

Build Week III – Malware Analysis

Tech-Team

**Andrea Dura Stigliano, Corrado Francesco Pio Li Quadri, Daniele Berardi,
Elena Kovalenko, Lisa Bonato, Luca Manna, Simone Greco**

INDICE

- **Introduzione del Malware**
- **Hybrid Analysis**
- **Strategic plan**
- **Giorno 1**
 - 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?
 - Quali librerie importa il Malware
- **Giorno 2**
 - Qual è lo scopo della funzione RegCreateKeyExA
 - Come vengono passati i parametri alla locazione indicata
 - Qual è l'oggetto rappresentato alla locazione indicata
 - Qual è significato delle istruzioni
 - Traduzione codice assembly
 - Qual è il valore del parametro "ValueName"
 - Funzionalità implementata dal Malware
- **Giorno 3**
 - Qual è il valore del parametro «ResourceName»
 - Funzionalità implementate dal malware
 - Identificazione della Funzionalità mediante analisi statica basica
 - Evidenze a supporto dell'identificazione
- **Giorno 4**
 - Cosa c'è all'interno della cartella dove è situato l'eseguibile del Malware?
 - Cosa è avvenuto?
 - Quale chiave di registro viene creata?
 - Quale valore viene associato alla chiave di registro creata?
 - Quale chiamata di sistema ha modificato il contenuto della cartella dove è presente l'eseguibile del Malware?
- **Giorno 5**
 - Cosa può succedere se il file .dll lecito viene sostituito con un file.dll malevolo, che intercetta i dati inseriti?
- **Grafico del funzionamento del Malware**
- **Conclusione**

Introduzione

Nella settimana della Build Week III ci occuperemo di analisi malware.

Durante questa settimana, il nostro team di esperti in sicurezza informatica affronterà una serie di tracce per analizzare in profondità il malware presente nel file eseguibile "Malware_Build_Week_U3".

Utilizzeremo una combinazione di strumenti e tecniche avanzate apprese durante le nostre lezioni teoriche per comprendere appieno il funzionamento di questo malware e per identificare le sue caratteristiche, le sue vulnerabilità e le sue potenziali minacce.

Il nostro lavoro sarà diviso in cinque giornate e tante task da risolvere:

Il primo giorno andremo a identificare il numero di parametri passati alla funzione main e il numero di variabili dichiarate all'interno di essa. Esaminare le sezioni presenti nel file eseguibile e le librerie importate dal malware.

Il secondo giorno dobbiamo comprendere lo scopo della funzione RegCreateKeyExA e analizzare come vengono passati i parametri ad essa.

Identificare l'oggetto rappresentato alla locazione indicata nel codice e tradurre le istruzioni in assembly.

Esaminare il valore del parametro "ValueName" e dedurre le funzionalità implementate dal malware.

Il terzo giorno andremo ad analizzare il valore del parametro "ResourceName" e identificare ulteriori funzionalità implementate dal malware. Utilizzare l'analisi statica di base per identificare le funzionalità del malware e cercare evidenze a supporto di tali identificazioni.

Il quarto giorno bisogna esaminare il contenuto della cartella dove è situato l'eseguibile del malware e identificare eventuali cambiamenti. Individuare la chiave di registro creata dal malware e il valore associato ad essa. Identificare la chiamata di sistema che ha modificato il contenuto della cartella dove è presente l'eseguibile del malware.

Il quinto ed ultimo giorno andremo a valutare le potenziali conseguenze dell'inserimento di un file .dll malevolo al posto di un file .dll lecito. Trarre conclusioni basate sull'analisi condotta durante la settimana e proporre eventuali raccomandazioni per mitigare i rischi associati al malware. Ogni giorno affronteremo una serie di domande specifiche per approfondire la nostra comprensione del malware e delle sue funzionalità, utilizzando una combinazione di analisi statiche, dinamiche e di reverse engineering. Alla fine della settimana, saremo in grado di fornire un'analisi completa del malware e delle relative minacce, insieme a raccomandazioni per la sicurezza informatica.

Prima di iniziare però andiamo ad effettuare un'analisi preliminare del Malware utilizzando una tecnica che viene chiamata Hybrid Analysis.



Ancor prima di testare con i tool appropriati il malware, il team di cybersecurity ha condotto un'analisi statica di base sul file eseguibile "Malware_Build_Week_U3" utilizzando Hybrid Analysis. L'obiettivo di questa analisi era comprendere il funzionamento e il comportamento del malware, nonché identificare la natura del malware.

L'**hybrid analysis** è una tecnica di analisi del malware che combina sia l'analisi statica che dinamica per ottenere una visione completa del comportamento del malware. In sostanza, durante un'analisi ibrida, il file sospetto viene esaminato sia in un ambiente statico (senza eseguirlo) che in un ambiente dinamico (eseguendo il file in un sandbox o in un ambiente controllato).

Analisi Statica: Durante l'analisi statica, vengono esaminati i metadati del file, come le firme digitali, i metodi di compressione, le sezioni del file e altro ancora, senza eseguire effettivamente il codice. Questo offre informazioni sulla struttura e le caratteristiche del file, ma non fornisce dettagli sul suo comportamento reale.

Analisi Dinamica: Durante l'analisi dinamica, il file sospetto viene eseguito in un ambiente controllato, come un sandbox, per monitorare il suo comportamento in tempo reale. Ciò può includere osservare le chiamate di sistema, l'interazione con i file di sistema, la comunicazione di rete e altro ancora. Questo fornisce informazioni preziose sulle azioni effettive del malware.

L'hybrid analysis combina questi due approcci per ottenere una comprensione più completa del malware, identificando le sue capacità, i suoi obiettivi e le sue modalità di azione. Questo approccio consente ai ricercatori di sicurezza di raccogliere informazioni approfondite sulle minacce informatiche e di sviluppare strategie efficaci di mitigazione e risposta agli attacchi.

The screenshot displays the Hybrid Analysis web interface. At the top, there's a navigation bar with the Hybrid Analysis logo and a menu icon. Below this, the page title is "Analysis Overview". On the right side, there's a red button labeled "malicious" and a "Request Report Deletion" link. The main content area shows submission details for "Lab11-01.exe".

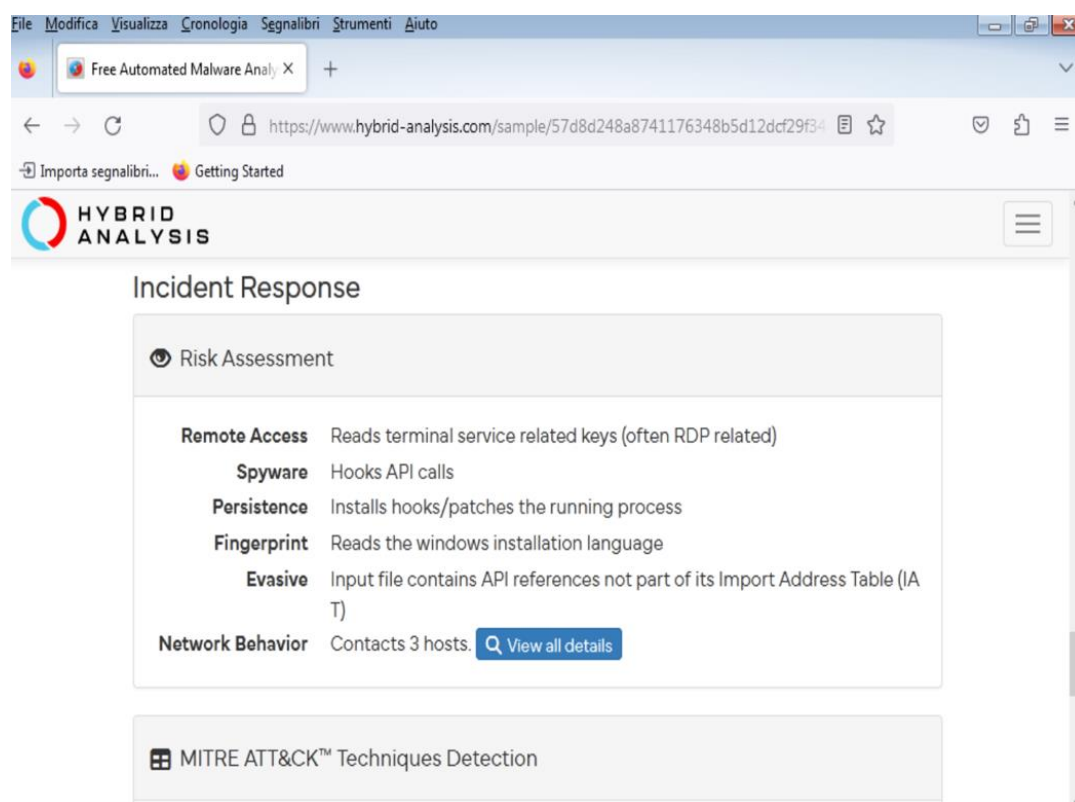
Submission	Lab11-01.exe
name:	Lab11-01.exe
Size:	52KiB
Type:	peexe executable
Mime:	application/x-dosexec
SHA256:	57d8d248a8741176348b5d12dcf29f34c8f48ede0ca13c30d12e5ba0384056d7
Operating System:	Windows
Last Anti-Virus Scan:	02/06/2024 13:40:54 (UTC)
Last Sandbox Report:	02/06/2024 13:40:51 (UTC)

On the right side of the submission details, there's a "Threat Score: 100/100" and "AV Detection: 88%". It also says "Labeled as: Malware" and includes tags "#lab11-01" and "#tag". There are social media sharing buttons for Link, Twitter, and E-Mail.

At the bottom of the page, there's a section titled "Anti-Virus Results" with a red button labeled "Refresh Required".

Risultati dell'Analisi Statica:

Incident Response: Il malware sembra essere in grado di rispondere agli eventi dell'incidente, indicando una potenziale capacità di reazione agli attacchi o ai comandi remoti.



Risk Assessment: Il malware è classificato come pericoloso con un punteggio di minaccia di 100/100 e una rilevazione antivirus dell'88%. È stato etichettato come malware e potenzialmente dannoso per il sistema.

Remote Access: Il malware legge le chiavi correlate al servizio Terminal Service, suggerendo una possibile attività di accesso remoto, spesso legata a protocolli di desktop remoto (RDP).

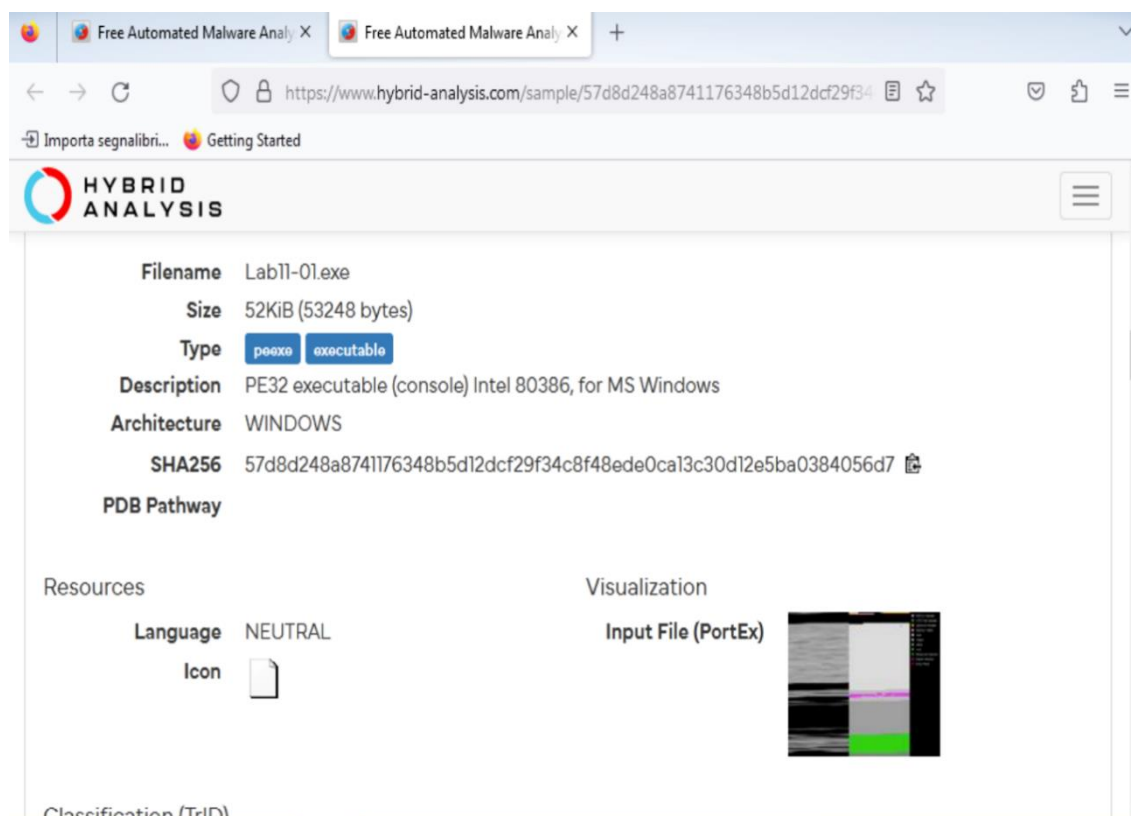
Spyware: Il malware ha funzionalità di intercettazione delle chiamate API, indicando un comportamento da spyware, con l'obiettivo di monitorare e raccogliere informazioni sull'attività del sistema.

Persistence: Il malware installa hook o patcha i processi in esecuzione, suggerendo una capacità di persistenza nel sistema, con il rischio di reinfezione dopo il riavvio.

Fingerprint: Il malware legge la lingua di installazione di Windows, indicando una capacità di ottenere informazioni sul sistema ospite.

Evasive: Il malware contiene riferimenti API che non fanno parte della sua Import Address Table (IAT), suggerendo un comportamento evasivo per eludere la rilevazione e l'analisi statica.

Network Behavior: Il malware contatta tre host esterni, indicando una potenziale attività di comunicazione con server di comando e controllo o altri sistemi remoti.

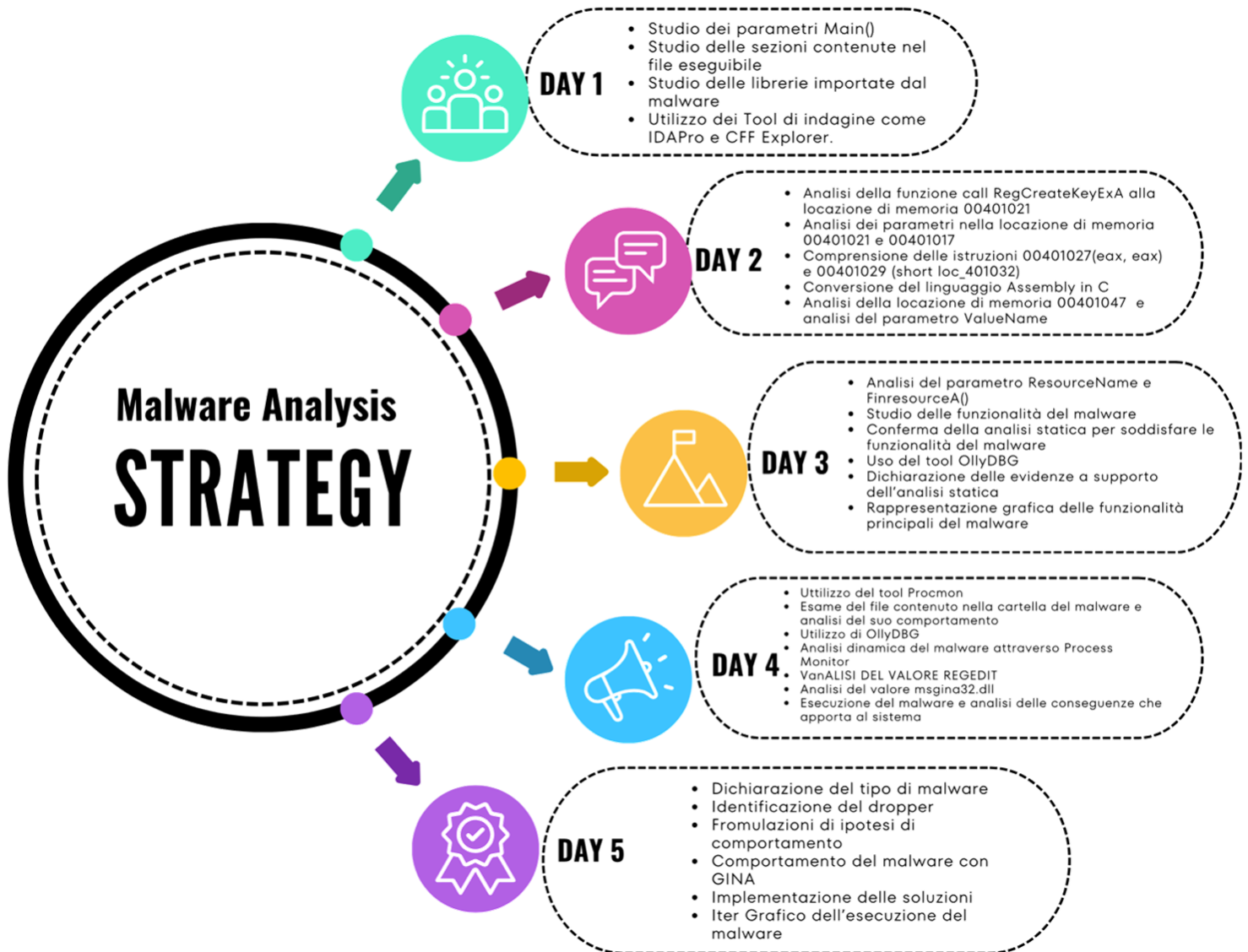


Dettagli del File Eseguiibile:

- Nome del File: Lab11-01.exe
- Dimensioni: 52KiB (53248 byte)
- Tipo di File: Eseguiibile PE32 (console) Intel 80386, per MS Windows
- Architettura: WINDOWS
- SHA256:
57d8d248a8741176348b5d12dcf29f34c8f48ede0ca13c30d12e5ba0384056d7
- Sezioni del File:
 - .text: Contiene codice eseguibile con un'entropia di 6.225.

L'analisi statica ha fornito una panoramica dettagliata delle funzionalità e del comportamento del malware. Sulla base dei risultati ottenuti, il malware sembra essere progettato per scopi dannosi, inclusi l'accesso remoto, il monitoraggio dell'attività del sistema e la persistenza nel sistema ospite. È essenziale adottare misure di mitigazione appropriate per proteggere i sistemi e reagire efficacemente agli attacchi del malware individuato. Ulteriori analisi dinamiche e investigazioni possono essere necessarie per comprendere appieno le capacità e le intenzioni del malware e sviluppare una strategia di risposta completa.

STRATEGIC PLAN



Build Week III – Malware Analysis

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:

1. Quanti parametri sono passati alla funzione Main()?
2. Quante variabili sono dichiarate all'interno della funzione Main()?
3. Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate
4. 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.

Seguendo la traccia andiamo ad avviare la nostra macchina Windows XP/7 per esaminare il malware.

Per risolvere la prima richiesta è stato necessario ottenere il codice Assembly del malware. Questo processo è stato realizzato mediante l'utilizzo di un disassembler, un software in grado di tradurre le istruzioni del linguaggio macchina dell'eseguibile nel linguaggio Assembly corrispondente.

Nel nostro caso, abbiamo utilizzato il tool **IDA Pro**, per ottenere il codice Assembly del malware. Questo strumento è ampiamente adottato dagli analisti di sicurezza e offre una traduzione completa delle istruzioni del linguaggio macchina in linguaggio Assembly. Oltre a ciò, IDA Pro fornisce funzionalità avanzate, come la rilevazione di funzioni e la visualizzazione chiara del flusso di controllo, semplificando notevolmente l'analisi del malware.

IDA Pro è un disassembler ampiamente diffuso e apprezzato nella comunità degli analisti di sicurezza. Esso supporta diversi formati di file eseguibili e offre agli analisti una serie di funzionalità intuitive per semplificare le attività di analisi. Oltre alla traduzione completa del linguaggio macchina in linguaggio Assembly, IDA Pro è in grado di identificare e fornire informazioni aggiuntive.

Questo include la rilevazione di funzioni, la visualizzazione di flussi di controllo, e l'organizzazione del codice in modo da renderlo più comprensibile per gli analisti.

Una volta avviato il tool siamo andati ad aprire il nostro malware cliccando l'icona della cartella e seguendo il path dell'eseguibile per poi cliccare su "Open".



Quando apriamo il malware con IDA Pro, l'interfaccia di analisi ci presenta il "Disassembly Panel" in modalità grafica nel pannello centrale. Per esaminare il codice Assembly dell'eseguibile, passiamo dalla visualizzazione grafica a quella testuale cliccando sulla barra spaziatrice. In questa modalità, IDA Pro mostra il codice Assembly, permettendoci di identificare facilmente le funzioni, evidenziate nel riquadro come la funzione **Main()**, le variabili locali e i parametri. Questo ci consente di esaminare in dettaglio la struttura del malware e comprendere meglio il suo funzionamento.

```
.text:00401100 ; ===== S U B R O U T I N E =====
.text:00401100
.text:00401100 ; Attributes: bp-based frame
.text:00401100
.text:00401100 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401100 _main          proc near          ; CODE XREF: start+AF↓p
.text:00401100
.text:00401100 hModule      = dword ptr -11Ch
.text:00401100 Data        = byte ptr -118h
.text:00401100 var_117     = byte ptr -117h
.text:00401100 var_8       = dword ptr -8
.text:00401100 var_4       = dword ptr -4
.text:00401100 argc        = dword ptr  8
.text:00401100 argv        = dword ptr  0Ch
.text:00401100 envp        = dword ptr 10h
.text:00401100
```


1. Quanti parametri sono passati alla funzione Main()?

I parametri che troviamo passati alla funzione Main() sono **"argc"**, **"argv"** e **"envp"**

```
.text:004011D0 argc      = dword ptr  8
.text:004011D0 argv      = dword ptr  0Ch
.text:004011D0 envp      = dword ptr  10h
```

"argc" (argument count) è un parametro di tipo intero, rappresenta il numero degli argomenti effettivamente passati al programma nella linee di comando con cui si invoca la sua esecuzione.

"argv" (argument vector) contiene informazioni sugli argomenti passati a un programma. Ogni elemento di questo array è un puntatore a una stringa di caratteri terminata da un null e rappresenta un argomento specifico fornito attraverso la linea di comando al programma.

"envp" (environment variable pointer) è un array di puntatori di caratteri che contiene informazioni sulle variabili di ambiente passate a un programma. Ogni elemento di questo array è un puntatore a una stringa di caratteri con terminazione null e segue il formato "nome=valore", rappresentando una specifica variabile di ambiente.

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

All'interno della funzione troviamo 5 variabili tra cui:

```
.text:004011D0 hModule    = dword ptr -11Ch
.text:004011D0 Data      = byte ptr  -118h
.text:004011D0 var_117    = byte ptr  -117h
.text:004011D0 var_8      = dword ptr  -8
.text:004011D0 var_4      = dword ptr  -4
```

"hModule" è una variabile utilizzata in Windows per rappresentare un identificatore (handle) associato a un modulo. Un modulo è un file che contiene codice o dati utilizzati da un programma, l'handle fornisce un modo per fare riferimento e accedere alle risorse o al codice specifico contenuto in quel modulo.

"Data" è una variabile flessibile adatta per memorizzare dati di qualsiasi tipo senza specificare un formato particolare.

"var_117" potrebbe rappresentare un'area di memoria che viene utilizzata per memorizzare dati temporanei o risultati intermedi durante l'esecuzione di un programma.

"Var_8" è una variabile che può contenere diversi tipi di dati in un programma.

"Var_4" è una variabile che può contenere diversi tipi di dati in un programma.

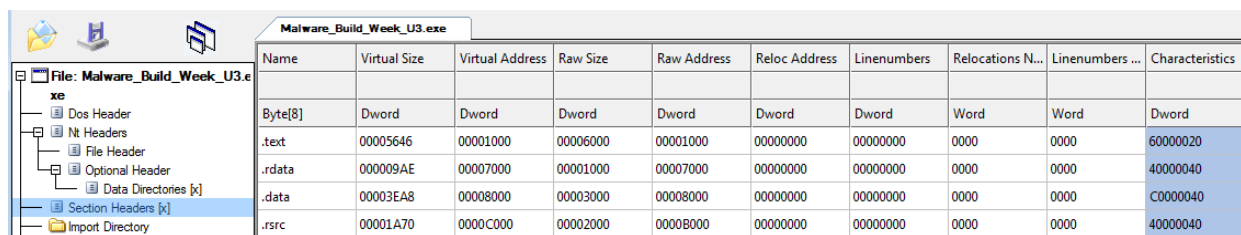
3. Quali sezioni sono presenti all'interno del file eseguibile? Descrivete brevemente almeno 2 di quelle identificate.

Per risolvere questa richiesta andremo ad utilizzare il tool **CFF Explorer**, che ci permetterà di identificare le librerie e le funzioni, che sono fondamentali per capire l'intento del malware.

CFF Explorer è uno strumento software utilizzato per analizzare e modificare i file eseguibili di Windows, come .DLL e .EXE. Fornisce una vasta gamma di funzionalità, tra cui la visualizzazione delle intestazioni PE, l'analisi delle funzioni esportate e importate, l'esplorazione delle risorse, la modifica dei manifesti e il disassemblaggio del codice assembly. È ampiamente utilizzato da sviluppatori, analisti di sicurezza e appassionati di reverse engineering per esaminare e manipolare i file binari.

Una volta avviato il tool siamo andati ad aprire il nostro malware cliccando l'icona della cartella e seguendo il path dell'eseguibile per poi cliccare su "Open".

Ci spostiamo nella "Section Headers[x]", per vedere tutte le informazioni sulle sezioni di cui viene composto il file eseguibile.



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

Troviamo 4 sezioni all'interno del file:

.text: la sezione «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: la sezione «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: la sezione «data» contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Ricordate che 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 dell'eseguibile.

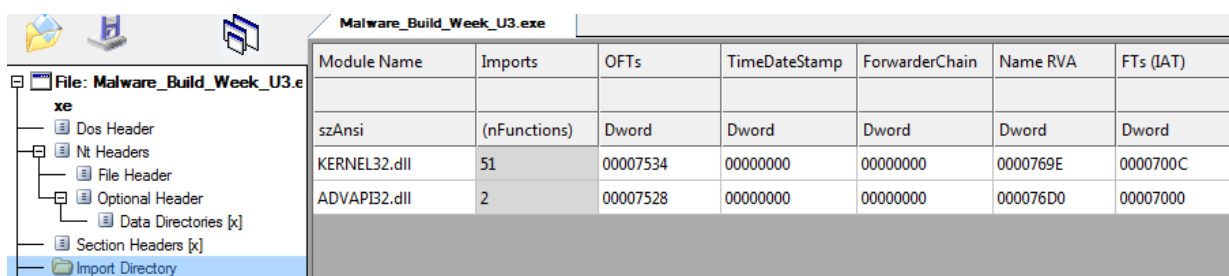
.rsrc: la sezione «rsrc» include le risorse utilizzate dall'eseguibile come ad esempio icone, immagini, menu e stringhe che non sono parte dell'eseguibile stesso.

4. 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.

Ci spostiamo nella “**Import Directory**”, per vedere tutte le informazioni sulle librerie importate dal file eseguibile.



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

KERNEL32.dll: è una delle principali librerie di sistema di Windows utilizzata per effettuare operazioni di base del sistema operativo.

Questa libreria offre funzioni di basso livello, come ad esempio la gestione della memoria, operazioni di input/output, manipolazione dei file, creazione e gestione di processi e thread e molte altre.

Advapi32.dll: La libreria advapi32.dll in ambienti Windows fornisce un'API essenziale per la gestione della sicurezza, account utente, crittografia, registrazione di sistema e servizi Windows.

Le sue funzioni supportano operazioni come l'autenticazione degli utenti, la crittografia dei dati, la manipolazione del Registro di sistema e la gestione dei servizi, rendendola cruciale per lo sviluppo di applicazioni Windows con requisiti di sicurezza avanzati.

Prima ancora di analizzare le funzioni chiariamo anche il concetto di Windows Api chiarendo la loro funzione e la loro natura.

La **Windows API** fornisce funzionalità native per interagire con componenti chiave del sistema operativo Windows. L'API è ampiamente utilizzata da molti, inclusi red teamers, threat actors, blue teamers, sviluppatori di software e fornitori di soluzioni.

L'API può integrarsi senza soluzione di continuità con il sistema Windows, offrendo una vasta gamma di casi d'uso.

È possibile vedere l'API Win32 utilizzata per lo sviluppo di strumenti offensivi e malware, l'ingegneria di EDR (Endpoint Detection & Response) e applicazioni software in generale.

Ecco alcuni esempi di Windows Api più comunemente utilizzati

Livello	Spiegazione
API	Un termine o teoria di alto livello usato per descrivere qualsiasi chiamata trovata nella struttura dell'API Win32.
File di intestazione o importazioni	Definisce le librerie da importare a tempo di esecuzione, definite da file di intestazione o importazioni di librerie. Utilizza puntatori per ottenere l'indirizzo della funzione.
DLL Core	Un gruppo di quattro DLL che definiscono le strutture delle chiamate. (KERNEL32, USER32 e ADVAPI32). Queste DLL definiscono servizi del kernel e dell'utente che non sono contenuti in un singolo sottosistema.
DLL Supplementari	Altre DLL definite come parte dell'API di Windows. Controlla sottosistemi separati del sistema operativo Windows. Circa altre 36 DLL definite. (NTDLL, COM, FVEAPI, ecc.)
Call Structures	Definisce la chiamata API stessa e i parametri della chiamata.
API calls	La chiamata API utilizzata all'interno di un programma, con gli indirizzi delle funzioni ottenuti dai puntatori.
Parametri di Input/Output	I valori dei parametri definiti dalle strutture di chiamata.

Dunque ora analizziamo i seguenti valori delle funzioni trovate all'interno del malware:

Malware_Build_Week_U3.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
000076D0	N/A	00007500	00007504	00007508	0000750C	00007510
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000
OFTs	FTs (IAT)	Hint	Name			
Dword	Dword	Word	szAnsi			
000076AC	000076AC	0186	RegSetValueExA			
000076BE	000076BE	015F	RegCreateKeyExA			

RegSetValueExA:

Questa funzione è utilizzata in Windows per impostare il valore di un'entry all'interno del Registro di sistema.

Implica che il malware sta cercando di modificare o aggiungere un valore all'interno del Registro di sistema per poter persistere nel sistema o modificare le configurazioni.

Potrebbe indicare che il malware sta cercando di modificare le impostazioni di avvio o di configurazione del sistema per eseguire operazioni dannose o ottenere accesso privilegiato.

RegCreateKeyExA:

Questa funzione è utilizzata per creare una nuova chiave o aprire una chiave esistente nel Registro di sistema.

Indica che il malware sta cercando di creare o aprire una nuova chiave nel registro di sistema per poter archiviare informazioni o configurazioni.

Potrebbe implicare che il malware stia cercando di stabilire una persistenza nel sistema o di modificare le impostazioni del sistema.

TimeStamp:

Questo valore indica la data e l'ora in cui il file eseguibile è stato compilato.

Implica semplicemente la data e l'ora in cui il file eseguibile è stato creato o compilato.

ForwarderChain:

Questo valore è utilizzato nei file di librerie dinamiche (DLL) di Windows e indica la catena di inoltro delle funzioni importate.

Indica la struttura della catena di inoltro delle funzioni all'interno della DLL.

000074F0 e 000074F4:

Questi sono valori di indirizzo in formato esadecimale all'interno del file eseguibile.

Potrebbero essere puntatori a dati o a codice all'interno del malware.

Dword:

Questo è un tipo di dati utilizzato nei sistemi Windows, rappresenta un valore a 32 bit (double word). Implica che i valori associati sono numeri a 32 bit.

Malware_Build_Week_U3.exe						
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
0000769E	N/A	000074EC	000074F0	000074F4	000074F8	000074FC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	51	00007534	00000000	00000000	0000769E	0000700C
ADVAPI32.dll	2	00007528	00000000	00000000	000076D0	00007000
OFTs	FTs (IAT)	Hint	Name			
Dword	Dword	Word	szAnsi			
00007632	00007632	0295	SizeofResource			
00007644	00007644	01D5	LockResource			
00007654	00007654	01C7	LoadResource			
00007622	00007622	028B	VirtualAlloc			
00007674	00007674	0124	GetModuleFileNameA			
0000768A	0000768A	0126	GetModuleHandleA			
00007612	00007612	00B6	FreeResource			
000078E6	000078E6	0034	CreateFileA			
000078F4	000078F4	00BF	GetCPInfo			
00007900	00007900	00B9	GetACP			
0000790A	0000790A	0131	GetOEMCP			
00007916	00007916	013E	GetProcAddress			
00007928	00007928	01C2	LoadLibraryA			

I successivi elenchi di funzioni API di Windows indicano le funzioni importate dal malware, le quali potrebbero essere utilizzate per vari scopi.

SizeofResource: Questa funzione restituisce la dimensione, in byte, di una risorsa all'interno di un file eseguibile o di una DLL.

LockResource: Utilizzata per ottenere un puntatore alla memoria che contiene i dati di una risorsa bloccata.

LoadResource: Carica una risorsa specificata all'interno di un modulo (file eseguibile o DLL).

Il comportamento del malware che utilizza queste funzioni indica che potrebbe essere coinvolto nel caricamento e nell'utilizzo di risorse all'interno del sistema, il che potrebbe essere parte di operazioni di caricamento di dati, asset, o di manipolazione delle risorse di sistema per scopi malevoli come l'esecuzione di codice dannoso o l'installazione di backdoor.

CreateFileA è utilizzato per creare o aprire file, **LoadLibraryA** è utilizzato per caricare una DLL, **GetProcAddress** permette a un programma di ottenere l'indirizzo di una specifica funzione da una DLL.

Il comportamento del malware in questo caso suggerisce che sta interagendo con il registro di sistema e utilizzando funzioni di sistema per ottenere informazioni o modificare le configurazioni del sistema. Potrebbe essere coinvolto in operazioni di persistenza, modifica delle impostazioni o esecuzione di azioni dannose.

L'analisi dei dati e delle funzioni trovate all'interno del malware suggerisce un comportamento tipico di un malware che cerca di ottenere accesso e controllo sul sistema compromesso attraverso la manipolazione del Registro di sistema e l'utilizzo di risorse di sistema.

Build Week III – Malware Analysis

Giorno 2:

Con riferimento al malware in analisi, spiegare:

1. Lo scopo della funzione chiamata alla locazione di memoria **00401021**
2. Come vengono passati i parametri alla funzione alla locazione **00401021**
3. Che oggetto rappresenta il parametro alla locazione **00401017**
4. Il significato delle istruzioni comprese tra gli indirizzi **00401027 e 00401029**
5. Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costruito C.
6. Valuta ora la chiamata alla locazione **00401047**, qual è il valore del parametro "ValueName"?

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

Per il raggiungimento della risoluzione del task 2 chiariamo anche i concetti che vi sono dietro i funzionamenti di un malware.

Il linguaggio Assembly è il livello più basso di linguaggio leggibile dall'uomo. È anche il livello più alto di linguaggio in cui un binario può essere decompilato in modo affidabile. Nel contesto dell'apprendimento dell'ingegneria inversa del malware, conoscere le basi del linguaggio Assembly è essenziale. Questo perché, quando otteniamo un campione di malware da analizzare, è molto probabile che sia un binario compilato. Non possiamo visualizzare il codice in **linguaggio C/C++** o in altri linguaggi perché non è disponibile per noi. Quello che possiamo fare, tuttavia, è decompilare il codice utilizzando un decompiler o un disassembler.

Risultati del file Malware Build_Week_Unit_3:

```
.text:00401000 ; ===== S U B R O U T I N E =====
.text:00401000
.text:00401000 ; Attributes: bp-based frame
.text:00401000
.text:00401000 ; int __cdecl sub_401000(BYTE *lpData, DWORD cbData)
.text:00401000 sub_401000      proc near                ; CODE XREF: _main+08↓p
.text:00401000
.text:00401000 hObject        = dword ptr -4
.text:00401000 lpData         = dword ptr  8
.text:00401000 cbData        = dword ptr 0Ch
.text:00401000
.text:00401000      push     ebp
.text:00401001      mov      ebp, esp
.text:00401003      push     ecx
.text:00401004      push     0                ; lpdwDisposition
.text:00401006      lea      eax, [ebp+hObject]
.text:00401009      push     eax                ; phkResult
.text:0040100A      push     0                ; lpSecurityAttributes
.text:0040100C      push     0F003Fh           ; samDesired
.text:00401011      push     0                ; dwOptions
.text:00401013      push     0                ; lpClass
.text:00401015      push     0                ; Reserved
.text:00401017      push     offset SubKey     ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
.text:0040101C      push     80000002h         ; hKey
.text:00401021      call     ds:RegCreateKeyExA
.text:00401027      test     eax, eax
.text:00401029      jz       short loc_401032
.text:0040102B      mov      eax, 1
.text:00401030      jmp      short loc_40107B
```

Uno **"stack di memoria"** è una struttura dati utilizzata in programmazione per gestire le chiamate di funzioni e le variabili locali. Funziona secondo il principio Last In, First Out (LIFO), dove l'ultimo elemento inserito è il primo a essere rimosso. Lo stack di memoria può riferirsi allo "stack di chiamate", che memorizza informazioni relative alle chiamate di funzioni, o allo "stack di dati", utilizzato per gestire variabili locali e temporanee. Questa struttura semplifica la gestione della memoria, consentendo un accesso efficiente e ordinato alle informazioni memorizzate.

Passaggi eseguiti:

Come primo passaggio andiamo ad identificare la funzione chiamata alla locazione **00401021** dello stack di memoria.

```
* .text:00401015      push     0                ; Reserved
* .text:00401017      push     offset SubKey     ; "SOFTWARE\\Microsoft\\Win
* .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
```

Come possiamo vedere nel riquadro rosso in figura alla locazione **00401021** troviamo la funzione **RegCreateKeyExA**.

1. Qual è lo scopo della funzione RegCreateKeyExA?

L'invocazione della funzione **RegCreateKeyExA** in questo contesto riveste un ruolo fondamentale nelle attività nefaste del software dannoso. Tale interfaccia di programmazione delle applicazioni di Windows non solo consente la creazione di nuove voci di registro, ma anche l'accesso a voci esistenti con specifici livelli di autorizzazione. Costituisce un mezzo diretto attraverso il quale il software dannoso può stabilire un punto di ancoraggio nel sistema compromesso, permettendo potenzialmente al malware di auto-avviarsi ad ogni riavvio del sistema, modificare il comportamento del sistema stesso o occultare altre sottochiavi e valori maligni. Questo potrebbe altresì essere impiegato per sovrascrivere le impostazioni di sicurezza, di rete o altre configurazioni, agevolando ulteriori azioni dannose del malware o garantendo la sua persistenza e occultamento nell'ambiente compromesso.

2. Come vengono passati i parametri alla locazione indicata?

I parametri vengono passati sullo stack tramite la chiamata push, seguendo il principio LIFO (Last In, First Out). Questo è uno standard comune per le chiamate di funzione in linguaggi come il C e consente di trasferire un numero variabile di parametri in modo ordinato. Inoltre, si ipotizza che il puntatore a una variabile 'phkResult' riceva l'handle della chiave aperta, presumibilmente una chiave di registro predefinita. Tuttavia, è rilevante notare che non viene richiamata la funzione RegCloseKey per liberare l'handle dopo il suo utilizzo. Questa mancanza può portare a perdite di risorse nel sistema e potenziali problemi di sicurezza, specialmente se il malware opera in modo persistente e crea un gran numero di chiavi senza liberarle correttamente.

```
*.text:00401000
*.text:00401001
*.text:00401003
*.text:00401004
*.text:00401006
*.text:00401009
*.text:0040100A
*.text:0040100C
*.text:00401011
*.text:00401013
*.text:00401015
*.text:00401017
*.text:0040101C
*.text:00401021

    push    ebp
    mov     ebp, esp
    push    ecx
    push    0                ; lpdwDisposition
    lea     eax, [ebp+hObject]
    push    eax              ; phkResult
    push    0                ; lpSecurityAttributes
    push    0F003Fh          ; samDesired
    push    0                ; dwOptions
    push    0                ; lpClass
    push    0                ; Reserved
    push    offset SubKey     ; "SOFTWARE\Microsoft\Windows NT\CurrentVe"...
    push    80000002h         ; hKey
    call    ds:RegCreateKeyExA
```

3. Qual è l'oggetto rappresentato alla locazione indicata?

```
.data:00408054 SubKey      db 'SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon',0
.data:00408054                                ; DATA XREF: sub_401000+17To
.data:0040808A                                align 4
```

Il parametro trasferito all'indirizzo 00401017 è un riferimento a una chiave di registro ampiamente conosciuta, utilizzata per avviare automaticamente le applicazioni durante l'avvio del sistema. Il malware sfrutta astutamente questa caratteristica per garantire l'esecuzione del suo codice senza richiedere l'interazione dell'utente, garantendo così la massima persistenza nel sistema. Sebbene questa tecnica sia comunemente impiegata in software legittimi per scopi come gli aggiornamenti automatici, nel contesto del malware rappresenta inequivocabilmente un segno di attività maligna.

4. Qual è significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029?

Questo codice Assembly x86 esegue un controllo di uguaglianza tra il valore contenuto nelle registrazioni EAX e EAX:

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

Queste istruzioni sono parte di un intricato processo di verifica dell'integrità in seguito a un'azione cruciale eseguita dal malware. Mediante un test sui bit del registro EAX, il software dannoso analizza se la precedente chiamata alla funzione **RegCreateKeyExA** ha avuto esito positivo. Questo controllo riveste un'importanza vitale nella logica di gestione del flusso del malware, poiché influenzerà le decisioni successive del programma. Nel caso in cui l'operazione non abbia avuto successo, il malware potrebbe intraprendere un percorso alternativo, tentare di rieseguire l'operazione o persino interrompere il proprio funzionamento per evitare la rilevazione da parte delle misure di sicurezza del sistema. Tale livello di adattabilità e reattività nel comportamento del malware evidenzia la sofisticatezza delle sue strategie per rimanere operativo e persistente nell'ambiente infetto.

L'istruzione "test" esegue una operazione bit a bit "and" tra i due valori e imposta i flag del processore in base al risultato. L'istruzione "jz" (jump if zero) esegue un salto condizionale all'indirizzo specificato "loc_401032" solo se lo Zero Flag è impostato a 1, il che significa che i due valori sono uguali. Quindi, questo codice sta confrontando il valore contenuto nella registrazione EAX con quello contenuto in EAX. Se i due valori sono uguali, il codice esegue un salto all'indirizzo "loc_401032", altrimenti continua eseguendo il codice successivo.

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

La traduzione del codice assembly fornito in un costruito C fornisce un equivalente ad alto livello più comprensibile per i programmatori e gli analisti di malware. Questa rappresentazione mostra come il malware sfrutti le istruzioni condizionali per prendere decisioni basate sul successo o fallimento delle sue operazioni, un concetto cruciale nella programmazione. La capacità del malware di adattarsi dinamicamente in base alle condizioni del sistema è evidenziata attraverso l'uso di costrutti condizionali nel codice C, riflettendo la sua natura reattiva e adattabile nell'ambiente infetto.

```
if (eax == 0) {  
    // Salta a loc_401032 se la chiamata a RegCreateKeyExA() è riuscita.  
} else {  
    eax = 1; // Imposta il valore di eax a 1 se la chiamata non è riuscita.  
}
```

6. Valutazione della chiamata alla locazione 00401047, qual è il valore del parametro "ValueName"?

```
.text:0040103C  
.text:0040103E  
.text:00401043  
.text:00401046  
.text:00401047
```

```
push    0 ; Reserved  
push    offset ValueName ; "GinaDLL"  
mov     eax, [ebp+hObject]  
push    eax ; hKey  
call    ds:RegSetValueExA
```

Nella chiamata alla locazione 00401047, che invoca la funzione RegSetValueExA, il valore del parametro "ValueName" è **"GinaDLL"**. Questo indica che il malware sta impostando o modificando il valore di una chiave di registro che ha a che fare con il processo di autenticazione di Windows (GinaDLL è un riferimento alla DLL di autenticazione grafica utilizzata nelle versioni precedenti di Windows).

Funzionalità implementata dal Malware in questa sezione

Il malware sta eseguendo un attacco mirato e sofisticato all'integrità del sistema operativo Windows. Attraverso l'uso della funzione RegSetValueExA, sta cercando di alterare i punti di estensione del sistema operativo per sovrascrivere la DLL di autenticazione grafica, che è un metodo utilizzato in passato per intercettare le credenziali di login o alterare il processo di autenticazione. Ciò potrebbe consentire al malware di catturare le credenziali di accesso, eseguire codice arbitrario al momento dell'autenticazione dell'utente, o persino bypassare alcuni meccanismi di sicurezza.

La modifica del valore **"GinaDLL"** è particolarmente preoccupante perché si riferisce a un componente che ha a che fare con il Graphical Identification and Authentication DLL (Gina).

In versioni precedenti di Windows fino a Windows XP, GINA era responsabile della gestione dei dialoghi di login; la sostituzione di questo componente potrebbe consentire al malware di controllare completamente l'autenticazione dell'utente.

Con l'alterazione della chiave di registro, il malware garantisce la sua esecuzione ogni volta che il sistema viene avviato, che è una tattica comune per assicurare la persistenza del malware. Questo consente al malware di continuare a operare indisturbato e di riattivarsi dopo ogni riavvio del sistema, aumentando così la difficoltà nella sua rimozione e rilevazione.

Il malware sta, quindi, implementando una strategia multifase: prima stabilisce la persistenza all'interno del sistema e poi mira a compromettere la sicurezza del processo di autenticazione di Windows. Le implicazioni di tali azioni sono significative e possono variare dalla semplice esfiltrazione di dati alla completa compromissione dell'accesso al sistema, offrendo agli attaccanti una varietà di vie per l'esecuzione di attività malevoli.

In sintesi, le funzionalità implementate dal malware in questa sezione dimostrano un attacco sofisticato e multilivello al sistema, con l'intento di ottenere persistenza, evitare il rilevamento e sfruttare le funzionalità del sistema operativo per scopi malevoli. Questo tipo di comportamento sottolinea l'importanza di una solida strategia di sicurezza informatica che includa non solo software antivirus e antimalware aggiornati, ma anche l'educazione degli utenti sui rischi associati all'apertura di file sospetti e la necessità di applicare regolarmente patch e aggiornamenti di sicurezza al proprio sistema operativo e software.

Build Week III – Malware Analysis

Giorno 3:

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

1. Qual è il valore del parametro «ResourceName» passato alla funzione FindResourceA();
2. 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?
3. È possibile identificare questa funzionalità utilizzando l'analisi statica basica?
4. 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.

Nel terzo giorno di lavoro il team si è riunito per risolvere i quesiti richiesti.

Il team ha nuovamente utilizzato **OllyDBG** tool per analizzare il malware Build_Week_U3.

OllyDBG è uno strumento di reverse engineering e debugging per Windows, utilizzato per analizzare e modificare il codice di programmi eseguibili. Offre funzionalità di disassemblaggio del codice assembly, tracciamento dell'esecuzione del programma, modifica dinamica della memoria, breakpoint e altre utilità utili per l'analisi dei malware, lo sviluppo del software e il reverse engineering. È ampiamente impiegato dagli esperti di sicurezza informatica e dagli sviluppatori per comprendere il comportamento dei programmi e risolvere i problemi di programmazione.

```
00401080 | 55          | PUSH EBP
00401081 | 8BEC        | MOV EBP,ESP
00401083 | 83EC 18     | SUB ESP,18
00401086 | 56          | PUSH ESI
00401087 | 57          | PUSH EDI
00401088 | C745 EC 000000 | MOV DWORD PTR SS:[EBP-14],0
0040108F | C745 E8 000000 | MOV DWORD PTR SS:[EBP-18],0
00401096 | C745 F8 000000 | MOV DWORD PTR SS:[EBP-8],0
0040109D | C745 F0 000000 | MOV DWORD PTR SS:[EBP-10],0
004010A4 | C745 F4 000000 | MOV DWORD PTR SS:[EBP-C],0
004010AB | 837D 08 00   | CMP DWORD PTR SS:[EBP+8],0
004010AF | 75 07       | JNZ SHORT Malware_.004010B8
004010B1 | 33C0        | XOR EAX,EAX
004010B3 | E9 07010000 | JMP Malware_.004011BF
> 004010B8 | A1 30804000 | MOV EAX,DWORD PTR DS:[408030]
004010BD | 50          | PUSH EAX
004010BE | 8B0D 34804000 | MOV ECX,DWORD PTR DS:[408034]
004010C4 | 51          | PUSH ECX
004010C5 | 8B55 08     | MOV EDX,DWORD PTR SS:[EBP+8]
004010C8 | 52          | PUSH EDX
004010C9 | FF15 28704000 | CALL DWORD PTR DS:[<&KERNEL32.FindResou
004010CF | 8945 EC     | MOV DWORD PTR SS:[EBP-14],EAX
004010D2 | 837D EC 00   | CMP DWORD PTR SS:[EBP-14],0
004010D6 | 75 07       | JNZ SHORT Malware_.004010DF
004010D8 | 33C0        | XOR EAX,EAX
004010DA | E9 E0000000 | JMP Malware_.004011BF
> 004010DF | 8B45 EC     | MOV EAX,DWORD PTR SS:[EBP-14]
004010E2 | 50          | PUSH EAX
004010E3 | 8B4D 08     | MOV ECX,DWORD PTR SS:[EBP+8]
004010E6 | 51          | PUSH ECX
004010E7 | FF15 14704000 | CALL DWORD PTR DS:[<&KERNEL32.LoadResou
004010ED | 8945 E8     | MOV DWORD PTR SS:[EBP-18],EAX
004010F0 | 837D E8 00   | CMP DWORD PTR SS:[EBP-18],0
004010F4 | 75 05       | JNZ SHORT Malware_.004010FB
> 004010F6 | E9 AA000000 | JMP Malware_.004011A5
> 004010F8 | 8B55 E8     | MOV EDX,DWORD PTR SS:[EBP-18]
004010FB | 52          | PUSH EDX
004010FC | FF15 10704000 | CALL DWORD PTR DS:[<&KERNEL32.LockResou
00401105 | 8945 F8     | MOV DWORD PTR SS:[EBP-8],EAX
00401108 | 837D F8 00   | CMP DWORD PTR SS:[EBP-8],0
0040110C | 75 05       | JNZ SHORT Malware_.00401113
> 0040110E | E9 92000000 | JMP Malware_.004011A5
> 00401113 | 8B45 EC     | MOV EAX,DWORD PTR SS:[EBP-14]
00401116 | 50          | PUSH EAX
00401117 | 8B4D 08     | MOV ECX,DWORD PTR SS:[EBP+8]
0040111A | 51          | PUSH ECX
0040111B | FF15 0C704000 | CALL DWORD PTR DS:[<&KERNEL32.SizeofRes
00401121 | 8945 F0     | MOV DWORD PTR SS:[EBP-10],EAX
00401124 | 837D F0 00   | CMP DWORD PTR SS:[EBP-10],0
00401128 | 77 02       | JA SHORT Malware_.0040112C
```

```
ResourceType => "BINARY"
Malware_.00408038
ResourceName => "TGAD" ←
hModule
FindResourceA
hResource
hModule
LoadResource
hResource
LockResource
hResource
hModule
SizeofResource
```

Esame dei Risultati:

1. **Quale è il valore del parametro ResourceName passato alla funzione FindResourceA()?**

Il primo quesito richiede di esaminare il valore del parametro **ResourceName** passato alla funzione **FindResourceA()**.

Da subito noteremo dalla figura i parametri in evidenza, indicano che

Il valore del parametro **ResourceName** passato alla funzione **FindResourceA()** nel codice fornito è **TGAD**.

2. **Che funzionalità è implementata dal Malware?:**

Il secondo quesito richiede di identificare la funzionalità implementata dal Malware.

Dall'analisi del codice, emerge che il malware sta cercando una risorsa denominata TGAD utilizzando le funzioni **FindResourceA()**, **LoadResource()**, **LockResource()** e **SizeofResource()**.

Dunque, questo suggerisce che il malware potrebbe essere progettato per accedere o manipolare una risorsa specifica all'interno del sistema.

```

.text:004010C8      push     edx                ; hModule
.text:004010C9      call    ds:FindResourceA
.text:004010CF      mov     [ebp+hResInfo], eax
.text:004010D2      cmp     [ebp+hResInfo], 0
.text:004010D6      jnz     short loc_4010DF
.text:004010D8      xor     eax, eax
.text:004010DA      jmp     loc_4011BF
.text:004010DF      ;
.text:004010DF      loc_4010DF:                ; CODE XREF: sub_401080+56↑j
.text:004010DF      mov     eax, [ebp+hResInfo]
.text:004010E2      push    eax                ; hResInfo
.text:004010E3      mov     ecx, [ebp+hModule]
.text:004010E6      push    ecx                ; hModule
.text:004010E7      call    ds:LoadResource
.text:004010ED      mov     [ebp+hResData], eax
.text:004010F0      cmp     [ebp+hResData], 0
.text:004010F6      push    edx                ; hResData
.text:004010FF      call    ds:LockResource
.text:00401105      mov     [ebp+Str], eax
.text:00401108      cmp     [ebp+Str], 0
.text:0040110C      jnz     short loc_401113
.text:0040110E      jmp     loc_4011A5
.text:00401113      ;
.text:00401113      loc_401113:                ; CODE XREF: sub_401080+8C↑j
.text:00401113      mov     eax, [ebp+hResInfo]
.text:00401116      push    eax                ; hResInfo
.text:00401117      mov     ecx, [ebp+hModule]
.text:0040111A      push    ecx                ; hModule
.text:0040111B      call    ds:SizeofResource
.text:00401121      mov     [ebp+Count], eax

```

3. E' possibile l'identificazione della Funzionalità mediante analisi statica basica?

Il team conferma che la funzionalità implementata dal Malware in esame può essere analizzata e percepita mediante l'analisi statica di base.

Infatti, utilizzando l'analisi statica basica, è possibile identificare la funzionalità del malware osservando le chiamate di funzione e le operazioni eseguite sulle risorse nel codice. Le chiamate alle funzioni di gestione delle risorse e il valore specifico del parametro ResourceName (TGAD) indicano che il malware sta cercando di lavorare con una risorsa specifica all'interno del sistema.

4. Quali evidenze a ci sono a supporto dell'identificazione della funzionalità: Quando vengono richieste le evidenze a supporto includono le seguenti:

- Utilizzo della funzione **FindResourceA()** con il parametro ResourceName impostato su TGAD. La funzione **FindResourceA()** è comunemente utilizzata in ambienti Windows per trovare una risorsa all'interno di un modulo (come un file eseguibile o una libreria condivisa) in base al suo nome. Il fatto che il malware utilizzi questa funzione e imponga il parametro ResourceName a TGAD indica chiaramente che il malware sta cercando una risorsa specifica chiamata TGAD all'interno del sistema.
- Chiamate successive alle funzioni **LoadResource()**, **LockResource()** e **SizeofResource()**, che indicano un'elaborazione o un'utilizzazione della risorsa trovata. Dopo aver individuato la risorsa utilizzando **FindResourceA()** con il parametro ResourceName impostato su TGAD, il malware procede con le chiamate alle funzioni **LoadResource()**, **LockResource()** e **SizeofResource()**. Queste funzioni sono comunemente utilizzate per ottenere informazioni sulla risorsa trovata e per accedere ai suoi dati. Le chiamate successive a queste funzioni indicano chiaramente che il malware sta elaborando o utilizzando attivamente la risorsa individuata.
- Il fatto che il valore del parametro "ResourceName" sia una stringa costante nel codice indica che il malware è progettato per cercare una risorsa specifica, indipendentemente dal contesto in cui è eseguito.
Questo suggerisce un comportamento deliberato e predefinito del malware, piuttosto che una ricerca casuale o dipendente dal contesto delle risorse.

Dopo l'analisi accurata della funzione del malware e il ritrovamento di una stringa ricordiamo il concetto che lo costituisce.

Una stringa è una sequenza di caratteri, come lettere, numeri e simboli, che viene utilizzata per rappresentare dati testuali in un programma o un linguaggio di programmazione. In molti linguaggi di programmazione, le stringhe sono considerate come tipi di dati fondamentali e sono supportate da una vasta gamma di funzionalità.

Le funzionalità principali delle stringhe:

Rappresentazione di dati testuali: Le stringhe vengono utilizzate per rappresentare dati testuali come parole, frasi, numeri di serie, indirizzi, ecc. Possono contenere qualsiasi carattere, inclusi lettere, numeri, spazi e simboli.

Manipolazione di stringhe: I linguaggi di programmazione forniscono un insieme di funzioni e metodi per manipolare le stringhe, come la concatenazione (unione di due o più stringhe), la ricerca di sottostringhe, la sostituzione, l'accesso a caratteri specifici e molto altro.

Input/output di stringhe: Le stringhe vengono spesso utilizzate per l'input e l'output di dati da e verso un programma. Ad esempio, le stringhe possono essere utilizzate per accettare l'input dell'utente da tastiera o per mostrare output su schermo o su file.

Manipolazione di file e risorse: Le stringhe vengono spesso utilizzate per rappresentare percorsi dei file o nomi di risorse all'interno di un programma. Possono essere utilizzate per accedere e manipolare file, directory o risorse all'interno di un sistema operativo.

Internazionalizzazione e localizzazione: Le stringhe possono essere utilizzate per supportare la localizzazione e l'internazionalizzazione di un programma, consentendo la rappresentazione di testo in diverse lingue e codifiche.

D'altro canto, nel contesto dei malware, le stringhe vengono spesso utilizzate per vari scopi, tra cui:

Identificazione delle risorse: Le stringhe possono essere utilizzate per identificare in modo univoco risorse all'interno di un malware o di un sistema, come file, processi, registry key, ecc.

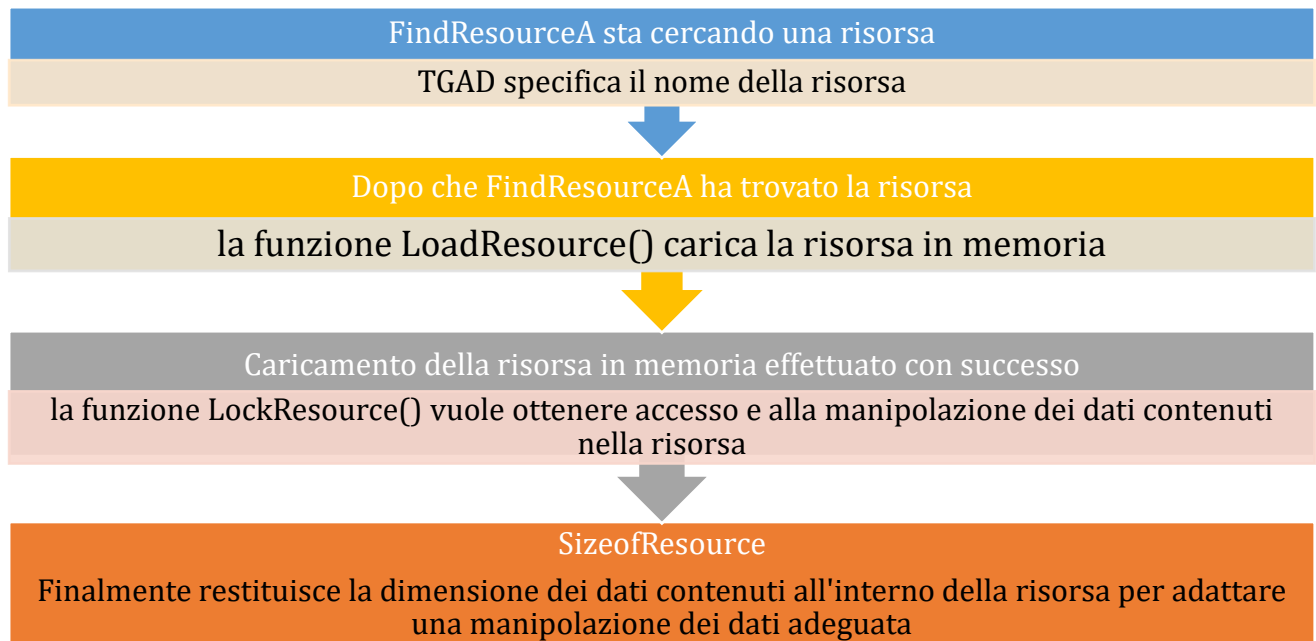
Comunicazione: Le stringhe possono essere utilizzate per la comunicazione con server di comando e controllo (C2) o altri componenti esterni del malware.

Codice malevolo nascosto: Le stringhe possono contenere payload criptati o comandi codificati che vengono poi decodificati e eseguiti dal malware durante l'esecuzione.

Messaggi di testo malevoli: Alcuni malware possono visualizzare messaggi di testo malevoli o minacciosi per intimorire gli utenti o convincerli a seguire istruzioni dannose.

In definitiva, le stringhe sono un elemento chiave nell'analisi dei malware poiché possono contenere informazioni utili per comprendere il comportamento, le capacità e gli obiettivi del malware stesso.

Diagramma di Flusso delle Funzionalità Principali



Esaminiamo più nel dettaglio il flusso delle funzioni coinvolte nel codice fornito e cosa potrebbe indicare riguardo al comportamento del malware.

FindResourceA(TGAD): La funzione FindResourceA() è utilizzata per cercare una risorsa all'interno di un modulo (ad esempio un file eseguibile) in base al suo nome. Nel nostro caso, il parametro TGAD specifica il nome della risorsa che il malware sta cercando. Questo potrebbe essere un file, una stringa, un'immagine o un altro tipo di risorsa all'interno del modulo.

LoadResource(): Dopo aver individuato la risorsa con FindResourceA(), la funzione LoadResource() viene utilizzata per caricare la risorsa in memoria. Questo passaggio potrebbe essere indicativo del fatto che il malware sta tentando di accedere o estrarre i dati contenuti nella risorsa identificata come TGAD.

LockResource(): Una volta caricata la risorsa, la funzione LockResource() viene utilizzata per ottenere un puntatore ai dati della risorsa caricata in memoria. Questo può indicare che il malware sta cercando di ottenere accesso ai dati contenuti nella risorsa per manipolarli o utilizzarli per scopi maligni.

SizeofResource(): La funzione SizeofResource() restituisce la dimensione della risorsa, ovvero la quantità di dati contenuti al suo interno. Questo passaggio potrebbe essere utile per il malware per calcolare quanto spazio di memoria è necessario per manipolare i dati della risorsa, o per determinare la complessità dell'operazione da eseguire su di essa.

Il malware sta cercando e ottenendo le dimensioni di una risorsa specifica nel sistema, potenzialmente tentando anche di bloccarla. Questo comportamento solleva sospetti poiché potrebbe indicare un tentativo di accesso non autorizzato ai dati sensibili o l'intenzione di compiere azioni dannose con la risorsa identificata come TGAD. Tuttavia, senza una comprensione dettagliata delle azioni del malware, è difficile prevedere le conseguenze precise di questo flusso di esecuzione.

Build Week III – Malware Analysis

Giorno 4:

Preparate l'ambiente ed i tool per l'esecuzione del Malware (suggerimento: avviate principalmente 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

- 1 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.

- 2 Quale chiave di registro viene creata?
- 3 Quale valore viene associato alla chiave di registro creata?
Passate ora alla visualizzazione dell'attività sul file system.
- 4 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.

Analisi dinamica del Malware

In questa giornata preparate l'ambiente ed i tool per l'esecuzione del Malware, andiamo ad utilizzare **Process Monitor** per eseguire un'analisi dinamica.

Il **Process Monitor** è uno strumento di sistema per Windows che fornisce una visione dettagliata delle attività del sistema operativo, inclusi registro di sistema, file, processi e attività di rete. Essenziale per identificare processi sospetti, monitorare le prestazioni del sistema e risolvere problemi di sistema in modo efficiente.

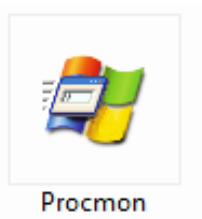
Con funzionalità avanzate come filtri e log delle attività, è un asset fondamentale per amministratori, sviluppatori e utenti avanzati.

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

Come sappiamo il malware è un file eseguibile che si trova nella cartella

C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3.

Eseguendo il malware, questo crea un file chiamato **msgina32.dll** nella stessa cartella, che è una DLL malevola che intercetta le credenziali dell'utente.



Andando ad analizzare anche il file creato dal malware utilizzando il tool di **OllyDBG** vediamo che il malware modifica anche il registro di sistema per rendere persistente la DLL, impostando il valore della chiave GinaDLL sotto **HKEY_LOCAL_MACHINE "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"** al percorso della DLL.

Address	Disassembly	Comment
00401000	55	PUSH EBP
00401001	8BEC	MOV EBP,ESP
00401003	51	PUSH ECX
00401004	6A 00	PUSH 0
00401006	8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]
00401009	50	PUSH EAX
0040100A	6A 00	PUSH 0
0040100C	68 3F00F00	PUSH 0F003F
00401011	6A 00	PUSH 0
00401013	6A 00	PUSH 0
00401015	6A 00	PUSH 0
00401017	68 54004000	PUSH Malware_.00400054
0040101C	68 02000000	PUSH 00000002
00401021	FF15 04704000	CALL DWORD PTR DS:[<&ADUAP132.RegCreateKeyExA>]

pDisposition = NULL

pHandle

pSecurity = NULL

Access = KEY_ALL_ACCESS

Options = REG_OPTION_NON_VOLATILE

Class = NULL

Reserved = 0

Subkey = "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"

hKey = HKEY_LOCAL_MACHINE

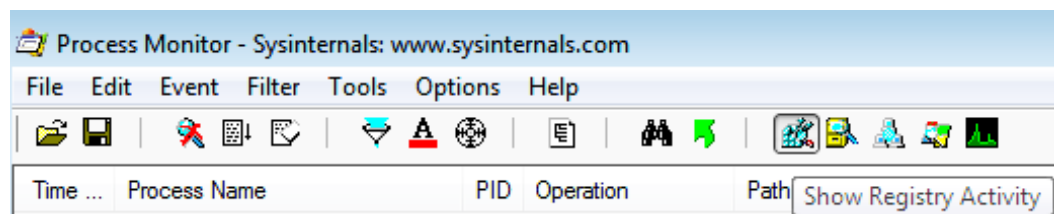
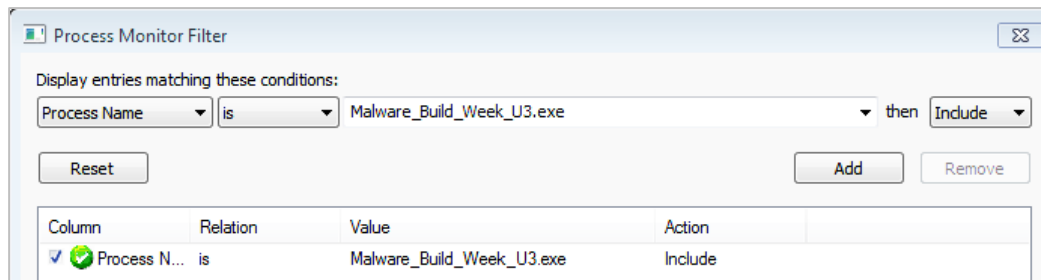
RegCreateKeyExA

Questo fa sì che la DLL venga caricata ogni volta che l'utente effettua il login o il logout.

2. Quale chiave di registro viene creata?

Analisi dinamica

Per analizzare il comportamento del malware, utilizziamo il tool Process Monitor, impostando il filtro per escludere tutti i processi tranne quello del malware, selezioniamo anche il filtro sull'interfaccia per visualizzare solo le attività di registro ed avviamo il malware.



Possiamo notare le seguenti modifiche apportate dal malware al sistema:

Creazione del file **msgina32.dll** nella cartella

C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3

Modifica del registro di sistema, impostando il valore della chiave GinaDLL sotto **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon** al percorso della DLL msgina32.dll

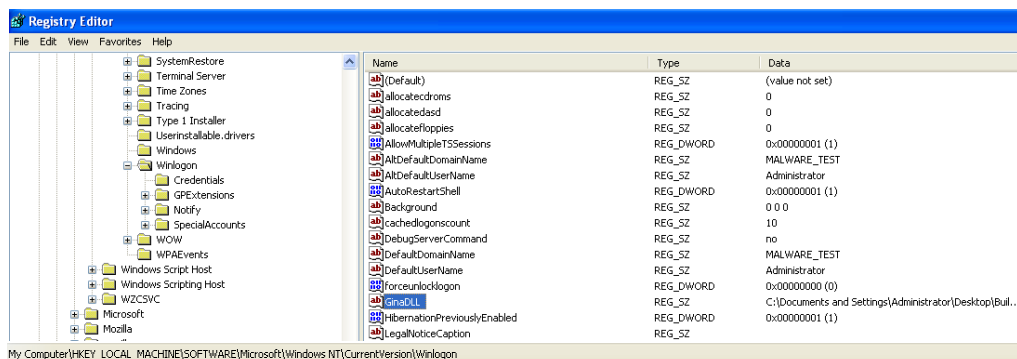
Time of Day	Process Name	PID	Operation	Path	Result	Detail
3:00:30.29143...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Malware_Build_Week_U3.exe	NAME NOT FOUND	Desired Access: Read
3:00:30.29250...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: Read
3:00:30.29250...	Malware_Build_Week_U3...	1648	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWORD, Le
3:00:30.29250...	Malware_Build_Week_U3...	1648	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
3:00:30.30097...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: Read
3:00:30.30102...	Malware_Build_Week_U3...	1648	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWORD, Le
3:00:30.30105...	Malware_Build_Week_U3...	1648	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
3:00:30.30109...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Secur32.dll	NAME NOT FOUND	Desired Access: Read
3:00:30.30114...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\RPCRT4.dll	NAME NOT FOUND	Desired Access: Read
3:00:30.30118...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ADVAPI32.dll	NAME NOT FOUND	Desired Access: Read
3:00:30.30122...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Access: Read
3:00:30.30125...	Malware_Build_Week_U3...	1648	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DWORD, Le
3:00:30.30127...	Malware_Build_Week_U3...	1648	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSUserEnabled	SUCCESS	Type: REG_DWORD, Le
3:00:30.30130...	Malware_Build_Week_U3...	1648	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	
3:00:30.30132...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: Read
3:00:30.30135...	Malware_Build_Week_U3...	1648	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTrack	NAME NOT FOUND	Length: 144
3:00:30.30137...	Malware_Build_Week_U3...	1648	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	
3:00:30.30139...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM	SUCCESS	Desired Access: Maximur
3:00:30.30142...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics	NAME NOT FOUND	Desired Access: Read
3:00:30.30146...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll	NAME NOT FOUND	Desired Access: Read
3:00:30.30153...	Malware_Build_Week_U3...	1648	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll	NAME NOT FOUND	Desired Access: Read
3:00:30.30584...	Malware_Build_Week_U3...	1648	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Access: All Acce
3:00:30.30588...	Malware_Build_Week_U3...	1648	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL	SUCCESS	Type: REG_SZ, Length:
3:00:30.30711...	Malware_Build_Week_U3...	1648	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	

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

Per poter rispondere a questa domanda andremo ad utilizzare **regedit** ovvero l'editor del registro di sistema, il quale è utilizzato per visualizzare e modificare i valori associati alle chiavi di registro nel Registro di sistema di Windows.

Seguendo il percorso trovato in precedenza, andiamo a trovare il file creato, quindi avviamo il registro di sistema con il comando Windows + R e digitiamo regedit, una volta aperto l'editor andiamo fino al file che ci interessa, quindi

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon



Il valore associato alla chiave di registro creata è

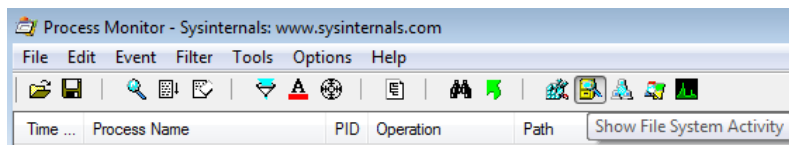
"C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll".

Questo valore indica il percorso del file msgina32.dll che viene utilizzato come Gina.DLL per l'interfaccia di autenticazione in Windows, Indica che è stata specificata un'interfaccia di autenticazione personalizzata (GINA) utilizzando il file msgina32.dll che si trova nel percorso specificato.

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

Per capire quale chiamata di sistema ha modificato il contenuto della cartella dove è presente il malware, andiamo ad utilizzare nuovamente Process Monitor, impostando il filtro per visualizzare solo le attività del sistema.

CreateFile è la chiamata del sistema dove è stato creato il file di **"msgina32.dll"**.



Time of Day	Process Name	PID	Operation	Path	Result	Detail
9/03/06 54487...	Malware_Build_Week_U3...	2708	QueryStandardInformationFile	C:\WINDOWS\system32\shlwapi.dll	SUCCESS	AllocationSize: 475,13
9/03/06 54490...	Malware_Build_Week_U3...	2708	FASTIO_RELEASE_FOR_SEC...	C:\WINDOWS\system32\shlwapi.dll	SUCCESS	
9/03/06 54492...	Malware_Build_Week_U3...	2708	CreateFileMapping	C:\WINDOWS\system32\shlwapi.dll	SUCCESS	
9/03/06 54493...	Malware_Build_Week_U3...	2708	FASTIO_RELEASE_FOR_SEC...	C:\WINDOWS\system32\shlwapi.dll	SUCCESS	SyncType: SyncTypeC
9/03/06 54522...	Malware_Build_Week_U3...	2708	CreateFile	C:\WINDOWS\system32\apphelp.dll	SUCCESS	Desired Access: Read
9/03/06 54537...	Malware_Build_Week_U3...	2708	CreateFileMapping	C:\WINDOWS\system32\apphelp.dll	SUCCESS	SyncType: SyncTypeC
9/03/06 54538...	Malware_Build_Week_U3...	2708	QueryStandardInformationFile	C:\WINDOWS\system32\apphelp.dll	SUCCESS	AllocationSize: 126,97
9/03/06 54540...	Malware_Build_Week_U3...	2708	FASTIO_RELEASE_FOR_SEC...	C:\WINDOWS\system32\apphelp.dll	SUCCESS	
9/03/06 54541...	Malware_Build_Week_U3...	2708	CreateFileMapping	C:\WINDOWS\system32\apphelp.dll	SUCCESS	SyncType: SyncTypeC
9/03/06 54542...	Malware_Build_Week_U3...	2708	FASTIO_RELEASE_FOR_SEC...	C:\WINDOWS\system32\apphelp.dll	SUCCESS	
9/03/06 54570...	Malware_Build_Week_U3...	2708	CreateFile	C:\Documents and Settings\Administrator\Local Settings\Temp\AAAA4_APPCOMPAT.TXT	NAME NOT FOUND	Desired Access: Read
9/03/06 54620...	Malware_Build_Week_U3...	2708	CreateFileMapping	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	SyncType: SyncTypeC
9/03/06 54621...	Malware_Build_Week_U3...	2708	QueryStandardInformationFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	AllocationSize: 8,132,1
9/03/06 54623...	Malware_Build_Week_U3...	2708	FASTIO_RELEASE_FOR_SEC...	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	
9/03/06 54633...	Malware_Build_Week_U3...	2708	CreateFileMapping	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	SyncType: SyncTypeC
9/03/06 54635...	Malware_Build_Week_U3...	2708	FASTIO_RELEASE_FOR_SEC...	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll	SUCCESS	
9/03/06 54688...	Malware_Build_Week_U3...	2708	CreateFile	C:\WINDOWS\AppPatch\sysmain.sdb	SUCCESS	Desired Access: Read
9/03/06 54688...	Malware_Build_Week_U3...	2708	CreateFileMapping	C:\WINDOWS\AppPatch\sysmain.sdb	SUCCESS	SyncType: SyncTypeC
9/03/06 54688...	Malware_Build_Week_U3...	2708	QueryStandardInformationFile	C:\WINDOWS\AppPatch\sysmain.sdb	SUCCESS	AllocationSize: 1,204,2
9/03/06 54688...	Malware_Build_Week_U3...	2708	FASTIO_RELEASE_FOR_SEC...	C:\WINDOWS\AppPatch\sysmain.sdb	SUCCESS	
9/03/06 54688...	Malware_Build_Week_U3...	2708	CreateFileMapping	C:\WINDOWS\AppPatch\sysmain.sdb	SUCCESS	SyncType: SyncTypeC
9/03/06 54687...	Malware_Build_Week_U3...	2708	FASTIO_RELEASE_FOR_SEC...	C:\WINDOWS\AppPatch\sysmain.sdb	SUCCESS	

Build Week III – Malware Analysis

Giorno 5:

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.

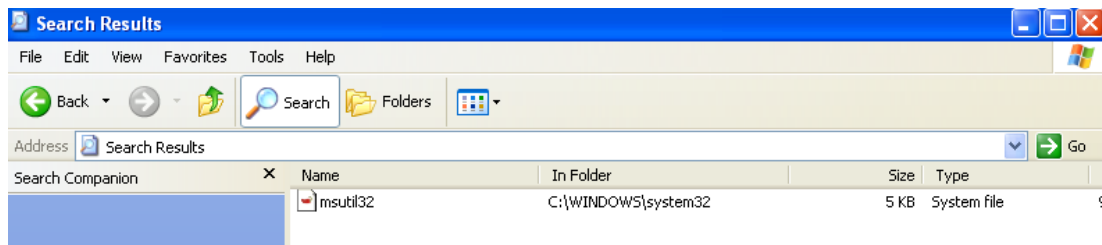
Utilizzando il tool di **IDA Pro** andiamo ad analizzare il file **msgina32.dll** che viene creato dal Malware, vediamo che il malware dropper in questione è stato identificato per la sua capacità di creare un file denominato **msutil32.sys** all'interno del sistema operativo bersaglio.

Questo comportamento è stato rilevato attraverso un'analisi del codice Assembly utilizzando IDA Pro.

Le azioni sospette eseguite dal malware includono: la creazione del file **msutil32.sys**, probabilmente contenente codice dannoso o funzionalità nocive.

```
.text:10001570
.text:10001570      mov     ecx, [esp+Format]
.text:10001574      sub     esp, 854h
.text:1000157A      lea     eax, [esp+854h+Args]
.text:10001581      lea     edx, [esp+854h+Dest]
.text:10001585      push    esi
.text:10001586      push    eax           ; Args
.text:10001587      push    ecx           ; Format
.text:10001588      push    800h          ; Count
.text:1000158D      push    edx           ; Dest
.text:1000158E      call    _vsprintf
.text:10001593      push    offset Mode    ; Mode
.text:10001598      push    offset Filename ; 'msutil32.sys'
.text:1000159D      call    _wopen
```

Andando a fare una ricerca del file nel nostro computer, abbiamo individuato il file **msutil32.sys** nella directory **C:\Windows\System32** e sappiamo che il malware dropper modifica la chiave di registro del sistema, possiamo dedurre, quindi, che il malware dropper utilizzi il file **msutil32.sys** come parte di una tattica per mantenere la persistenza nel sistema.



A questo punto sappiamo che il malware modifica la chiave di registro del sistema per garantire la sua esecuzione automatica, possiamo concludere che il malware dropper sta cercando di mantenere la persistenza nel sistema e di eseguire attività dannose ad ogni avvio del sistema. Questo comportamento indica chiaramente l'intenzione del malware di compromettere la sicurezza del sistema e potenzialmente eseguire ulteriori attività dannose, come il furto di dati sensibili o il danneggiamento dei file di sistema.

È importante rimuovere il file **msutil32.sys** e ripristinare la chiave di registro modificata per mitigare il rischio di danni al sistema, inoltre adottare misure di sicurezza aggiuntive, come l'aggiornamento dei software antivirus e l'implementazione di politiche di sicurezza più rigorose, per mitigare il rischio di infezione da parte di questo tipo di malware.

Procediamo per rispondere ai quesiti

Nell'ultimo giorno di lavoro al team è stato richiesto di formulare ipotesi sul possibile comportamento del malware una volta penetrato all'interno del sistema e in particolare infettando GINA l'interfaccia grafica lecita che consente agli utenti di effettuare l'accesso con il processo di autenticazione di password e username.

Prima di intraprendere le ipotesi formulate, spieghiamo con esattezza cosa sono i malware in generale e cosa potrebbero provocare all'interno di un network.

Inoltre spieghiamo GINA e le sue peculiarità.

Definizione di malware

I malware, abbreviazione per malicious software, sono programmi o codice malevolo progettati per infiltrarsi, danneggiare o controllare un computer o un sistema informatico senza il consenso dell'utente.

Tipologie, caratteristiche e scopi dei malware



Virus: Un virus è un tipo di programma per computer che, quando eseguito, si replica e si attacca ad altri file eseguibili, come ad esempio un documento, inserendo il proprio codice. La maggior parte dei virus richiede l'interazione dell'utente finale per attivarsi e possono essere scritti per agire in una data o un'ora specifica. I virus possono essere relativamente innocui, come quelli che mostrano un'immagine divertente. Oppure possono essere distruttivi, come quelli che modificano o cancellano dati. I virus possono anche essere programmati per mutare al fine di evitare la rilevazione. La maggior parte dei virus si diffonde tramite chiavette USB, dischi ottici, condivisioni di rete o e-mail.



Worm: Questo è un tipo di malware che si replica al fine di diffondersi da un computer all'altro. A differenza di un virus, che richiede un programma host per eseguirsi, i worm possono funzionare autonomamente. Oltre all'infezione iniziale dell'host, non richiedono la partecipazione dell'utente e possono diffondersi molto rapidamente sulla rete.

I worm condividono schemi simili: sfruttano vulnerabilità di sistema, hanno un modo per propagarsi e contengono tutti un codice malevolo (payload) per causare danni ai sistemi informatici o alle reti. I worm sono responsabili di alcuni degli attacchi più devastanti su Internet. Nel 2001, il worm Code Red aveva infettato oltre 300.000 server in appena 19 ore.



Trojan Horse (Trojan): Questo malware esegue operazioni dannose mascherando la sua vera intenzione. Potrebbe sembrare legittimo ma, in realtà, è molto pericoloso. I Trojan sfruttano i privilegi dell'utente e si trovano più spesso in file di immagine, file audio o giochi.

A differenza dei virus, i Trojan non si auto-replicano ma agiscono come esca per far passare software dannoso inosservato agli utenti ignari.



Spyware: Progettato per tracciare e spiare l'utente, lo spyware monitora la sua attività online e può registrare ogni tasto premuto sulla tastiera, oltre a catturare praticamente tutti i suoi dati, comprese informazioni personali sensibili come i dettagli del suo conto bancario online. Lo spyware fa ciò modificando le impostazioni di sicurezza sui suoi dispositivi. Spesso si raggruppa con software legittimo o cavalli di Troia.



Adware: L'adware viene spesso installato con alcune versioni di software ed è progettato per consegnare automaticamente annunci pubblicitari a un utente, più spesso su un browser web. Lo riconosci quando lo vedi! È difficile ignorarlo quando ti trovi di fronte a costanti annunci pop-up sullo schermo. È comune che l'adware venga fornito con lo spyware.



Ransomware: Questo malware è progettato per tenere in ostaggio un sistema informatico o i dati che contiene fino a quando non viene effettuato un pagamento. Di solito, il ransomware funziona criptando i tuoi dati in modo che non possa accedervi. Alcune versioni di ransomware possono sfruttare specifiche vulnerabilità di sistema per bloccarlo. Il ransomware viene spesso diffuso attraverso e-mail di phishing che ti incoraggiano a scaricare un allegato dannoso o attraverso una vulnerabilità del software.



Dropper: Un dropper è un tipo di malware progettato per installare e distribuire altri componenti dannosi sul sistema compromesso. Può sfruttare vulnerabilità di sicurezza o ingannare l'utente per eseguire il caricamento dei malware. Proprio quest'ultimo il dropper è il malware di nostro interesse ed è ciò che abbiamo scoperto durante l'analisi, il malware si comporta e ha le stesse funzioni di un dropper.

Proseguiamo la relazione con le funzionalità di GINA ed esaminiamo la minaccia rappresentata dalla sostituzione del file .dll di GINA con un file .dll malevolo e le implicazioni di sicurezza associate. Inoltre, presenteremo un'ipotesi sul funzionamento del malware dropper e le possibili soluzioni per mitigare e risolvere la minaccia su sistemi Windows 7 e XP.



GINA, acronimo di "Graphical Identification and Authentication," è un componente del sistema operativo Microsoft Windows progettato per gestire l'interfaccia di autenticazione degli utenti. L'utilizzo principale di GINA è quello di fornire un'interfaccia grafica per l'autenticazione degli utenti durante il processo di accesso al sistema operativo.

Nel nostro caso l'inserimento del valore "GinaDLL" nella chiave **HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon** suscita particolare preoccupazione.

Tale modifica potrebbe indicare un tentativo di compromettere o manipolare il processo di autenticazione di Windows, un attacco che potrebbe causare gravi problemi di sicurezza, come il furto di credenziali.

Il malware, infatti, associa il valore "GinaDLL" a un percorso di file, indicando un tentativo di sostituire o intercettare una DLL utilizzata da Windows durante l'autenticazione degli utenti. Inoltre, nel file System sono presenti le chiamate di sistema CreateFile() e WriteFile() che indicano modifiche al contenuto della cartella dell'eseguibile del malware, che coinvolge la creazione o la modifica del file "msgina32.dll".

Analisi della minaccia

Ipotesi:

Implicazioni della sostituzione del file .dll di GINA

Se il file .dll legittimo di GINA venisse sostituito con un file .dll malevolo, il malware potrebbe intercettare i dati inseriti dagli utenti durante il processo di autenticazione. Questo potrebbe includere username e password, consentendo agli attaccanti di compromettere le credenziali degli utenti e ottenere accesso non autorizzato ai sistemi.

Profilo del malware e delle sue funzionalità

Il malware che sostituisce il file .dll di GINA è categorizzato come un dropper, un tipo di malware progettato per rilasciare o installare altri componenti dannosi sul sistema compromesso.

Le sue funzionalità e peculiarità includono:



Intercezione dei dati: Il dropper potrebbe intercettare e raccogliere le credenziali inserite dagli utenti durante l'autenticazione tramite GINA.



Furtività: Potrebbe operare in modo stealth, mascherando la sua presenza per evitare la rilevazione da parte delle soluzioni antivirus e anti-malware.



Persistenza: Potrebbe installare componenti aggiuntivi sul sistema per garantire un accesso continuato o per avviarsi automaticamente ad ogni avvio del sistema.



Comunicazione con server remoto: Potrebbe trasmettere le credenziali rubate o altri dati sensibili a un server controllato dagli attaccanti.

Dopo aver menzionato i tipici comportamenti di un dropper e i possibili danni che crea all'interno di un sistema vediamo invece quali potrebbero essere le soluzioni che possiamo implementare per contrastare questa minaccia.

Rappresentazione grafica del comportamento del dropper

Soluzioni proposte

Soluzione 1: Mitigazione della minaccia



Monitoraggio dell'integrità del file .dll: Implementare un sistema di monitoraggio per rilevare eventuali modifiche al file .dll di GINA e avvisare gli amministratori di sistema in caso di alterazioni sospette.



Firma digitale e certificati: Utilizzare firme digitali e certificati per verificare l'autenticità dei file .dll di GINA, impedendo la sostituzione da parte di file non autorizzati.

Soluzione 2: Risoluzione della minaccia su sistemi Windows 7 e XP

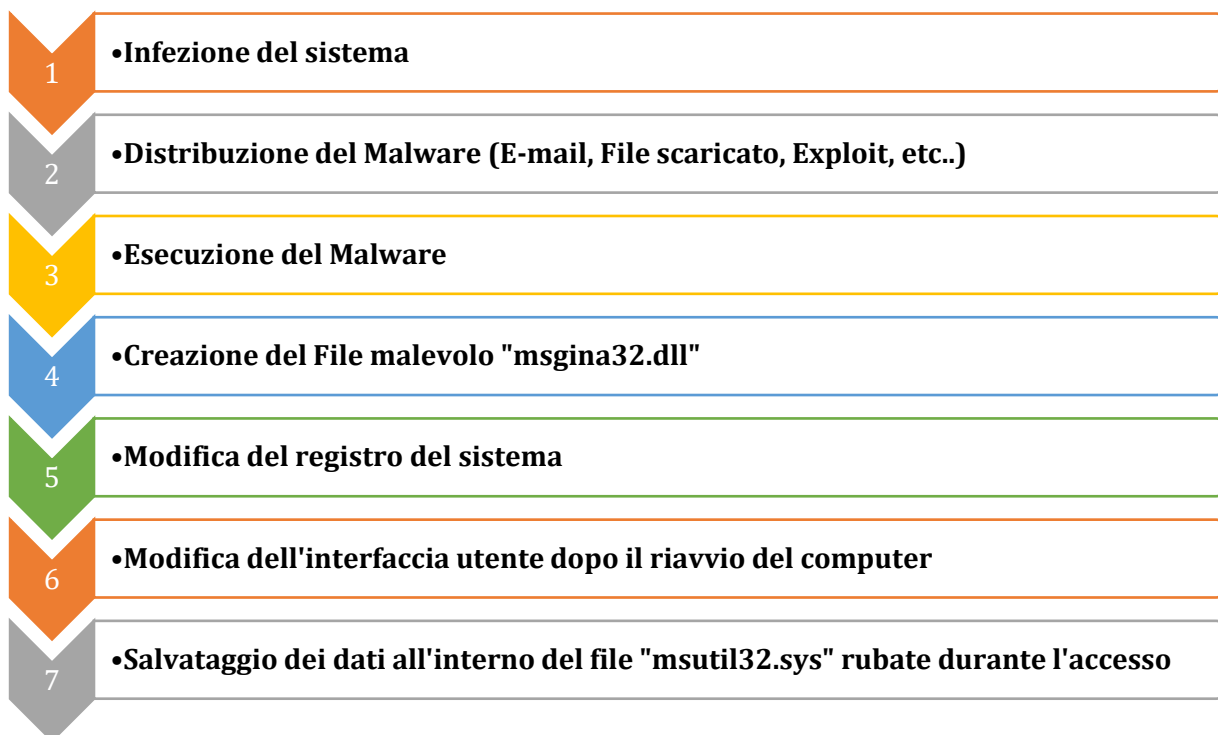


Analisi antivirus e rimozione: Utilizzare software antivirus aggiornati per identificare e rimuovere il dropper e altre componenti dannose dal sistema compromesso.



Ripristino dei file .dll originali: Sostituire i file .dll compromessi con versioni legittime da una fonte sicura e autorizzata.

Grafico del funzionamento del Malware



Conclusioni

La sostituzione del file.dll di GINA con un file .dll malevolo rappresenta una minaccia significativa per la sicurezza dei sistemi Windows in quanto compromette direttamente il processo di autenticazione degli utenti. GINA, l'interfaccia grafica lecita utilizzata per l'inserimento delle credenziali durante l'accesso al sistema operativo, svolge un ruolo critico nella verifica dell'identità degli utenti e nell'assicurare che solo persone autorizzate possano accedere al sistema.

La manipolazione di questo componente può consentire al malware di intercettare e registrare le credenziali degli utenti, come nomi utente e password, compromettendo così l'integrità dei dati sensibili e delle risorse del sistema. Inoltre, potrebbe consentire agli attaccanti di ottenere accesso non autorizzato ai sistemi compromessi, aprendo la strada a ulteriori attacchi o al furto di informazioni sensibili.

Per mitigare questa minaccia, è fondamentale adottare misure di sicurezza appropriate, come la verifica dell'integrità dei file di sistema, l'implementazione di controlli di accesso avanzati e la costante vigilanza sui comportamenti anomali del sistema. Inoltre, è essenziale effettuare regolarmente aggiornamenti di sicurezza e utilizzare strumenti antivirus e antimalware aggiornati per rilevare e rimuovere eventuali minacce in modo tempestivo.

In definitiva, proteggere la integrità di GINA e dei file di sistema associati è cruciale per garantire un ambiente informatico sicuro e proteggere i dati sensibili degli utenti e dell'organizzazione nel suo complesso.