**Pratica S10L5**

Traccia: Con riferimento al file Malware\_U3\_W2\_L5 presente all’interno della cartella «Esercizio\_Pratico\_U3\_W2\_L5 » sul desktop della macchina virtuale dedicata per l’analisi dei malware, rispondere ai seguenti quesiti:

1. Quali librerievengono importate dal file eseguibile?

2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli, altri costrutti )

4. Ipotizzare il comportamento della funzionalità implementata

5. BONUS fare tabella con significato delle singole righe di codice assembly.

Une image contenant texte, ligne, diagramme, logiciel

Description générée automatiquement

1. **librerie importate dal file eseguibile:**

Le librerie importate dal file eseguibile sono KERNEL32.dll e WININET.dll.

Queste sono librerie di sistema standard di Windows comunemente utilizzate per una varietà di funzionalità, tra cui gestione dei processi, gestione dei file, comunicazione di rete e altro ancora. Il fatto che il malware importi queste librerie suggerisce che potrebbe coinvolgere operazioni legate alla gestione dei file, alla comunicazione di rete o ad altre funzionalità di basso livello fornite da queste librerie.

***KERNEL32.dll***

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

La libreria KERNEL32.dll è una delle librerie principali del sistema operativo Windows. Contiene numerose funzioni di basso livello che gestiscono vari aspetti del sistema operativo, tra cui gestione dei processi, gestione dei file, gestione della memoria, gestione dei thread e altro ancora. Alcune delle funzioni più comuni all'interno di KERNEL32.dll includono:

* Creazione e gestione dei processi e dei thread: Funzioni come CreateProcess, CreateThread, WaitForSingleObject, ecc., vengono utilizzate per creare e gestire processi e thread.
* Gestione dei file e delle directory: Funzioni come CreateFile, ReadFile, WriteFile, CloseHandle, ecc., vengono utilizzate per la gestione dei file e delle directory.
* Gestione della memoria: Funzioni come VirtualAlloc, VirtualFree, ecc., vengono utilizzate per l'allocazione e la gestione della memoria.
* Gestione del tempo: Funzioni come GetSystemTime, GetLocalTime, ecc., vengono utilizzate per ottenere informazioni sul tempo di sistema.
* Gestione delle eccezioni: Funzioni come SetUnhandledExceptionFilter, ecc., vengono utilizzate per la gestione delle eccezioni.
* Gestione delle librerie DLL: Funzioni come LoadLibrary, GetProcAddress, ecc., vengono utilizzate per la gestione delle librerie di collegamento dinamico (DLL).

Queste sono solo alcune delle funzioni principali offerte da KERNEL32.dll. Il fatto che il malware importi questa libreria suggerisce che potrebbe utilizzare una o più di queste funzioni per eseguire operazioni nel sistema operativo Windows.

***WININET.dll***

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

La libreria WININET.dll è una libreria di sistema di Windows che fornisce funzionalità per la comunicazione di rete tramite i protocolli HTTP, HTTPS, FTP e altri protocolli Internet. Questa libreria è ampiamente utilizzata per consentire alle applicazioni di effettuare richieste e ricevere risposte da server remoti su Internet.

Alcune delle funzioni principali fornite da WININET.dll includono:

* Connessione e sessioni: Funzioni come InternetOpen, InternetConnect e InternetOpenUrl vengono utilizzate per aprire una sessione di comunicazione con un server remoto tramite un determinato protocollo.
* Richieste e risposte HTTP: Funzioni come HttpOpenRequest, HttpSendRequest e HttpQueryInfo vengono utilizzate per inviare richieste HTTP a un server e ricevere risposte.
* Download e upload di file: WININET.dll fornisce funzionalità per scaricare e caricare file da e verso server remoti tramite protocolli come HTTP e FTP.
* Gestione dei cookie: Le funzioni fornite consentono la gestione dei cookie HTTP, inclusa la lettura, la scrittura e l'eliminazione dei cookie.
* Gestione delle cache: La libreria permette la gestione della cache delle pagine Web, consentendo di controllare la memorizzazione temporanea dei dati scaricati dal Web.

Il fatto che il malware importi questa libreria suggerisce che potrebbe essere coinvolto in operazioni di rete, come il download di file aggiuntivi, l'invio di dati a un server remoto o la comunicazione con altri sistemi tramite protocolli Internet.

1. **Sezioni di cui si compone il file eseguibile del malware**

Le sezioni di un file eseguibile sono fondamentali per comprendere come il programma viene organizzato e come viene gestita la sua esecuzione. Le tre sezioni seguente sono quelle del nostro file eseguibile; ***.text, .rdata e .data***, sono particolarmente significative:

.***text***: Questa è la sezione del codice eseguibile. Contiene le istruzioni CPU che definiscono la logica del programma, il suo comportamento e le operazioni che eseguirà. Quando il programma viene eseguito, il contenuto di questa sezione viene caricato nella memoria del sistema e viene eseguito dall'unità di elaborazione centrale (CPU). Nel contesto di un malware, la sezione .text potrebbe contenere il codice malevolo che esegue operazioni dannose sul sistema infetto, come la propagazione, la raccolta di informazioni sensibili o il danneggiamento del sistema.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

***.rdata:*** Questa sezione contiene dati di sola lettura, che sono in genere costanti o stringhe utilizzate dal programma durante l'esecuzione ma che non possono essere modificati. Ad esempio, questa sezione potrebbe contenere stringhe di testo utilizzate per scopi di output, messaggi di errore o costanti utilizzate all'interno del programma. Nel caso di un malware, potrebbe contenere informazioni costanti utilizzate dal codice malevolo, come indirizzi IP o stringhe crittografiche.

Une image contenant texte, logiciel, nombre, Icône d’ordinateur

Description générée automatiquement

***.data***: Questa sezione contiene dati variabili che possono essere modificati durante l'esecuzione del programma. Include variabili globali, dati inizializzati e non inizializzati che il programma utilizza durante il suo funzionamento. Nel caso di un malware, questa sezione potrebbe contenere dati utilizzati per mantenere lo stato del malware, come contatori, informazioni di configurazione o variabili di controllo.

Une image contenant capture d’écran, texte, logiciel, Icône d’ordinateur

Description générée automatiquement

In generale, queste sezioni svolgono ruoli specifici nel funzionamento complessivo del programma, fornendo sia il codice eseguibile che i dati necessari per l'esecuzione e la funzionalità del programma. Nel nostro caso chi è di un malware, queste sezioni possono contenere sia elementi legittimi che codice malevolo, e l'analisi delle sezioni può fornire importanti indizi sulla natura e sul comportamento del malware.

1. **Identificazione dei costrutti noti (creazione dello stack, eventuali cicli, altri costrutti )**

* ***Creazione dello stack***: Le istruzioni ***push ebp e mov ebp, esp*** sono utilizzate per creare un nuovo frame di stack all'inizio di una funzione. Questo è un costrutto comune per impostare il punto di riferimento per le variabili locali e i parametri della funzione.
* ***Eventuali cicli***: la presenza di cicli potrebbe essere presente all'interno delle funzioni chiamate o in parti del codice non mostrate
* ***Condizioni e salti***: Le istruzioni jz e cmp sono utilizzate per controllare le condizioni e decidere il flusso del programma in base a esse. Questi sono costrutti comuni per eseguire operazioni condizionali.
* ***Altri costrutti***:

*Chiamate di funzioni:* L'istruzione call ds:InternetGetConnectedState viene utilizzata per chiamare la funzione InternetGetConnectedState. Questo è un costrutto noto per invocare una routine esterna per eseguire un'operazione specifica.

*Condizioni e salti:* Le istruzioni jz e cmp sono utilizzate per controllare le condizioni e decidere il flusso del programma in base a esse. Questi sono costrutti comuni per eseguire operazioni condizionali.

*Gestione dello stack:* Le istruzioni push e pop sono utilizzate per gestire i dati nello stack. In questo caso, push 0, push offset aSuccessInterne e push offset aError1\_1NoInte sono utilizzate per inserire valori nello stack.

Non sono presenti cicli direttamente identificabili tra le istruzioni fornite.

Inoltre, i malware possono includere tecniche avanzate per mascherare la propria presenza e rendere più difficile l'analisi, ad esempio utilizzando tecniche di "polimorfismo" o "metamorfismo" per modificare il proprio codice dinamicamente e sfuggire alla rilevazione. In questi casi, identificare i costrutti noti potrebbe richiedere un'analisi più approfondita e sofisticata del codice.

1. **Ipotizzare il comportamento della funzionalità implementata**

Basandomi sulle istruzioni di assembly fornite, posso tentare di ipotizzare il comportamento della funzionalità implementata.

Ecco una possibile interpretazione:

* ***Inizializzazione del contesto della funzione***: Le istruzioni push ebp e mov ebp, esp vengono utilizzate per inizializzare il contesto della funzione, creando un nuovo frame di stack.
* ***Controllo della connessione Internet***: Il codice chiama la funzione InternetGetConnectedState per verificare lo stato della connessione Internet del sistema. Il risultato viene memorizzato nella variabile locale [ebp+var\_4].
* ***Decisione basata sullo stato della connessione***: Viene eseguito un controllo per determinare se il sistema è connesso a Internet o meno. Se [ebp+var\_4] è zero, il programma salta all'etichetta Loc\_40102B, che sembra gestire il caso in cui non c'è connessione a Internet.
* ***Messaggio di successo o errore***: Se la connessione a Internet è attiva, viene visualizzato un messaggio di successo ("Success: Internet Connection\n"). Altrimenti, se non c'è connessione a Internet, viene visualizzato un messaggio di errore ("Error 1.1: No Internet\n").
* ***Pulizia dello stack e ritorno***: Dopo la visualizzazione del messaggio, lo stack viene pulito utilizzando l'istruzione add esp, 4. Successivamente, il programma imposta il registro eax a 1 e salta all'etichetta Loc\_40103A. Dopo questo, il contesto della funzione viene ripristinato e la funzione ritorna al punto di chiamata utilizzando le istruzioni mov esp, ebp, pop ebp e retn.

***“In sintesi, questa sequenza di istruzioni sembra essere parte di un'implementazione che verifica lo stato della connessione Internet e agisce di conseguenza, visualizzando un messaggio di successo se la connessione è attiva e un messaggio di errore se non lo è.”***

1. ***Tabella con significato delle singole righe di codice assembly.***

|  |  |
| --- | --- |
| **Codice Assembly** | **Significato** |
| push ebp | spinge il valore del registro ebp nello stack. Solitamente, questo viene fatto all'inizio di una funzione per salvare il contesto del frame precedente. |
| mov ebp, esp | copia il valore del registro esp (stack pointer) nel registro ebp (base pointer). Questo stabilisce un nuovo frame di stack per la funzione corrente. |
| Push ecx | spinge il valore del registro ecx nello stack. Il registro ecx può essere utilizzato per vari scopi, inclusi contatori di cicli o puntatori a dati. |
| Push 0 ; dwReserved | spinge il valore 0 nello stack. Potrebbe essere utilizzato come parametro per una funzione. |
| Push 0 ; IpdwFlags | spinge il valore 0 nello stack. Potrebbe anche essere utilizzato come parametro per una funzione diversa. |
| Call ds:InternetGetConnectedState | chiama la funzione InternetGetConnectedState per verificare lo stato della connessione Internet. |
| Mov [ebp+var\_4], eax | memorizza il valore di eax (probabilmente il risultato della chiamata a InternetGetConnectedState) nella posizione di memoria [ebp+var\_4]. |
| jz [ebp+var\_4], 0 | salta all'etichetta loc\_40102B se il valore in [ebp+var\_4] è zero (presumibilmente indicando che non c'è connessione Internet). |
| Cmp Short loc\_40102B | confronta due valori e salta all'etichetta loc\_40102B se il confronto è vero. |
|  |  |
| Push offset aSuccessInterne ; ‘’Success :Internet Connection\n’’ | spinge l'offset dell'etichetta "Success: Internet Connection\n" nello stack, presumibilmente per essere utilizzato come argomento di una funzione. |
| Call sub\_40117F | chiama la subroutine sub\_40117F. |
| Add esp, 4 | aggiunge 4 a esp per ripulire lo stack dopo la chiamata. |
| Mov eax, 1 | sposta il valore 1 nel registro eax. |
| Jmp short loc\_40103A | salta all'etichetta loc\_40103A. |
|  |  |
| Loc\_40102B : ; ‘’Error 1.1 : No Internet\n’’ | Etichetta di destinazione per il salto condizionato. |
| Push offset aError1\_1NoInte | spinge l'offset dell'etichetta "Error 1.1: No Internet\n" nello stack. |
| Call Sub\_40117F | chiama la subroutine sub\_40117F. |
| Add esp, 4 | aggiunge 4 a esp per ripulire lo stack dopo la chiamata. |
| Xor eax, eax | un'operazione di XOR tra eax ed eax, azzerando il registro eax. |
|  |  |
| Loc\_40103A: | Etichetta di destinazione per il salto incondizionato. |
| Mov esp, ebp | ripristina il valore originale di esp. |
| Pop ebp | ripristina il valore originale di ebp. |
| retn | ritorna al punto di chiamata della funzione. |
| Sub\_401000 endp | Fine della subroutine sub\_401000. |