che il malware sta cercando di ottenere le dimensioni della risorsa bloccata in memoria per eseguire operazioni specifiche su di essa.

È possibile identificare questa funzionalità utilizzando l'analisi statica basica? In caso di risposta affermativa, elencare le evidenze a supporto.

Utilizzando l'analisi statica basica, potremmo identificare la funzionalità legata al recupero e all'utilizzo di risorse dai moduli nel codice. Le evidenze a supporto potrebbero includere le chiamate di funzione come FindResourceA, LoadResource e SizeofResource, che sono comunemente utilizzate per manipolare risorse nei moduli.

Svolgendo un'analisi statica di base utilizzando lo strumento CFF Explorer, è possibile esaminare le funzioni e le librerie che vengono importate dal malware. Questo ci permette di ottenere una visione dettagliata delle risorse esterne utilizzate dal software dannoso per il suo funzionamento.

The screenshot shows the CFF Explorer interface with the file 'Malware_Build_Week_U3.exe' loaded. The left pane displays the file's structure with nodes like 'File: Malware_Build_Week_U3.exe', 'Dos Header', 'Nt Headers', 'Optional Header', 'Data Directories [x]', 'Section Headers [x]', and 'Import Directory'. The right pane contains a table titled 'Imports' with columns: Module Name, Imports, OFTs, TimeStamp, ForwarderChain, Name RVA, and FTs (IAT). A blue box highlights the 'Import Directory' row for 'KERNEL32.dll' at address 00007534, which corresponds to the 'szAnsi' entry in the table.

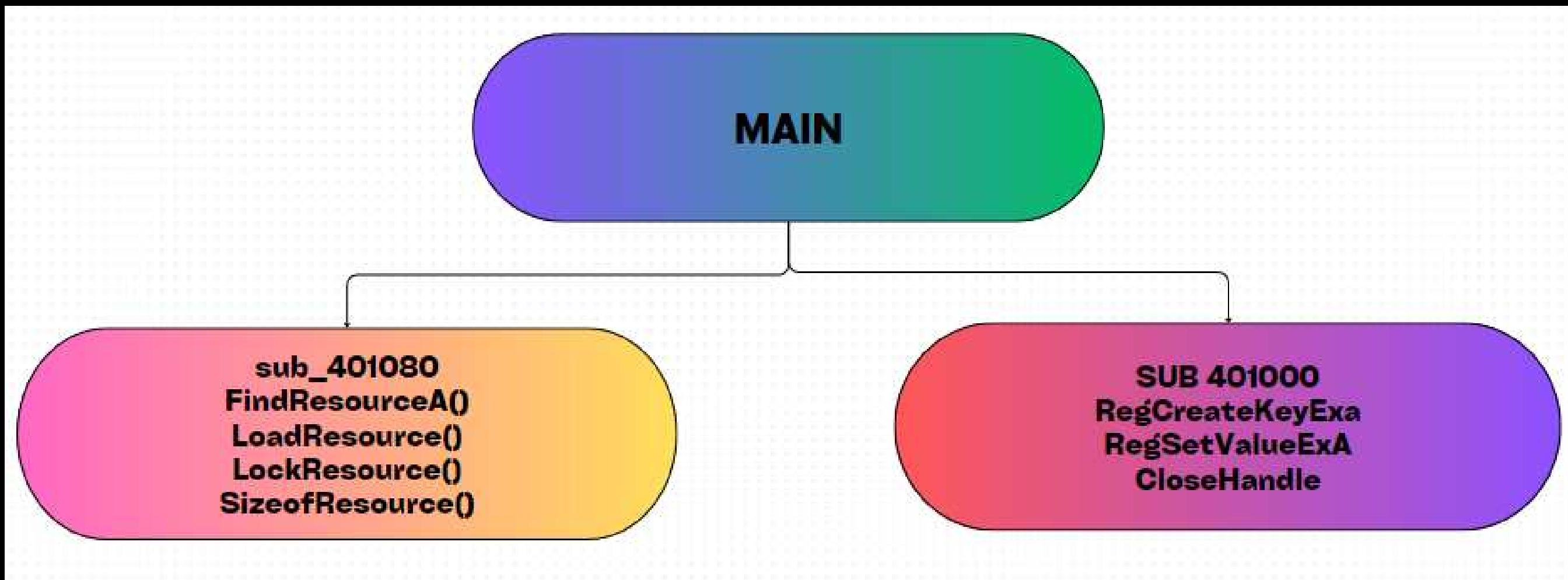
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	0000769E	0000700C
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000

The screenshot shows the CFF Explorer interface with the file 'Malware_Build_Week_U3.exe' loaded. The left pane displays the file's structure with nodes like 'File: Malware_Build_Week_U3.exe', 'Dos Header', 'Nt Headers', 'Optional Header', 'Data Directories [x]', 'Section Headers [x]', and 'Import Directory'. The right pane contains a table titled 'Imports' with columns: OFTs, FTs (IAT), Hint, and Name. A red box highlights several entries: 'SizeofResource' at address 00007632, 'LockResource' at address 00007644, 'LoadResource' at address 00007654, 'FindResourceA' at address 00007664, and 'TerminateProcess' at address 00007734. These are likely the functions identified as manipulating resources.

OFTs	FTs (IAT)	Hint	Name
00007570	00007048	00007734	00007736
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	0105	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	0018	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion
000076FE	000076FE	007D	ExitProcess
0000770C	0000770C	019F	HeapFree
00007718	00007718	011A	GetLastError
00007728	00007728	02DF	WriteFile
00007734	00007734	029E	TerminateProcess

Come possiamo notare dall'immagine sopra, le funzioni che abbiamo analizzato in precedenza sono evidenziate in rosso.

Qui di seguito è presentato un diagramma di flusso semplificato che comprende le tre funzioni principali coinvolte:

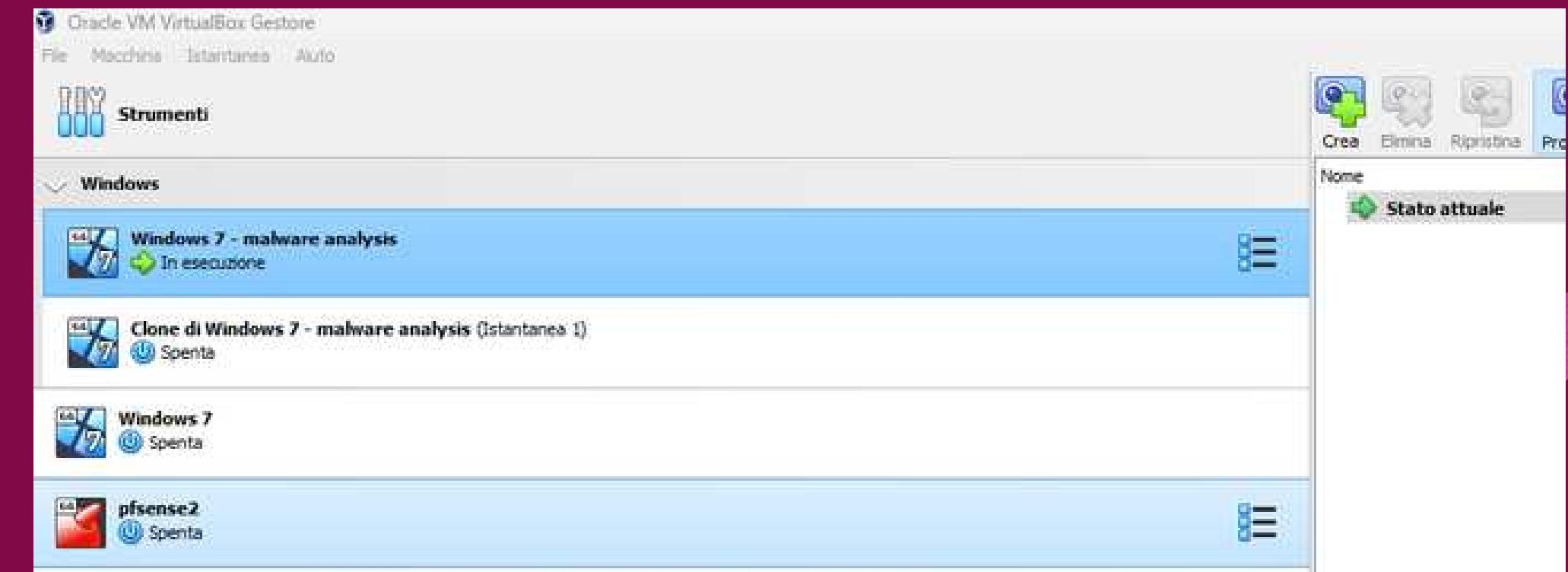


Dopo aver esaminato le chiamate di funzione nel codice in questione, sembra che il malware che stiamo analizzando potrebbe essere un **dropper**. I dropper sono programmi dannosi progettati per installare altri tipi di malware, come virus o backdoor, su un sistema compromesso.

Giorno 4

Analisi Dinamica del Malware

Per l'analisi dinamica del Malware
dobbiamo come prima cosa
occuparci della configurazione della
VM.

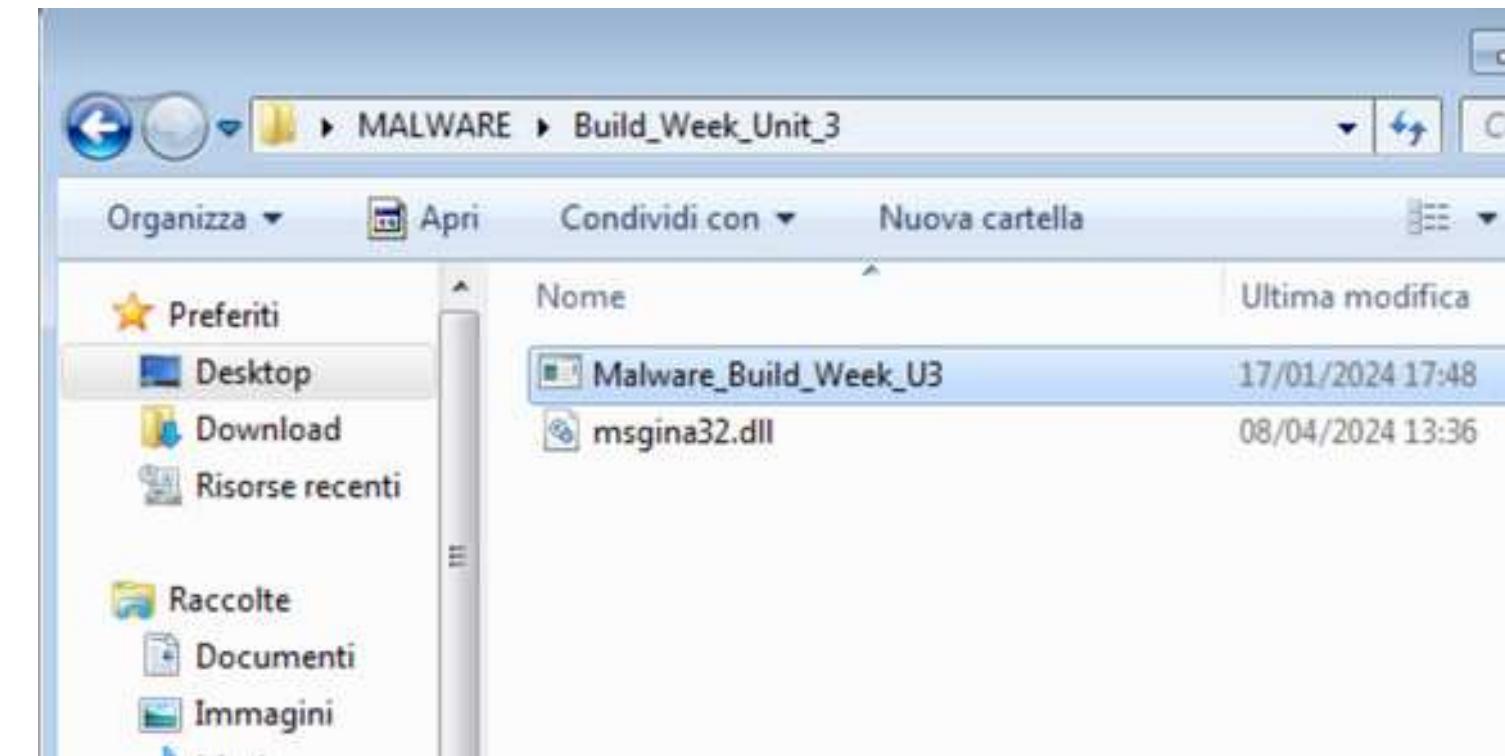


Tra le buone pratiche da adottare per configurare un ambiente sicuro troviamo:

- **Configurazione scheda di rete:** L'ambiente di test non deve avere accesso diretto ad internet, né alle altre macchine presenti sulla rete. Dunque per la configurazione ideale bisognerebbe eliminare le interfacce di rete e abilitare un'interfaccia di rete interna
- **Dispositivi USB:** Quando un dispositivo USB viene collegato alla macchina fisica, esso può essere riconosciuto anche dall'ambiente di test. Al fine di evitare questo comportamento è buona pratica non abilitare o disabilitare il controller USB
- **Cartelle Condivise:** Esse potrebbero essere usate dal malware per propagarsi al di fuori del laboratorio causando danni alla macchina principale e alle macchine sulla rete domestica. Di conseguenza, è consigliato non condividere cartelle tra host e guest
- **Creare Istantanee:** Analizzando i malware spesso capita di arrecare danno all'ambiente di test; è una buona pratica creare delle istantanee della macchina virtuale nel suo stato iniziale, così da ripristinarlo qualora ce ne fosse bisogno.

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

Il team ha effettuato l'esecuzione del Malware che si trova al path:
C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3.



Dopo l'esecuzione abbiamo potuto notare che si crea un file chiamato **msgina32.dll** nella stessa cartella, che è un .dll malevolo che intercetta le credenziali dell'utente.



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

Proseguendo l'analisi con un altro tool - **OllyDBG** - possiamo vedere che il malware va a modificare il registro di sistema per ottenere la persistenza, impostando il valore della chiave GinaDLL sotto HKEY_LOCAL_MACHINE "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" al percorso del dll. In questo modo il dll viene caricato ad ogni logon o logoff effettuato dall'utente.

The screenshot shows assembly code from OllyDbg. The assembly code includes:

- MOV EBP, ESP
- PUSH ECX
- PUSH 0
- LEA EAX, DWORD PTR SS:[EBP-4]
- PUSH EAX
- PUSH 0
- PUSH 0F003F
- PUSH 0
- PUSH 0
- PUSH 0
- PUSH 0
- PUSH Malware_.00408054
- PUSH 00000002
- CALL DWORD PTR DS:[&ADUAPI32.RegCreateKeyExA] - **RegCreateKeyExA**
- TEST EAX, EAX
- JE SHORT Malware_.00401032
- MOV EAX, 1
- JMP SHORT Malware_.0040107B
- MOV ECX, DWORD PTR SS:[EBP+C]
- PUSH ECX
- MOV EDX, DWORD PTR SS:[EBP+8]
- PUSH EDX
- PUSH 1
- PUSH 0
- PUSH Malware_.0040804C
- MOV EAX, DWORD PTR SS:[EBP-4]
- PUSH EAX
- CALL DWORD PTR DS:[&ADUAPI32.RegSetValueExA] - **RegSetValueExA**
- TEST ENV ENV

On the right, the parameters for **RegCreateKeyExA** are listed:

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

On the right, the parameters for **RegSetValueExA** are listed:

- BufSize
- Buffer
- ValueType = REG_SZ
- Reserved = 0
- ValueName = "GinaDLL"
- hKey



Quale chiave di registro viene creata?

Il team ha proseguito con l'analisi del malware su Procmon impostando un filtro per escludere tutti i processi diversi dal malware, e selezionando inoltre il filtro per visualizzare solo le attività di registro.

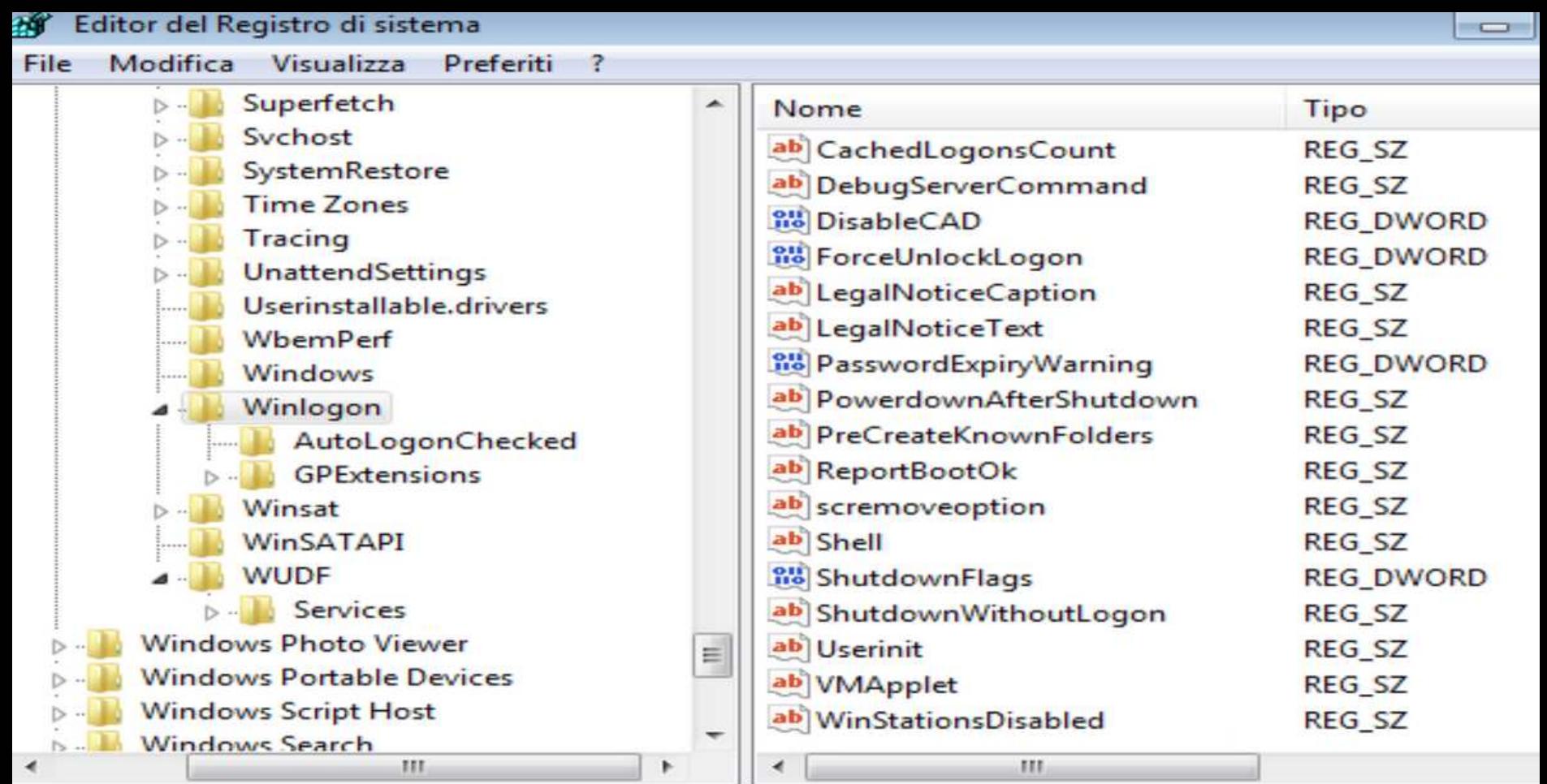
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Diagnos	NAME NOT FOUND	Desired Access: R...
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegQueryKey	HKLM	SUCCESS	Query: HandleTag...
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All...
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	KeySetInformation...
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Query: HandleTag...
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	ACCESS DENIED	Type: REG_SZ, Le...
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS	
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS	
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	SUCCESS	
14:54:... [!] Malware_Build_Week_U3.exe	2948	RegCloseKey	HKLM	SUCCESS	

Possiamo notare le seguenti modifiche apportate dal malware al sistema:

- Creazione del file msgina32.dll nella cartella C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3.
- Modifica del registro di sistema, impostando il valore della chiave GinaDLL sotto HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon al percorso del dll msgina32.dll > notiamo però **ACCESS DENIED**

Quale valore viene associato alla chiave di registro?

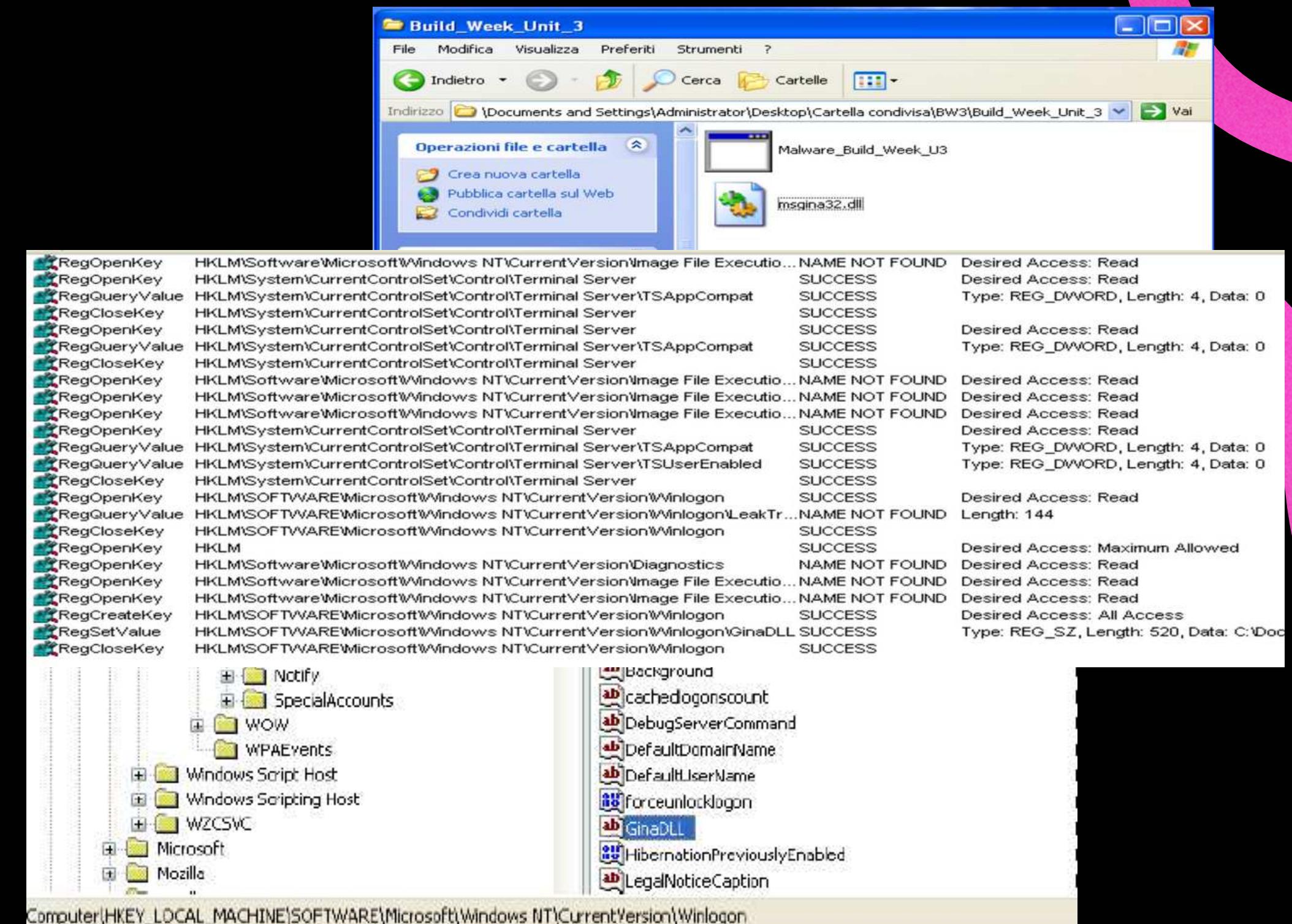
Il prossimo tool da utilizzare è regedit. Questo è un editor del registro di sistema che ci permette di visualizzare i valori associati alle chiavi di registro nel SO Windows.



In questo caso non troviamo GinaDLL, questo ci ha portato ad approfondire la nostra indagine spostandoci su una VM diversa: WindowsXP. Dove abbiamo effettivamente riscontrato il funzionamento corretto del Malware.

Funzionamento Malware su WindowsXP

- Creazione file msgina32.dll nella stessa cartella del malware
- Modifica del registro di sistema, impostando il valore della chiave GinaDLL sotto HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon



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

Tornando a Procmon possiamo comprendere quale chiamata di sistema ha modificato il contenuto della cartella all'interno della quale troviamo il malware; impostiamo il filtro per le attività del sistema:

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\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryBasicInfor...	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll	FILE LOCKED WITH ONLY READERS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\vmsgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\vmsgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\vmsgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\vmsgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryNameInfo...	C:\Windows\System32\apisetschema.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	OpenHandle	C:\Windows\System32\apisetschema.dll	SUCCESS

- CreateFile è la chiamata di sistema con la quale è stato creato il file “vmsgina32.dll”

Giorno 5

GINA (Graphical identification and authentication)

GINA.dll, acronimo di *Graphical Identification and Authentication*, è una componente di Windows che è stata utilizzata in passato per gestire il processo di autenticazione degli utenti. Si tratta di un componente critico del sistema operativo Windows, specialmente nelle versioni precedenti come Windows XP e Windows Server 2003.

La funzione principale di *GINA.dll* era quella di fornire un'interfaccia grafica per l'autenticazione degli utenti durante il processo di accesso al sistema. In sostanza, gestiva la schermata di login in cui gli utenti inserivano le proprie credenziali (nome utente e password) per accedere al sistema.

Tuttavia, con l'introduzione di Windows Vista e versioni successive del sistema operativo, il meccanismo di autenticazione è stato completamente rivisto e il concetto di GINA.dll è stato sostituito da quello di **Credential Providers**. I *Credential Providers* offrono maggiore flessibilità e supporto per una varietà di metodi di autenticazione, inclusi quelli biometrici e basati su certificati.



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

Quando il file GINA.dll legittimo, che gestisce l'autenticazione degli utenti, viene sostituito da un file .dll malevolo progettato per intercettare i dati inseriti, si aprono le porte a una serie di gravi conseguenze:

- **Compromissione della sicurezza:** Il file malevolo potrebbe essere progettato per registrare in modo clandestino i dati di accesso degli utenti, come nomi utente e password, mentre vengono inseriti. Queste informazioni sensibili potrebbero essere inviate a un server remoto controllato dall'attaccante, mettendo a rischio la sicurezza dell'intero sistema e delle informazioni utente.
- **Accesso non autorizzato:** L'attaccante potrebbe utilizzare le credenziali ottenute per accedere in modo non autorizzato al sistema o ai servizi a cui l'utente è registrato. Ciò potrebbe portare a ulteriori violazioni della sicurezza, al furto di dati sensibili o alla compromissione di risorse critiche.



[Back to Agenda Page](#)



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

- **Vulnerabilità del sistema:** La presenza di un file .dll malevolo nel sistema può aprire la strada ad altre forme di attacco o compromissione della sicurezza. Ad esempio, l'attaccante potrebbe utilizzare il file per installare ulteriori malware o per ottenere un accesso persistente al sistema, rendendo difficile la sua rimozione e il ripristino della sicurezza.

In sintesi, la sostituzione del file GINA.dll con un file .dll malevolo rappresenta una grave minaccia per la sicurezza informatica, con il potenziale di compromettere i dati sensibili degli utenti, consentire accessi non autorizzati e causare danni significativi al sistema e agli utenti coinvolti. È cruciale adottare misure di sicurezza robuste, come l'uso di software antivirus e il monitoraggio attivo dei file di sistema, per prevenire e rilevare tali attacchi.



[Back to Agenda Page](#)



Subroutine di WINLOGON

Winlogon è un componente critico del sistema operativo Windows che gestisce il processo di login e logout degli utenti, assicurando un'esperienza di accesso sicura e affidabile. È responsabile della gestione delle interfacce di autenticazione, del caricamento dei profili utente, della gestione delle sessioni e del controllo degli eventi di sistema correlati all'accesso e alla disconnessione degli utenti.

Andando ad analizzare il nuovo file GINA.dll con IDA possiamo vedere come la libreria abbia per sua natura il controllo dell'intero sistema di autenticazione della macchina andando ad eseguire arbitrariamente delle subroutine del componente Winlogon.

```
.text:10001350 ; ===== S U B R O U T I N E =====
.text:10001350
.text:10001350
.text:10001350
.text:10001350
.text:10001350 WlxLoggedOnSAS public WlxLoggedOnSAS
.text:10001350 proc near ; DATA XREF: .rdata:off_1000234810
.text:10001350     push    offset aWlxloggedons_0 ; "WlxLoggedOnSAS"
.text:10001355     call    sub_10001000
.text:1000135A     jmp    eax
.text:1000135A WlxLoggedOnSAS endp
```

Tra le funzioni trovate le più interessanti sono:

- La funzione WlxLoggedOnSAS: viene chiamata quando un utente esegue un'azione Secure Attention Sequence (SAS) mentre è già effettuato il login al sistema. Un SAS può essere un'azione come premere Ctrl+Alt+Canc o effettuare un cambio utente. Quando viene attivata un'azione SAS mentre un utente è già connesso, WlxLoggedOnSAS gestisce l'evento e può eseguire azioni specifiche o modificare il comportamento del sistema in risposta a tale evento. Ad esempio, potrebbe mostrare una finestra di dialogo per confermare l'azione richiesta dall'utente o avviare un'applicazione specifica. Seguendo queste best practice è possibile ridurre significativamente il rischio di infezione da malware dropper e proteggere efficacemente i sistemi e i dati dall'attività dannosa dei malware

Subroutine di WINLOGON

La funzione WlxLogoff: viene chiamata quando un utente esegue il logout dal sistema. Questa funzione gestisce il processo di disconnessione dell'utente, assicurandosi che tutte le risorse associate alla sessione utente vengano liberate correttamente e che l'ambiente di lavoro sia ripulito in modo appropriato.

```
.text:10001360 ; ====== S U B R O U T I N E ======
.text:10001360
.text:10001360
.text:10001360
.text:10001360
.text:10001360 WlxLogoff
.text:10001360
.text:10001365
.text:1000136A
.text:1000136A WlxLogoff
.text:10001360
```

```
public WlxLogoff
proc near
push offset aWlxlogoff_0 ; DATA XREF: .rdata:off_1000234810
call sub_10001000
jmp eax
endp
```

La funzione WlxNegotiate: viene chiamata durante il processo di autenticazione dell'utente per negoziare i parametri di autenticazione tra Winlogon e un processo di autenticazione personalizzato, come un modulo di autenticazione fornito da terze parti. Questa negoziazione è importante perché permette al processo di autenticazione personalizzato di comunicare i suoi requisiti e capacità a Winlogon e di stabilire un protocollo di autenticazione sicuro e appropriato.

```
.text:10001370 ; ====== S U B R O U T I N E ======
.text:10001370
.text:10001370
.text:10001370
.text:10001370
.text:10001370 WlxNegotiate
.text:10001370
.text:10001375
.text:1000137A
.text:1000137A WlxNegotiate
.text:1000137A
```

```
public WlxNegotiate
proc near
push offset aWlxnegotiate_0 ; DATA XREF: .rdata:off_1000234810
call sub_10001000
jmp eax
endp
```

Subroutine di WINLOGON

La funzione WlxNetworkProviderLoad è parte del sottosistema Winlogon di Windows e viene chiamata per caricare un provider di rete durante il processo di autenticazione dell'utente. Tale provider può essere utilizzato per consentire l'accesso a risorse di rete, come file condivisi o stampanti, una volta eseguito l'accesso al sistema. La funzione WlxNetworkProviderLoad garantisce che il provider di rete sia disponibile e caricato correttamente, in modo che l'utente possa accedere alle risorse in modo sicuro e affidabile durante la sessione di lavoro.

Tutte le operazioni viste fin ora potrebbero sembrare lecite visto il funzionamento che svolge normalmente GINA, tuttavia nelle ultime righe di codice riscontriamo che la libreria effettua una chiamata ad un file di sistema denominato msutil32.sys. Questo file sembra avere un nome del tutto lecito, solitamente i file di sistema hanno nomi ben noti e standard, ciò nonostante non c'è alcuna conoscenza diffusa di un file di sistema denominato msutil32.sys. Potrebbe trattarsi di un file malevolo mascherato con un nome simile a uno dei file di sistema di Windows per eludere la rilevazione.

```
.text:10001380 ; ====== S U B R O U T I N E ======
.text:10001380
.text:10001380
.text:10001380
.text:10001380          public WlxNetworkProviderLoad
.text:10001380 WlxNetworkProviderLoad proc near      ; DATA XREF: .rdata:OFF_10002348↓
.text:10001380             push    offset aWlxnetworkpr_0 ; "WlxNetworkProviderLoad"
.text:10001380             call    sub_10001000
.text:10001380             jmp    eax
.text:1000138A WlxNetworkProviderLoad endp
.text:1000138A
.text:1000138A ; -----
.text:1000138C             align 10h
.text:10001390 ; Exported entry 45. WlxReconnectNotify
.text:10001390
```

```
.text:10001593
.text:10001598
.text:1000159D
push    offset Mode      ; Mode
push    offset Filename ; "msutil32.sys"
call    _wfopen
```

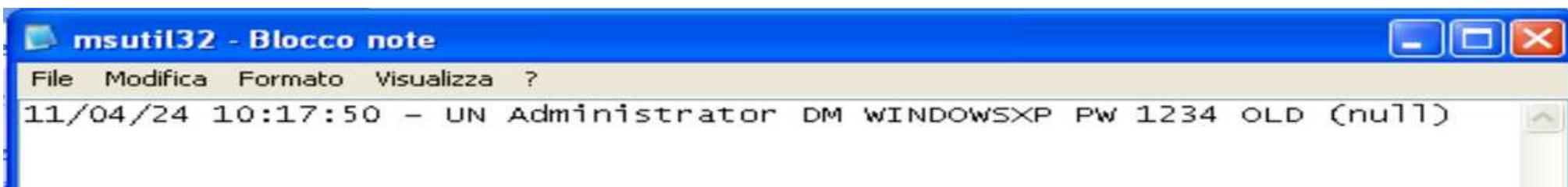
Proseguendo con l'analisi vediamo che la libreria apre il file per andarci a scrivere il nome utente, il dominio, la password e la vecchia password.

```
push    offset Mode      ; Mode
push    offset Filename  ; "msutil32.sys"
call    _wfopen
mov     esi, eax
add    esp, 18h
test   esi, esi
jz     loc_1000164F
```

```
lea     eax, [esp+858h+Dest]
push   edi
lea     ecx, [esp+85Ch+Buffer]
push   eax
push   ecx      ; Buffer
call   _wstrtime
add    esp, 4
lea     edx, [esp+860h+var_828]
push   eax
push   edx      ; Buffer
call   _wstrdate
add    esp, 4
push   eax
push   offset Format  ; "%s %s - %s "
push   esi      ; File
call   _fwprintf
mov     edi, [esp+870h+dwMessageId]
add    esp, 14h
test   edi, edi
jz     short loc_10001637
```

Formatta i valori raccolti all'interno di una stringa e la passa alla subroutine in text:10001570. La subroutine formatta un'altra stringa e la registra in C:\Windows\System32\msutil32.sys."

Verifichiamo che *msutil32.sys* è un file di testo semplice senza codifica, che memorizza le credenziali nel seguente formato:
Data - Ora - Nome Utente - Gruppo - Password - Vecchia Password



Conclusioni

Best Practices:

- **Mantenere software e sistemi aggiornati:** Gli aggiornamenti regolari del sistema operativo, dei browser web e del software antivirus sono fondamentali per proteggere il sistema da vulnerabilità note che potrebbero essere sfruttate dai malware dropper
- **Educazione degli utenti:** Fornire una formazione adeguata agli utenti sull'importanza di evitare link sospetti, scaricare file da fonti non attendibili e aprire allegati di posta elettronica provenienti da mittenti sconosciuti può ridurre significativamente il rischio di infezione da malware dropper
- **Utilizzo di software antivirus:** L'installazione e l'aggiornamento di software antivirus affidabili possono aiutare a rilevare e rimuovere i malware dropper prima che possano danneggiare il sistema
- **Monitoraggio del traffico di rete:** L'implementazione di strumenti di monitoraggio del traffico di rete può consentire di individuare attività sospette o comportamenti anomali all'interno della rete, inclusi i tentativi di distribuzione di malware dropper
- **Backup regolari dei dati:** Mantenere backup regolari dei dati critici su dispositivi di storage separati può aiutare a ripristinare rapidamente i dati in caso di infezione da malware dropper o di altri attacchi informatici
- **Utilizzo di filtri antispam:** Configurare filtri antispam efficaci per ridurre la probabilità che gli utenti ricevano email contenenti malware dropper o altri contenuti dannosi
- **Implementazione di politiche di accesso e autorizzazione:** Limitare l'accesso ai dati e ai sistemi solo agli utenti autorizzati e applicare il principio del privilegio minimo (PoLP) può ridurre la superficie di attacco per i malware dropper
- **Analisi approfondita dei file sospetti:** Prima di aprire o eseguire un file scaricato da Internet o ricevuto via email, è consigliabile eseguire una scansione antivirus approfondita per rilevare eventuali malware.

Seguendo queste *best practice* è possibile ridurre significativamente il rischio di infezione da malware dropper e proteggere efficacemente i sistemi e i dati dall'attività dannosa dei malware

Grazie per l'attenzione



[Back to Agenda Page](#)