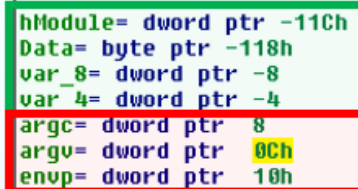


# BUILD WEEK 3

## Giorno 1

**Tasks 1-2:** Quanti parametri sono passati alla funzione main()?  
Quante variabili sono dichiarate all'interno della funzione main()?

Tramite l'utilizzo di IDA pro vediamo come vengono passati alla funzione main() i seguenti parametri e variabili :



```
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
```

Precedentemente abbiamo evidenziato le variabili con il rettangolo verde, mentre i parametri con quello rosso.

### Variabili:

una variabile è un area della memoria che contiene un particolare dato.

- **hModule**: variabile di tipo dword pointer posizionata a -11Ch rispetto all'EBP (indirizzo di base dello stack).
- **Data**: variabile di tipo byte pointer posizionata a -118h rispetto all'EBP.
- **var\_8**: variabile di tipo dword pointer posizionata a -8 rispetto all'EBP.
- **var\_4**: variabile di tipo dword pointer posizionata a -4 rispetto all'EBP.

### Parametri:

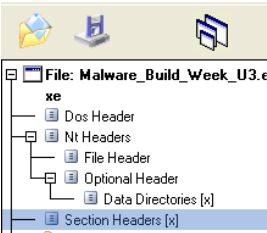
un parametro è un valore che la funzione prevede di passare quando viene chiamata.

- **argc**: rappresenta il numero totale di argomenti passati a un programma dalla riga di comando quando viene eseguito.
- **argv**: array di puntatori a stringhe, dove ogni elemento dell'array contiene un argomento passato al programma.
- **envp**: array di puntatori a stringhe che rappresentano le variabili passate al programma.

**Task 3** : Quali sezioni sono presenti all'interno del file eseguibile? Descrivete quelle identificate.

Le sezioni di cui si compone il software hanno un preciso scopo e possono aiutare a raccogliere informazioni sull'eseguibile in fase di analisi.

All'interno del file eseguibile possiamo vedere le seguenti sezioni:



Malware_Build_Week_U3.exe									
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	0000B000	00000000	00000000	0000	0000	40000040

Possiamo identificare quattro Sections all'interno del malware:

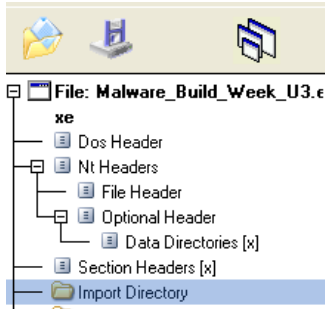
- **.text**: contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.
- **.rdata**: include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

**Task 4:** Quali librerie importa il Malware?

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.

La conoscenza delle librerie è fondamentale per l'analisi dei malware e sono contenute nell'header dell'eseguibile. Di seguito possiamo identificare quelle importate:



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	000076D0	00007000

1) **KERNEL32.DLL** : libreria usata per le funzioni principali per interagire con il sistema operativo. Un malware potrebbe sfruttare tale libreria per manipolare i file e per accedere la gestione della

memoria.

Tra le funzioni contenute nella libreria KERNEL32.dll possiamo notarne quattro in particolare:

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

- **FindResource()**: funzione utilizzata per individuare una risorsa specifica in un modulo eseguibile (come un file .exe o una DLL);
- **LoadResource()**: funzione utilizzata per caricare una risorsa identificata da un handle restituito da FindResource;
- **LockResource()**: funzione utilizzata per ottenere un puntatore ai dati di una risorsa caricata con LoadResource;
- **SizeOfResource()**: funzione che restituisce la dimensione (in byte) di una risorsa identificata da un handle restituito da FindResource e caricata con LoadResource.

Queste APIs permettono di localizzare all'interno della sezione «risorse» il malware da estrarre, e successivamente da caricare in memoria per l'esecuzione o da salvare sul disco per esecuzione futura.

Possiamo quindi supporre che si tratti di un **dropper**.

Essi si distinguono principalmente per l'utilizzo di queste funzioni per estrarre il malware contenuto nella sezione delle risorse.

Un **dropper** è un programma malevolo che contiene al suo interno un malware.

Nel momento in cui viene eseguito, un dropper inizia la sua esecuzione ed estrae il malware che contiene per salvarlo sul disco.

Generalmente il malware incluso nel dropper è contenuto nella sezione «.rss» dell'eseguibile, ovvero nella sezione risorse (talvolta anche identificata con .rsrc).

2) **ADVAPI32.dll** : libreria che contiene le funzioni per interagire con i servizi ed i registri del sistema operativo Microsoft, tramite la quale un malware potrebbe creare nuovi account utente, accedere al registro di sistema e crittografare o decrittografare dati sensibili.

Le funzioni contenute nella libreria ADVAPI32.dll sono due:

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

I malware utilizzano il registro di Windows e ne modificano le chiavi per ottenere la **'persistenza'**: l'eseguibile andrà ad aggiungersi alle entry dei programmi che andranno ad avviarsi automaticamente all'avvio della macchina. Quelle presenti nel malware sono alcune delle più comuni:

- **RegCreateKeyEx**: permette di creare una chiave del registro al fine di modificarla. Nel caso in cui dovesse già esistere, andrà ad aprirla.  
Accetta come parametri la chiave di registro da aprire.
- **RegSetValueEx**: permette di aggiungere un nuovo valore all'interno del registro e settare i rispettivi dati. Accetta come parametri la chiave, sottochiave e il dato da inserire.

## Giorno 2

**Task 1:** Qual è lo scopo della funzione chiamata alla locazione di memoria 00401021?

```
.text:00401021 | call ds:RegCreateKeyExA
```

La funzione **RegCreateKeyEx** fa parte dell'API di Windows e viene utilizzata per creare o aprire una chiave del registro di sistema in base al percorso specificato.  
Se la chiave esiste già, viene aperta. Se non esiste, viene creata.

**Task 2:** Come vengono passati i parametri alla funzione alla locazione 00401021?

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

I diversi parametri richiesti dalla funzione **RegCreateKeyEx** vengono passati tramite l'istruzione PUSH. In questo modo, vengono memorizzati i valori delle diverse variabili vengono memorizzate nello stack in attesa di essere richiamate dalla funzione.

**Task 3:** Che oggetto rappresenta il parametro alla locazione 00401017?

```
.text:00401017      push    offset SubKey      ; "SOFTWARE\\Microsoft\\Windows
.text:0040101C      push    80000002h         ; hKey

db 'SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon',0
; DATA XREF: sub_401000+17fo
```

L'offset SubKey viene utilizzato per ottenere l'indirizzo di memoria di una determinata variabile. In questo caso, 'push offset SubKey' restituisce l'indirizzo di memoria della variabile SubKey e lo sposta nello stack.

Si può notare che il parametro passato sia

**"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon":**

**Winlogon (Windows LogOn Process)** è un processo di sistema di Microsoft Windows che gestisce il logon ovvero l'autenticazione delle credenziali all'utente e altri startup.

Il processo si occupa anche di avviare la sessione utente e la shell.

**Task 4:** Qual è il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029?

```
.text:00401027      test     eax, eax
.text:00401029      jz       short loc_401032
```

L'istruzione **test** si riferisce a un'operazione che compara due valori o registri.

Viene effettuata un'operazione logica AND tra i due operandi senza memorizzare il risultato, ma imposta le apposite flag del processore.

La regola principale dell'operazione AND è la seguente:

- Se entrambi i bit sono 1, il risultato è 1.
- In tutti gli altri casi, il risultato è 0.

Il **Jump Zero Flag (JZ)** è un'istruzione condizionale che effettua un salto ad una certa posizione del programma solo se il risultato dell'ultima operazione aritmetica o logica è zero.

Se lo zero flag è impostato, il salto viene eseguito, portando l'esecuzione del programma a una nuova locazione di memoria specificata.

Rifacendoci all'istruzione del programma se il salto viene eseguito ci porta alla locazione di memoria 401032, altrimenti proseguirà con le istruzioni seguenti al jz.

```
.text:0040102B      mov     eax, 1
.text:00401030      jmp     short loc_40107B
```

L'istruzione **mov** è utilizzata per copiare dati da una locazione di memoria ad un'altra locazione o registro di memoria.

nel nostro caso viene copiato il valore 1 nel registro di memoria eax e successivamente viene effettuato un jump alla locazione di memoria 40107B.

L'istruzione **jmp** è utilizzata per eseguire un salto incondizionato all'indirizzo di destinazione specificato, indipendentemente dallo stato delle flag del processore o di altri fattori.

Rifacendoci all'istruzione del programma vediamo che arrivati a questo punto l'istruzione effettua un jump incondizionato nella locazione 40107B.

**Task 5:** Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C.

```
if (eax==0){  
    goto loc_401032;  
} else{  
    eax=1;  
    goto loc_40107b;  
}
```

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

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

L'offset ValueName viene utilizzato per ottenere l'indirizzo di memoria di una determinata variabile.

In questo caso, "push offset ValueName" restituisce l'indirizzo di memoria della variabile ValueName e lo sposta nello stack.

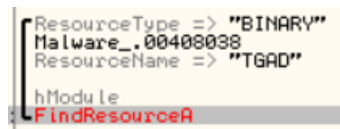
Si può notare che il valore di ValueName passato allo stack corrisponde a **'GinaDLL'**.

## Giorno 3

**Task 1:** 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()?



Utilizzando OllyDBG, si può notare il valore del parametro ResourceName, il quale è **"TGAD"**.  
(che non è altro una risorsa utilizzata dal dropper)

**Task 2:** Che funzionalità sta implementando il Malware?

<code>.text:004010C9</code>	<code>call</code>	<code>ds:FindResourceA</code>
<code>.text:004010E7</code>	<code>call</code>	<code>ds:LoadResource</code>
<code>.text:004010FF</code>	<code>call</code>	<code>ds:LockResource</code>
<code>.text:0040111B</code>	<code>call</code>	<code>ds:SizeofResource</code>

- **FindResource()**: funzione utilizzata per individuare una risorsa specifica in un modulo eseguibile (come un file .exe o una DLL);
- **LoadResource()**: funzione utilizzata per caricare una risorsa identificata da un handle restituito da FindResource;
- **LockResource()**: funzione utilizzata per ottenere un puntatore ai dati di una risorsa caricata con LoadResource;
- **SizeOfResource()**: funzione che restituisce la dimensione (in byte) di una risorsa identificata da un handle restituito da FindResource e caricata con LoadResource.

Queste APIs permettono di localizzare all'interno della sezione «risorse» il malware da estrarre, e successivamente da caricare in memoria per l'esecuzione o da salvare sul disco per esecuzione futura.

A seguito di queste indagini possiamo confermare ulteriormente la nostra ipotesi che si tratti di un **dropper**, in quanto utilizza queste funzioni per estrarre il malware contenuto nella sezione delle risorse.

Ricordiamo che, un **dropper**, è un programma malevolo che estrae il malware che contiene per salvarlo sul disco e che, normalmente, file incluso nel dropper è contenuto nella sezione «.rss» dell'eseguibile (come spiegato al giorno 1).

**Task 3:** È possibile identificare questa funzionalità utilizzando l'analisi statica basica?

Con l'utilizzo di **CFFExplorer**, è possibile identificare le librerie implementate e le corrispondenti

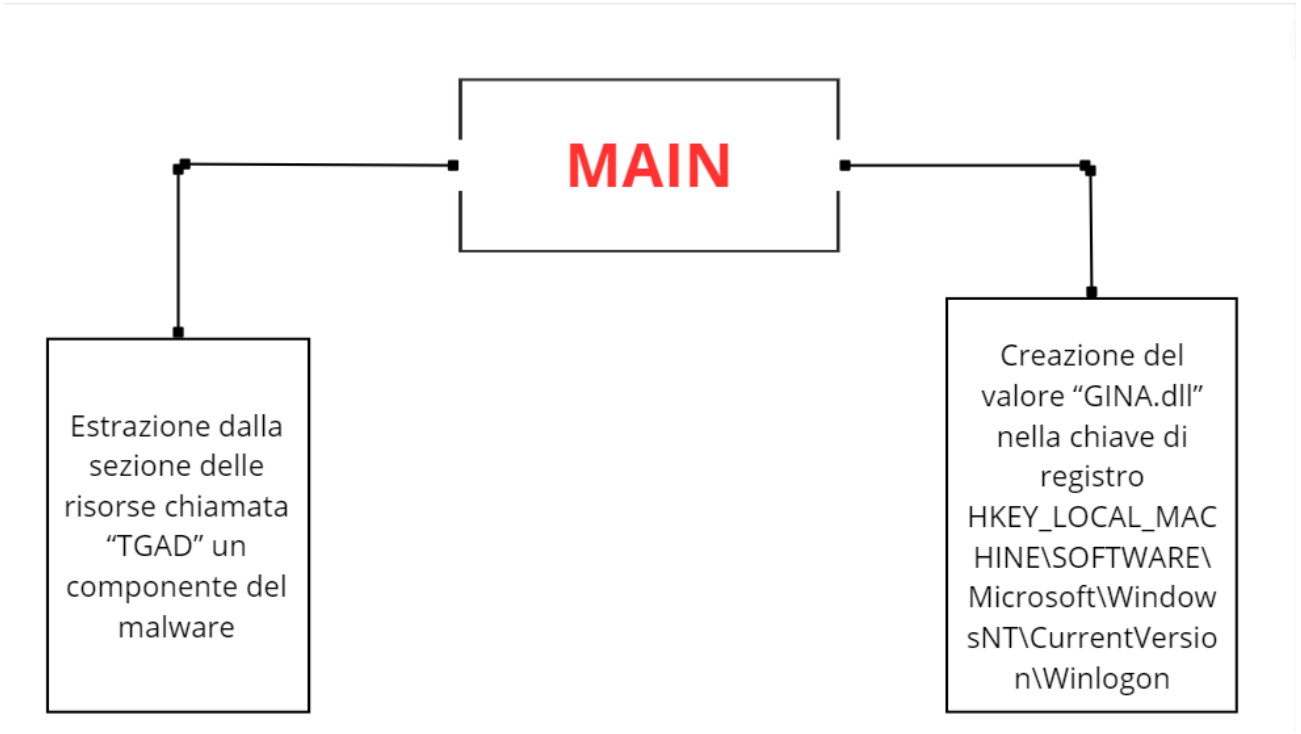
funzioni.  
In questo modo possiamo supporre l'obiettivo del malware senza avviarlo.

**Task 4:** In caso di risposta affermativa, elencare le evidenze a supporto.

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

In questo caso, le evidenze che riusciamo a riscontrare, sono le diverse funzioni contenute nella libreria KERNEL32.dll.  
Troviamo, per l'appunto, tutte le funzioni utilizzate da un **dropper**.

**Task 5:** Disegnare un diagramma di flusso (inserite all'interno dei box solo le informazioni circa le funzionalità principali) che comprenda le 3 funzioni.



Si possono distinguere tre funzioni che si occupano del malware:

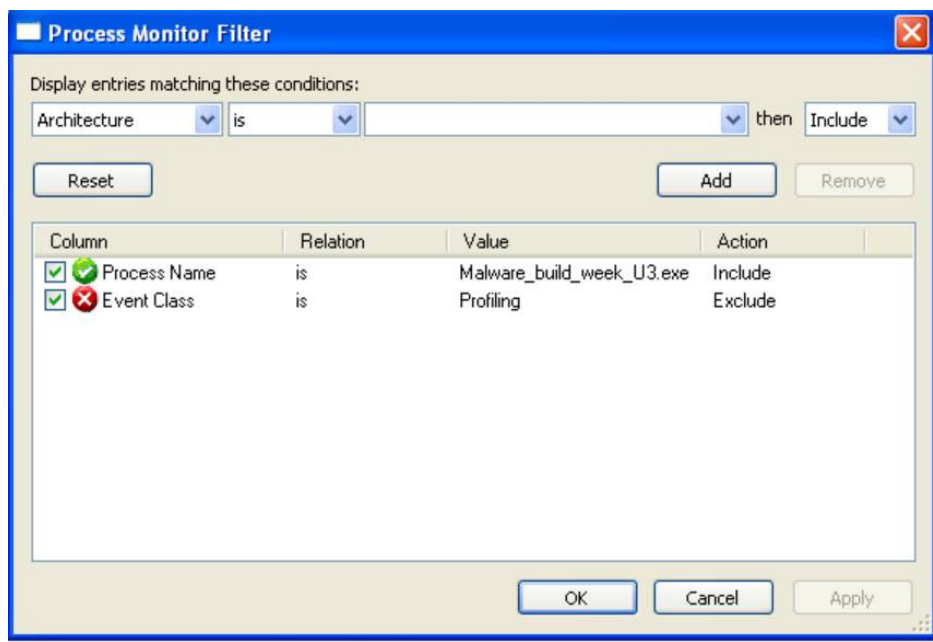
- La **funzione Main**.
- La funzione che si occupa di estrarre la risorsa "**ResourceName**" (in questo caso TGAD).
- La funzione che va a modificare il registro "Winlogon" creando il valore "**GINA.dll**".



# Giorno 4

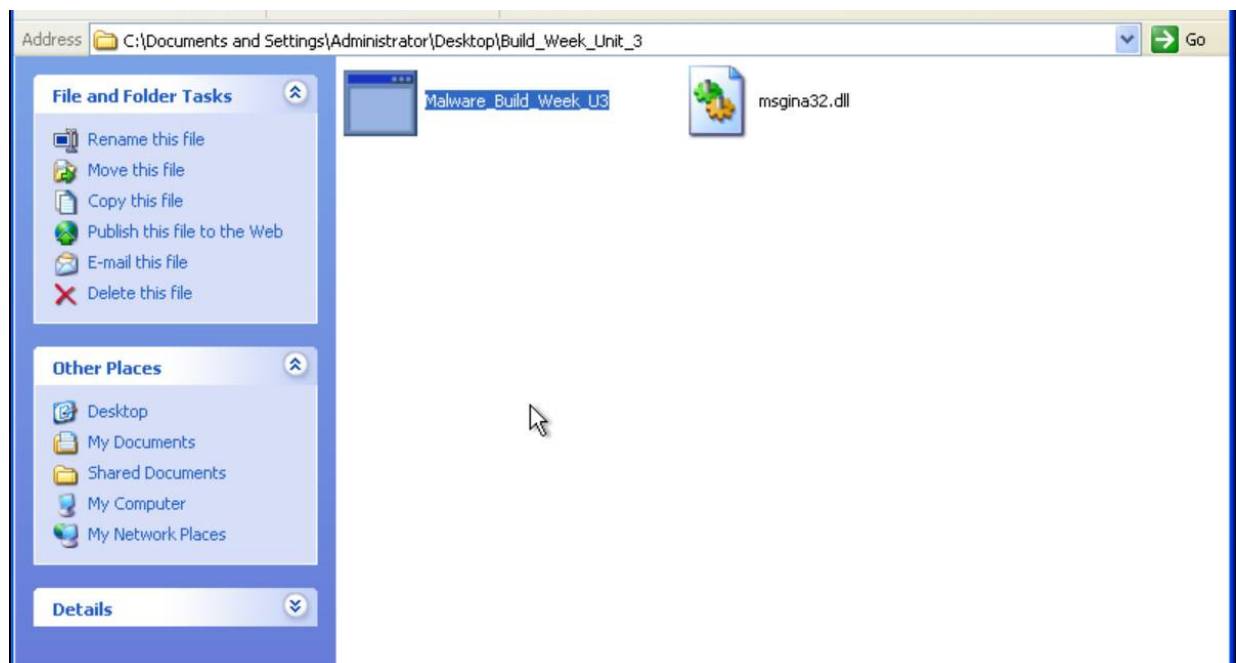
**Task 1:** A seguito dell'esecuzione del malware, cosa notate all'interno della cartella dove è situato l'eseguibile del Malware?

Una volta impostato un ambiente sicuro per l'esecuzione del malware (in questo caso la macchina virtuale Windows XP), abbiamo avviato Process Monitor e impostato i filtri in modo da analizzare il comportamento del malware all'avvio.



**Task 2:** A seguito dell'esecuzione del malware, cosa notate all'interno della cartella dove è situato l'eseguibile del Malware?

Prima di passare a Process Monitor, possiamo subito notare che all'interno della cartella dove è presente l'eseguibile viene creato un nuovo file chiamato "msgina32.dll". A seguito delle nostre ipotesi sul comportamento del malware come dropper, possiamo affermare che abbia scaricato sulla macchina una nuova libreria sconosciuta



**Tasks 3-4:** Quale chiave di registro viene creata? E quale valore viene associato ad essa?

10:02:47....	Malware_Build_Week_U3....	2608	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS

Tramite ProcMon, monitorando le attività dei registri, possiamo notare che il malware esegue tre precise istruzioni che si occupano di aprire una chiave di registro, modificarla e infine chiuderla. Analizzando il path su cui il malware va ad agire, viene creata la chiave di registro “Winlogon” e successivamente la modifica col valore “GinaDLL”.

**Task 5:** Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?

10:02:47....	Malware_Build_Week_U3....	2608	IRP_MJ_CREATE	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	IRP_MJ_CREATE	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	IRP_MJ_CLEANUP	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	IRP_MJ_CLOSE	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	IRP_MJ_WRITE	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	FAST_IO_WRITE	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	FAST IO DIS
10:02:47....	Malware_Build_Week_U3....	2608	IRP_MJ_WRITE	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	IRP_MJ_CLEANUP	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS
10:02:47....	Malware_Build_Week_U3....	2608	IRP_MJ_CLOSE	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS

Passando alla visualizzazione dell'attività sul file system, si possono notare delle istruzioni che vanno a modificare il contenuto della cartella dove è presente il file malevolo. La prima funzione “IRP\_MJ\_CREATE” crea la libreria “msgina32.dll” nella cartella del malware, mentre le seguenti funzioni “IRP\_MJ\_WRITE” modificano il contenuto della libreria stessa.

**Task 6:** Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware.

A valle dell'analisi statica e dinamica possiamo affermare che il malware estrae la libreria "msgina32.dll" (dalla sezione ".rsrc") nella sua stessa cartella.

Successivamente va a modificare la chiave di registro "Winlogon", che gestisce le funzioni di interfaccia indipendenti dai criteri di autenticazione (<https://learn.microsoft.com/it-it/windows/win32/secauthn/winlogon>), e ad assegnare il valore "GinaDLL". Possiamo quindi affermare che sia veritiera l'ipotesi che il malware sia un dropper che, nel momento in cui viene eseguito, estrae il file malevolo che contiene per salvarlo sulla macchina vittima.

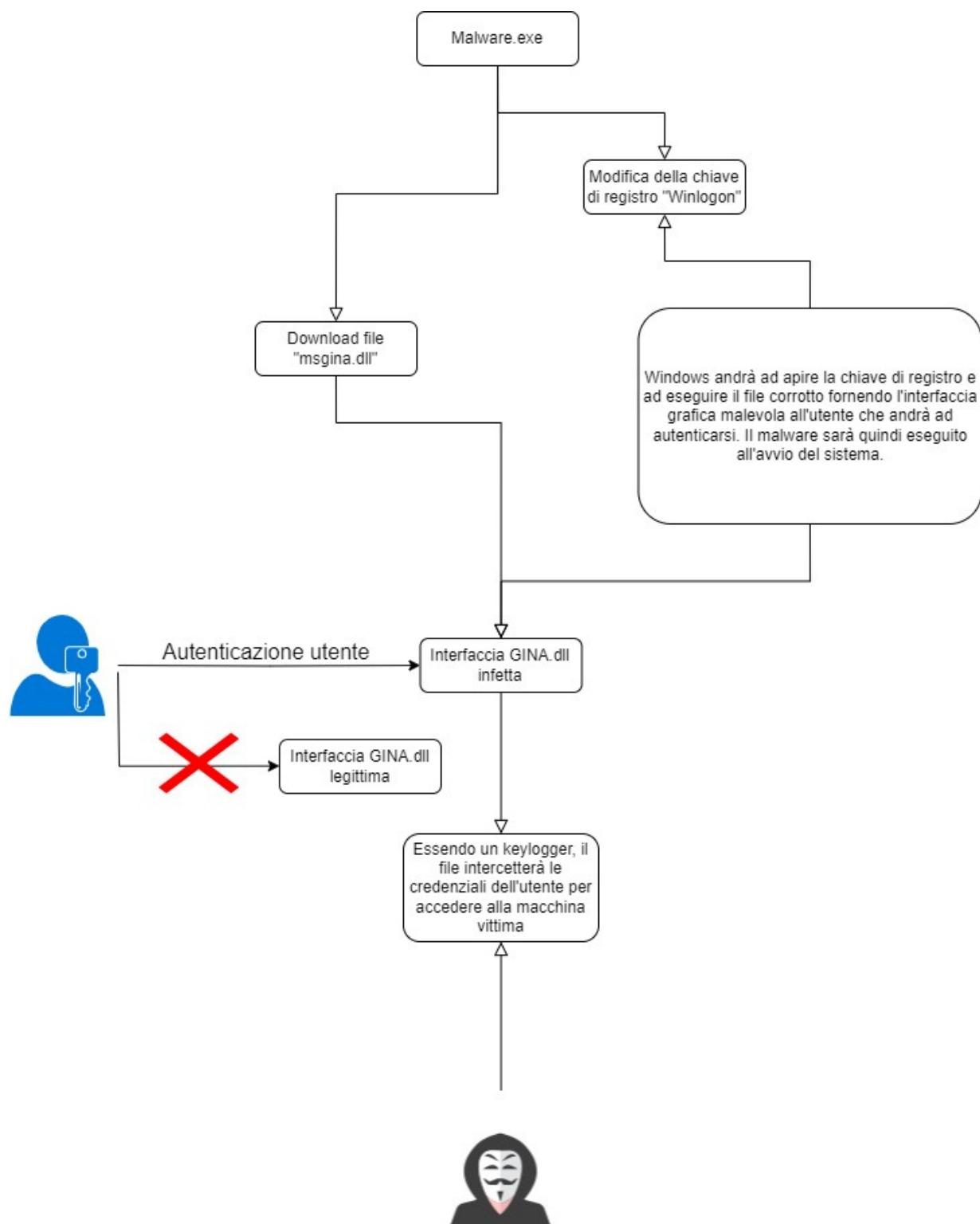
## Giorno 5

**Task 1:** Cosa può succedere se il file .dll lecito viene sostituito con un file .dll malevolo, che intercetta i dati inseriti?

Non ci resta che capire quale sia il funzionamento di "msgina32.dll".

GINA (Graphic authentication & authentication) è un componente lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica. Così facendo il malware va a sostituire la libreria ufficiale permettendo a quello nuovo di intercettare i dati inseriti dall'utente tramite una visualizzazione grafica uguale a quella del sistema: questo ci porta ad identificare il PE scaricato come un "keylogger", un particolare malware creato al fine di rubare le credenziali di accesso tramite tutto quello che l'utente andrà a digitare sulla tastiera.

Di seguito abbiamo creato un grafico che rappresenta lo scopo e i passaggi ad "alto livello":



## Bonus 1: analisi con Anyrun

**Task:** Analizzare l'analisi di anyrun e spiegarla in un piccolo report.  
Traduzione tra anyrun ed italiano per un eventuale cliente / manager.

A seguito di un'analisi di anyrun su un malware sconosciuto, andiamo ad impostare un piccolo report con le informazioni essenziali.

## 1) Attività chiave del malware:

Parte del dropper:

- Viene scaricato dall'utente il file Uqzqkjvt.exe.
- Uqzqkjvt.exe scarica un secondo file malevolo RegAsm.exe.

Parte dello spyware:

- RegAsm.exe è anche conosciuto come "Agent Tesla" (Agent Tesla è uno spyware che raccoglie informazioni sulle azioni delle sue vittime registrando i tasti premuti e le interazioni dell'utente. Viene erroneamente pubblicizzato come un software legittimo su un sito web dedicato dove viene venduto questo malware.).
- Il malware sta creando nuovi file e cartelle nella directory dell'utente in modo da ottenere persistenza e quindi nascondere le sue azioni all'interno del sistema.
- Il malware sta leggendo le informazioni del computer vittima.
- Il malware sta cercando di ottenere informazioni sulle configurazioni di rete del sistema compromesso ed utilizzarle in modo da adattare il suo comportamento.
- Individua la porta SMTP (25) e si connette a quest'ultima in modo da avviare un canale di comunicazione.
- Agent Tesla estrae le credenziali dai log e dai registri a seguito di un'azione di keylogging.
- Tutti i dati estratti vengono poi inviati ad un server Command and Control sotto il controllo dell'attaccante attraverso SMTP.

## 2) Come difendersi:

- Investire sulla formazione del personale aziendale (consigliato l'utilizzo di Phishing Attack Simulation: un tool per la simulazione di attacchi phishing)
- Saper riconoscere una mail di phishing dai certificati SPF, DKIM e DMARC  
(**SPF**: specifica quali host possono inviare messaggi da un determinato dominio tramite l'indirizzo IP.  
**DKIM**: controlla se l'email non è stata compromessa consentendo al mittente di firmare elettronicamente l'email con una chiave pubblica, che i destinatari possono utilizzare per decrittografare l'hash.  
**DMARC**: è uno standard di sicurezza utile al filtraggio email, che utilizza i due precedenti certificati per verificare la legittimità di un messaggio.).

## Illustrazione del funzionamento e origine del malware:



Durante l’analisi si è venuti a conoscenza di una mail associata a questo malware (asia@asiaparadisehotel.com).

Da questa evidenza, si è iniziato a supporre che il malware venisse trasmesso tramite campagne di phishing.

Le vittime venivano ingannate e portate a scaricare il file “Uqzqkqvjt.exe”, un dropper presente nella email.

Una volta avviato, il malware scaricava un secondo file malevolo “RegAsm.exe” che fungeva da spyware e keylogger, con l’obiettivo di rubare tutti i dati relativi alle configurazioni internet (quindi eventuali cookies e password salvate sul browser) ed inviarli al server dell’attaccante.

## Bonus 2: Parte 1

**Task:** Analizzare il file “calcolatriceinnovativa50.exe.zip” con gli strumenti che conoscete andando a confermare che è un malware.

Abbiamo iniziato con l’utilizzo di VirusTotal per eseguire una prima analisi statica del malware. Dopo aver inserito il codice hash, abbiamo trovato le seguenti informazioni:

53  
/ 71

Community Score

53 security vendors and 2 sandboxes flagged this file as malicious

Reanalyze Similar More

b8ed129eb56c68cec1661206c313c6eab2e20e4b9223336f7edf661c9956e81a

Size 112.50 KB

Last Analysis Date 20 hours ago

EXE

CALC.EXE

peexe idle checks-user-input

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 1

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

Popular threat label 1 trojan.swrort/cryptz

Threat categories trojan

Family labels swrort cryptz marte

Security vendors' analysis 1

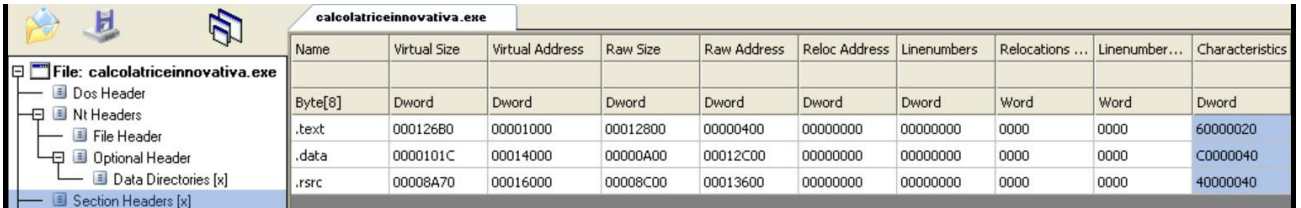
Do you want to automate checks?

Alibaba	1 Trojan:Win32/CobaltStrike.5c89	ALYac	1 Trojan.CryptZ.Marte.1.Gen
Antiy-AVL	1 Trojan/Win32.Rozena	Arcabit	1 Trojan.CryptZ.Marte.1.Gen
Avast	1 Win32:SwPatch [Wrm]	AVG	1 Win32:SwPatch [Wrm]
Avira (no cloud)	1 TR/Patched.Gen2	BitDefender	1 Trojan.CryptZ.Marte.1.Gen
BitDefenderTheta	1 Gen:NN.ZexaF.36608.hm0@ayKeBUjc	Bkav Pro	1 W32.AIDetectMalware
ClamAV	1 Win.Trojan.MSShellcode-6360730-0	CrowdStrike Falcon	1 Win/malicious_confidence_100% (W)

Possiamo ipotizzare che il file malevolo sia un “Trojan” che nasconde al suo interno il file per infettare la macchina vittima. Continuando con l’analisi statica, abbiamo utilizzato CFFExplorer per identificare le sezioni e le librerie importate del malware:

**Sezioni:**

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



The screenshot shows the CFF Explorer interface. On the left, a tree view displays the file structure: File: calcolatriceinnovativa.exe, Dos Header, Nt Headers, File Header, Optional Header, Data Directories [x], and Section Headers [x]. The main pane shows a table of sections:

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	000126B0	00001000	00012800	00000400	00000000	00000000	0000	0000	60000020
.data	0000101C	00014000	00000A00	00012C00	00000000	00000000	0000	0000	C0000040
.rsrc	00008A70	00016000	00008C00	00013600	00000000	00000000	0000	0000	40000040

**Librerie:**

- **SHELL32.dll**: contiene funzioni che gestiscono diverse componenti dell'interfaccia utente di Windows, inclusi elementi del desktop, icone, menu contestuali e barre degli strumenti;
- **msvcrt.dll**: contiene le funzioni per la manipolazione delle stringhe o di allocazioni di memoria;
- **ADVAPI32.dll**: contiene le funzioni per interagire con registri e servizi di Windows;
- **KERNEL32.dll**: contiene le funzioni principali per interagire con il sistema operativo. Un malware potrebbe sfruttare tale libreria per manipolare i file e per accedere la gestione della memoria;
- **GDI32.dll**: contiene le funzioni per l’implementazione e manipolazione della grafica e dei suoi effetti;
- **USER32.dll**: contiene un insieme di funzioni e procedure di interfaccia utente (UI) che consentono alle applicazioni di interagire con l'utente e gestire elementi dell'interfaccia grafica del sistema operativo.



calcolatriceinnovativa.exe						
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
00012242	N/A	00011F80	00011F84	00011F88	00011F8C	00011F90
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
SHELL32.dll	1	00012CA8	FFFFFFFF	FFFFFFFF	00012E42	0000109C
msvcrt.dll	26	00012DC8	FFFFFFFF	FFFFFFFF	00012F60	0000118C
ADVAPI32.dll	3	00012C0C	FFFFFFFF	FFFFFFFF	00012FFC	00001000
KERNEL32.dll	30	00012C2C	FFFFFFFF	FFFFFFFF	000131D4	00001020
GDI32.dll	3	00012C1C	FFFFFFFF	FFFFFFFF	0001320C	00001010
USER32.dll	69	00012CB0	FFFFFFFF	FFFFFFFF	000136A4	000010A4
OFTs	FTs (IAT)	Hint	Name			
Dword	Dword	Word	szAnsi			
00012E34	7748E3DB	0094	ShellAboutW			

Dopo aver ultimato i passaggi di analisi statica, è possibile procedere con quella dinamica avviando il file e lavorando sempre in ambiente sicuro. Anche attraverso ProcMon possiamo notare che l'eseguibile importa le librerie trovate in precedenza.

2:30:56.84821...	c..	464	Load Image	C:\Documents and Settings\Administrator\Desktop\public-archived\4-823\calcolatriceinnovativa.exe	SUCCESS	Image Base: 0x1000000, Image Size: 0x10000
2:30:56.84928...	c..	464	Load Image	C:\WINDOWS\system32\ntdll.dll	SUCCESS	Image Base: 0x7c900000, Image Size: 0x40000
2:30:56.98157...	c..	464	Load Image	C:\WINDOWS\system32\kernel32.dll	SUCCESS	Image Base: 0x7c800000, Image Size: 0x60000
2:30:56.99200...	c..	464	Load Image	C:\WINDOWS\system32\shell32.dll	SUCCESS	Image Base: 0x7c9c0000, Image Size: 0x817000
2:30:57.00413...	c..	464	Load Image	C:\WINDOWS\system32\advapi32.dll	SUCCESS	Image Base: 0x77dd0000, Image Size: 0x3b000
2:30:57.04135...	c..	464	Load Image	C:\WINDOWS\system32\user32.dll	SUCCESS	Image Base: 0x77e70000, Image Size: 0x32000
2:30:57.07240...	c..	464	Load Image	C:\WINDOWS\system32\gdi32.dll	SUCCESS	Image Base: 0x771e0000, Image Size: 0x11000
2:30:57.10924...	c..	464	Load Image	C:\WINDOWS\system32\ole32.dll	SUCCESS	Image Base: 0x771f0000, Image Size: 0x49000
2:30:57.13173...	c..	464	Load Image	C:\WINDOWS\system32\ole32.dll	SUCCESS	Image Base: 0x7e410000, Image Size: 0x31000
2:30:57.16173...	c..	464	Load Image	C:\WINDOWS\system32\ole32.dll	SUCCESS	Image Base: 0x77c10000, Image Size: 0x58000
2:30:57.17207...	c..	464	Load Image	C:\WINDOWS\system32\ole32.dll	SUCCESS	Image Base: 0x77160000, Image Size: 0x76000
2:30:57.55204...	c..	464	Load Image	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2500.5512_x-ww_35d4ce83\comctl32.dll	SUCCESS	Image Base: 0x773d0000, Image Size: 0x103000

Analizzando l'eseguibile con OllyDBG, abbiamo notato che il malware passa dei valori (ovvero i pulsanti presenti sulla tastiera) a diverse variabili di tipo char. Data l'analisi sulle librerie, abbiamo notato che il malware ottiene la persistenza. Unendo le due informazioni abbiamo dedotto che potrebbe trattarsi di uno spyware (keylogger).

0101202F	74	DB 74	CHAR 't'
01012030	7D	DB 7D	CHAR 'j'
01012031	0E	DB 0E	
01012032	C1	DB C1	
01012033	55	DB 55	CHAR 'U'
01012034	D8	DB D8	
01012035	. 8B45 F1	MOV EAX,DWORD PTR SS:[EBP-F]	
01012038	. 8B5D E0	MOV EBX,DWORD PTR SS:[EBP-20]	
0101203B	. 024D 0C	ADD CL,BYTE PTR SS:[EBP+C]	
0101203E	. EB 95	JMP SHORT calcolat.01011F05	
01012040	83	DB 83	
01012041	47	DB 47	
01012042	EB	DB EB	CHAR 'G'
01012043	04	DB 04	
01012044	83	DB 83	
01012045	45	DB 45	CHAR 'E'
01012046	EC	DB EC	
01012047	04	DB 04	
01012048	FF	DB FF	
01012049	4D	DB 4D	CHAR 'M'
0101204A	F8	DB F8	
0101204B	10	DB 10	
0101204C	6F	DB 6F	CHAR 'o'
0101204D	F8	DB F8	
0101204E	00	DB 00	
0101204F	0F	DB 0F	
01012050	8F	DB 8F	
01012051	4B	DB 4B	CHAR 'K'
01012052	FF	DB FF	
01012053	65	DB 65	
01012054	FF	DB FF	
01012055	A3	DB A3	
01012056	55	DB 55	CHAR 'U'
01012057	1C	DB 1C	
01012058	FF	DB FF	
01012059	95	DB 95	
0101205A	F4	DB F4	
0101205B	7B	DB 7B	
0101205C	7D	DB 7D	CHAR '{'
0101205D	F4	DB F4	CHAR '}'
0101205E	00	DB 00	
0101205F	0F	DB 0F	
01012060	8F	DB 8F	



# Bonus 2: Parte 2

**Task:** dato il file “AmicoNerd”, Il nostro compito è convincere il dipendente che è malevolo.

Come fatto in precedenza, abbiamo iniziato la nostra analisi partendo dal web tool VirusTotal per capire se effettivamente risulta dannoso.

52

/ 72

Community Score

52 security vendors and 1 sandbox flagged this file as malicious

Reanalyze Similar More

c6603d416dfc48894eda35d9a9a8523bdf9823e215ab926783ce6848aa8a62c4

Size722.69 KBLast Analysis Date23 days ago

AutoPico.exe

peexeassemblyoverlayrevoked-certruntime-modulesinvalid-signaturesigneddetect-debug-environmentchecks-network-adapterslong-sleepsdirect-cpu-clock-accessvia-torcalls-wmi

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY20 +

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

Popular threat labelhacktool.autokms/rpchook

Threat categorieshacktoolpua Trojan

Family labelsautokmsrpchookkmsactivator

Security vendors' analysis

Do you want to automate checks?

AhnLab-V3	HackTool/Win.AutoKMS.C948312	ALYac	Application.Hacktool.KMSActivator.AQ
AntiV-AVL	RiskWare[NetTool]/Win64.RPCHook	Arcabit	Application.Hacktool.KMSActivator.AQ
Avast	Win32:MiscX-gen [PUP]	AVG	Win32:MiscX-gen [PUP]
BitDefender	Application.Hacktool.KMSActivator.AQ	BitDefenderTheta	Gen:NN.ZemsiIF.36792.Tm1@a8vJERd
ClamAV	Win.Tool.Kmsactivator-9811695-0	CrowdStrike Falcon	Win/grayware_confidence_100% (W)

Da questa schermata possiamo notare due notazioni particolari:

- **hacktool.autokms/rpchook**: cercando a fondo, esso può essere usato per trovare le “crack” o le “patch” per software non registrati di Microsoft e, la maggior parte dei programmi antivirus, lo identificano come malware (<https://www.microsoft.com/en-US/wdsi/threats/malware-encyclopedia-description?name=hacktool:win32/autokms>);
- Il termine "riskware": si riferisce a software che, sebbene non sia dannoso per il computer o i dati dell'utente, può comunque rappresentare un rischio o una minaccia potenziale. In genere, il riskware non è progettato per danneggiare direttamente un sistema, ma potrebbe avere funzionalità che potrebbero essere utilizzate in modo malevolo.

Per aggiungere informazioni all'analisi basta utilizzare ancora una volta CFFExplorer così da poter entrare nel dettaglio delle sezioni e delle librerie importate:

### Sezioni:

- **.text**: contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto;
- **.rsrc**: include le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell'eseguibile stesso;
- **.rloc**: Contiene informazioni sulla rilocazione, che sono necessarie quando un file eseguibile deve essere caricato in un'area diversa della memoria rispetto a quella in cui è stato

originariamente compilato. Questo è particolarmente importante per garantire che i programmi siano in grado di eseguire correttamente quando vengono caricati in diverse posizioni di memoria.

File: AmicoNerd.exe

- Dos Header
- Nt Headers
- File Header
- Optional Header
  - Data Directories [x]
- Section Headers [x]

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	000B2484	00002000	000B2600	00000200	00000000	00000000	0000	0000	60000020
.rsrc	00000E40	000B6000	00001000	000B2800	00000000	00000000	0000	0000	40000040
.reloc	0000000C	000B8000	00000200	000B3800	00000000	00000000	0000	0000	42000040

Librerie:

- **mscorlib.dll**: è libreria di sistema associata al Common Language Runtime (CLR) di Microsoft. Il CLR è un componente del framework .NET che gestisce l'esecuzione di programmi scritti in linguaggi come C# o VB.NET. Essa fornisce funzionalità fondamentali per il supporto del runtime .NET