CS0424IT — ESERCITAZIONE S11L4
ANALISI COMPORTAMENTO MALWARE

Simone La Porta

GitHub

---

22 agosto 2024

INDICE

1 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.

Il codice Assembly fornito è il seguente:

```
.text: 00401010    push eax
.text: 00401014    push ebx
.text: 00401018    push ecx
.text: 0040101C    push WH_Mouse           ; hook to Mouse
.text: 0040101F    call SetWindowsHook()   ; sets a hook for mouse events
.text: 00401040    XOR ECX,ECX             ; clears the ECX register
.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()         ; copies the malware to the startup
    folder
```

2 SVOLGIMENTO

2.1 *Tipo di Malware*

In base alle chiamate di funzione identificate nel codice Assembly, è possibile dedurre che il tipo di malware analizzato sia un **Keylogger** o **Trojan** con capacità di persistenza. Questa deduzione è basata sulla presenza della chiamata `SetWindowsHook()`, utilizzata tipicamente per intercettare eventi come input della tastiera o del mouse.

2.2 *Chiamate di funzione*

- `call SetWindowsHook()` - Chiamata di funzione per impostare un hook su un evento di sistema, che permette di monitorare o intercettare eventi come l'input del mouse o della tastiera. Nel codice analizzato, è stato specificato un hook del mouse (`WH_Mouse`), il che indica 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.

- `call CopyFile()` - Chiamata alla funzione API di Windows che copia un file, in questo caso il malware, nella cartella di avvio per ottenere la persistenza e garantendo così l'esecuzione automatica del codice malevolo a ogni avvio del sistema.

2.3 *Metodo di persistenza*

Il malware utilizza la funzione `CopyFile()` per copiare se stesso in una cartella di avvio del sistema (`startup_folder_system`), garantendo che venga eseguito automaticamente all'avvio del sistema. Questo è un metodo comune utilizzato dai malware per ottenere la persistenza su un sistema operativo.

2.4 *Analisi a basso livello delle singole istruzioni*

L'analisi a basso livello delle singole istruzioni del codice Assembly permette di comprendere meglio le operazioni specifiche eseguite dal malware. Di seguito, analizziamo le istruzioni una per una:

- `push eax, ebx, ecx` - Queste istruzioni salvano il contenuto dei registri `EAX`, `EBX`, e `ECX` nello stack. Questo viene fatto per preservare il loro stato prima di eseguire ulteriori operazioni che potrebbero modificarli. È una pratica comune nei programmi Assembly, specialmente quando si fanno chiamate a funzioni che potrebbero alterare i registri.

- `push WH_Mouse` - Questa istruzione spinge 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 click). Questo parametro sarà passato alla funzione `SetWindowsHook()` per specificare il tipo di evento da monitorare.

- `call SetWindowsHook()` - Questa è una chiamata di funzione a una API di Windows. `SetWindowsHook()` viene utilizzata per installare un hook, ovvero un "gancio" che intercetta determinati eventi del sistema, come movimenti del mouse o pressioni di tasti. In questo caso, si sta impostando un hook sugli eventi del mouse, il che suggerisce che il malware potrebbe essere un keylogger o uno strumento di monitoraggio.

- `XOR ECX, ECX` - L'operazione XOR tra un registro e se stesso ha l'effetto di azzerare il registro, poiché ogni bit è XOR con il suo equivalente, che è identico. Questo è un metodo rapido per azzerare un registro e viene spesso usato per pulire i registri prima di un'operazione successiva.

- `mov ecx, [EDI]` - Questa istruzione sposta il valore contenuto nell'indirizzo puntato dal registro `EDI` nel registro `ECX`. In questo contesto, `EDI` sembra puntare al percorso della cartella di avvio del sistema (`startup_folder_system`), preparando `ECX` a contenere questo percorso.

- `mov edx, [ESI]` - Simile all'istruzione precedente, questa istruzione carica nel registro `EDX` il valore contenuto nell'indirizzo puntato da `ESI`. `ESI` sembra puntare al percorso del file malware stesso (`path_to_Malware`).

- `push ecx, edx` - Queste istruzioni spingono i valori di `ECX` (che ora contiene il percorso di destinazione) e `EDX` (che contiene il percorso del file da copiare) nello stack, preparandoli come parametri per la successiva chiamata a funzione.

- `call CopyFile()` - Questa è una chiamata a una API di Windows che copia un file da una destinazione a un'altra. In questo caso, il malware si copia nella cartella di avvio del

sistema, assicurandosi di essere eseguito ogni volta che il sistema si avvia, garantendo così la persistenza.