



GIUGNO 2024

REPORT BUILDWEEK

Team 1

Scritto da:
Andrea Di Benedetto
Samuele Aversa
Lorenzo Franchi
Federico Biggi
Mario Reitano

INDICE

Day 1 | Report 1.0

<i>Introduzione</i>	2
<i>Traccia</i>	3
<i>Malware Analysis</i>	4
<i>Tool Utilizzati</i>	6
<i>Analisi dei Parametri</i>	7
<i>Analisi delle Variabili</i>	8
<i>Sezioni dei Malware</i>	9
<i>Librerie del Malware</i>	10
<i>Funzioni del Malware</i>	11
<i>Conclusioni</i>	14

Day 2 | Report 2.0

<i>Traccia</i>	15
<i>Apertura del file</i>	16
<i>Memoria e parametri</i>	17
<i>Parametro</i>	18
<i>Analisi Istruzioni</i>	19
<i>Codice Assembly in C</i>	19
<i>Valore di <ValueName></i> ..	20
<i>Conclusioni</i>	21

Day 3 | Report 3.0

<i>Traccia</i>	22
<i>Valore <ResourceName>...</i>	23
<i>Funzionalità Malware</i>	25
<i>Analisi basica statica</i>	26
<i>Evidenze a supporto</i>	27
<i>Diagramma di flusso</i>	28
<i>Conclusioni</i>	29

Day 4 | Report 4.0

<i>Traccia</i>	30
<i>Preparazione VM</i>	31
<i>Pre-Esecuzione</i>	32
<i>Esecuzione Malware</i>	33
<i>Registri Windows</i>	34
<i>File System</i>	36
<i>Conclusioni</i>	37

Day 5 | Report 5.0

<i>Traccia</i>	38
<i>Gina</i>	39
<i>Conseguenza</i>	40
<i>Diagramma</i>	41
<i>Analisi del diagramma</i>	42
<i>Conclusioni</i>	43
<i>Team</i>	44

INTRODUZIONE

L'analisi del malware nella cartella Build_Week_Unit_3 si svolge in più fasi. Inizialmente, si esamina il file eseguibile Malware_Build_Week_U3, identificando i parametri e le variabili della funzione Main(), descrivendo le sezioni principali del file e analizzando le librerie importate per formulare ipotesi sulle funzionalità del malware.

Successivamente, si approfondiscono specifiche funzioni del malware: la funzione a 00401021 viene analizzata per comprenderne lo scopo e il passaggio dei parametri, mentre le istruzioni tra 00401027 e 00401029 vengono esaminate e tradotte in costrutti C. Si valuta anche il parametro "ValueName" alla locazione 00401047.

L'analisi continua con l'esame delle routine tra 00401080 e 00401128 per determinare il valore del parametro "ResourceName" e le funzionalità implementate dal malware.

Viene poi eseguito il malware in un ambiente controllato utilizzando Process Monitor per osservare le modifiche al sistema, analizzando le attività sul registro di Windows e sul file system per identificare le chiavi di registro create e le modifiche alle cartelle.

Infine, si esplora l'uso di GINA (Graphical Identification and Authentication) e si ipotizza l'impatto della sostituzione di un file dll lecito con uno malevolo. Questa analisi complessiva permette di delineare il comportamento del malware e le sue principali funzionalità.

TRACCIA 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? Descrivete brevemente almeno 2 di quelle identificate
- 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.



MALWARE ANALYSIS

L'analisi del malware o «malware analysis» è l'insieme di competenze e tecniche che permettono ad un analista della sicurezza informatica di indagare accuratamente un malware per studiare e capire esattamente il suo comportamento al fine di rimuoverlo dal sistema. L'analisi dei malware si basa su due tecniche principali:

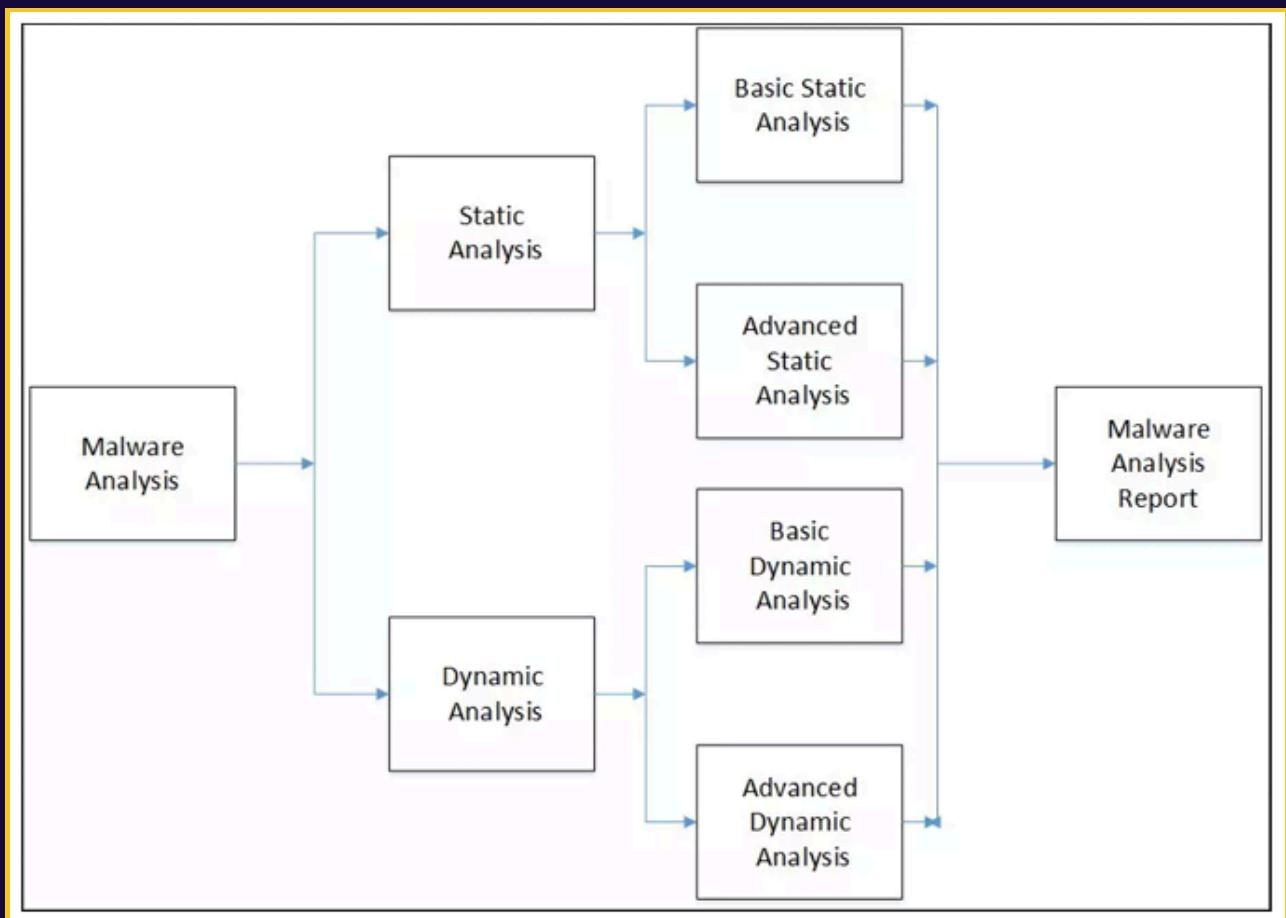
- **Analisi statica:** fornisce tecniche e strumenti per analizzare il comportamento di un software malevolo senza la necessità di eseguirlo.
- **Analisi dinamica:** presuppone l'esecuzione del malware in ambiente controllato

A loro volta queste due tecniche si dividono ulteriormente in altre due categorie: **basica** e **avanzata**.

- L'**analisi statica basica**: consiste nell'esaminare un eseguibile senza vedere le istruzioni che lo compongono. Lo scopo di tale analisi è di confermare se un dato file è malevolo e fornire informazioni generiche circa le sue funzionalità.
- L'**analisi statica avanzata**: presuppone la conoscenza dei fondamenti di «reverse-engineering» al fine di identificare il comportamento di un malware a partire dall'analisi delle istruzioni che lo compongono. In questa fase vengono utilizzati dei tool chiamati «disassambler» che ricevono in input un file eseguibile e restituiscono in output il linguaggio «assembly».
- L'**analisi dinamica basica**: presuppone l'esecuzione del malware in modo tale da osservare il suo comportamento sul sistema infetto al fine di rimuovere l'infezione. I malware devono essere eseguiti in ambiente sicuro e controllato in modo tale da eliminare ogni rischio di arrecare danno a sistemi o all'intera rete.
- L'**analisi dinamica avanzata**: presuppone la conoscenza dei debugger per esaminare lo stato di un programma durante l'esecuzione.

QUADRO RIASSUNTIVO

Le tecniche descritte sopra sono tra di loro complementari, per un'analisi efficace i risultati delle analisi statiche devono essere poi confermate dai risultati delle analisi dinamiche.



Lo studio e la comprensione del comportamento esatto di un malware è un compito piuttosto complicato. Capire preventivamente il tipo di malware da alcuni caratteri generali può dunque aiutare nella comprensione del comportamento generale e capire come può aver colpito un sistema bersaglio.

TOOL UTILIZZATI

L'obiettivo principale della malware analysis è determinare la natura, l'origine e l'impatto del malware, nonché sviluppare misure per prevenirlo e rimuoverlo. Il risultato della malware analysis è una comprensione dettagliata del malware, che può essere utilizzata per sviluppare soluzioni di sicurezza, aggiornare firme antivirus, creare regole di rilevamento e migliorare le difese complessive contro le minacce informatiche.

Per questo esercizio sono stati utilizzati due tool molto noti per l'analisi dei malware: IDA Pro, CFF Explorer.

IDA PRO

IDA Pro è un disassembler, utilizzato per l'analisi e l'ingegneria inversa di software.

Ida Pro riesce a mettere a disposizione degli analisti una serie di caratteristiche intuitive per semplificare le attività. Infatti, oltre alla traduzione completa del linguaggio macchina di un eseguibile in assembly, IDA identifica:

- Funzioni / chiamate di funzione
- Analisi dello stack
- Variabili locali e parametri

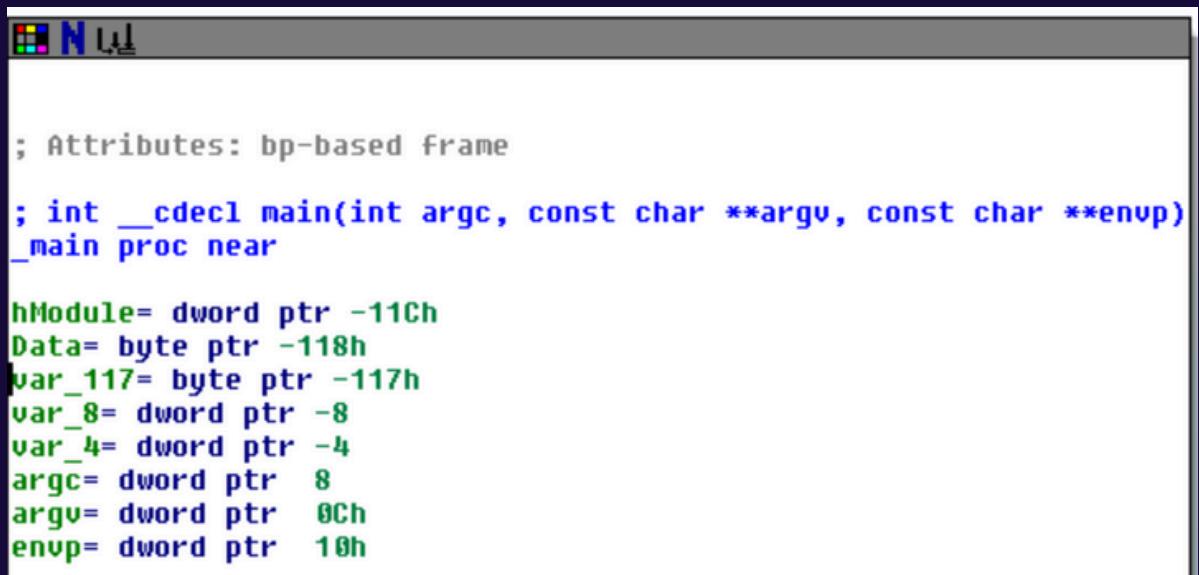
CFF EXPLORER

CFF Explorer è uno strumento di analisi e modifica di file eseguibili per il sistema operativo Windows. Questo strumento è particolarmente utile per esaminare, modificare o correggere file eseguibili PE (Portable Executable), che sono i formati dei file eseguibili standard su Windows. Viene utilizzato nell'analisi statica di base di un malware, per vedere in maniera rapida le sezioni di cui è composto e quali librerie e funzioni vengono importate.

ANALISI DEI PARAMETRI

All'interno della funzione main() sono dichiarati tre parametri:

- **argc**: Questo parametro rappresenta il numero di argomenti passati al programma dalla linea di comando. Include il nome del programma stesso come primo argomento.
- **argv**: Questo è un array di puntatori a stringhe (caratteri), ognuna delle quali contiene uno degli argomenti passati al programma dalla linea di comando. Il primo elemento (argv[0]) è il nome del programma, e i successivi elementi sono gli argomenti effettivi.
- **envp**: Questo è un array di puntatori a stringhe che rappresentano le variabili d'ambiente. Ogni stringa è una variabile d'ambiente con il formato NOME=VALORE. Questo array è terminato da un puntatore nullo.



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

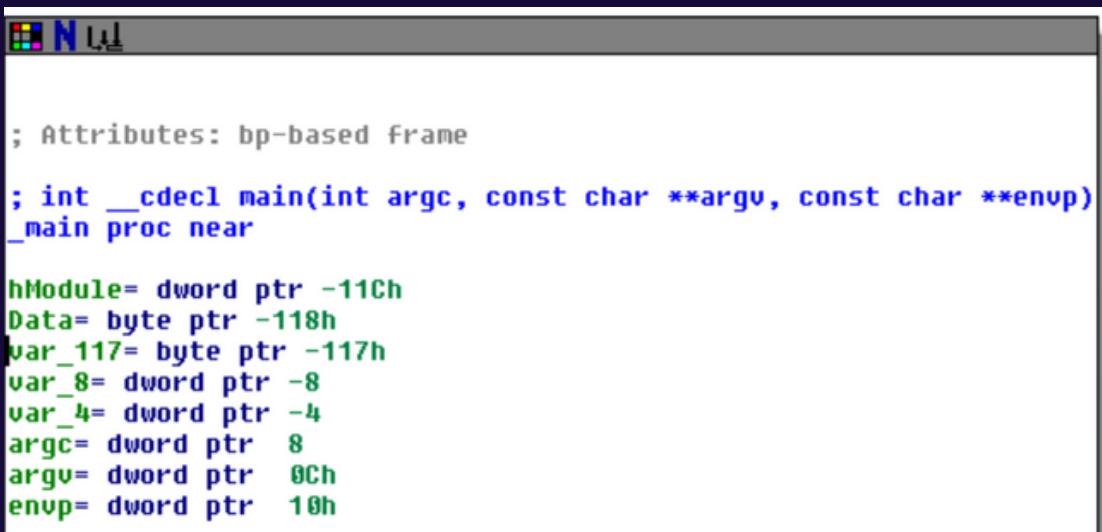
```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

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
```

ANALISI DELLE VARIABILI

All'interno della funzione main() sono dichiarate cinque variabili locali. Queste variabili sono utilizzate per memorizzare informazioni necessarie all'esecuzione del programma. Di seguito sono elencate e descritte le variabili locali:

- **hModule**: Questa variabile è di tipo dword ptr e viene utilizzata per memorizzare un handle al modulo. Gli handle ai moduli sono utilizzati per identificare un modulo specifico all'interno di un'applicazione.
- **Data**: Anche questa è una variabile di tipo byte ptr. La sua funzione specifica non è chiara dal nome, ma generalmente potrebbe essere utilizzata per memorizzare dati temporanei o buffer di byte.
- **var_117**: Questa variabile è di tipo byte ptr e, come suggerisce il nome, è probabilmente una variabile temporanea o di uso generale per memorizzare dati di tipo byte. Il nome è generato automaticamente dal decompilatore, il che indica che il suo utilizzo specifico non è stato determinato.
- **var_8**: Questa variabile è di tipo dword ptr e il suo utilizzo specifico non è evidente dal nome. Può essere utilizzata per vari scopi, come contatori, flag o per memorizzare indirizzi di memoria.
- **var_4**: Anche questa è una variabile di tipo dword ptr, utilizzata per scopi generici simili a var_8.



```

; 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

```

SEZIONI DEL MALWARE

Per visualizzare le sezioni all'interno del malware, ci avvaliamo dell'uso di **CFF Explorer**.

Nella sezione **Sections Headers**, presentata qui sotto, possiamo vedere che il malware è composto da 4 sezioni:

- **.text**: Contiene il codice eseguibile (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato.
- **.rdata**: Contiene dati di sola lettura, solitamente le informazioni riguardanti le librerie e le funzioni importate.
- **.data**: Contiene dati/ variabili globali, che devono essere disponibili da qualsiasi parte del programma.
- **.rsrc**: Include dati di risorse come ad esempio icone, immagini, menu e stringhe che non sono parte dell'eseguibile stesso.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations 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

LIBRERIE IMPORTATE DAL MALWARE

In Import Directory su CFF Explorer possiamo notare 2 librerie che vengono richiamate dal malware.

L'eseguibile importa due librerie:

- KERNEL32.dll**: Una libreria essenziale di Windows che fornisce funzioni di basso livello per la gestione della memoria, dei processi, dei file e degli input/output. Gestisce anche il tempo e le date, la sincronizzazione dei thread e le risorse di sistema. Questa DLL è fondamentale per il corretto funzionamento delle applicazioni e delle operazioni di sistema.
- ADVAPI32.dll**: Una libreria di Windows che offre funzioni avanzate per la gestione dei servizi, del registro di sistema, della sicurezza e della crittografia. Fornisce anche strumenti per il controllo degli accessi e la gestione degli eventi. Questa DLL è cruciale per operazioni di sistema avanzate e per la protezione del sistema operativo.

Malware_Build_Week_U3.exe			
Module Name	Imports	OFTs	TimeDate
szAnsi	(nFunctions)	Dword	Dword
KERNEL32.dll	51	00007534	00000000
ADVAPI32.dll	2	00007528	00000000

FUNZIONI LIBRERIE

Analizzando la libreria Kernel32.dll possiamo notare alcune funzioni:

- **SizeofResource**: Questa funzione recupera la dimensione, in byte, di una risorsa specifica all'interno di un modulo eseguibile. È utilizzata per determinare quanto spazio occupa una risorsa caricata in memoria.
- **LockResource**: Questa funzione ottiene un puntatore ai dati di una risorsa caricata in memoria. Una volta che una risorsa è stata caricata con LoadResource, LockResource consente di accedere ai dati effettivi della risorsa.
- **LoadResource**: Questa funzione carica una risorsa specificata in memoria. Viene utilizzata per caricare risorse come bitmap, icone, stringhe o altre risorse definite in un modulo eseguibile.

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

Queste funzioni sono utilizzate insieme per gestire e accedere alle risorse incorporate all'interno di eseguibili e librerie di Windows.

FUNZIONI LIBRERIE

Analizzando la libreria Kernel32.dll possiamo notare alcune funzioni:

- **CreateFileA**: Questa funzione apre o crea un file, un dispositivo I/O, un flusso di comunicazione o una risorsa di dati. Restituisce un handle che può essere utilizzato per accedere all'oggetto.
- **LoadLibraryA**: Questa funzione carica una libreria dinamica (DLL) specificata nel processo chiamante. Restituisce un handle alla DLL che può essere utilizzato con altre funzioni.
- **GetProcAddress**: Questa funzione recupera l'indirizzo di una funzione esportata o di una variabile da una DLL specificata. Viene utilizzata per chiamare funzioni della DLL caricate in fase di esecuzione.

00078E6	000078E6	0034	CreateFileA
00078F4	000078F4	00BF	GetCPInfo
0007900	00007900	00B9	GetACP
000790A	0000790A	0131	GetOEMCP
0007916	00007916	013E	GetProcAddress
0007928	00007928	01C2	LoadLibraryA
0007938	00007938	0261	SetEndOfFile
0007948	00007948	0218	ReadFile
0007954	00007954	01E4	MultiByteToWideChar

Queste funzioni sono fondamentali per la gestione di file, il caricamento di librerie dinamiche e l'accesso alle funzioni esportate dalle DLL in Windows.

FUNZIONI LIBRERIE

Tra le funzioni importate dal malware possiamo vederne alcune del ADVAPI32.dll:

- **RegSetValueExA:** permette di aggiungere un nuovo valore all'interno del registro e di settare i rispettivi dati. Accetta come parametri la chiave, la sottochiave e il dato da inserire. Ciò implica che il malware tenta di modificare o aggiungere un valore all'interno del Registro di sistema per poter persistere nel sistema o modificare le configurazioni.
- **RegCreateKeyExA:** viene usata per creare una nuova chiave o aprire una chiave esistente nel Registro di sistema. Potrebbe voler dire che il malware tenta di stabilire una persistenza all'interno del sistema o tenta di modificare le impostazioni di sistema.

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

CONCLUSIONI

Attraverso queste funzioni possiamo ipotizzare che il comportamento del malware riguardi il caricamento e l'uso di risorse all'interno del sistema, questo può farci pensare al caricamento di dati, asset e alla manipolazione delle risorse di sistema per l'esecuzione di codice malevolo o l'installazione di una backdoor. Inoltre, le funzioni ci permettono anche di comprendere che il malware interagisce con il registro di sistema e usa le funzioni di sistema per acquisire informazioni o probabilmente per modificare configurazioni di sistema. Dunque, l'analisi effettuata ci porta ad ipotizzare un comportamento tipico di un malware che tenta di ottenere l'accesso e il controllo del sistema attraverso la manipolazione del registro di sistema e l'uso delle risorse di sistema; infatti il registro di Windows è utilizzato per salvare informazioni sul sistema operativo come ad esempio configurazioni del sistema stesso o delle applicazioni che girano sul sistema e i malware lo usano spesso per ottenere la «persistenza».

TRACCIA GIORNO 2

Con riferimento al Malware in analisi, spiegare:

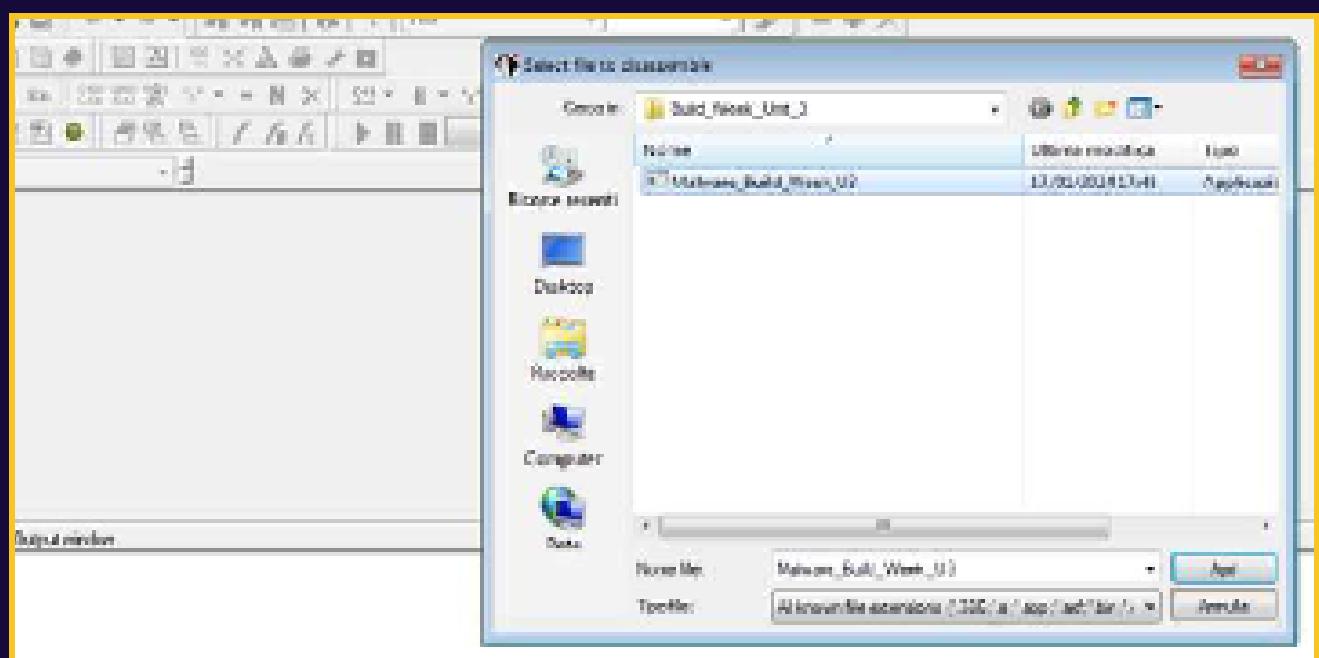
- Lo scopo della funzione chiamata alla locazione di memoria 00401021
- Come vengono passati i parametri alla funzione alla locazione 00401021 ;
- Che oggetto rappresenta il parametro alla locazione 00401017
- Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029 (se serve, valutate anche un'altra o altre due righe assembly).
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto 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.

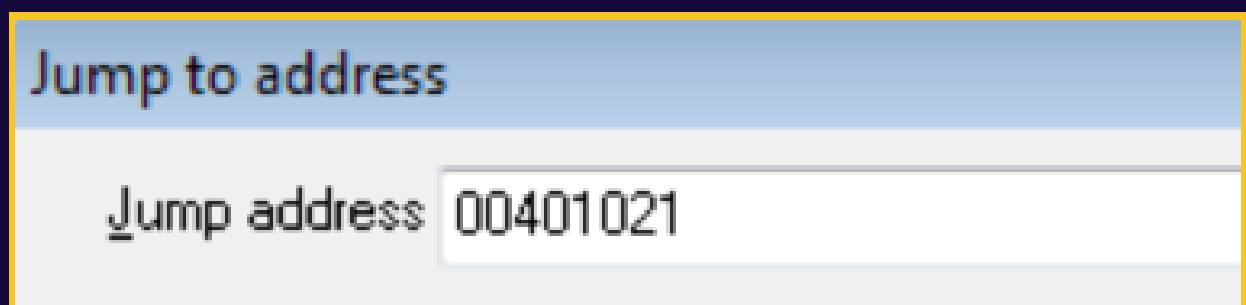


APERTURA DEL FILE

Per iniziare l'analisi del l'eseguibile dannoso, avviamo il software **IDA PRO** ed apriamo il malware.



Come primo step bisogna analizzare l'indirizzo **00401021** quindi tramite il Jump Address di IDA possiamo andare direttamente all'indirizzo specifico.



MEMORIA E PARAMETRI

La funzione chiamata alla locazione di memoria 00401021 è RegCreateKeyExA. Questa è una funzione dell'API di Windows che crea una chiave di registro o apre una chiave di registro esistente.

Scopo di RegCreateKeyExA:

- Aprire una chiave del registro di sistema se esiste.
- Creare una nuova chiave del registro di sistema se non esiste.
- Restituire un handle alla chiave aperta o creata.
- Restituire un valore che indica se la chiave è stata aperta o creata.

```
.text:00401021    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
```

Come vengono passati i parametri alla funzione alla locazione **00401021** ? I parametri vengono passati alla funzione RegCreateKeyExA tramite lo stack, utilizzando l'istruzione push per ciascun parametro. I parametri passati sono, in ordine:

- **hKey**: 0x80000002 (HKEY_LOCAL_MACHINE)
- **lpSubKey**: puntatore alla stringa "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon"
- **Reserved**: 0
- **lpClass**: 0
- **dwOptions**: 0
- **samDesired**: KEY_ALL_ACCESS
- **lpSecurityAttributes**: 0
- **phkResult**: puntatore a una variabile che riceve l'handle della chiave aperta/creata
- **lpdwDisposition**: puntatore a una variabile che riceve il valore di creazione (REG_CREATED_NEW_KEY o REG_OPENED_EXISTING_KEY)

PARAMETRO

Che oggetto rappresenta il parametro alla locazione **00401017**?

Il parametro alla locazione 00401017 è il secondo parametro passato alla funzione **RegCreateKeyExA** e rappresenta l'indirizzo della stringa **IpSubKey**. Questa stringa specifica il nome della chiave del registro che deve essere creata o aperta.

Nell'immagine, l'offset di questa stringa è etichettato come SubKey e punta a "**SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon**".

Questo è il percorso all'interno del registro di sistema che la funzione tenterà di aprire o creare.

A seguire una tabella per avere un sunto sui primi tre punti trattati:

Conclusione

Abbiamo esaminato il codice assembly per comprendere il funzionamento della chiamata alla funzione `RegCreateKeyExA` alla locazione di memoria `00401021`.

Scopo della Funzione: La funzione `RegCreateKeyExA` viene utilizzata per creare una chiave del registro di sistema o aprire una chiave esistente. Questo è fondamentale per modificare le impostazioni del sistema operativo memorizzate nel registro di Windows.

Passaggio dei Parametri: I parametri per la funzione sono passati tramite lo stack usando istruzioni `push`. Questo metodo ordina i parametri secondo l'ordine richiesto dalla funzione, permettendo così la corretta esecuzione della chiamata.

Parametro Specifico (IpSubKey): Alla locazione `00401017` viene passato il parametro `1psubkey`, che rappresenta l'indirizzo della stringa che specifica il percorso della chiave del registro ("SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon"). Questo parametro indica la chiave che deve essere creata o aperta dalla funzione.

Questi punti insieme illustrano come il programma interagisce con il registro di Windows per configurare o modificare chiavi specifiche, esemplificando l'uso diretto delle API di sistema a basso livello.

ANALISI ISTRUZIONI

tra gli indirizzi 00401027 e 00401029

Le istruzioni tra gli indirizzi 00401027 e 00401029 sono:

- **test eax, eax**: Testa il valore di eax impostando i flag del processore in base al risultato.
- **jz short loc_401032**: Questa istruzione salta alla locazione 401032 se il risultato del test è zero, cioè se eax è zero. In questo contesto, eax contiene il codice di ritorno della funzione RegCreateKeyExA. Se eax è zero, la funzione ha avuto successo.

Quindi, il codice verifica se la funzione RegCreateKeyExA ha avuto successo. Se sì, procede all'indirizzo 401032, altrimenti salta all'indirizzo 40107B.

CODICE ASSEMBLY IN CODICE C

Il segmento di codice Assembly tra 00401027 e 00401029 può essere tradotto in C come:

```
if (RegCreateKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\Microsoft\Windows NT", ...) == ERROR_SUCCESS) {
    goto loc_401032
} else {
    goto loc_40107B
}
```

Il codice tenta di creare o aprire la chiave di registro "SOFTWARE\Microsoft\Windows NT" sotto HKEY_LOCAL_MACHINE usando RegCreateKeyExA. Se l'operazione ha successo, esegue il codice associato a loc_401032; in caso contrario, esegue il codice associato a loc_40107B. L'istruzione if-else determina quale blocco di codice eseguire basandosi sul risultato della funzione.

VALORE DI <VALUENAME>

Alla locazione **00401047**, l'istruzione **push offset ValueName** carica l'offset della variabile **ValueName** nello stack. Questo fa parte della preparazione per la chiamata alla funzione **RegSetValueExA**, che serve a impostare il valore di una voce di registro di Windows.

Il commento accanto all'istruzione indica che **ValueName** contiene la stringa "**GinaDLL**". Questa stringa rappresenta il nome della chiave di registro che il malware sta cercando di creare o modificare. La funzione **RegSetValueExA** richiede l'handle della chiave di registro (**hKey**), il nome del valore da impostare (**ValueName**), il tipo di dati del valore e il valore effettivo da impostare.

L'istruzione successiva, **call ds:RegSetValueExA**, effettua la chiamata alla funzione con i parametri preparati. Il parametro **ValueName**, contenente "**GinaDLL**", specifica la voce del registro che sarà modificata o creata. "**GinaDLL**" è spesso associato a modifiche alla Graphical Identification and Authentication (**GINA**) su sistemi Windows, utilizzato dai malware per intercettare credenziali di accesso o altre attività dannose.

In sintesi, alla locazione **00401047**, il valore del parametro **ValueName** passato alla funzione **RegSetValueExA** è "**GinaDLL**", indicando che il malware sta tentando di manipolare questa chiave di registro per ottenere persistenza o eseguire altre azioni dannose.

```
.text:0040103C          push    $0                      ; Reserved
.text:0040103E          push    offset ValueName ; "GinaDLL"
.text:00401043          mov     eax, [ebp+hObject]
.text:00401046          push    eax                     ; hKey
.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                     ; hObject
.text:00401055          call    ds:CloseHandle
.text:0040105B          mov     eax, 1
```

CONCLUSIONI

Durante l'analisi del malware nella cartella Build_Week_Unit_3, abbiamo esaminato il codice assembly, focalizzandoci sulla funzione `RegCreateKeyExA` alla locazione `00401021`, utilizzata per creare o aprire chiavi del registro di sistema, modificando così le impostazioni di Windows.

I parametri per la funzione sono passati tramite lo stack usando istruzioni `push`. Alla locazione `00401017`, il parametro `lpSubKey`, che contiene il percorso della chiave del registro ("SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon"), viene passato alla funzione.

L'analisi delle istruzioni tra `00401027` e `00401029` ha mostrato come la funzione verifichi il successo della chiamata attraverso le istruzioni `test eax, eax` e `jz short loc_401032`. In caso di successo, il codice procede con ulteriori operazioni.

Alla locazione `00401047`, abbiamo visto come il parametro `ValueName` con il valore `"GinaDLL"` venga passato alla funzione `RegSetValueExA`, indicando una chiave di registro specifica che il malware tenta di modificare. Questa è spesso associata alla Graphical Identification and Authentication (GINA) su sistemi Windows, utilizzata dai malware per intercettare credenziali o altre attività dannose.

In sintesi, il malware interagisce con il registro di Windows per configurare o modificare chiavi specifiche, utilizzando le API di sistema per ottenere persistenza o eseguire altre azioni dannose. Questo studio fornisce una comprensione chiara del funzionamento interno del malware e delle sue tecniche per compromettere il sistema.

TRACCIA GIORNO 3

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.



VALORE DI <RESOURCENAME>

Per poter identificare il valore del parametro <ResourceName> passato alla funzione **FindResourceA()** è stato usato inizialmente lo strumento IDA Pro. Analizzando la parte iniziale della subroutine si sono identificati i parametri passati sullo stack per la funzione FindResourceA. Tale analisi non ha permesso di identificare il valore del parametro ResourceName.

```

t:00401000 ; 
t:00401004 ; 
t:00401008 loc_401008: ; CODE XREF: sub_401000+4
t:0040100B     mov    eax, lpType           ; lpType
t:0040100D     push   eax                ; lpType
t:00401011     mov    ecx, lpName          ; lpName
t:00401013     push   ecx                ; lpName
t:00401015     mov    edx, [ebp+hModule]    ; hModule
t:00401017     push   edx                ; hModule
t:00401019     call   ds:FindResourceA    ; FindResourceA
t:0040101F     mov    [ebp+hResInfo], eax ; hResInfo
t:00401021     cmd    [ebp+hResInfo].0   ; hResInfo
t:00401024     mov    ecx, lpName          ; lpName
t:00401026     ecx    ; lpName
t:00401028     edd    ; LPCSTR lpName
t:0040102A     dd offset aTgad         ; DATA XREF: sub_401000+4
t:0040102C     db ""TGAD""             ; "TGAD"

```

Per ovviare al problema è stato utilizzato un altro strumento per l'analisi dei malware, Olly DBG.

OllyDBG è un debugger, ovvero un software che permette di analizzare l'esecuzione di un eseguibile, evidenziando nel dettaglio le modifiche che esso subisce in tempo reale:

- come cambiano gli indirizzi di memoria
- Come cambia il contenuto dei registri della CPU
- Come si modificano i parametri di una funzione
- Come si modificano le variabili di una funzione.

VALORE DI <RESOURCENAME>

Con l'utilizzo del debugger OllyDBG è risultato più facile l'identificazione del valore del parametro **ResourceName**. Come si osserva dalla figura in basso, nel Disassembler window, che è la finestra in cui è mostrato il codice eseguito dalla CPU, si trovano dei commenti lasciati dal debugger stesso.

Si nota come il valore del parametro **ResourceName** è **TGAD**.

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 30004000	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.FindResourceA]	
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 E00000000	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.LoadResourceA]	
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_.004010FF	

ResourceType => "BINARY"
 Malware_.00408038
 ResourceName => "TGAD"
 hModule FindResourceA
 hResource
 hModule LoadResourceA

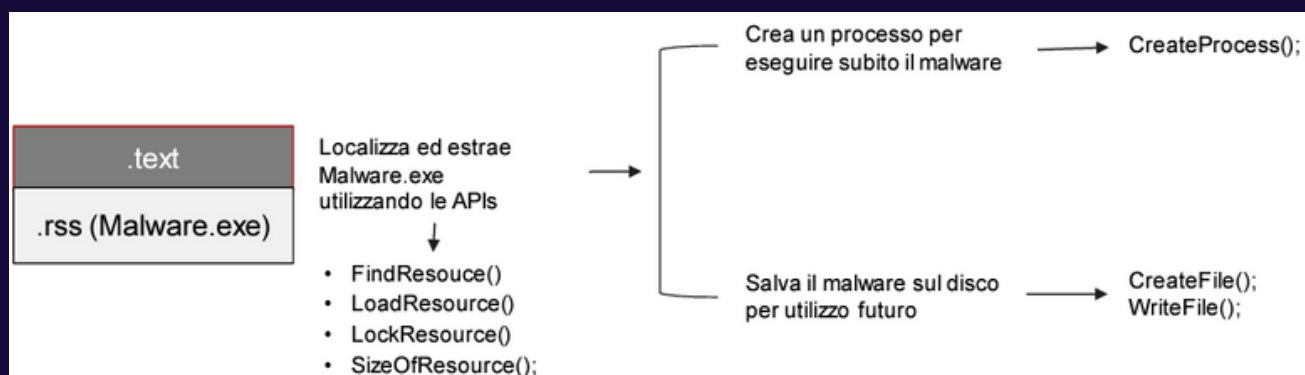
FUNZIONALITÀ MALWARE

Il susseguirsi delle chiamate di funzione che effettua il Malware nella sezione delimitata dalle locazioni 00401080 e 00401128 suggerisce un comportamento da **dropper**.

Il dropper è un programma malevolo che contiene al suo interno un malware. Nel momento in cui viene eseguito, un dropper inizia la sua esecuzione ed estrae il malware che contiene per salvarlo sul disco. Per estrarre il malware contenuto nella sezione delle risorse, i dropper utilizzano delle APIs come ad esempio:

- FindResource()
- LoadResource()
- LockResource()
- SizeOfResource()

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



Si potrebbe schematizzare un dropper come in figura sopra. Esso prima utilizza il set di APIs di Windows per estrarre il malware contenuto nel segmento risorse. Successivamente può creare un processo per eseguire immediatamente il malware, oppure salvare il malware sul disco, e magari eseguirlo in un secondo momento.

FUNZIONALITÀ TRAMITE ANALISI BASICA STATICÀ

Sì, è possibile identificare alcune funzionalità di questo eseguibile utilizzando l'analisi statica basica di CFFexplorer. L'immagine che possiamo vedere mostra la struttura delle importazioni delle funzioni di un file eseguibile, il che può rivelare molto sul comportamento del programma. Nella prossima pagina mostriremo alcune evidenze a supporto

Evidenze dall'Analisi Statica

- Modulo di Importazione e Funzioni Importate:
 - Il programma importa funzioni da KERNEL32.dll (51 funzioni) e ADVAPI32.dll (2 funzioni). Questi moduli sono tipicamente utilizzati per operazioni di base del sistema operativo e per l'accesso alle API di Windows.

Prosegue nella pagina seguente

The screenshot shows two tables of imported functions. The top table lists imports from KERNEL32.dll and ADVAPI32.dll, with KERNEL32.dll having 51 imports and ADVAPI32.dll having 2 imports. The bottom table lists specific functions from KERNEL32.dll, including VirtualAlloc, LoadResource, LockResource, SizeofResource, and GetProcAddress.

szAnsi	(# Functions)	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000
ADVAPI32.dll	2	00007528	00000000	00000000

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	

EVIDENZE A SUPPORTO

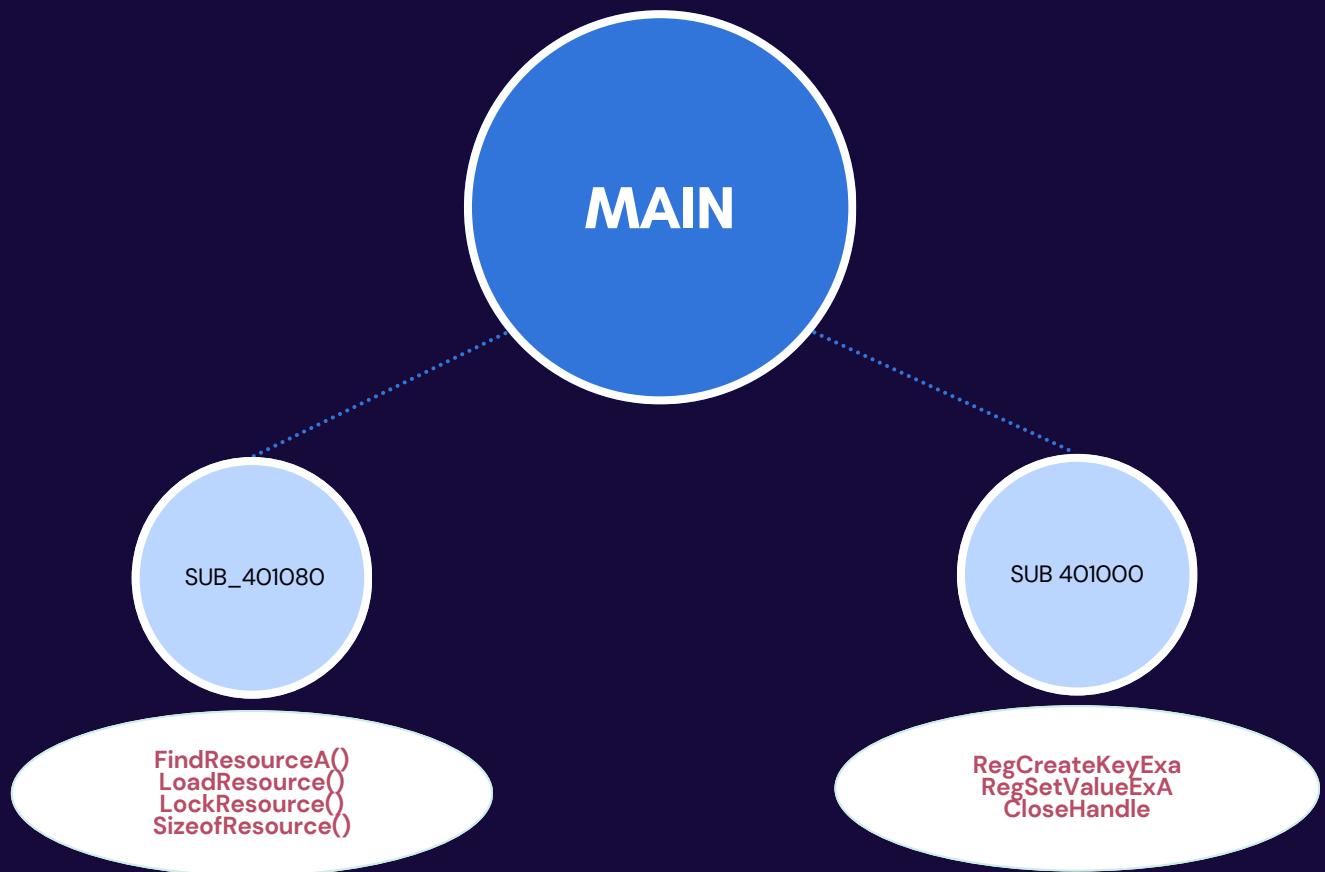
- Funzioni Specifiche Importate:
 - **szAnsi**: Utilizzato per la manipolazione di stringhe.
 - **SizeofResource, LockResource, LoadResource**: Queste funzioni suggeriscono che il programma potrebbe manipolare risorse incluse nell'eseguibile (es. caricamento di dati embedded come immagini, configurazioni, ecc.).
 - **VirtualAlloc**: Questa funzione è utilizzata per l'allocazione di memoria virtuale, il che potrebbe indicare operazioni di memoria dinamica come l'allocazione di buffer di dati o l'iniezione di codice.

Le evidenze a supporto sono:

- KERNEL32.dll (51 funzioni) e ADVAPI32.dll (2 funzioni) suggeriscono operazioni di sistema di basso livello.
- Funzioni Importate:
- Manipolazione di Risorse: Funzioni come SizeofResource, LockResource, e LoadResource indicano gestione di risorse incorporate.
- Allocazione di Memoria: VirtualAlloc suggerisce uso di memoria dinamica, potenzialmente per buffer di dati o iniezione di codice.
- Tabelle di Indirizzi:

DIAGRAMMA DI FLUSSO

Il diagramma di flusso mostra la struttura funzionale di un'applicazione software con il nodo "MAIN" al centro, collegato a due nodi secondari. Il nodo "SUB_401080" gestisce la ricerca, il caricamento, il blocco e la misurazione delle risorse, mentre il nodo "SUB_401000" è responsabile per la creazione, la modifica e la chiusura delle chiavi nel registro di sistema di Windows.



Questa rappresentazione aiuta a visualizzare l'architettura del programma e le interazioni tra le sue componenti principali.

CONCLUSIONI

Parametro «ResourceName»:

Il parametro `ResourceName` passato a `FindResourceA()` è "TGAD", utilizzato per identificare specifiche risorse all'interno dell'applicativo, potenzialmente impiegate per attività malevole.

Sequenza di Chiamate di Funzione:

Le chiamate a `FindResourceA()`, `LoadResource()`, `LockResource()` e `SizeofResource()` suggeriscono che il malware abbia le funzionalità di un dropper. Le funzioni chiamate indicano un'operazione di accesso e manipolazione di risorse interne, suggerendo tentativi di esecuzione o modifica di codice nascosto.

Funzionalità Malware e Analisi Statica:

Queste funzioni sono tipiche nelle tecniche di steganografia o occultamento di codice malevolo all'interno di risorse apparentemente legittime, complicando la rilevazione da parte di antivirus. L'uso di librerie come `KERNEL32.dll` e `ADVAPI32.dll` merita ulteriori indagini per comprendere meglio le interazioni del malware con il sistema ospite.

In sintesi:

Il malware sembra sfruttare risorse interne per occultare o eseguire codice dannoso, utilizzando metodi che potrebbero eludere il rilevamento e realizzare attività nocive in modo discreto. Questo comportamento suggerisce la necessità di ulteriori analisi per determinare l'esatta natura e lo scopo delle risorse manipolate.

TRACCIA GIORNO 4

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



PREPARAZIONE VM

Per effettuare l'analisi dei processi del malware, è meglio configurare il nostro ambiente virtuale in modo tale da isolarlo per evitare il diffondersi del malware nella rete o nel nostro sistema principale:

- Isolamento della Rete:
 - Impostiamo la rete su "Rete interna", in modo tale da evitare che il malware possa infettare altri dispositivi.
- Snapshot:
 - Prima di eseguire il malware, creiamo uno snapshot della VM così da poter ripristinare rapidamente la VM allo stato precedente in caso di problemi.
- Funzioni di Integrazione:
 - Disabilitiamo il copia-incolla e il drag-and-drop tra macchina host e macchina virtuale.
 - Impediamo la condivisione di cartelle tra host e guest così da prevenire il passaggio accidentale di file infetti.
- Monitoraggio:
 - Useremo poi strumenti di monitoraggio come Process Monitor per monitorare i processi attivi del malware.

PRE-ESECUZIONE MALWARE

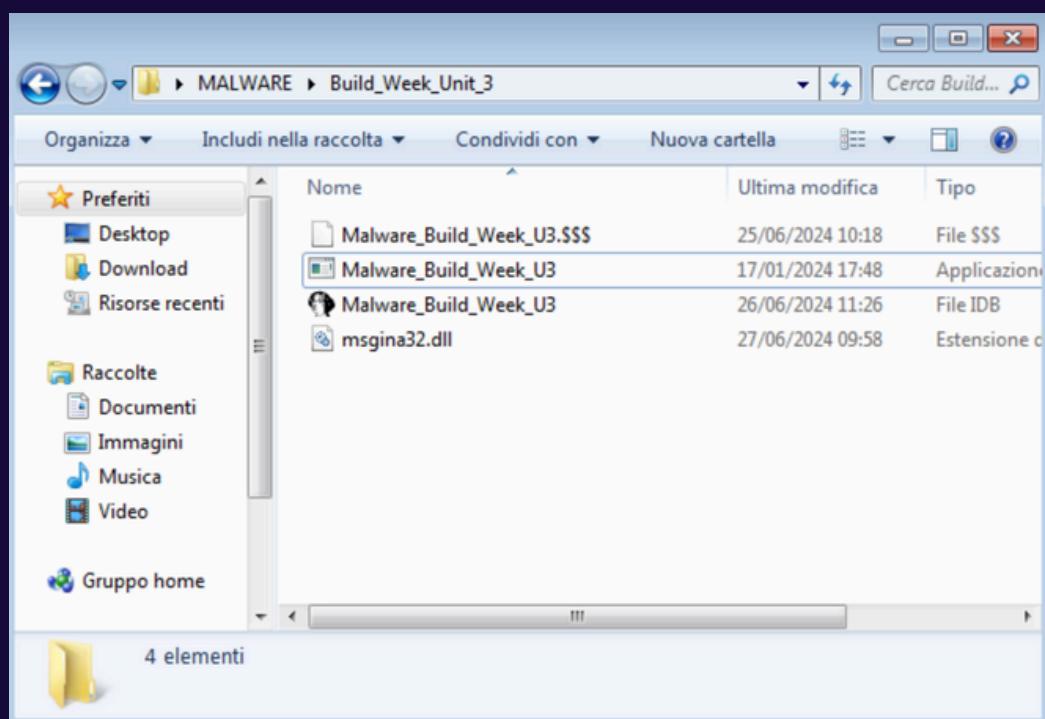
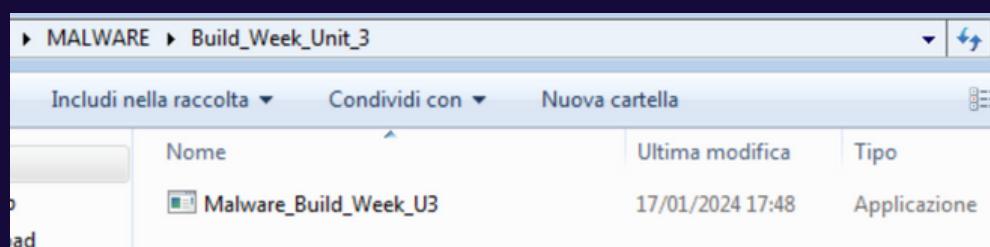
Abbiamo configurato Process Monitor applicando un filtro per visualizzare solo i processi eseguiti da Malware_Build_Week_U3.exe prima di avviare il malware. Questo ci ha permesso di concentrarci esclusivamente sulle attività generate dal malware, escludendo gli altri processi del sistema. Monitorando in tempo reale, abbiamo osservato i file creati, modificati o eliminati e le chiavi di registro coinvolte. Questa analisi ci ha fornito una chiara comprensione del comportamento del malware e delle sue interazioni con il sistema operativo, aiutandoci a identificare le sue funzionalità principali.

Display entries matching these conditions:

Process Name	is	Malware_Build_Week_U3.exe	then	Include
<input type="button" value="Reset"/>		<input type="button" value="Add"/> <input type="button" value="Remove"/>		
Column	Relation	Value	Action	
✓	Process N...	Malware_Build_...	Include	
✓	Process N...	Procmon.exe	Exclude	
✓	Process N...	Procexp.exe	Exclude	
✓	Process N...	Autoruns.exe	Exclude	
✓	Process N...	Procmon64.exe	Exclude	
✓	Process N...	Procexp64.exe	Exclude	
✓	Process N...	System	Exclude	

ESECUZIONE MALWARE

Prima di avviare il malware all'interno della nostra macchina virtuale, abbiamo avviato il tool Procmon, il quale ci permette di vedere i processi in tempo reale e filtrare quelli di nostro interesse. Una volta avviato il malware, abbiamo subito notato che sono stati creati tre file, ma in particolare poniamo la nostra attenzione su `msgina32.dll`, in quanto ha la stessa estensione delle librerie analizzate precedentemente. Questo ci porta a ipotizzare che questo elemento sia cruciale per il funzionamento del malware e, pertanto, sarà il nostro soggetto di analisi.



ATTIVITÀ REGISTRO WINDOWS

Per filtrare i processi e le attività legate al registro di Windows sono state sfruttate le funzionalità di Procmon.

Time ...	Process Name	PID	Operation	Path	Result	Detail
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\Software\Microsoft\Windows N...	SUCCESS	Desired Access: Q...
09:58:...	Malware_Build_...	2832	RegQueryValue	HKLM\SOFTWARE\MICROSOFT\WIN...NAME NOT FOUND	Length: 1.024	
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
09:58:...	Malware_Build_...	2832	RegQueryValue	HKLM\System\CurrentControlSet\Contr... NAME NOT FOUND	Length: 1.024	
09:58:...	Malware_Build_...	2832	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\SOFTWARE\Microsoft\WOW64 NAME NOT FOUND	Desired Access: Q...	
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\Software\Wow6432Node\Micro...	SUCCESS	Desired Access: Q...
09:58:...	Malware_Build_...	2832	RegSetInfoKey	HKLM\SOFTWARE\MICROSOFT\WIN...SUCCESS		KeySetInformation...
09:58:...	Malware_Build_...	2832	RegQueryValue	HKLM\SOFTWARE\MICROSOFT\WIN...NAME NOT FOUND	Length: 1.024	
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
09:58:...	Malware_Build_...	2832	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	KeySetInformation...
09:58:...	Malware_Build_...	2832	RegQueryValue	HKLM\System\CurrentControlSet\Contr... NAME NOT FOUND	Length: 1.024	
09:58:...	Malware_Build_...	2832	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
09:58:...	Malware_Build_...	2832	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
09:58:	Malware_Build_...	2832	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr... SUCCESS		KeySetInformation...

Grazie al filtraggio dei processi è risultato più facile identificare la chiave di registro creata. Si nota come l'operazione <RegSetValue> sia identificabile come quella che crea una nuova chiave di registro:

HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon.

948	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: Maximum Allowed, Granted Access: Read
948	RegOpenKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
948	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
948	RegOpenKey	HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Dia...	NAME NOT FOUND	Desired Access: Read
948	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0
948	RegCreateKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion	SUCCESS	Desired Access: All Access, Disposition: REG_OPENED_EXISTING_KEY
948	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0
948	RegQueryKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion	SUCCESS	Query: HandleTags, HandleTags: 0x400
948	RegSetValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion	ACCESS DENIED	Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_3\msigma32
948	RegCloseKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion	SUCCESS	
948	RegCloseKey	HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\Im...	SUCCESS	
948	RegCloseKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	SUCCESS	
948	RegCloseKey	HKLM	SUCCESS	

ATTIVITÀ REGISTRO WINDOWS

Il fatto che ci sia "ACCESS DENIED" significa che il tentativo di impostare il valore è stato negato dal sistema. Di conseguenza, il valore non viene effettivamente scritto nella chiave di registro.

Pertanto, il valore della chiave di registro non viene aggiornato con il nuovo valore desiderato. In altre parole, nonostante il valore che si tenta di associare sia

C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll,

a causa dell'accesso negato, questo valore non viene applicato e la chiave di registro non viene modificata.

```
NAME NOT FOUND Desired Access: Read
SUCCESS      Query: HandleTags, HandleTags: 0x0
SUCCESS      Desired Access: All Access, Disposition: REG_OPENED_EXISTING_KEY
SUCCESS      KeySetInformationClass: KeySetHandleTagsInformation, Length: 0
SUCCESS      Query: HandleTags, HandleTags: 0x400
ACCESS DENIED  Type: REG_SZ, Length: 520, Data: C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32
SUCCESS
SUCCESS
```

Il valore

C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll indica il percorso del file msgina32.dll, usato come GINADLL per l'interfaccia di autenticazione in Windows.

Questo indica che è stata specificata un'interfaccia di autenticazione personalizzata (GINA) utilizzando il file che si trova nel percorso specificato.

ATTIVITÀ FILE SYSTEM

Passando successivamente alla visualizzazione su Procmon, abbiamo filtrato i processi per visualizzare solo quelli relativi al file system. Analizzando questi ultimi, abbiamo dedotto che la chiamata di sistema che ha modificato il contenuto della cartella dove è presente l'eseguibile è "**CreateFile**" ed è proprio questa che ha creato il file msgina32.dll.

Time ...	Process Name	PID	Operation	Path	Result
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	ReadFile	C:\Windows\System32\wow64win.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryBasicInfor...	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll	FILE LOCKED WITH ONLY READERS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFileMapp...	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Windows\SysWOW64\sechost.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CreateFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	WriteFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	CloseFile	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\msgina32.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryNameInfo...	C:\Windows\System32\apisetschemas.dll	SUCCESS
13:36...	Malware_Build_Week_U3.exe	1900	QueryNameInfo...	C:\Users\user\Desktop\MALWARE\Build_Week_Unit_3\Malware_Build_Week_U3.exe	SUCCESS

CONCLUSIONI

Funzionamento del malware:

Grazie all'analisi effettuata con **ProcMonitor** e le analisi dei giorni precedenti, possiamo intuire che il malware vada a creare e sostituire **msgina32.dll**, un file lecito di Windows che permette l'autenticazione degli utenti tramite interfaccia grafica.

TRACCIA GIORNO 5

GINA (Graphical identification and authentication) è un componente legato 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 che usate anche voi per accedere alla macchina virtuale.

- Cosa può succedere se il file .dll legato 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

(GRAPHICAL IDENTIFICATION AND AUTHENTICATION)

La libreria GINA.dll (Graphical Identification and Authentication) è una componente di Windows che gestisce il processo di autenticazione e identificazione degli utenti durante l'accesso al sistema operativo. Utilizzata principalmente nelle versioni di Windows precedenti a Windows Vista, come Windows XP e Windows 2000, GINA.dll è responsabile della schermata di login che gli utenti vedono quando accedono al loro computer.

GINA.dll permette agli sviluppatori di personalizzare la procedura di accesso, inclusi elementi come la schermata di autenticazione e i prompt per la password. Quando un utente tenta di accedere, GINA.dll gestisce l'input delle credenziali, come nome utente e password, e le invia al sistema per la verifica. Se le credenziali sono corrette, GINA.dll comunica con il sistema operativo per avviare una sessione utente, caricare il profilo dell'utente e configurare l'ambiente di lavoro. In caso contrario, visualizza un messaggio di errore e richiede all'utente di reinserire le credenziali.

Oltre alla gestione delle credenziali standard, GINA.dll supporta anche modalità di autenticazione alternative, come l'uso di smart card, sistemi biometrici o autenticazione basata su rete, rendendola molto versatile per personalizzazioni di sicurezza specifiche.

Con l'introduzione di Windows Vista, Microsoft ha sostituito GINA.dll con un nuovo sistema chiamato Credential Provider. Questo nuovo sistema offre maggiore flessibilità, sicurezza e interoperabilità con i moderni metodi di autenticazione, rendendo GINA.dll obsoleta. In sostanza, mentre GINA.dll era essenziale per la gestione del login e dell'autenticazione in Windows fino a XP, è stata rimpiazzata da un modello più avanzato e sicuro nelle versioni successive del sistema operativo.

CONSEGUENZE

Conseguenze dell'esecuzione del malware

Possibili conseguenze della sostituzione del file GINA lecito con un file .dll malevolo:

- **Intercettazione delle credenziali:**

- Il componente GINA di Windows è responsabile della gestione dell'autenticazione degli utenti tramite un'interfaccia grafica.
- Questo componente permette agli utenti di inserire username e password nel riquadro di login.
- Se il file GINA lecito viene sostituito con un file .dll malevolo, il malware può intercettare tutte le credenziali inserite dagli utenti.

- **Accesso non autorizzato:**

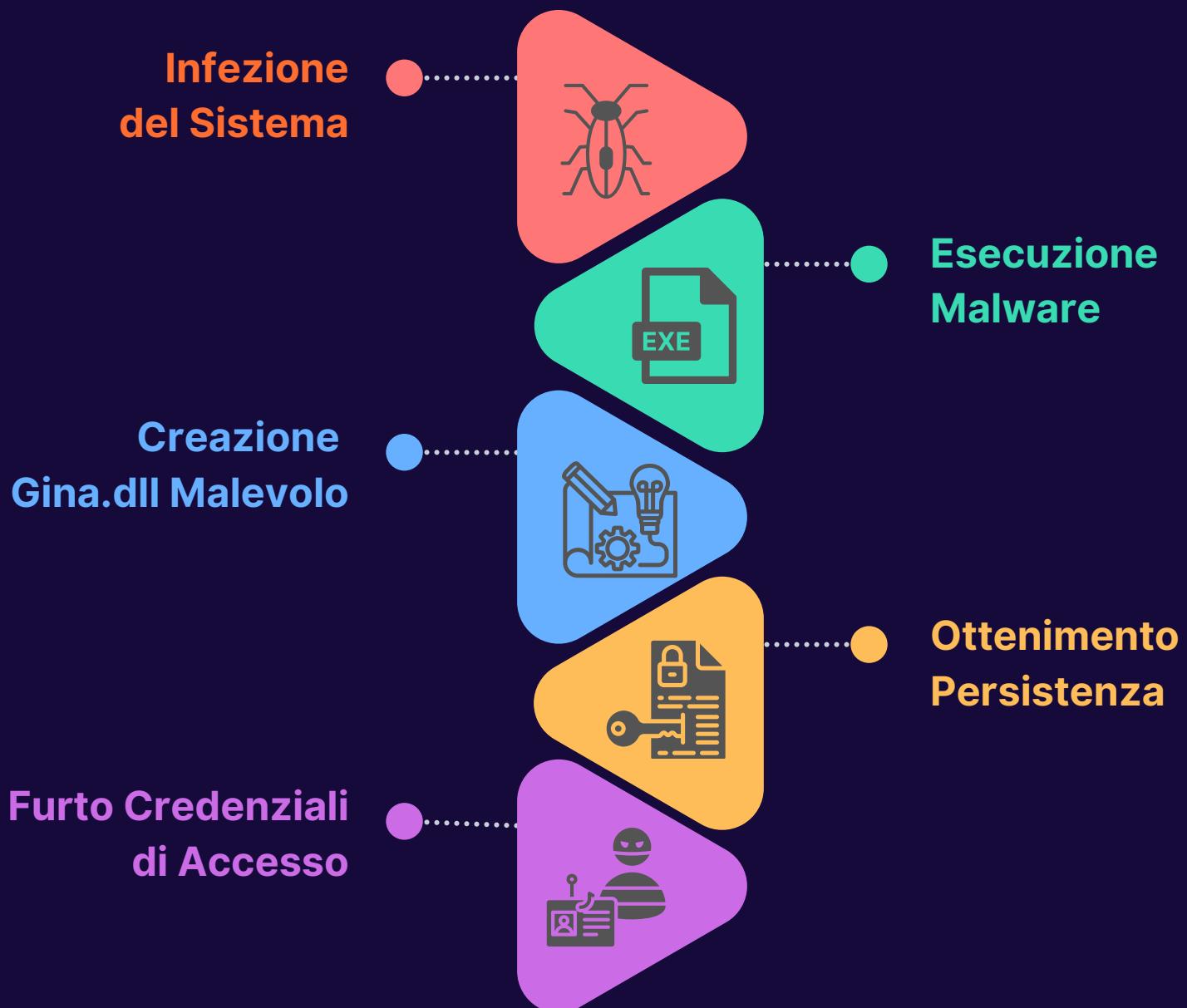
- Le credenziali intercattate possono essere utilizzate dagli attaccanti per ottenere accesso non autorizzato al sistema.
- Questo accesso può includere privilegi elevati, permettendo agli attaccanti di eseguire ulteriori attacchi o modificare configurazioni critiche del sistema.

- **Compromissione della sicurezza:**

- Il malware potrebbe sfruttare le credenziali rubate per installare ulteriori software malevoli o eseguire comandi dannosi.
- Questo comprometterebbe ulteriormente la sicurezza della macchina virtuale e dei dati in essa contenuti.

In sintesi, la sostituzione del file GINA lecito con una versione malevola rappresenta una grave minaccia alla sicurezza, poiché permette agli attaccanti di intercettare informazioni sensibili e ottenere accesso non autorizzato al sistema.

Diagramma di Flusso



DESCRIZIONE DIAGRAMMA

Analisi dei punti del diagramma:

- **Infezione del Sistema:** Il primo passo di un attacco malware è l'infezione del sistema target. Questo può avvenire attraverso vari vettori come email di phishing, download da siti web infetti, o sfruttando vulnerabilità del sistema operativo o di applicazioni installate.
- **Esecuzione Malware:** Una volta che il malware è stato introdotto nel sistema, viene eseguito. Il malware può essere eseguito automaticamente attraverso script o programmi che si attivano all'avvio del sistema, o manualmente dall'utente che clicca su un file eseguibile apparentemente innocuo.
- **Creazione Gina.dll Malevolo:** Il malware crea e tenta di sostituire il file GINA.dll con una versione malevola. GINA (Graphical Identification and Authentication) è un componente di Windows che gestisce l'autenticazione degli utenti tramite l'interfaccia grafica. La versione malevola di GINA.dll intercetta le credenziali di accesso degli utenti, permettendo agli attaccanti di ottenere username e password.
- **Ottenimento Persistenza:** Il malware implementa meccanismi per mantenere l'accesso al sistema compromesso. Questo può includere la modifica delle chiavi di registro per eseguire il malware all'avvio del sistema: in questo caso viene creata la chiave di registro "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon".
- **Furto Credenziali di Accesso:** Infine, il malware utilizza il file GINA.dll malevolo per rubare le credenziali di accesso degli utenti. Quando gli utenti tentano di accedere al sistema, le loro credenziali vengono intercettate e inviate agli attaccanti, permettendo loro di ottenere accesso non autorizzato al sistema e ad altre risorse protette.

CONCLUSIONI

L'analisi dettagliata del malware presente nella cartella Build_Week_Unit_3 ha rivelato diverse informazioni critiche riguardo al suo funzionamento e alle sue intenzioni dannose. Inizialmente, attraverso l'esame del file eseguibile Malware_Build_Week_U3, sono stati identificati i parametri e le variabili utilizzate dalla funzione Main(), insieme alle principali sezioni del file e alle librerie importate. Questo ha permesso di formulare ipotesi preliminari sulle potenziali funzionalità del malware.

L'approfondimento successivo ha coinvolto l'analisi della funzione alla locazione 00401021, rivelando lo scopo e il passaggio dei parametri, oltre a esaminare le istruzioni tra 00401027 e 00401029 e tradurle in costrutti C. La valutazione del parametro "ValueName" alla locazione 00401047 ha fornito ulteriori dettagli sul comportamento specifico del malware.

L'analisi delle routine tra 00401080 e 00401128 ha permesso di determinare il valore del parametro "ResourceName" e di comprendere le funzionalità implementate dal malware. Eseguendo il malware in un ambiente controllato con Process Monitor, sono state osservate le modifiche al sistema, identificando le attività sul registro di Windows e sul file system, inclusa la creazione di chiavi di registro e la modifica delle cartelle.

Infine, l'esplorazione dell'uso di GINA (Graphical Identification and Authentication) ha evidenziato l'impatto potenziale della sostituzione di un file dll lecito con uno malevolo, permettendo al malware di intercettare credenziali di accesso e ottenere persistenza nel sistema.

In conclusione, questa analisi completa ha delineato il comportamento del malware, mostrando come esso utilizzi vari meccanismi per infettare il sistema, eseguire codice malevolo, ottenere persistenza e rubare informazioni sensibili. Le informazioni ottenute forniscono una solida base per sviluppare strategie di difesa e mitigazione contro tali minacce, migliorando la sicurezza complessiva del sistema.

Team 1



Samuele
Aversa



Andrea
Di Benedetto



Lorenzo
Franchi



Federico
Biggi



Mario
Reitano
