

BUILD WEEK 3 PROJECT

Indice

- *RoadMap*
- *Giorno 1*
- *Giorno 2*
- *Giorno 3*
- *Giorno 4*
- *Giorno 5*
- *Conclusioni*



RoadMap



Giorno 1

Traccia

Con riferimento al file eseguibile Malware_Build_Week_U3, rispondete 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?
- Quali librerie importa il Malware? Per ognuna delle librerie impostate, fate delle ipotesi sulla base della sola analisi statica delle funzionalità che il Malware potrebbe implementare. Utilizzare le funzioni che sono richiamate all'interno delle librerie per supportare le vostre ipotesi

IDA PRO

Approfondimento

Uno degli strumenti che utilizzeremo oggi è IDA Pro.

IDA Pro è un potente ambiente di disassemblaggio e analisi per il reverse engineering di programmi binari. È uno strumento ampiamente utilizzato dagli analisti di sicurezza, ricercatori di malware e sviluppatori di software per esaminare il codice eseguibile e comprendere il funzionamento interno di programmi compilati.

CFF Explorer

Approfondimento

Uno degli strumenti che utilizzeremo oggi è CFF Explorer.

CFF Explorer è un software avanzato di esplorazione di file PE (Portable Executable). È utilizzato principalmente per analizzare: eseguibili in formato PE come gli EXE Windows, DLL (Dynamic Link Libraries) e altri formati correlati

Quanti parametri sono passati alla funzione Main()?

All'interno della funzione main sono stati passati 3 parametri.

Come è possibile vedere nell'immagine sottostante, si trovano all'interno di
“int __cdecl main(int argc, const char *argv[], const char *envp[])”

```
.text:004011D0 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:004011D0 _main          proc near               ; CODE XREF: start+AF↓p
.text:004011D0
.text:004011D0 hModule        = dword ptr -11Ch
.text:004011D0 Data           = byte ptr -118h
.text:004011D0 var_8          = dword ptr -8
.text:004011D0 var_4          = dword ptr -4
.text:004011D0 argc           = dword ptr  8
.text:004011D0 argv           = dword ptr  0Ch
.text:004011D0 envp           = dword ptr  10h
```

- **argc**: Questo parametro rappresenta il numero di argomenti della riga di comando passati al programma. È un intero che indica il numero totale di argomenti passati, incluso il nome del programma stesso.
- **argv**: Questo è un array di stringhe (puntatori a caratteri) che contiene gli argomenti effettivi della riga di comando passati al programma. argv[0] è il nome del programma, e argv[1] è il primo argomento della riga di comando, e così via.
- **envp**: Questo è un array di stringhe che contiene le variabili di ambiente disponibili per il processo. Questo parametro non è sempre utilizzato o richiesto in ogni implementazione della funzione main, ma la sua presenza indica che il programma potrebbe utilizzare o modificare le variabili d'ambiente.

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

All'interno della funzione main sono dichiarate **3** variabili locali.

- **var_C:** (dword ptr -0Ch) Questo indica una variabile a 32 bit (o 4 byte) posizionata a 12 byte (C in esadecimale) prima del frame pointer.
- **var_8:** (dword ptr -8) Una variabile a 32 bit a 8 byte prima del frame pointer.
- **var_4:** (dword ptr -4) Una variabile a 32 bit a 4 byte prima del frame pointer.

```
.text:004011D0 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:004011D0 _main          proc near               ; CODE XREF: start+AF↓p
.text:004011D6
.text:004011D0 hModule        = dword ptr -11Ch
.text:004011D0 Data           = byte ptr -118h
.text:004011D0 var_8          = dword ptr -8
.text:004011D0 var_4          = dword ptr -4
.text:004011D0 argc           = dword ptr 8
.text:004011D0 argv           = dword ptr 0Ch
.text:004011D0 envp           = dword ptr 10h
.text:004011D0
```

Quali sezioni sono presenti all'interno del file eseguibile?

Le sezioni presenti nel file eseguibile sono:

.text: Contiene il codice eseguibile del programma.

.rdata: Contiene dati di sola lettura, come costanti e importazioni di funzioni.

.data: Contiene dati inizializzati che possono essere modificati durante l'esecuzione del programma.

.rsrc: Contiene le risorse del programma, come icone, menu e dialoghi.

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

In queste immagini vengono mostrate le funzioni chiamate dal malware, appartenenti alla libreria kernel32.dll

00007632	00007632	0295	SizeofResource				
00007644	00007644	01D5	LockResource				
00007654	00007654	01C7	LoadResource				
00007622	00007622	02B8	VirtualAlloc				
00007674	00007674	0124	GetModuleFileNameA				
0000768A	0000768A	0126	GetModuleHandleA	0000786E	0000786E	0198	HeapCreate
00007612	00007612	00B6	FreeResource	0000787C	0000787C	028F	VirtualFree
00007664	00007664	00A3	FindResourceA	0000788A	0000788A	022F	RtlUnwind
00007604	00007604	001B	CloseHandle	00007896	00007896	0199	HeapAlloc
000076DE	000076DE	00CA	GetCommandLineA	000078A2	000078A2	01A2	HeapReAlloc
000076F0	000076F0	0174	GetVersion	000078B0	000078B0	027C	SetStdHandle
000076FE	000076FE	007D	ExitProcess	000078C0	000078C0	00AA	FlushFileBuffers
0000770C	0000770C	019F	HeapFree	000078D4	000078D4	026A	SetFilePointer
00007718	00007718	011A	GetLastError	000078E6	000078E6	0034	CreateFileA
00007728	00007728	02DF	WriteFile	000078F4	000078F4	00BF	GetCPInfo
00007734	00007734	029E	TerminateProcess	00007900	00007900	0089	GetACP
00007748	00007748	00F7	GetCurrentProcess	0000790A	0000790A	0131	GetOEMCP
0000775C	0000775C	02AD	UnhandledExceptionFilter	00007916	00007916	013E	GetProcAddress
00007778	00007778	00B2	FreeEnvironmentStringsA	00007928	00007928	01C2	LoadLibraryA
00007792	00007792	00B3	FreeEnvironmentStringsW	00007938	00007938	0261	SetEndOfFile
000077AC	000077AC	02D2	WideCharToMultiByte	00007948	00007948	0218	ReadFile
000077C2	000077C2	0106	GetEnvironmentStrings	00007954	00007954	01E4	MultiByteToWideChar
000077DA	000077DA	0108	GetEnvironmentStringsW	0000796A	0000796A	01BF	LCMapStringA
000077F4	000077F4	026D	SetHandleCount	0000797A	0000797A	01C0	LCMapStringW
00007806	00007806	0152	GetStdHandle	00007984	0000798A	0153	GetStringTypeA
00007816	00007816	0115	GetFileType	0000799C	0000799C	0156	GetStringTypeW
00007824	00007824	0150	GetStartupInfoA				
00007836	00007836	0109	GetEnvironmentVariableA				
00007850	00007850	0175	GetVersionExA				
00007860	00007860	019D	HeapDestroy				

In questa immagine vengono mostrate le funzioni chiamate dal malware, appartenenti alla libreria advapi32.dll

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

Quali librerie importa il Malware?

Il malware importa 2 librerie, kernel32.dll e advapi32.dll

Module Name	Imports	OFTs	TimeDateStamp	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

kernel32.dll: Fornisce funzioni di basso livello che gestiscono la memoria, i file, i processi e altri aspetti fondamentali del sistema operativo. Come ad esempio funzioni per la gestione dei processi, la sincronizzazione, l'allocazione di memoria e la gestione delle eccezioni.

advapi32.dll: Contiene funzioni per la gestione dei servizi, la sicurezza e le operazioni relative al registro di sistema. Fornisce un'interfaccia per interagire con il sottosistema di sicurezza di Windows. Include funzionalità per la gestione degli account utente, l'accesso alle informazioni sul sistema e la manipolazione del registro di sistema.

Ipotesi sul funzionamento del malware

La presenza e l'uso delle funzioni `SizeofResource()`, `LoadResource()`, e `FindResource()` indicano che il malware sta potenzialmente accedendo a risorse incorporate nel file eseguibile, che si trovano nella sezione `.rsrc`. Queste funzioni sono tipicamente utilizzate per estrarre queste risorse durante l'esecuzione del programma.

L'ipotesi che abbiamo sviluppato è che si tratti di un "dropper", un programma apparentemente innocuo che, quando eseguito, installa un malware sul computer vittima. Utilizzando le risorse incorporate, il dropper può evitare di essere rilevato da alcuni tipi di analisi statica, poiché il payload dannoso non è direttamente visibile nel codice del programma, ma piuttosto nascosto all'interno delle risorse del file.

Ipotesi sul funzionamento del malware

Queste funzioni possono essere utilizzate da un "dropper" per i seguenti scopi:

FindResource(): Trova la posizione di una risorsa specifica nel file eseguibile. Nel contesto del malware, questa risorsa potrebbe essere un altro pezzo di codice dannoso o un payload che il dropper è destinato a installare sul computer vittima.

SizeofResource(): Determina la dimensione della risorsa trovata. Questo è importante per sapere quanto spazio allocare quando si estraе la risorsa.

LoadResource(): Carica la risorsa in memoria per utilizzarla o eseguirla. Nel caso di un dropper, questa funzione sarebbe usata per caricare il payload in memoria prima di eseguirlo o scriverlo sul disco.

Giorno 2

Traccia

Con riferimento al Malware in analisi, spiegare:

- Lo scopo della funzione chiamata alla locazione di memoria 00401021
- Come vengono passati i parametri alla funzione alla locazione 00401021
- Che oggetto rappresenta il parametro alla locazione 00401017
- Il significato delle istruzioni comprese tra gli indirizzi 00401027 e 00401029
- Con riferimento all'ultimo quesito, tradurre il codice Assembly nel corrispondente costrutto C.
- 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.

Scopo della Funzione alla Locazione di Memoria 00401021

* .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	

La chiamata alla funzione *RegCreateKeyExA()* a questa locazione è cruciale nelle operazioni del malware.

Questa API di Windows è utilizzata per creare nuove chiavi di registro e accedere a chiavi esistenti con livelli di accesso specifici. È un mezzo diretto per il malware di istituire un punto di ancoraggio nel sistema infetto, consentendo potenzialmente al malware di auto-avviarsi ogni volta che il sistema viene acceso, modificando il comportamento di sistema o nascondendo altre sotto-chiavi e valori malevoli.

Questo potrebbe essere usato anche per sovrascrivere impostazioni di sicurezza, di rete o altre configurazioni che potrebbero facilitare operazioni di malware successive o garantire che il malware rimanga nascosto e persistente.

Passaggio dei Parametri alla Funzione alla Locazione 00401021

```
* .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
```

I parametri sono passati tramite lo stack, un'area di memoria che funziona con il metodo **LIFO** (Last In, First Out). Questo metodo è standard per le chiamate di funzione in molti linguaggi di programmazione, compreso il C, e consente di passare un numero variabile di parametri a una funzione. La natura ordinata di questo processo è critica per la corretta esecuzione della funzione, poiché ogni parametro deve essere recuperato nello stesso ordine in cui è stato inserito. Questo processo è inoltre fondamentale per la sicurezza, poiché errori nel manipolare lo stack possono portare a vulnerabilità come buffer overflow.

Oggetto Rappresentato dal Parametro alla Locazione 00401017

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

Il parametro passato alla locazione 00401017 è un riferimento a una chiave di registro ben nota, utilizzata per eseguire automaticamente applicazioni all'avvio. Il malware si sta approfittando di questa caratteristica per assicurare che il suo codice venga eseguito senza l'intervento dell'utente, massimizzando la sua persistenza nel sistema. Questa tecnica è comunemente usata in software legittimi per funzionalità come aggiornamenti automatici, ma nel caso del malware, è un chiaro segno di attività malevola.

Significato delle Istruzioni tra gli Indirizzi 00401027 e 00401029

• .text:00401027	test	eax, eax
• .text:00401029	jz	short loc_401032
• .text:0040102B	mov	eax, 1
• .text:00401030	jmp	short loc_40107B

Queste istruzioni rappresentano un controllo di integrità dopo un'operazione critica.

Utilizzando un test sui bit del registro EAX, il malware determina se la precedente chiamata alla *RegCreateKeyExA()* è stata eseguita con successo. Questo è essenziale per la logica di controllo del flusso del malware, che potrebbe alterare il suo comportamento a seconda dell'esito dell'operazione. Se l'operazione non riesce, potrebbe tentare un percorso alternativo, riprovare l'operazione o terminare per evitare la rilevazione.

Traduzione del Codice Assembly in Costrutto C

La traduzione del codice assembly fornito nel costrutto C fornisce un equivalente ad alto livello che è più comprensibile per i programmatori e analisti di malware. Mostra come il malware utilizza le istruzioni condizionali per prendere decisioni basate sul successo o fallimento delle sue operazioni, un concetto fondamentale nella programmazione.

```
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  
}
```

Il valore del parametro “ValueName” alla locazione 00401047

```
* .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 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.

Funzionalità implementata dal Malware in questa sezione

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.

Giorno 3

Traccia

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

- Qual è il valore del parametro “ResourceName” passato alla funzione FindResourceA();
- Il susseguirsi delle chiamate di funzione che effettua il Malware in questa sezione di codice l'abbiamo visto durante le lezioni teoriche. Che funzionalità sta implementando il Malware?
- È possibile identificare questa funzionalità utilizzando l'analisi statica basica?
- 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 principale) che comprende le 3 funzioni.

Olly DBG

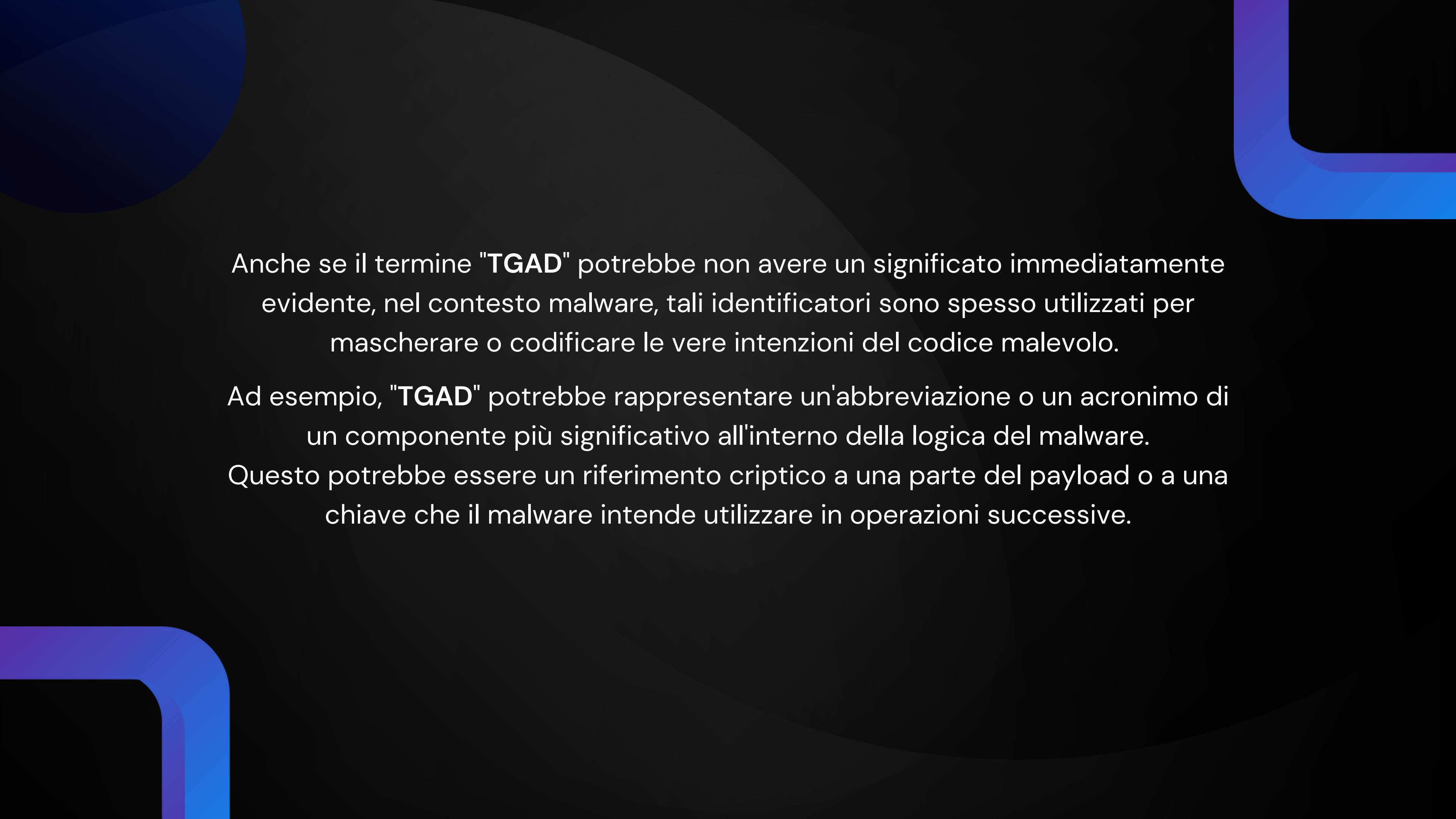
Approfondimento

OllyDbg è un debugger noto per la sua capacità di analizzare il codice assembly dei programmi eseguibili per scoprire bug e vulnerabilità o per comprendere meglio il funzionamento interno di un'applicazione. Strumento preferito dai reverse engineer, offre funzionalità potenti come l'analisi del codice, l'esecuzione step-by-step, i breakpoint condizionali e l'ispezione diretta delle registrazioni e della memoria. Attraverso un'interfaccia grafica intuitivo. È uno strumento estremamente potente per la diagnostica di software, la ricerca di sicurezza e l'apprendimento dell'ingegneria inversa, rendendolo indispensabile per sviluppatori, analisti di sicurezza e appassionati di tecnologia.

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

Nell'analisi malware, il parametro "ResourceName" usato nella chiamata alla funzione FindResourceA è di cruciale importanza. Questo parametro funge da identificativo per una risorsa incorporata nell'eseguibile del malware, e in questo caso specifico è stato identificato come "TGAD".

004010B0	> 7F 00000000	MOV EBP,ESP	ResourceType => "BINARY"
004010BE	. 8B0D 34804000	MOV ECX,DWORD PTR DS:[408034]	Malware_.00408038
004010C4	. 51	PUSH ECX	ResourceName => "TGAD"
004010C5	. 8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
004010C8	. 52	PUSH EDX	
004010C9	. FF15 28704000	CALL DWORD PTR DS:[&KERNEL32.FindResou:	hModule
004010CF	. 8945 EC	MOV DWORD PTR SS:[EBP-14],EAX	FindResourceA
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	. 8B40 08	MOV ECX,DWORD PTR SS:[EBP+8]	
004010E6	. 51	PUSH ECX	
004010E7	. FF15 14704000	CALL DWORD PTR DS:[&KERNEL32.LoadResou:	hResource
004010ED	. 8945 E8	MOV DWORD PTR SS:[EBP-18],EAX	hModule
004010F0	. 837D E8 00	CMP DWORD PTR SS:[EBP-18],0	LoadResource
004010F4	.~75 05	JNZ SHORT Malware_.004010FB	
004010F6	.~E9 AA000000	JMP Malware_.004011A5	
004010FB	> 8B55 E8	MOV EDX,DWORD PTR SS:[EBP-18]	
004010FE	. 52	PUSH EDX	
004010FF	. FF15 10704000	CALL DWORD PTR DS:[&KERNEL32.LockResou:	nHandles
00401105	. 8945 F8	MOV DWORD PTR SS:[EBP-8],EAX	SetHandleCount
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	. 8B40 08	MOV ECX,DWORD PTR SS:[EBP+8]	
0040111A	. 51	PUSH ECX	
0040111B	. FF15 0C704000	CALL DWORD PTR DS:[&KERNEL32.SizeofRes:	hResource
			hModule
			SizeofResource



Anche se il termine "**TGAD**" potrebbe non avere un significato immediatamente evidente, nel contesto malware, tali identificatori sono spesso utilizzati per mascherare o codificare le vere intenzioni del codice malevolo.

Ad esempio, "**TGAD**" potrebbe rappresentare un'abbreviazione o un acronimo di un componente più significativo all'interno della logica del malware.

Questo potrebbe essere un riferimento criptico a una parte del payload o a una chiave che il malware intende utilizzare in operazioni successive.

Funzionalità Implementata dal Malware

Dalla disamina delle istruzioni presenti nelle locazioni di memoria indicate, è evidente che il malware sta eseguendo una sequenza operativa caratteristica dei dropper. Questa sequenza è composta dalle seguenti chiamate API di Windows: FindResourceA, LoadResource, e SizeOfResource.

FindResourceA:

Questa funzione è utilizzata per localizzare una risorsa incorporata all'interno dell'eseguibile, in questo caso identificata dal nome "TGAD". La presenza di questa chiamata è indicativa del tentativo del malware di trovare un componente specifico che è stato strategicamente occultato all'interno del suo codice eseguibile, il quale potrebbe essere un secondo stadio di un attacco o un payload aggiuntivo.

LoadResource:

Dopo aver identificato la risorsa con FindResourceA, il malware utilizza LoadResource per caricare il contenuto della risorsa identificata in memoria. Ciò è tipico dei dropper, che spesso hanno bisogno di caricare componenti dannosi in memoria per eseguirli o per eseguire ulteriori operazioni su di essi.

SizeOfResource:

Per garantire che il componente malevolo sia stato caricato correttamente e per conoscere la quantità di memoria da allocare, il malware invoca SizeOfResource. Questa funzione restituisce la dimensione della risorsa specificata, che è un dato necessario per la corretta gestione della memoria durante l'estrazione e l'eventuale esecuzione del payload.

Funzionalità Implementata dal Malware

Le funzionalità che il malware sta implementando con questa sequenza di chiamate sono, quindi, la ricerca, il caricamento e il dimensionamento di un componente nascosto. Questa attività di routine è comunemente utilizzata dai malware per esfiltrare o eseguire codice dannoso senza destare sospetti, poiché la risorsa non viene eseguita direttamente dal disco, ma viene invece decompressa o assemblata in memoria.

L'uso di risorse binarie nascoste è un metodo astuto per eludere la rilevazione basata su firma, poiché il payload non appare come un file eseguibile separato sul disco rigido del sistema infetto. Inoltre, l'identificazione di una risorsa attraverso un nome come "TGAD" potrebbe indicare l'uso di tecniche di steganografia o di compressione per mascherare ulteriormente la natura della risorsa incorporata.

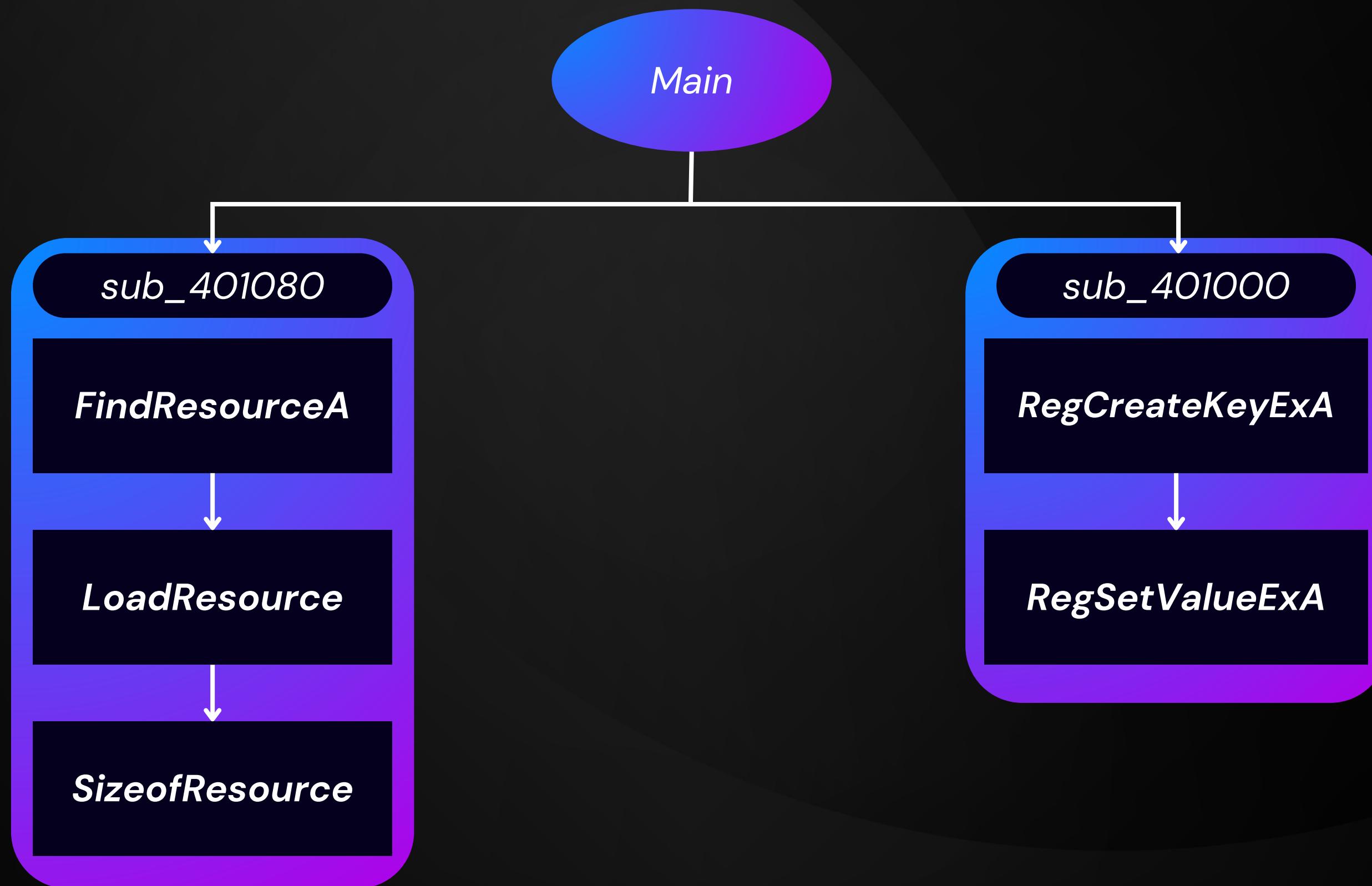
In conclusione, le chiamate sequenziali a ***FindResourceA***, ***LoadResource***, e ***SizeOfResource*** rivelano che il malware sta eseguendo le operazioni di un dropper, con l'intento di distribuire componenti dannosi nascosti all'interno del suo corpo eseguibile. Questo comportamento è coerente con tecniche avanzate di evasione e persistenza, che consentono al malware di rimanere latente e attivo nel sistema compromesso, eseguendo operazioni dannose senza essere facilmente rilevato da strumenti di sicurezza tradizionali.

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

Sì, è possibile identificare questa funzionalità attraverso l'analisi statica. Le evidenze a supporto includono:

- Il riferimento alla risorsa “TGAD” nel codice, che potrebbe essere una stringa utilizzata per identificare un componente specifico all'interno del file eseguibile.
- La sequenza ordinata di chiamate a funzioni API di Windows che sono comunemente usate per manipolare le risorse all'interno di un eseguibile: *FindResourceA*, *LoadResource*, e *SizeOfResource*.
- Il pattern di comportamento che segue la routine di un dropper, un tipo di malware che estrae e installa un payload più pericoloso.

Disegnare un diagramma di flusso che comprende le 3 funzioni viste in precedenza



Analisi statica

Approfondimento

L'analisi statica è una tecnica fondamentale nell'identificazione del comportamento dei malware senza eseguirli in un ambiente vivo. Questo metodo si basa sull'ispezione del codice sorgente o, più comunemente nel caso dei malware, del codice eseguibile disassemblato o decompilato. L'obiettivo è comprendere le funzionalità del malware e le potenziali minacce che esso rappresenta per i sistemi informatici.

Giorno 4

Traccia

Preparare l'ambiente ed i tools per l'esecuzione del Malware (suggerimento: avviate principalmente Process Monitor ed assicurate di eliminare ogni filtro cliccando sul "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

Analisi Malware

Per un'analisi professionale e dettagliata delle attività di un malware, è essenziale preparare un ambiente controllato che isoli il malware e protegga la rete e i sistemi circostanti da potenziali danni. Questo di solito comporta l'uso di una macchina virtuale o di un ambiente sandbox, in cui il malware può essere eseguito in sicurezza. I tool di monitoraggio come Process Monitor di Sysinternals sono poi configurati per registrare tutte le attività di sistema correlate al processo del malware, permettendo agli analisti di tracciare le sue azioni in dettaglio.

Process Monitor

Approfondimento

Process Monitor è uno strumento avanzato per monitorare in tempo reale le attività di file system, registro di sistema e processi su Windows. Offre funzionalità di filtraggio dettagliate e un'interfaccia intuitiva, rendendolo essenziale per l'analisi delle prestazioni e il troubleshooting.

Permette di rilevare modifiche al sistema, comportamento di malware e risolvere problemi di configurazione, grazie alla registrazione di accessi ai file e al registro. Utilizzato da amministratori, sviluppatori e tecnici, facilita l'identificazione di colli di bottiglia e conflitti, grazie anche alla cattura di stack trace e filtri personalizzabili.

Immagine 1

10:59:13.8932...	Malware_Build_Week_U3....	4020	Process Start	
10:59:13.8932...	Malware_Build_Week_U3....	4020	Thread Create	
10:59:13.8951...	Malware_Build_Week_U3....	4020	QueryNameInformationFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe
10:59:13.8954...	Malware_Build_Week_U3....	4020	Load Image	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe
10:59:13.8956...	Malware_Build_Week_U3....	4020	Load Image	C:\WINDOWS\system32\ntdll.dll
10:59:13.8956...	Malware_Build_Week_U3....	4020	QueryNameInformationFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\Malware_Build_Week_U3.exe
10:59:13.8959...	Malware_Build_Week_U3....	4020	CreateFile	C:\WINDOWS\Prefetch\MALWARE_BUILD_WEEK_U3.EXE-0E171D0F.pf
10:59:13.8960...	Malware_Build_Week_U3....	4020	QueryStandardInformationFile	C:\WINDOWS\Prefetch\MALWARE_BUILD_WEEK_U3.EXE-0E171D0F.pf
10:59:13.8962...	Malware_Build_Week_U3....	4020	ReadFile	C:\WINDOWS\Prefetch\MALWARE_BUILD_WEEK_U3.EXE-0E171D0F.pf
10:59:13.8988...	Malware_Build_Week_U3....	4020	CloseFile	C:\WINDOWS\Prefetch\MALWARE_BUILD_WEEK_U3.EXE-0E171D0F.pf
10:59:13.8990...	Malware_Build_Week_U3....	4020	CreateFile	C:
10:59:13.8990...	Malware_Build_Week_U3....	4020	QueryInformationVolume	C:
10:59:13.8991...	Malware_Build_Week_U3....	4020	FileSystemControl	C:
10:59:13.9005...	Malware_Build_Week_U3....	4020	CreateFile	C:\
10:59:13.9005...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\
10:59:13.9006...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\
10:59:13.9021...	Malware_Build_Week_U3....	4020	CloseFile	C:\
10:59:13.9022...	Malware_Build_Week_U3....	4020	CreateFile	C:\DOCUMENTS AND SETTINGS
10:59:13.9023...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\Documents and Settings
10:59:13.9030...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\Documents and Settings
10:59:13.9031...	Malware_Build_Week_U3....	4020	CloseFile	C:\Documents and Settings
10:59:13.9038...	Malware_Build_Week_U3....	4020	CreateFile	C:\Documents and Settings\ADMINISTRATOR
10:59:13.9039...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\Documents and Settings\Administrator
10:59:13.9040...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\Documents and Settings\Administrator
10:59:13.9047...	Malware_Build_Week_U3....	4020	CloseFile	C:\Documents and Settings\Administrator
10:59:13.9049...	Malware_Build_Week_U3....	4020	CreateFile	C:\Documents and Settings\Administrator\Desktop
10:59:13.9050...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\Documents and Settings\Administrator\Desktop
10:59:13.9079...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\Documents and Settings\Administrator\Desktop
10:59:13.9080...	Malware_Build_Week_U3....	4020	CloseFile	C:\Documents and Settings\Administrator\Desktop
10:59:13.9102...	Malware_Build_Week_U3....	4020	CreateFile	C:\Documents and Settings\Administrator\Desktop\BUILD_WEEK_UNIT_3
10:59:13.9103...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:59:13.9105...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:59:13.9121...	Malware_Build_Week_U3....	4020	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:59:13.9123...	Malware_Build_Week_U3....	4020	CreateFile	C:\WINDOWS
10:59:13.9155...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\WINDOWS
10:59:13.9157...	Malware_Build_Week_U3....	4020	QueryDirectory	C:\WINDOWS
10:59:13.9158...	Malware_Build_Week_U3....	4020	CloseFile	C:\WINDOWS
10:59:13.9177...	Malware_Build_Week_U3....	4020	CreateFile	C:\WINDOWS\AppPatch

Immagine 2

10:59:14.0291...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:59:14.0291...	Malware_Build_Week_U3...	4020	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAAppCompat
10:59:14.0291...	Malware_Build_Week_U3...	4020	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:59:14.0389...	Malware_Build_Week_U3...	4020	Load Image	C:\WINDOWS\system32\advapi32.dll
10:59:14.0392...	Malware_Build_Week_U3...	4020	Load Image	C:\WINDOWS\system32\rpcrt4.dll
10:59:14.0395...	Malware_Build_Week_U3...	4020	Load Image	C:\WINDOWS\system32\secur32.dll
10:59:14.0397...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:59:14.0397...	Malware_Build_Week_U3...	4020	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAAppCompat
10:59:14.0397...	Malware_Build_Week_U3...	4020	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:59:14.0398...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Secur32.dll
10:59:14.0398...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\RPCRT4.dll
10:59:14.0399...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ADVAPI32.dll
10:59:14.0399...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:59:14.0399...	Malware_Build_Week_U3...	4020	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAAppCompat
10:59:14.0399...	Malware_Build_Week_U3...	4020	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSUserEnabled
10:59:14.0400...	Malware_Build_Week_U3...	4020	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server
10:59:14.0400...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:59:14.0400...	Malware_Build_Week_U3...	4020	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTrack
10:59:14.0400...	Malware_Build_Week_U3...	4020	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:59:14.0400...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM
10:59:14.0400...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics
10:59:14.0401...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll
10:59:14.0401...	Malware_Build_Week_U3...	4020	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll
10:59:14.0407...	Malware_Build_Week_U3...	4020	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:59:14.0409...	Malware_Build_Week_U3...	4020	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:59:14.0410...	Malware_Build_Week_U3...	4020	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:59:14.0421...	Malware_Build_Week_U3...	4020	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:59:14.0445...	Malware_Build_Week_U3...	4020	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:59:14.0448...	Malware_Build_Week_U3...	4020	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:59:14.0455...	Malware_Build_Week_U3...	4020	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:59:14.0455...	Malware_Build_Week_U3...	4020	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
10:59:14.0458...	Malware_Build_Week_U3...	4020	SetEndOfFileInformationFile	C:\WINDOWS\system32\config\software.LOG
10:59:14.0462...	Malware_Build_Week_U3...	4020	SetEndOfFileInformationFile	C:\WINDOWS\system32\config\software.LOG
10:59:14.0466...	Malware_Build_Week_U3...	4020	SetEndOfFileInformationFile	C:\WINDOWS\system32\config\software.LOG
10:59:14.0468...	Malware_Build_Week_U3...	4020	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:59:14.0471...	Malware_Build_Week_U3...	4020	Thread Exit	
10:59:14.0471...	Malware_Build_Week_U3...	4020	Process Exit	
10:59:14.0472...	Malware_Build_Week_U3...	4020	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3

Osservazioni Rilevanti:

Creazione di File Non Autorizzati:

Dopo l'avvio del malware, viene osservata la creazione di un file chiamato "msgina32.dll" all'interno della directory dell'eseguibile. Questa DLL non fa parte delle librerie standard di Windows, indicando un'azione malevola. Il nome del file suggerisce un tentativo di mimetizzarsi come un componente legittimo di Windows, sfruttando la familiarità del nome per nascondere le sue attività.

10:59:14.0407...	Malware_Build_Week_U3...	4020	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:59:14.0409...	Malware_Build_Week_U3...	4020	CreateFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:59:14.0410...	Malware_Build_Week_U3...	4020	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3
10:59:14.0421...	Malware_Build_Week_U3...	4020	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:59:14.0445...	Malware_Build_Week_U3...	4020	WriteFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll
10:59:14.0448...	Malware_Build_Week_U3...	4020	CloseFile	C:\Documents and Settings\Administrator\Desktop\Build_Week_Unit_3\msgina32.dll

Osservazioni Rilevanti:

Manipolazione del Registro di Sistema:

Le attività di Process Monitor evidenziano modifiche critiche alle chiavi di registro. L'aggiunta del valore "**GinaDLL**" alla chiave "*HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon*" è particolarmente preoccupante. Questo cambio potrebbe indicare un tentativo di intercettare o alterare il processo di autenticazione di Windows, un vettore di attacco che può portare a gravi compromissioni della sicurezza, come il furto di credenziali.

10:59:14.0455...	 Malware_Build_Week_U3...	4020	 RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
10:59:14.0455...	 Malware_Build_Week_U3...	4020	 RegSetValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL

Osservazioni Rilevanti:

Interazione con il File System:

Le chiamate alle funzioni ***CreateFile()*** e ***WriteFile()*** mostrano che il malware non solo crea nuovi file ma li manipola attivamente. Queste azioni suggeriscono un tentativo di stabilire la persistenza del malware nel sistema, un passo essenziale per assicurare operazioni a lungo termine e resistere ai tentativi di rimozione.

Conclusioni

L'osservazione diretta della directory dell'eseguibile del malware rivela la creazione del file "msgina32.dll". Questo conferma l'attuazione di una delle tattiche più insidiose nel repertorio del malware: l'uso di dropper per distribuire e attivare componenti dannosi. La concomitante manipolazione delle chiavi di registro rafforza l'evidenza che il malware mira a modificare il processo di autenticazione e a garantirsi la persistenza nel sistema.

Giorno 5

Traccia

GINA (Graphical identification & 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.

Sostituzione file .dll

La sostituzione di un file .dll legittimo relativo al componente GINA di Windows con una variante malevola introduce significative vulnerabilità di sicurezza che possono essere sfruttate in diversi modi per compromettere sia l'integrità del sistema che la privacy degli utenti. Di seguito, viene approfondita e dettagliata ulteriormente l'analisi dei rischi associati:

- Intercezione delle Credenziali**
- Esecuzione di Codice Arbitrario**
- Elevazione dei Privilegi**
- Movimenti Laterali**
- Fuga di Dati**

Intercezione delle Credenziali

La cattura di credenziali tramite un .dll malevolo è un attacco diretto alla confidenzialità dell'utente. L'attaccante potrebbe utilizzare le credenziali rubate per eseguire azioni malevoli sotto l'identità dell'utente, compresi il furto di identità, transazioni finanziarie fraudolente e l'accesso a informazioni sensibili e personali.

Esecuzione di Codice Arbitrario

Il .dll malevolo potrebbe sfruttare vulnerabilità di sicurezza per eseguire attacchi sofisticati, come l'inserimento di payload indesiderati o la modifica dei processi del sistema. Il codice arbitrario può anche essere utilizzato per creare una porta di ingresso per attacchi futuri, aumentando la durata dell'esposizione del sistema a minacce esterne.

Elevazione dei Privilegi

Ottenere privilegi elevati è uno degli obiettivi principali degli attacchi informatici, poiché fornisce un controllo quasi illimitato sull'intero sistema. Un attaccante potrebbe utilizzare questi privilegi per disabilitare i sistemi di sicurezza, manipolare log e registri per nascondere la propria presenza o eseguire azioni distruttive senza rilevamento.

Movimenti Laterali

Questa tattica è particolarmente dannosa in ambienti di rete dove l'attaccante può sfruttare una singola compromissione per accedere a più risorse di rete, aumentando la portata e l'impatto dell'attacco.

Fuga di Dati

La violazione della confidenzialità dei dati può avere conseguenze devastanti, con la potenziale divulgazione di segreti commerciali, dati personali sensibili e altre informazioni riservate, portando a perdite finanziarie e danni all'immagine per le organizzazioni colpite.

In sintesi, sostituire una componente di autenticazione critica come GINA con una versione malevola può aprire la porta a una vasta gamma di attività illecite e dannose.

La risposta a tali minacce richiede un approccio olistico alla sicurezza, che combina misure tecniche, operative e organizzative per difendere efficacemente contro attacchi sofisticati e mitigare i rischi associati.

Misure di Mitigazione e Strategie di Difesa:

Strategie Proattive di Sicurezza

Implementazione di sistemi di rilevamento e prevenzione delle intrusioni, insieme a soluzioni di sicurezza endpoint avanzate, per monitorare e bloccare attività sospette in tempo reale.

Autenticazione Multifattore (MFA)

L'Autenticazione a più fattori (MFA) è una tecnologia di sicurezza che richiede almeno due passaggi per verificare l'identità dell'utente. Questi possono includere una combinazione di password, token o app, e dati biometrici come impronte digitali o riconoscimento del volto.

Hardening del Sistema

Questo processo comporta il rafforzamento della configurazione del sistema operativo e delle applicazioni per ridurre le vulnerabilità.

Isolamento e Segmentazione della Rete

imitare la capacità di un attaccante di muoversi lateralmente all'interno di una rete attraverso l'uso di firewall, VLAN e altre tecnologie di segmentazione può contenere un attacco e ridurre l'impatto su sistemi non compromessi.

Il team



Giulio Zanet



Riccardo Lattanzi



Giuseppe Lupoi



Davide Caldirola



Maria Huapaya



Alex Fiorillo



Michael Bonifazzi





GRAZIE
per aver letto la presentazione