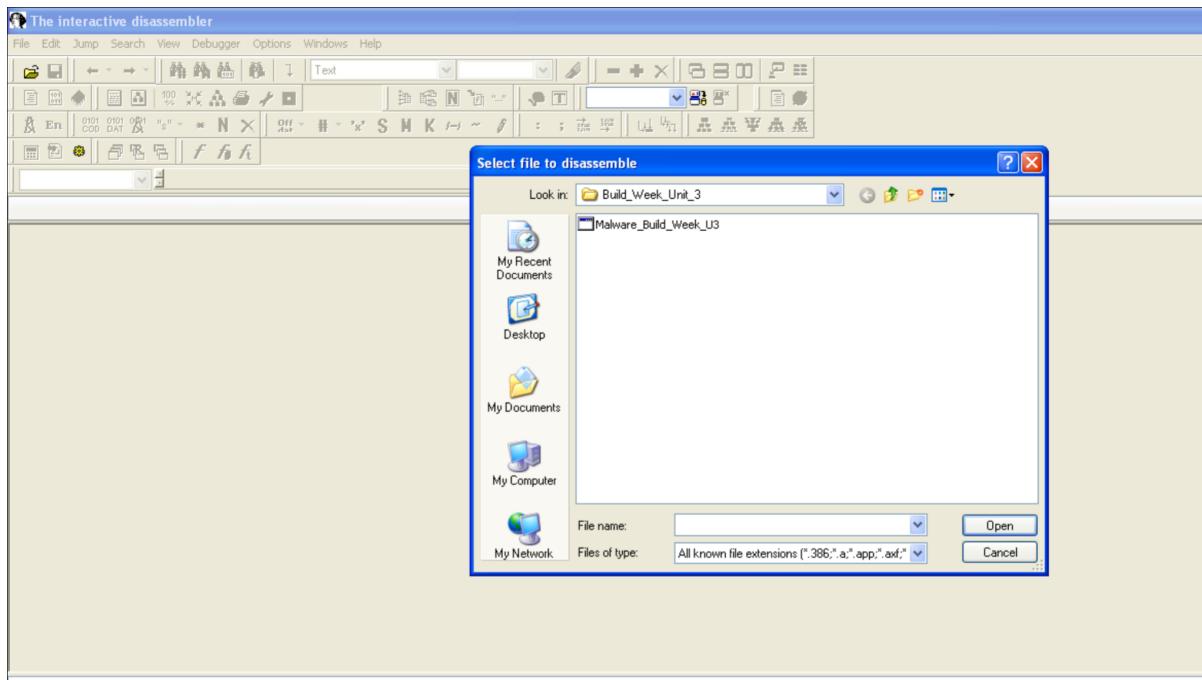


Progetto Modulo 6

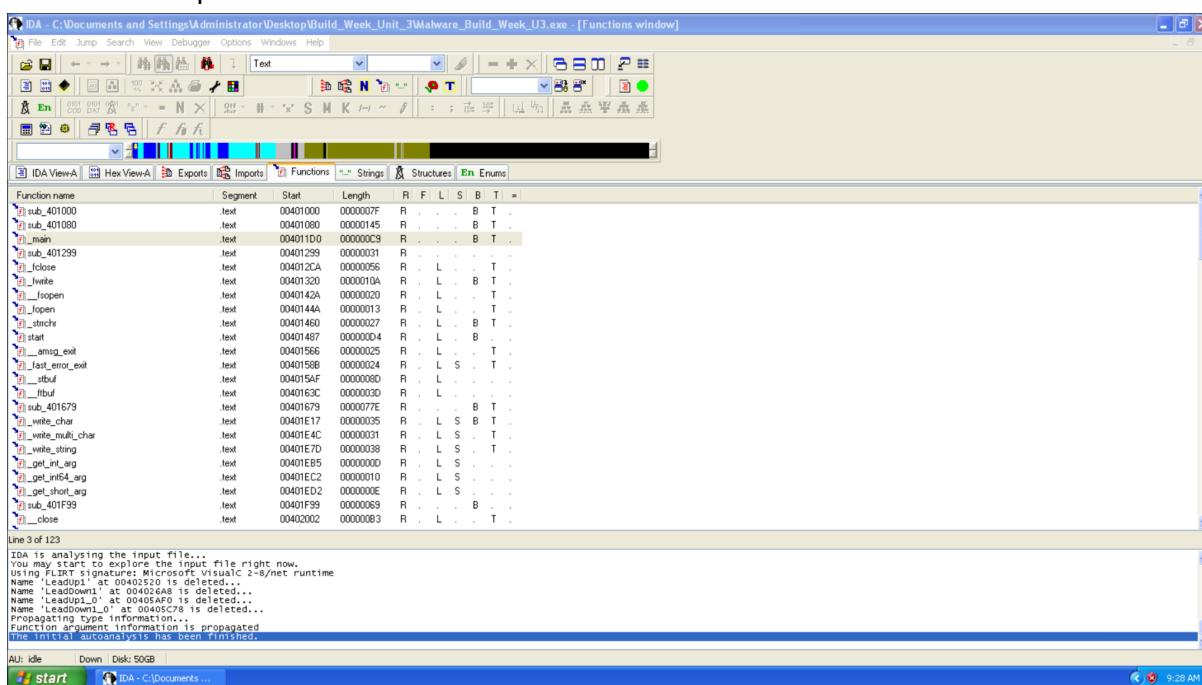
Malware Analysis

Nel progetto di questo modulo ci viene dato un malware da analizzare attraverso i vari software utilizzati nel corso di questo modulo.

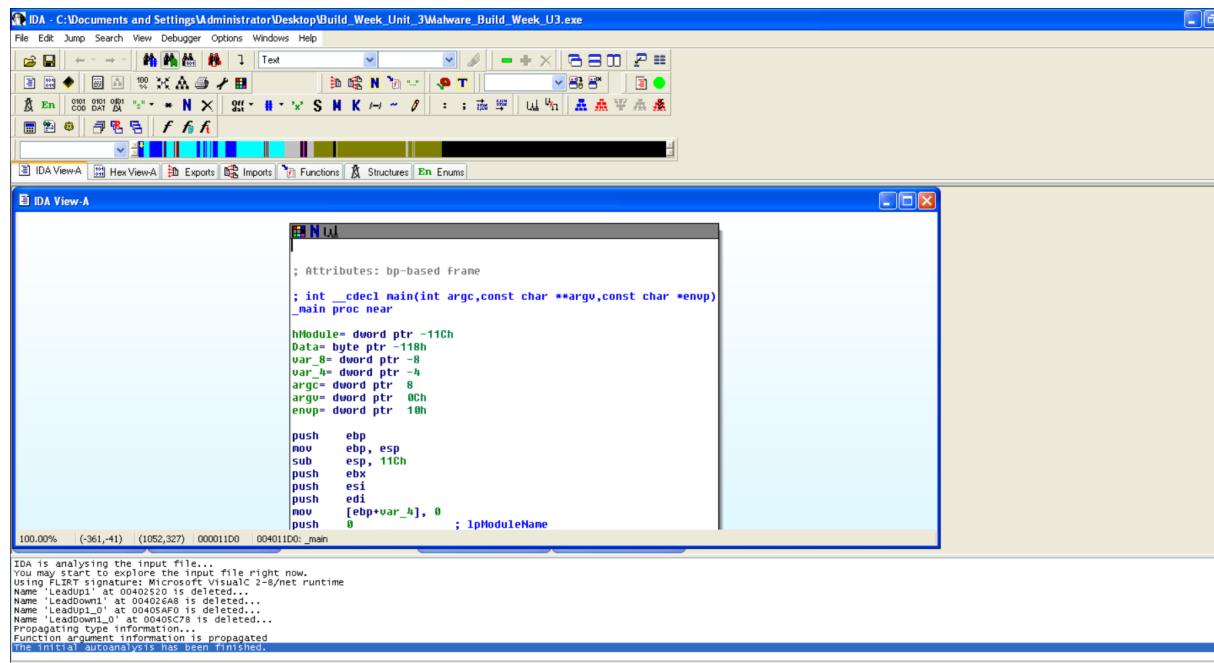
Cominciamo con l'analisi statica del malware, apriamo il nostro malware all'interno di IDA.



La prima richiesta è quella di individuare quali parametri sono passati alla funzione Main. Selezioniamo quindi il menù che racchiude tutte le funzioni



dopo di che andiamo a selezionare main e facendo doppio click



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

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

hModule= dword ptr -11Ch
Data= byte ptr -11Bh
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
push    edi
mov     [ebp+var_4], 0
push    0 ; lpModuleName
```

Below the assembly window, a status bar displays: 100.00% (-361,-41) (1052,327) 000011D0 004011D0:_main.

IDA is analyzing the input file right now.
You can start by opening the input file right now.
Using FLIRT signatures: Microsoft VisualC 2-8/net runtime
Name 'Leadup1_0' at 00405420 is deleted...
Name 'Leaddown1_0' at 00405440 is deleted...
Name 'Leadup1_0' at 004054F0 is deleted...
Name 'Leaddown1_0' at 00405C78 is deleted...
Propagating types...
Function argument information is propagated
The initial autoanalysis has been finished.

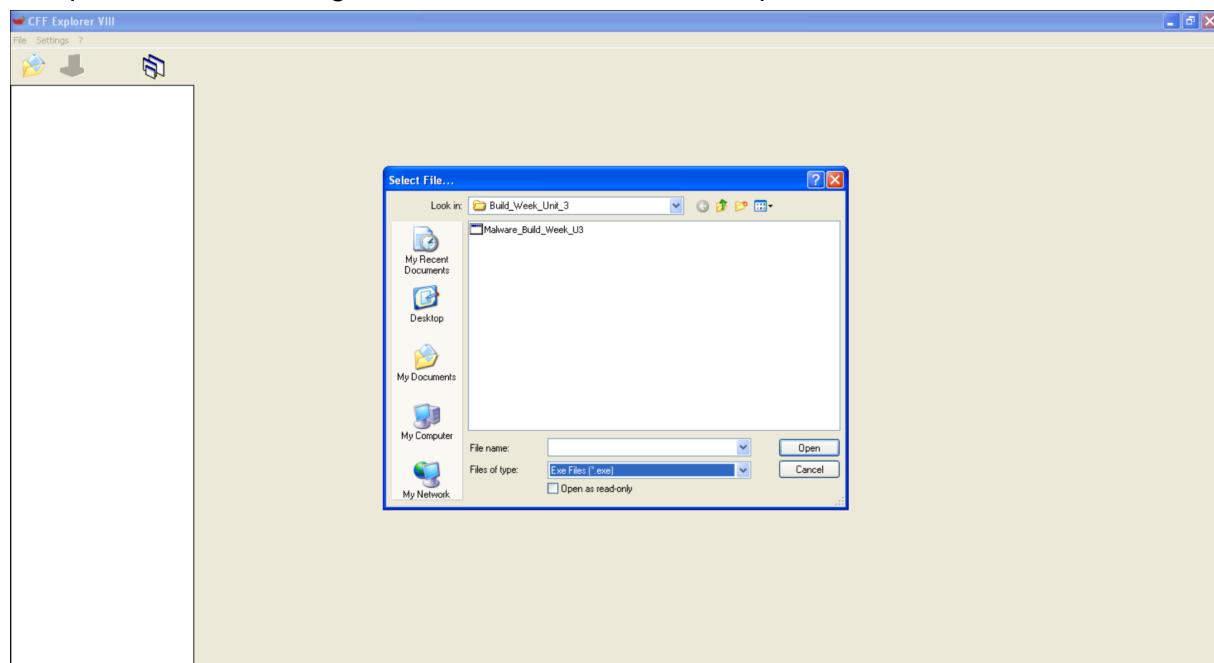
ci troviamo davanti a questa schermata.

Da qui è facile individuare Parametri e Variabili. Sappiamo infatti che le variabili hanno un valore negativo, mentre i parametri hanno un valore positivo.

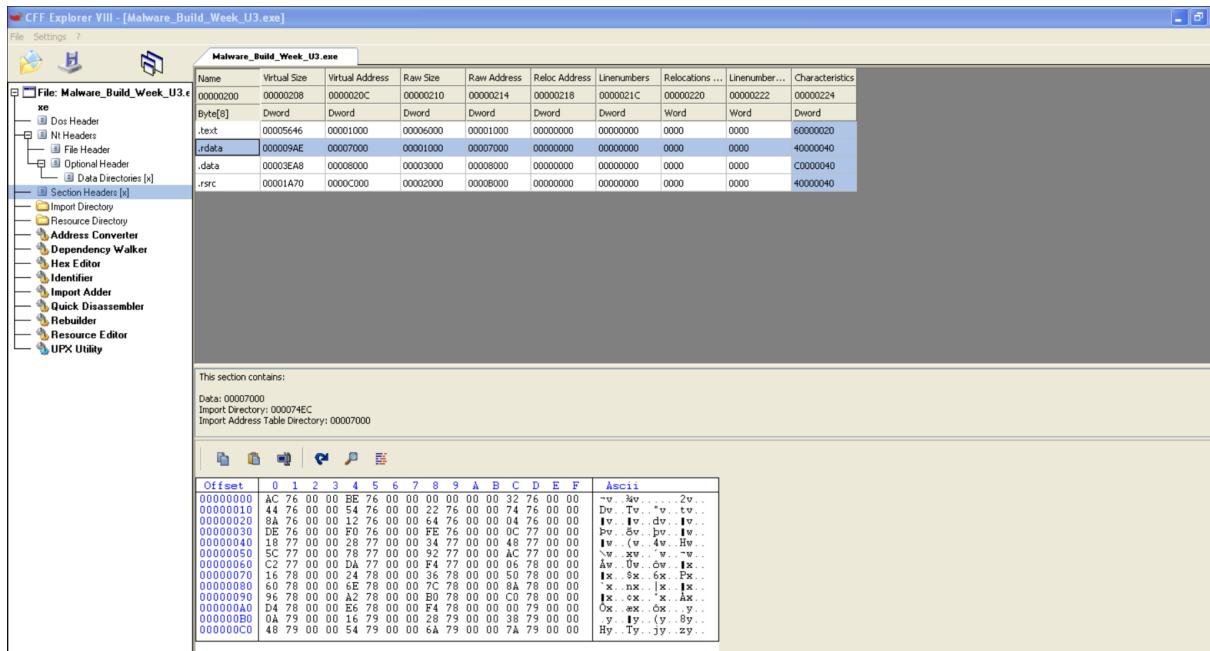
Quindi ci troviamo di fronte a 3 parametri : argc, argv, envp
e 4 variabili hModule, var_8, var_4, data.

Ci viene chiesto poi di identificare le sezioni presenti all'interno del file.

Per questo abbiamo bisogno di un altro software : CFF Explorer.



Una volta aperto il file esegibile del malware sotto analisi andiamo subito ad analizzare le sezioni cliccando su Section Headers nel menù a sinistra



Vediamo quindi che ci troviamo di fronte a 4 sezioni : .text, .rdata, .data, .rsrc.

.text:

- Questa sezione contiene il codice eseguibile del programma, ovvero le istruzioni che la CPU eseguirà. È dove risiede il codice del programma.

.data:

- La sezione .data è destinata a contenere dati inizializzati, come variabili globali e statiche, che vengono allocate in memoria durante l'esecuzione del programma. Questi dati hanno un valore iniziale specificato nel codice sorgente.

.rdata:

- Questa sezione è spesso utilizzata per archiviare dati di sola lettura. Ad esempio, potrebbe contenere stringhe costanti o tabelle di lookup che non devono essere modificate durante l'esecuzione del programma.

.rsrc:

- La sezione .rsrc è spesso associata a file eseguibili di Windows e contiene risorse, come icone, bitmap, stringhe localizzate e altre informazioni non eseguibili necessarie per l'applicazione. Queste risorse possono essere richiamate e utilizzate durante l'esecuzione del programma.

Queste sezioni sono parte integrante della struttura di un file eseguibile e contribuiscono al funzionamento complessivo del programma.

La traccia ci chiede poi di identificare quali librerie importa il malware.

Continuando a usare CFF Explorer andiamo quindi nel menù “Import Directory” e

Module Name	Imports	OFTs	TimeStamp	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	00007600	00007000

troviamo le due Dll importate : [kernel32.dll](#) e [advapi32.dll](#).

[kernel32.dll](#) è una libreria di collegamento dinamico (DLL) di sistema in ambienti operativi Microsoft Windows. Questa libreria è fondamentale per il funzionamento del sistema operativo e offre una vasta gamma di funzionalità utilizzate dai programmi in esecuzione su Windows.

[ADVAPI32.dll](#) è un'altra libreria di collegamento dinamico (DLL) di sistema in ambienti operativi Microsoft Windows. Questa libreria fornisce funzionalità avanzate per la programmazione dell'interfaccia a livello di sistema, concentrandosi principalmente su servizi avanzati di sicurezza, gestione delle credenziali, crittografia e altre operazioni relative alla sicurezza e all'amministrazione del sistema.

L'importazione di [kernel32.dll](#) e [advapi32.dll](#) da parte di un malware suggerisce diverse possibili attività:

Accesso a Funzionalità di Sistema:

- Il malware potrebbe cercare di manipolare processi, gestire la memoria e eseguire operazioni di file fondamentali.

Manipolazione della Sicurezza:

- L'uso di [advapi32.dll](#) indica l'interesse del malware nelle operazioni di sicurezza, come gestione delle credenziali e crittografia.

Installazione di Servizi o Modifiche al Registro di Sistema:

- Potrebbe essere coinvolto nell'installazione di servizi di sistema o apportare modifiche al Registro di sistema.

Attività di Monitoraggio e Controllo:

- Il malware potrebbe cercare di raccogliere informazioni sensibili, monitorare l'attività di sistema o eseguire azioni con privilegi elevati.

Camuffamento e Autodifesa:

- L'importazione delle librerie può servire a mimetizzarsi come processi legittimi o attuare strategie di difesa contro la rilevazione.

Interazione con Servizi di Windows:

- L'uso di `advapi32.dll` suggerisce interazioni con servizi di Windows, come creazione o configurazione di servizi.

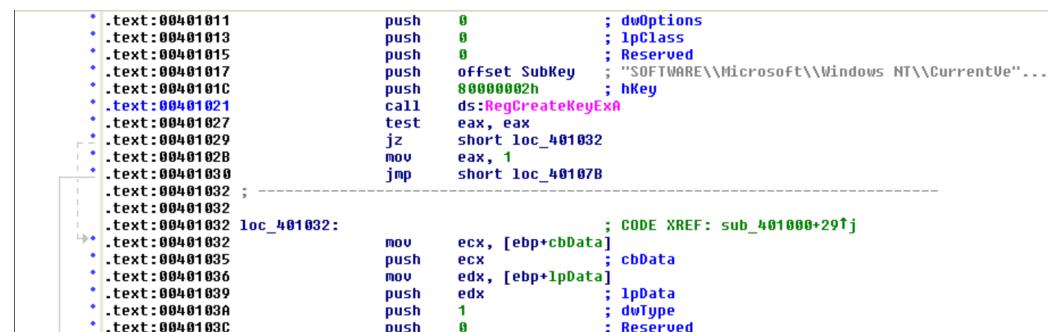
L'analisi complessiva dovrebbe considerare il comportamento complessivo del malware, valutando le azioni intraprese e l'interazione con il sistema operativo. L'importazione di queste librerie è un indicatore, ma non è sufficiente per stabilire la malignità di un programma.

Malware Analysis

Con riferimento al Malware in analisi, spiegare:

- 1)Lo scopo della funzione chiamata alla locazione di memoria 00401021
- 2)Come vengono passati i parametri alla funzione alla locazione 00401021;

Prendendo in riferimento la locazione di memoria indicata dalla traccia



```
00401011 push 0 ; dwOptions
00401013 push 0 ; lpClass
00401015 push 0 ; Reserved
00401017 push offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion"
0040101C push 80000002h ; hKey
00401021 call ds:RegCreateKeyExA
00401027 test eax, eax
00401029 jz short loc_401032
0040102B mov eax, 1
00401030 jmp short loc_40107B
00401032 ; -
00401032 loc_401032:
00401032 mov ecx, [ebp+cbData] ; CODE XREF: sub_401000+29↑j
00401035 push ecx ; cbData
00401036 mov edx, [ebp+lpData]
00401039 push edx ; lpData
0040103A push 1 ; dwType
0040103C push 0 ; Reserved
```

possiamo notare che viene chiamata la funzione `RegCreateKeyExA` che ha come compito creare o aprire una chiave nel Registro di sistema. Questa funzione è inclusa nella libreria `advapi32.dll` ed è spesso utilizzata per manipolare le voci del Registro di sistema.

I parametri alla funzione vengono passati tramite l'istruzione `push`.

L'istruzione `push` viene utilizzata nell'assembly x86 per inserire un valore sulla pila. Nella programmazione in assembly, la pila spesso viene utilizzata per passare parametri a funzioni e per gestire lo stack frame (il frame di chiamata di una funzione). L'operazione `push` inserisce un valore nella cima della pila.

- 3)Che oggetto rappresenta il parametro alla locazione 00401017



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

L'istruzione push offset SubKey in assembly inserisce l'indirizzo (offset) della variabile o etichetta denominata SubKey sulla cima della pila. Questo è comunemente utilizzato quando si passano parametri a una funzione.

4) Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029.

```
.text:00401027          test    eax, eax
F: .text:00401029          jz     loc_401032
.etc...
```

L'istruzione **test** in assembly è un'operazione logica che effettua un AND bit a bit tra due operandi senza memorizzare il risultato.

L'istruzione **test** modifica il registro delle bandiere nel processore senza memorizzare il risultato effettivo dell'AND bit a bit tra gli operandi. In particolare, il **zero flag** (ZF) viene impostato o cancellato in base al risultato dell'operazione.

Se il risultato dell'AND è zero, il **zero flag** viene impostato a 1. Se il risultato è diverso da zero, il **zero flag** viene impostato a 0.

Dopo aver effettuato questo AND, del registro EAX con se stesso viene effettuato un salto condizionato, JZ infatti sta a significare JUMP ZERO e salterà alla locazione 401032 se lo Zero Flag ha come valore 1.

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



The screenshot shows a debugger interface with two panes. The left pane displays assembly code, and the right pane displays the corresponding C code. The assembly code is:

```
C: > Users > andch > OneDrive > Desktop > Epicode > C prova.c > ...
1 int main()
2 {
3     int a ;
4     int b;
5     int c;
6     if(a==0)
7     {
8         b=c;
9     }
10    else
11    {
12        a=1;
13    }
14
15 return 0;
16
17 }
```

The C code is:

```
int main()
{
    int a ;
    int b;
    int c;
    if(a==0)
    {
        b=c;
    }
    else
    {
        a=1;
    }
}

return 0;
```

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

The screenshot shows the assembly code for a Windows registry operation. The code is written in Intel Hexadecimal Assembly. Key instructions include:

- push ecx ; cbData
- mov edx, [ebp+lpData] ; lpData
- push 1 ; dwType
- push 0 ; Reserved
- push offset ValueName ; "GinaDLL"
- mov eax, [ebp+hObject] ; hKey
- call ds:RegSetValueExA
- test eax, eax
- jz short loc_401062
- mov ecx, [ebp+hObject]
- push eax ; hObject
- call ds:CloseHandle
- mov eax, 1
- jnp short loc_40107B

Below the assembly code, there is a call to a function at address 00401062:

```
00001062 00401000:loc_401062
IDA is analysing the input file...
You may start to explore the binary file right now.
Using FLIRT signatures from Microsoft Visual C++ & .NET runtime
Name 'Leadup1' at 00402520 is deleted...
Name 'Leaddown1' at 00402648 is deleted...
Name 'Leadup1_0' at 004040F0 is deleted...
Name 'Leaddown1_0' at 00404C78 is deleted...
```

L'istruzione in questa posizione di memoria va a richiamare l'API RegSetValueExA di Windows utilizzata per impostare il valore di una voce di registro. Questa funzione è parte della libreria di sistema advapi32.dll e non è una singola istruzione di assembly, ma piuttosto una chiamata a funzione da un programma scritto in un linguaggio di alto livello come C o C++.

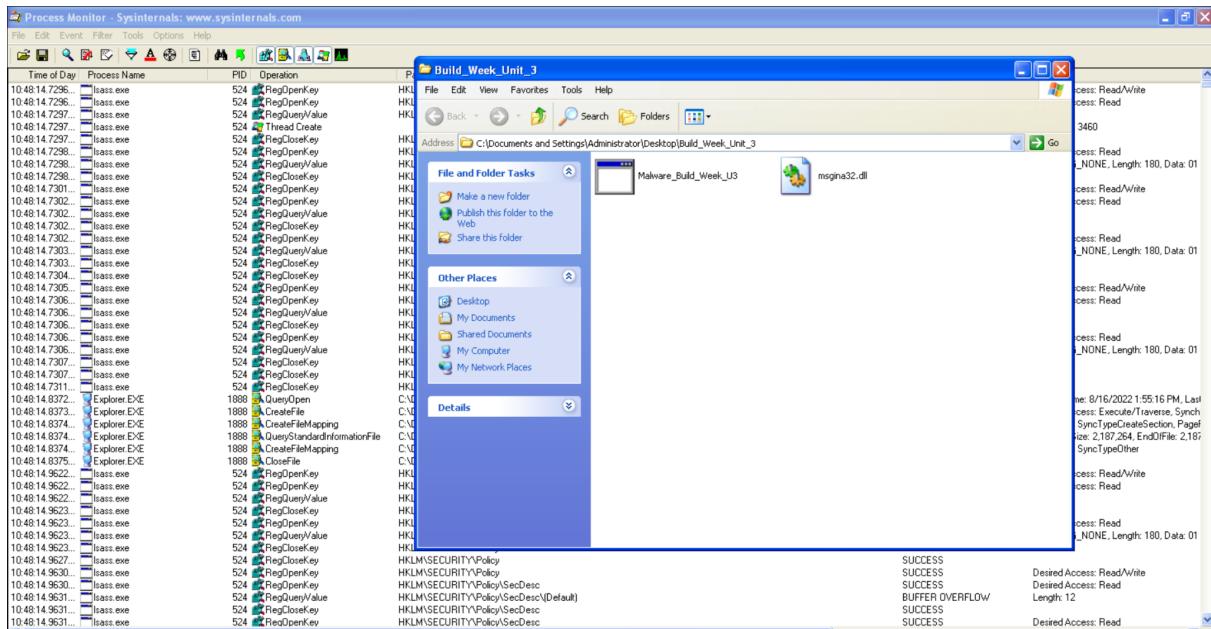
Se guardiamo attentamente qualche riga più sopra possiamo notare un push OffSet ValueName "Gina DLL" che viene usato come parametro.

Analisi Dinamica

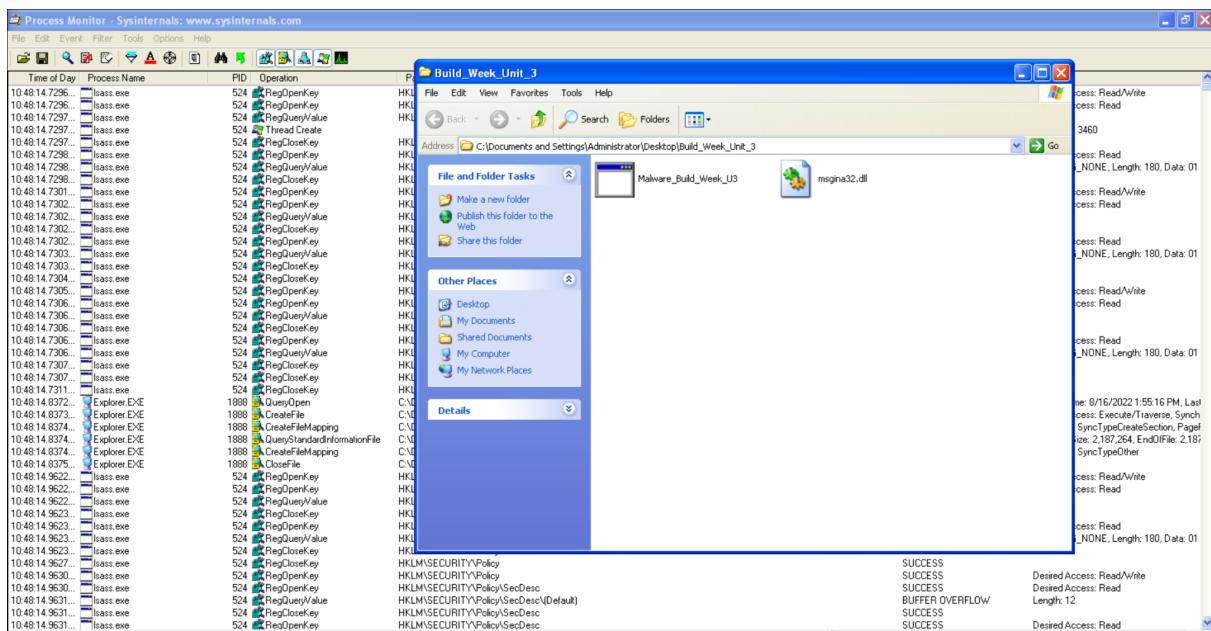
Passiamo adesso all'analisi Dinamica.

Apriamo Process Monitor (ProcMon) un programma che ci aiuterà a capire cosa succede in tempo reale quando andiamo a eseguire il malware.

Dopo aver cliccato sull'eseguibile notiamo subito che è stato creato un altro file MsGina32.dll all'interno della cartella dove si trova il malware.



Andiamo adesso a filtrare i processi che sono stati creati dal malware



Dopo aver applicato il filtro notiamo la presenza di:

Numerose attività di sistema tra cui la creazione della chiave di registro
HKLM\SOFTWARE\Windows\Microsoft\WindowsNT\CurrentVersion\Winlogon
 Alla quale poi viene associato il valore GINA DLL.

Tier of Day	Process Name	PID	Operation	Path	Result	Detail
10:49:46:1657.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSoftware\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Malware_Build_Week_U3.exe	NAME NOT FOUND	Desired Access: Read
10:49:46:1677.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSystem\CurrentControlSet\Services\Terminal Server	SUCCESS	Desired Access: Read
10:49:46:1678.	Malware_Build_Week_U3	3592	RegQueryValue	HKLMLSystem\CurrentControlSet\Services\Terminal Server\TSSAppCompat	SUCCESS	Type: REG_DWORD, Length:
10:49:46:1678.	Malware_Build_Week_U3	3592	RegCloseKey	HKLMLSystem\CurrentControlSet\Services\Terminal Server\TSSAppCompat	SUCCESS	Desired Access: Read
10:49:46:1778.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSystem\CurrentControlSet\Services\Terminal Server	SUCCESS	Desired Access: Read
10:49:46:1778.	Malware_Build_Week_U3	3592	RegQueryValue	HKLMLSystem\CurrentControlSet\Services\Terminal Server\TSSAppCompat	SUCCESS	Type: REG_DWORD, Length:
10:49:46:1778.	Malware_Build_Week_U3	3592	RegCloseKey	HKLMLSystem\CurrentControlSet\Services\Terminal Server\TSSAppCompat	SUCCESS	Desired Access: Read
10:49:46:1790.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSoftware\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Secur32.dll	NAME NOT FOUND	Desired Access: Read
10:49:46:1791.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSoftware\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\IPCFilter14.dll	NAME NOT FOUND	Desired Access: Read
10:49:46:1791.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSoftware\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\VIDVAP132.dll	NAME NOT FOUND	Desired Access: Read
10:49:46:1791.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSystem\CurrentControlSet\Services\Terminal Server	SUCCESS	Desired Access: Read
10:49:46:1791.	Malware_Build_Week_U3	3592	RegQueryValue	HKLMLSystem\CurrentControlSet\Services\Terminal Server\TSSAppCompat	SUCCESS	Type: REG_DWORD, Length:
10:49:46:1791.	Malware_Build_Week_U3	3592	RegCloseKey	HKLMLSystem\CurrentControlSet\Services\Terminal Server\TSSAppCompat	SUCCESS	Desired Access: Read
10:49:46:1792.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSystem\CurrentControlSet\Services\Terminal Server	SUCCESS	Desired Access: Read
10:49:46:1792.	Malware_Build_Week_U3	3592	RegQueryValue	HKLMLSystem\CurrentControlSet\Services\Terminal Server\TSSAppCompat	SUCCESS	Type: REG_DWORD, Length:
10:49:46:1792.	Malware_Build_Week_U3	3592	RegCloseKey	HKLMLSystem\CurrentControlSet\Services\Terminal Server\TSSAppCompat	SUCCESS	Desired Access: Read
10:49:46:1793.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	NAME NOT FOUND	Desired Access: Read
10:49:46:1793.	Malware_Build_Week_U3	3592	RegQueryValue	HKLMLSOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTrack	SUCCESS	Desired Access: Maximum Allowable
10:49:46:1793.	Malware_Build_Week_U3	3592	RegCloseKey	HKLMLSOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	NAME NOT FOUND	Desired Access: Read
10:49:46:1794.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSOFTWARE\Microsoft\Windows NT\CurrentVersion\DeviceInstall	NAME NOT FOUND	Desired Access: Read
10:49:46:1794.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSoftware\Microsoft\Windows NT\CurrentVersion\DeviceInstall	NAME NOT FOUND	Desired Access: Read
10:49:46:1794.	Malware_Build_Week_U3	3592	RegCreateKey	HKLMLSoftware\Microsoft\Windows NT\CurrentVersion\DeviceInstall\Wmi	NAME NOT FOUND	Desired Access: Read
10:49:46:1794.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSoftware\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Wmi.dll	NAME NOT FOUND	Desired Access: Read
10:49:46:1795.	Malware_Build_Week_U3	3592	RegOpenKey	HKLMLSoftware\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All Access
10:49:46:1795.	Malware_Build_Week_U3	3592	RegCreateKey	HKLMLSOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GaudI	SUCCESS	Type: REG_SZ, Length: 5004
10:49:46:1898.	Malware_Build_Week_U3	3592	RegCloseKey	HKLMLSOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	

Possiamo vedere poi che sempre il malware ha creato il file che abbiamo notato subito prima dopo aver cliccato sul malware con le seguenti chiamate :

10:49:46:1656.	inmemory_cbs_w32_v3...	3592	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	
10:49:46:1670.	Malware_Build_Week_U3	3592	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	Desired Access: Execute/Traverse, Synchronizing, Control: FSCTL_IS_VOLUME_MOUNTED
10:49:46:1671.	Malware_Build_Week_U3	3592	QueryOpen	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe	NAME NOT FOUND	
10:49:46:1671.	Malware_Build_Week_U3	3592	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\migrn32.dll	SUCCESS	Desired Access: Generic Write, Read Attributes
10:49:46:1763.	Malware_Build_Week_U3	3592	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\migrn32.dll	SUCCESS	Desired Access: Synchronize, Disposition: I
10:49:46:1765..	Malware_Build_Week_U3	3592	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	
10:49:46:1765..	Malware_Build_Week_U3	3592	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	
10:49:46:1775..	Malware_Build_Week_U3	3592	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\migrn32.dll	SUCCESS	Offset: 0, Length: 4,096
10:49:46:1790..	Malware_Build_Week_U3	3592	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\migrn32.dll	SUCCESS	Offset: 4,096, Length: 2,560
10:49:46:1790..	Malware_Build_Week_U3	3592	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\migrn32.dll	SUCCESS	
10:49:46:1793..	Malware_Build_Week_U3	3592	SetEndOfFileInformation	C:\Windows\system32\config\software\LOG	SUCCESS	SetEndOfFile: 8,192
10:49:46:1793..	Malware_Build_Week_U3	3592	SetEndOfFileInformation	C:\Windows\system32\config\software\LOG	SUCCESS	SetEndOfFile: 8,192
10:49:46:1795..	Malware_Build_Week_U3	3592	SetEndOfFileInformation	C:\Windows\system32\config\software\LOG	SUCCESS	SetEndOfFile: 16,384
10:49:46:1798..	Malware_Build_Week_U3	3592	SetEndOfFileInformation	C:\Windows\system32\config\software\LOG	SUCCESS	SetEndOfFile: 20,480
10:49:46:1801..	Malware_Build_Week_U3	3592	SetEndOfFileInformation	C:\Windows\system32\config\software\LOG	SUCCESS	SetEndOfFile: 24,576
10:49:46:1812..	Malware_Build_Week_U3	3592	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS	

Conclusioni

Stando a quanto raccolto con l'analisi statica e dinamica possiamo dedurre tranquillamente che si tratta di un malware che andava a interagire con la dll GINA (Graphical Identification and Authentication) che è stata un componente critico nei sistemi operativi Windows precedenti a Windows Vista. La sua funzione principale era gestire l'autenticazione degli utenti durante il processo di accesso al sistema.

In particolare, **gina.dll** è stata utilizzata per implementare l'interfaccia di autenticazione del sistema operativo. Quando un utente tenta di accedere a un sistema Windows, **gina.dll** viene coinvolta nella gestione della schermata di accesso (interfaccia di login) e nelle attività di autenticazione. Le GINA permettevano agli sviluppatori di personalizzare il processo di accesso, ad esempio, per supportare metodi di autenticazione personalizzati o dispositivi di sicurezza aggiuntivi.

Con l'introduzione di Windows Vista, la GINA è stata rimpiazzata dal componente **Credential Provider**, che ha introdotto un modello più flessibile e sicuro per la gestione delle credenziali e delle interfacce di autenticazione.

Con tutta probabilità quindi interagendo con questo malware il nostro pc verrebbe infettato e fornirebbe la possibilità ad altri utenti di connettersi al nostro dispositivo.

