Analisi Comportamento Malware | Marco Malizia - DataShields

Traccia

Il codice seguente mostra un estratto del codice di un malware. Identificate:
1. Il tipo di Malware in base alle chiamate di funzione utilizzate.
2. Evidenziate le chiamate di funzione principali aggiungendo una descrizione per ognuna di essa.
3. Il metodo utilizzato dal Malware per ottenere la persistenza sul sistema operativo.
4. BONUS: Effettuare anche un'analisi basso livello delle singole istruzioni.

.text: 00401010	push eax	
.text: 00401014	push ebx	
.text: 00401018	push ecx	
.text: 0040101C	push WH_Mouse	; hook to Mouse
.text: 0040101F	call SetWindowsHook()	
.text: 00401040	XOR ECX,ECX	
.text: 00401044	mov ecx, [EDI]	EDI = «path to startup_folder_system»
.text: 00401048	mov edx, [ESI]	ESI = path_to_Malware
.text: 0040104C	push ecx	; destination folder
.text: 0040104F	push edx	; file to be copied
.text: 00401054	call CopyFile();	

Svolgimento

1. Il tipo di Malware in base alle chiamate di funzione utilizzate.

Sulla base delle chiamate di funzione identificate nel codice Assembly, si può dedurre che il malware in questione sia un Keylogger o un Trojan con capacità di persistenza. Questa deduzione è supportata dalla presenza della chiamata `SetWindowsHook()`, che viene comunemente utilizzata per intercettare eventi come input da tastiera o mouse.

2. Evidenziate le chiamate di funzione principali aggiungendo una descrizione per ognuna di essa.

a. **call SetWindowsHook()**: Questa chiamata di funzione consente di impostare un hook su un evento di sistema, permettendo di monitorare o intercettare eventi come l'input del mouse o della tastiera. Nel codice analizzato, è stato specificato un hook per il mouse (`WH_MOUSE`), il che suggerisce che il malware sta cercando di monitorare o manipolare gli eventi del mouse. L'obiettivo potrebbe essere l'intercettazione degli input dell'utente o la raccolta di dati sensibili inseriti tramite dispositivi di puntamento.
b. **call CopyFile()**: Questa chiamata alla funzione API di Windows copia un file, in questo caso il malware, nella cartella di avvio del sistema. Questo passaggio consente al malware di ottenere persistenza, garantendo l'esecuzione automatica del codice malevolo a ogni avvio del sistema.

3. Il metodo utilizzato dal Malware per ottenere la persistenza sul sistema operativo.

Il malware utilizza la funzione `CopyFile()` per duplicarsi all'interno di una cartella di avvio del sistema, assicurandosi così di essere eseguito automaticamente ogni volta che il sistema viene avviato. Questo è un metodo comunemente impiegato dai malware per ottenere la persistenza su un sistema operativo.ù

4. Effettuare anche un’analisi basso livello delle singole istruzioni.

L'analisi a basso livello delle istruzioni del codice Assembly consente di comprendere nel dettaglio le operazioni specifiche eseguite dal malware. Di seguito, analizziamo le singole istruzioni:

- **push eax, ebx, ecx**: Queste istruzioni salvano i contenuti dei registri EAX, EBX e ECX nello stack, preservandone lo stato prima di eseguire altre operazioni che potrebbero modificarli. Questo è un approccio comune nei programmi Assembly, soprattutto durante le chiamate a funzioni che potrebbero alterare i registri.
- **push WH_MOUSE**: Questa istruzione inserisce il valore costante `WH_MOUSE` nello stack. `WH_MOUSE` è un codice che indica al sistema operativo di intercettare gli eventi

del mouse, come movimenti o clic. Questo parametro verrà passato alla funzione `SetWindowsHook()` per specificare il tipo di evento da monitorare.

- **call SetWindowsHook()**: Questa è una chiamata a una API di Windows, utilizzata per installare un "hook", ovvero un meccanismo che intercetta determinati eventi di sistema, come i movimenti del mouse o la pressione di tasti. In questo caso, l'hook viene impostato per gli eventi del mouse, suggerendo che il malware potrebbe essere un keylogger o uno strumento di monitoraggio.
- **XOR ECX, ECX**: L'operazione XOR tra un registro e se stesso azzera il registro, poiché ogni bit viene XORato con il suo equivalente. Questo è un metodo rapido per azzerare un registro, spesso utilizzato per pulire i registri prima di un'operazione successiva.
- **mov ecx, [EDI]**: Questa istruzione sposta nel registro ECX il valore contenuto nell'indirizzo puntato da EDI. In questo contesto, EDI sembra puntare al percorso della cartella di avvio del sistema, preparando ECX a contenere questo percorso.
- **mov edx, [ESI]**: Simile all'istruzione precedente, questa operazione carica nel registro EDX il valore contenuto nell'indirizzo puntato da ESI. ESI sembra puntare al percorso del file del malware.
- **push ecx, edx**: Queste istruzioni inseriscono i valori di ECX (che contiene il percorso di destinazione) e di EDX (che contiene il percorso del file da copiare) nello stack, preparandoli come parametri per la successiva chiamata a funzione.
- **call CopyFile()**: Questa chiamata a una API di Windows copia un file da una posizione all'altra. In questo caso, il malware si duplica nella cartella di avvio del sistema, garantendo la propria esecuzione automatica a ogni avvio del sistema, ottenendo così persistenza.