

Esercizio Fine M6 – Galli Alessandro

1) Iniziamo eseguendo l'analisi statica del malware. L'**analisi statica** si riferisce all'ispezione del codice sorgente o del codice binario di un programma (in questo caso, un malware) per identificarne la funzionalità, le caratteristiche e le potenziali minacce senza eseguirlo. Questo approccio si contrappone all'**analisi dinamica**, dove il codice viene eseguito in un ambiente controllato (sandbox) per osservare il suo comportamento.

L'**analisi statica** **basica** consiste nell'esaminare un eseguibile senza vedere le istruzioni che lo compongono e la sua funzione è confermare se un dato file è malevolo e fornire informazioni generiche circa le sue funzionalità. L'**analisi statica** **avanzata** presuppone la conoscenza dei fondamenti di «reverse-engineering» al fine di identificare il comportamento di un malware a partire dall'analisi delle istruzioni che lo compongono. questo passaggio è essenziale per capire esattamente cosa fa il malware a livello di istruzioni della cpu. si possono inoltre estrarre stringhe di testo, url, chiavi di cifratura, e altre risorse dal codice del malware, che possono indicare il suo comportamento o intento e se ne può esaminare il codice relativo alla rete per comprendere come il malware comunica.

Prima di procedere è meglio assicurarsi che sia di fatto un malware, estraendone l'**hash** con **md5deep** e controllando su **VirusTotal** la sua reputazione che si basa su vari riscontri di software antivirus.

In questa prima analisi posso vedere che il virus è noto, si tratta di un malware di tipo **Trojan** compilato in data 11-06-2011 in C++, analizzato l'ultima volta in data 17 aprile 2024. è un malware progettato per colpire la macchina Intel 386 e processori successivi.

Ha **5255 entry points**, **4 sezioni** ed importa **2 librerie**:

kernel32.dll - una delle librerie fondamentali di windows, contiene numerose funzioni che gestiscono la memoria, i processi e i thread. i malware la utilizzano per manipolare i processi e per accedere a diverse api di sistema ed ottenere persistenza.

advapi32.dll - fornisce funzioni relative alla sicurezza e alla gestione di account, che i malware possono sfruttare per modificare permessi, accedere a token di sicurezza e alterare il registro di sistema, ad esempio per essere avviati all'avvio del sistema operativo.

Confermiamo tutti questi dati tramite tool dedicati come **Cffexplorer** e **IDA pro**.

Hash con md5deep

```

C:\> Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd Desktop
C:\Documents and Settings\Administrator\Desktop>cd md5deep-4.3
C:\Documents and Settings\Administrator\Desktop\md5deep-4.3>md5deep "C:\Documents and Settings\Administrator\Desktop\Esercizio_Pratico_U3_W3_L3\Malware_U3_W3_L3.exe"
251f4d0caf6eadae453488f9c9c0ea95 C:\Documents and Settings\Administrator\Desktop\Esercizio_Pratico_U3_W3_L3\Malware_U3_W3_L3.exe
C:\Documents and Settings\Administrator\Desktop\md5deep-4.3>

```

VirusTotal

43
/ 70

Community Score

43/70 security vendors flagged this file as malicious

f153dfacec09dd69809c3bbf68270a38ee3701f44220c7bf181c14a68c138133

Lab09-02.exe

Size24.00 KB

Last Analysis Date2 months ago

EXE

peexechecks-user-inputidlearmadillo

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY11

Join our Community

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

Popular threat labeltrojan.genericrxt/neanvzc

Threat categoriestrojan

Family labelsgenericrxtneanvzrr002c0plk20

Security vendors' analysis

Do you want to automate checks?

Alibaba	Trojan:Win32/Generic.9d3f6ef1	AliCloud	Trojan
ALYac	Application.Agent.AHB	Antiy-AVL	Trojan/Win32.BTSGeneric
Arcabit	Application.Agent.AHB	Avast	Win32:TrojanX-gen [Trj]
AVG	Win32:TrojanX-gen [Trj]	BitDefender	Application.Agent.AHB
BitDefenderTheta	Gen:NN.ZexaF.36806.bmW@aapI0K	Bkav Pro	W32.Common.D47F939F
Cybereason	Malicious.caf6ea	Cylance	Unsafe
DeepInStinct	MALICIOUS	Emsisoft	Application.Agent.AHB (B)
eScan	Application.Agent.AHB	ESET-NOD32	A Variant Of Generik.NEANVZC
Fortinet	W32/Generic_PUA_OK.NEANVZC!tr	GData	Application.Agent.AHB
Gridinsoft (no cloud)	Trojan.Win32.Agent.vb1s1	Ikarus	PUA.Agent
Kingsoft	Malware.kb.a.933	Lionic	Trojan.Win32.Generic.41c
Malwarebytes	Malware.AI.393618554	MAX	Malware (ai Score=99)
MaxSecure	Trojan.Malware.7164915.susgen	McAfee Scanner	TiIF153DFACEC09
NANO-Antivirus	Trojan.Win32.MlwGen.daocxo	Palo Alto Networks	Generic.ml
Sangfor Engine Zero	Trojan.Win32.Agent.V7s1	SecureAge	Malicious
Skyhigh (SWG)	GenericRXET-IEI251F4D0CAF6E	Sophos	Generic Reputation PUA (PUA)
Symantec	ML.Attribute.HighConfidence	Trellix (ENS)	GenericRXET-IEI251F4D0CAF6E

Cff explorer:

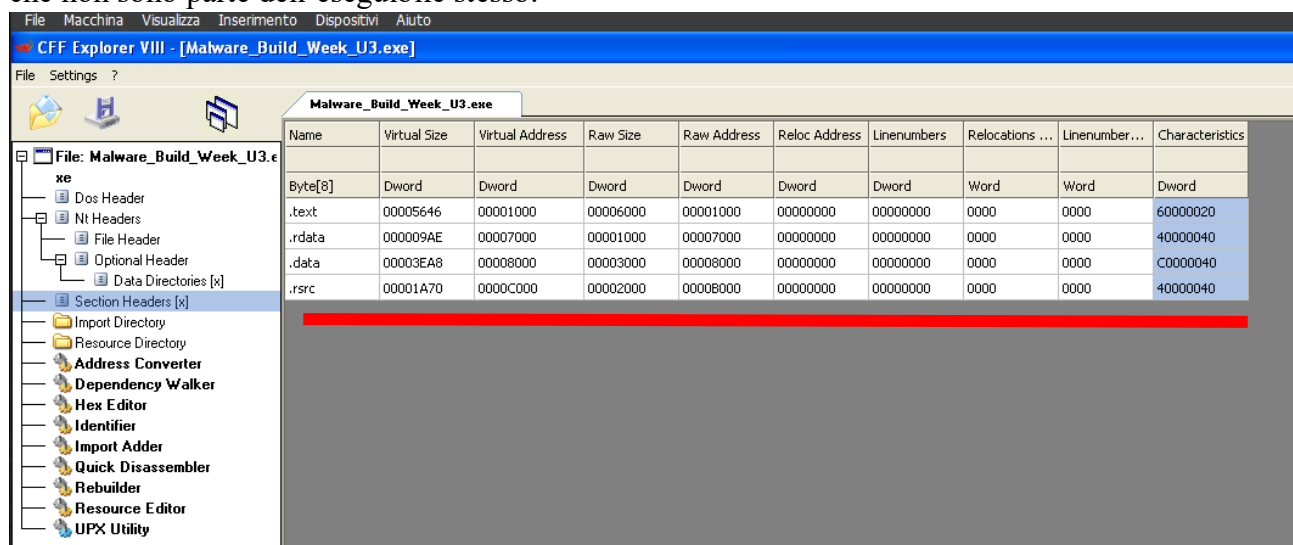
Per controllare le funzioni importate ed esportate da un malware, si può utilizzare Cff explorer, un tool dedicato all'analisi dei malware. Spostandosi su «**import directory**» si possono controllare le librerie e le funzioni importate, mentre su «**section headers**» si possono vedere le sezioni presenti all'interno del file eseguibile:

.text - contiene le righe di codice, istruzioni, che la cpu eseguirà all'avvio del malware.

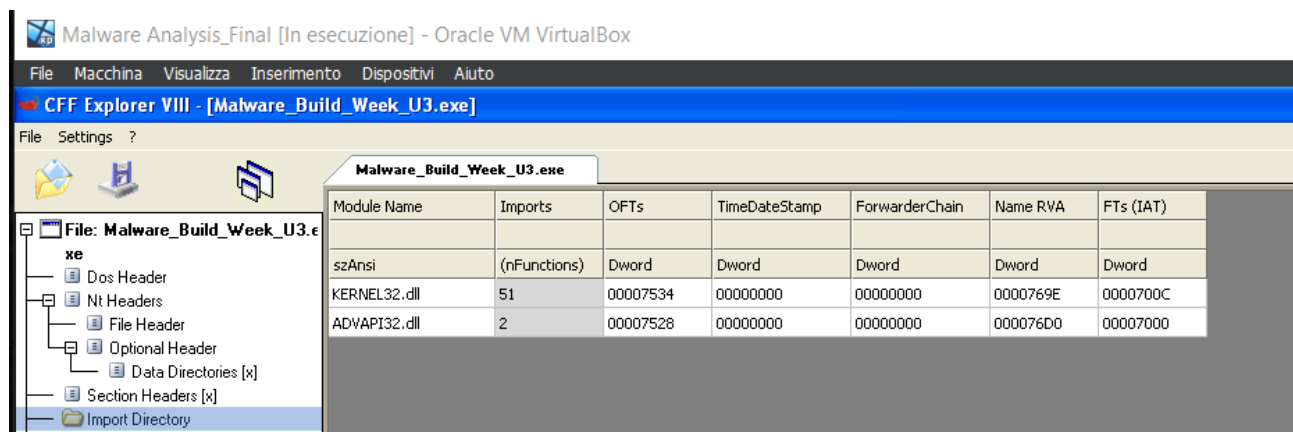
.rdata - sezione che include le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, dati che il programma legge mentre è in funzione.

.data - contiene i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma e possono essere modificati.

.rsrc - include le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell'eseguibile stesso.



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	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	00008000	00000000	00000000	0000	0000	40000040



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

CFF Explorer VIII - [Malware_Build_Week_U3.exe]

File Settings ?

Malware_Build_Week_U3.exe

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

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA

Malware Analysis_Final [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

CFF Explorer VIII - [Malware_Build_Week_U3.exe]

File Settings ?

Malware_Build_Week_U3.exe

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

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

Per quel che riguarda le due librerie importate, che sono **kernel32.dll** (1) e **advapi32.dll** (2), posso ipotizzare che il malware cerchi di ottenere persistenza e modificare le chiavi di registro (**RegSetValueExA/RegCreateKeyExA**) per poter essere eseguito in autonomia. Inoltre la presenza di funzioni come **SizeOfResource/LockResource/LoadResource/FindResourceA** fa presupporre che sia un dropper, cioè un malware che contiene al suo interno un altro malware, ed utilizza queste apis che permettono di localizzare all'interno della sezione «risorse» il malware da estrarre, e successivamente da caricare in memoria per l'esecuzione.

IDA Pro:

Per informazioni su **variabili locali** e **parametri** della funzione `main()`, useremo **IDA Pro** (interactive disassembler professional) che è uno strumento avanzato di reverse engineering software che offre capacità di disassemblaggio, debugging e analisi statica.

Variabili locali: le variabili locali in una funzione assembly sono tipicamente allocate nello stack.

Questo è spesso fatto attraverso istruzioni di tipo `push` all'inizio di una funzione o con un'istruzione `sub` che aumenta il puntatore dello stack (`sp`) per creare spazio.

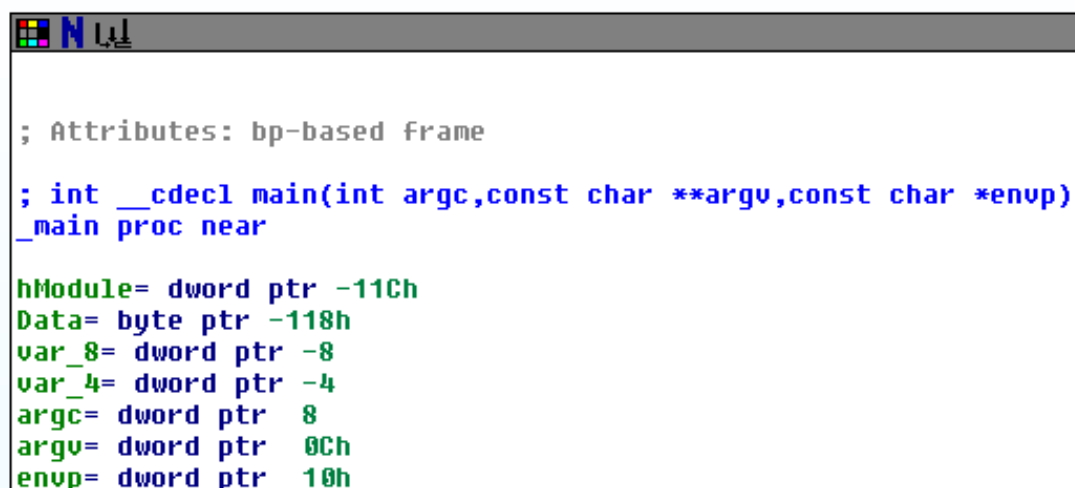
Parametri: solitamente i parametri sono passati ai registri o attraverso l'uso dello stack prima della chiamata della funzione. I commenti nel codice possono fornire indicazioni su dove e quanti parametri sono passati.

Nell'assembly, le etichette che iniziano con `var_` tendono a indicare variabili locali, mentre quelle che iniziano con `arg_` indicano argomenti o parametri passati alla funzione.

Valori offset: gli offset (come `var_54h`) sono utilizzati per accedere a dati specifici sullo stack. Gli offset negativi rispetto all'indirizzo base del frame della funzione (ad esempio `ebp` su x86): di solito indicano variabili locali, mentre gli offset positivi indicano parametri.

In questo caso le **variabili locali** in evidenza che possiamo rilevare ad un'occhiata **sono 4**: `hmodule` / `data` / `var_8` / `var_4`.

Mentre i **parametri** sono 3 - `argc` / `argv` / `envp`.



```
; 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 -118h
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

Proseguendo con l'analisi statica del codice del malware, troviamo alla locazione di memoria **00401021** la funzione **regcreatekeyexa**, che è una funzione nota che fa parte delle API di Windows che permettono alle applicazioni di interagire con il registro di Windows. Queste funzioni sono utilizzate per creare nuove chiavi di registro o per aprire chiavi esistenti per modificare i loro valori.

Nello specifico il malware può utilizzare **regcreatekeyexa** per ottenere persistenza creando nuove chiavi di registro o modificando chiavi esistenti ed assicurandosi che il codice malevolo venga eseguito ogni volta che il sistema viene avviato. Per esempio, potrebbe aggiungere una voce nella chiave run per eseguire automaticamente il malware all'avvio del sistema.

Riguardo il metodo in cui vengono passati i parametri alla suddetta funzione, nel modulo si è visto che la convenzione di chiamata più comune in molti sistemi operativi quando si utilizza l'architettura x86 è «pushare» i parametri sullo stack prima della chiamata (call) alla funzione **regcreatekeyexa**. I parametri verranno letti dalla funzione in ordine inverso, quindi a partire da **hkey** per finire con **lpdwdisposition**.

```
* .text:00401017      push    offset SubKey      ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
* .text:0040101C      push    80000002h         ; hKey
* .text:00401021      call    ds:RegCreateKeyExA
```

Troviamo alla locazione di memoria **00401017** - **push offset subkey** - l'indirizzo della sottochiave di registro che viene spinto nello stack. Il nome vicino al codice indica che potrebbe essere il nome della sottochiave che verrà creata dal processo **hkey** in **winlogon**, componente del sistema operativo Windows che si occupa della gestione della sessione di accesso (login) e disconnessione (logout) degli utenti.

```
* .text:00401027      test     eax, eax
* .text:00401029      jz       short loc_401032
* .text:0040102B      mov      eax, 1
* .text:00401030      jmp      short loc_40107B
* .text:00401032      ; -----
* .text:00401032      loc_401032:      mov      ecx, [ebp+cbData] ; CODE XREF: sub_401000+29f:
* .text:00401032
```

Cerchiamo ora di comprendere il significato delle **istruzioni comprese tra gli indirizzi 00401027 e 00401029** vediamo:

- un'istruzione condizionale **test**, che è simile all'istruzione **and** logico bit a bit, ma non va a modificare il contenuto degli operandi (cioè **eax** e se stesso). Modifica invece il flag **zf** (zero flag) del registro **eflags**, che viene settato ad 1 se e solo se il risultato dell'**and** è 0. (**0 and 1 = 0 * 1 = 0 -> zeroflag = 1**). Viene di fatto utilizzato per controllare se un valore è zero o meno. se **test** è zero, lo **zf** è 1.
- Un **conditional jump di tipo jz**, che nel flusso di controllo salta ad una determinata locazione di memoria (in questo caso **00401032**) se **zf** è pari a uno. A meno che il valore contenuto nel registro **eax** non sia 0, il salto non avverrà.

Questa operazione equivale ad un **ciclo if in C** come quello seguente (qui, **eax** rappresenta una variabile in **c** che contiene il valore che era nel registro **eax**):

```
if (eax==0)
{
// vai a loc_401032
}
else
{
// riprova a fare un'operazione
}
```

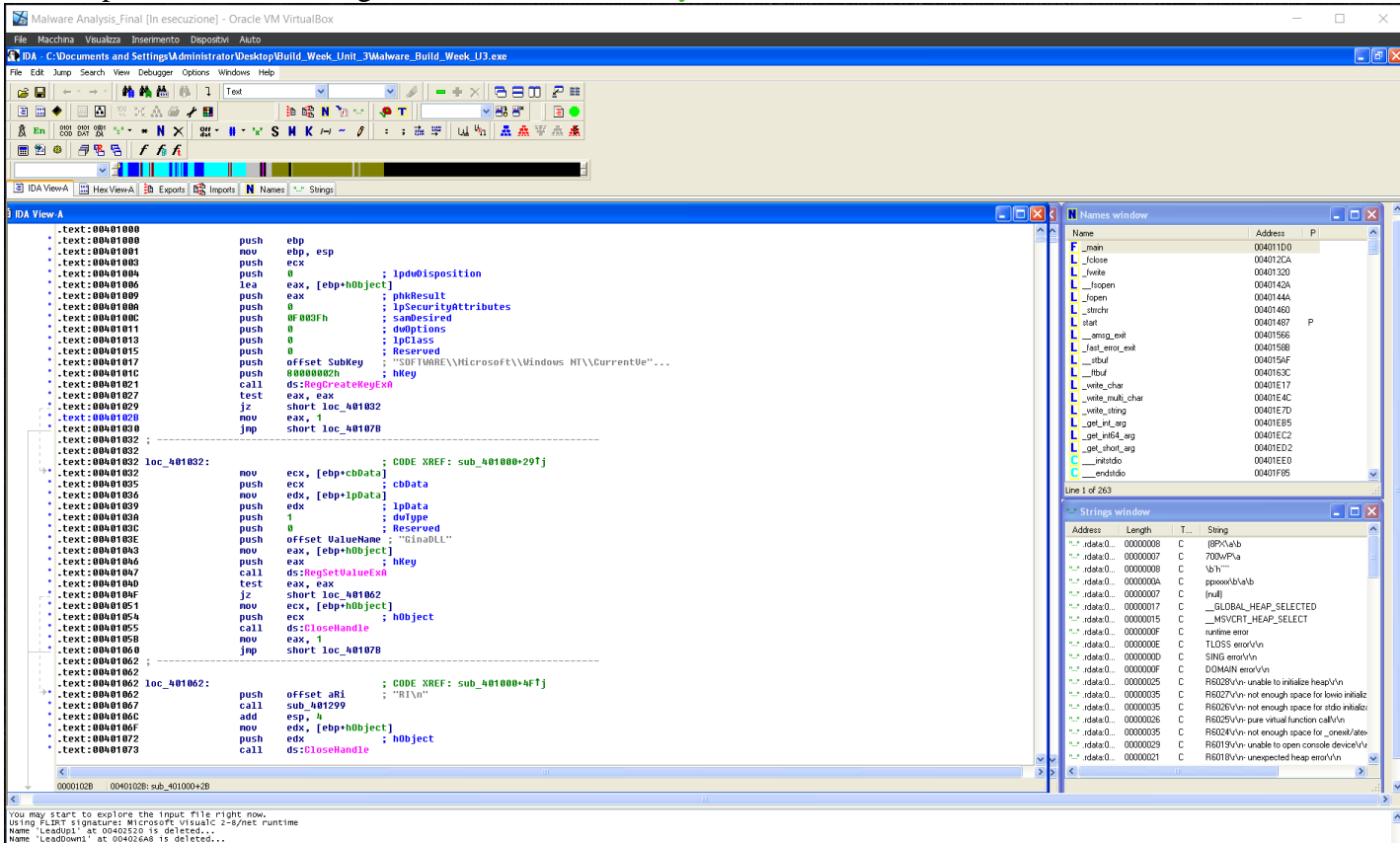
```

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

```

Nella chiamata alla locazione **00401047** si può vedere che si tratta di una call alla funzione **RegSetValueEx**, che è una funzione dell'API di windows che imposta il valore di una voce nel registro di sistema. Il prefisso **ds:** indica che l'indirizzo della funzione è preso dal registro ds, che è il segmento dati.

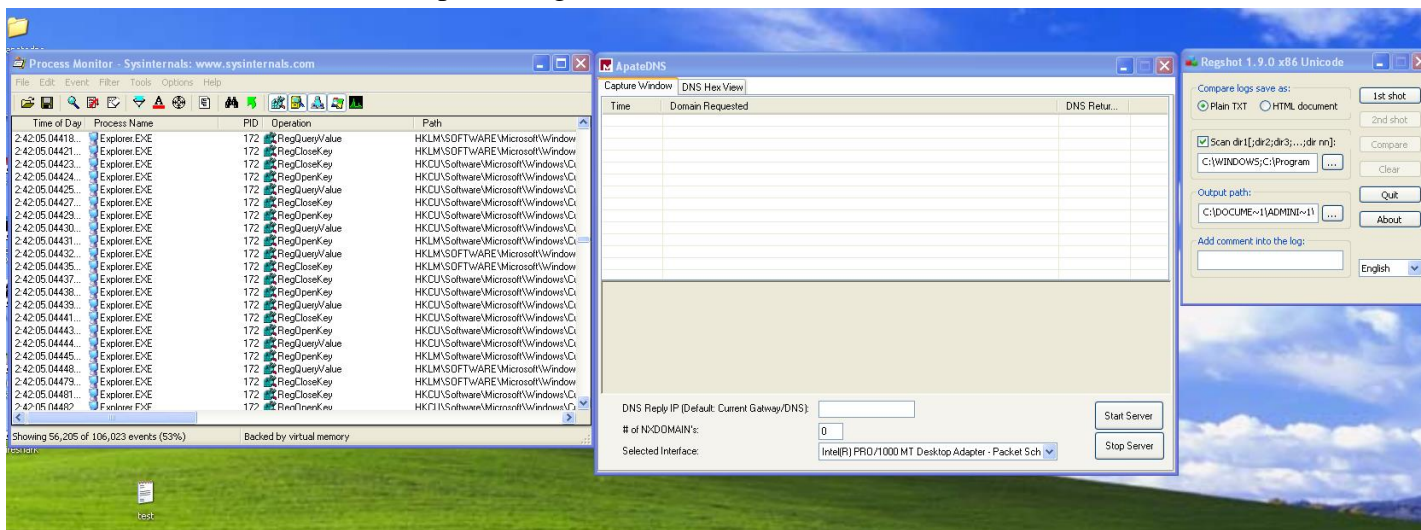
Questa funzione, quindi, impartisce istruzioni affinché **offset ValueName** abbia valore **"GinaDLL"** in una specifica chiave di registro identificata da **hKey**.



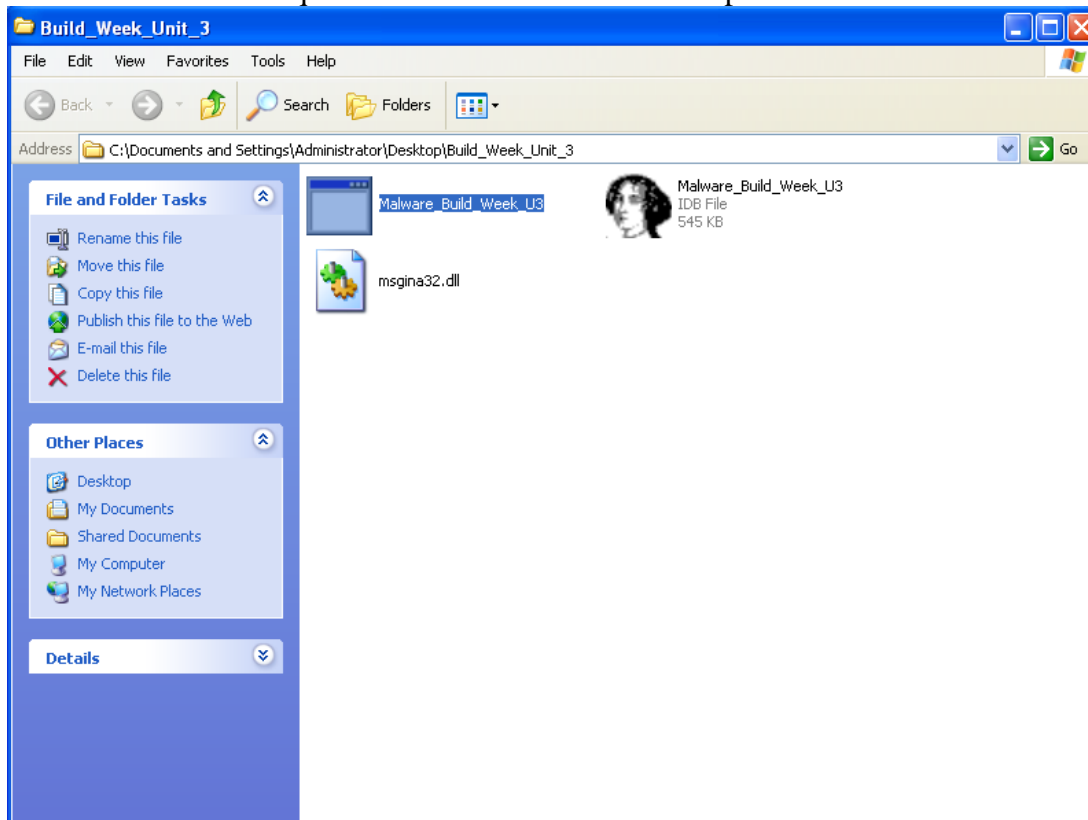
2)Proseguiamo effettuando l'analisi dinamica, ricreando un'istantanea da VirtualBox della macchina Windows XP prima di iniziare, per poter ripristinare in caso di problemi, e visto che andrò ad eseguire il malware mi assicuro di rispettare i seguenti accorgimenti:

- Disattivare controller usb
- Disattivare comunicazione con la rete (scheda di rete interna)
- Disabilitare la condivisione delle cartelle
- Disabilitare appunti condivisi (copia / incolla)

Prima di aprire il malware, è meglio aprire **Procmon**, al fine di identificare eventuali azioni del malware su processi e thread, e modifiche registro; si avvia anche **Regshot** per confrontare tramite screenshot (prima / dopo) le modifiche che avverranno a livello di sistema, e si imposta un ip statico per vedere tramite **ApateDNS** le chiamate che il malware andrà a fare nel web (quali siti). Avendolo isolato dalla rete ovviamente non potrà eseguire tutte le sue funzioni.



Vediamo subito che all'interno della cartella dove è situato l'exe del malware viene creato un file, ovvero **msgina.dll** (acronimo di "Microsoft Graphical Identification and Authentication") che gestisce il processo di accesso, e nello specifico l'interfaccia utente di logon interattiva, che include la schermata di accesso classica di inserimento nome utente / password. Funziona nel contesto del processo **winlogon** e viene caricato all'inizio del processo di avvio del sistema. È responsabile di fornire procedure personalizzabili per l'identificazione e l'autenticazione degli utenti, ed è proprio andando a modificare questo file che il malware ottiene persistenza all'avvio del sistema operativo.



Dopo un confronto ottenuto tramite [Regshot](#) posso vedere che sono stati modificati 113 elementi tra cui 20 chiavi aggiunte, 2 cancellate; valori aggiunti 38, cancellati 18, modificati 23.

```
HKU\S-1-5-21-1993962763-1606980848-72534
HKU\S-1-5-21-1993962763-1606980848-72534
```

```
-----
Files [attributes?] modified: 23
-----
```

```
-----
Values modified: 38
-----
```

```
Regshot 1.9.0 x86 Unicode
Comments:
Datetime: 2024/8/11 14:24:56 , 2024/8/11 14:26:06
Computer: MALWARE_TEST , MALWARE_TEST
Username: Administrator , Administrator
```

```
-----
Keys deleted: 2
-----
```

```
HKLM\SYSTEM\ControlSet001\Services\PROCMON24\Enum
HKLM\SYSTEM\CurrentControlSet\Services\PROCMON24\Enum
```

```
-----
Keys added: 20
-----
```

```
HKLM\SOFTWARE\Classes\Applications\idag.exe
HKLM\SOFTWARE\Classes\Applications\idag.exe\shell
HKLM\SOFTWARE\Classes\Applications\idag.exe\shell\open
HKLM\SOFTWARE\Classes\Applications\idag.exe\shell\open\command
HKLM\SYSTEM\ControlSet001\Control\Print\Printers
HKLM\SYSTEM\ControlSet001\Control\Print\Printers\Microsoft XPS
HKLM\SYSTEM\ControlSet001\Control\Print\Printers\Microsoft XPS
HKLM\SYSTEM\ControlSet001\Control\Print\Printers\Microsoft XPS
HKLM\SYSTEM\ControlSet001\Enum\Root\LEGACY_PROCMON24\0000\Contr
HKLM\SYSTEM\CurrentControlSet\Control\Print\Printers
HKLM\SYSTEM\CurrentControlSet\Control\Print\Printers\Microsoft
HKLM\SYSTEM\CurrentControlSet\Control\Print\Printers\Microsoft
HKLM\SYSTEM\CurrentControlSet\Control\Print\Printers\Microsoft
HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_PROCMON24\0000\C
HKU\S-1-5-21-1993962763-1606980848-725345543-500\Software\Micro
HKU\S-1-5-21-1993962763-1606980848-725345543-500\Software\Micro
HKU\S-1-5-21-1993962763-1606980848-725345543-500\Software\Micro
HKU\S-1-5-21-1993962763-1606980848-725345543-500\Software\Micro
```

```
-----
Values deleted: 18
-----
```

```
HKLM\SYSTEM\ControlSet001\Services\aec\Enum\0: "Sw\{4245ff73-1d
HKLM\SYSTEM\ControlSet001\Services\DMusic\Enum\0: "Sw\{8c07dd50
HKLM\SYSTEM\ControlSet001\Services\drmkau\Enum\0: "Sw\{eec12db
HKLM\SYSTEM\ControlSet001\Services\kmixer\Enum\0: "Sw\{b7eafdc0
HKLM\SYSTEM\ControlSet001\Services\PROCMON24\Enum\0: "Root\LEGA
HKLM\SYSTEM\ControlSet001\Services\PROCMON24\Enum\Count: 0x0000
HKLM\SYSTEM\ControlSet001\Services\PROCMON24\Enum\NextInstance:
HKLM\SYSTEM\ControlSet001\Services\splitter\Enum\0: "Sw\{2f412a
HKLM\SYSTEM\ControlSet001\Services\swmidi\Enum\0: "Sw\{6c1b9f60
HKLM\SYSTEM\CurrentControlSet\Services\aec\Enum\0: "Sw\{4245ff7
HKLM\SYSTEM\CurrentControlSet\Services\DMusic\Enum\0: "Sw\{8c07
HKLM\SYSTEM\CurrentControlSet\Services\drmkau\Enum\0: "Sw\{eec
HKLM\SYSTEM\CurrentControlSet\Services\kmixer\Enum\0: "Sw\{b7ea
HKLM\SYSTEM\CurrentControlSet\Services\PROCMON24\Enum\0: "Root\
HKLM\SYSTEM\CurrentControlSet\Services\PROCMON24\Enum\Count: 0x
HKLM\SYSTEM\CurrentControlSet\Services\PROCMON24\Enum\NextInsta
HKLM\SYSTEM\CurrentControlSet\Services\splitter\Enum\0: "Sw\{2f
```

Riassumendo i dati raccolti dall'analisi statica e dinamica, posso dedurre che:

- Il malware è identificabile come un Trojan/Dropper.
- All'avvio, genera un file sospetto nella cartella in cui si trova il suo eseguibile.
- Tenta di eseguire un'escalation dei privilegi per ottenere diritti amministrativi.
- Ottiene persistenza nel sistema (avvio automatico ad ogni riavvio di Windows) modificando una chiave di MSGINA32.DLL, che è coinvolta nel processo di login gestito da Winlogon.
- Esegue tecniche di evasione nascondendosi in un processo legittimo.
- Ha la capacità di accedere alle credenziali degli utenti.