Funzionalità dei Malware

Obiettivo: Analizzare una parte di codice in Assembly.

1) Traccia

Il codice dato dall'esercizio è il seguente

Figura 1:

.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();	

2) Tipo di Malware

La prima richiesta dell'esercizio è di identificare il tipo di **Malware** che il codice ci pone davanti. Per prima cosa noto che è presente una chiamata alla funzione **SetWindowsHook**, ovvero installa un "*uncino*" (*hook* appunto) che si occupa di monitorare gli eventi che esegue una data periferica; in questo caso nella sezione *Commenti* è presente un "**hook to Mouse**" che suggerisce appunto l'utilizzo dell'uncino al mouse che verrà allertato ogni volta che l'utente colpito utilizzerà tale periferica, salvando le informazioni su un file di *log*.

Figura 1:

.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();	

Questo tipo di **Malware** suppongo si tratti di un **Keylogger** e potrebbe essere utilizzato per vari scopi come il monitorare le attività degli utenti colpiti, rubare le loro informazioni personali, eseguire attacchi di phishing o controllare in remoto il sistema vittima.

2) Chiamate di Funzione

Successivamente viene richiesto dalla traccia di individuare le **Chiamate di Funzione** del codice.

Figura 1:

.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();	

Possiamo osservare due Chiamate:

- **Copyfile**: questa funzione, come suggerisce il nome, viene utilizzata per copiare un file esistente in un nuovo file. Se utilizzata per copiare un file crittografato, tenta di crittografare il file con le chiavi usate nella crittografia del file di origine. In caso non riesca ad usare quelle del file di origine tenta di crittografare con delle chiavi predefinite. Notiamo inoltre che nelle istruzioni **PUSH** vengono inseriti nello stack il file da copiare e la cartella di destinazione.

- **SetWindowsHook**: come detto in precedenza la funzione ha il compito di installare un “uncino” alla periferica che viene indicata, in questo caso il Mouse (**PUSH WH_Mouse**), che viene monitorato da un probabile hacker.

3) Persistenza

Come ulteriore richiesta viene indicato lo spiegare come fa il codice a creare una **Persistenza**.

Figura 1:

.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();	

Come detto nel punto precedente i primi due push della funzione **CopyFile** (partendo da basso) inseriscono nello stack il file da copiare e la cartella di destinazione; successivamente viene assegnato ad entrambi un valore con l'istruzione **mov**.
Notiamo infatti che sempre della sezione commenti a **edx** viene assegnato il *Path* del **Malware** (quindi è il file da copiare) e a **ecx** viene assegnato il *Path* dove esso verrà salvato. La destinazione è la cartella **startup_folder_system** che, come il nome suggerisce, si tratta probabilmente cartella dei Registri che vengono eseguiti all'avvio del Sistema Operativo o quando eseguiamo il login.

4) Analisi Istruzioni

Ultima richiesta della traccia è di analizzare le singole istruzioni del codice. Nei punti precedenti ho già spiegato la maggior parte delle istruzioni di cui è composto il codice per approfondire le richieste della traccia.

Le istruzioni partendo da basso sono:

- **call CopyFile**: effettua una chiamata alla funzione **CopyFile** che è utilizzata per copiare un file esistente in un nuovo file.

- i **push ecx** e **edx**: fissano nello stack i valori di **ecx** e **edx**.

- **mov** su **ecx** e **edx** assegnano i valori dei registri **ESI** e **EDI** ai due parametri (in questo caso il **Malware** da copiare e il percorso dove deve essere copiato).

- **XOR ecx, ecx**: con l'operatore logico **XOR** si resetta il valore di **ecx**.

- **call SetWindowsHook**: effettua una chiamata alla funzione **SetWindowsHook** che come ho detto in precedenza installa un "uncino" che monitora gli eventi riferiti alla periferica designata.

- **push WH_Mouse**: qui si indica il tipo di hook da installare, ovvero sulla periferica **Mouse** e da restituire alla funzione **GetMessage** o **PeekMessage**.

- **push ecx, ebx, eax**: inserisce nello stack i valori dei tre **Registri** indicati.