S11-L4

(Bonaldi Cristian)

Malware analysis - Funzionalità dei Malware

Traccia:

La figura nella slide successiva mostra un estratto del codice di un malware. Identificate:

1. **Il tipo di malware in base alle chiamate di funzione utilizzate.**
2. **Le principali chiamate di funzione evidenziandole e aggiungendo una descrizione per ognuna.**
3. **Il metodo utilizzato dal malware per ottenere la persistenza sul sistema operativo.**

BONUS: Effettuare anche un'analisi a basso livello delle singole istruzioni.

Esercizio e sviluppo:

1.

Il malware pusha WH_MOUSE (0040101C) nello stack, dopodiché effettua una chiamata alla funzione **SetWindowsHook** (0040101F).

Questo sta a significare che il malware in esame sta installando un hook (ovvero un meccanismo per cui un applicazione è in grado di intercettare eventi, come i movimenti del mouse in questo caso) di Windows per essere in grado di manipolarli e/o bloccarli.
La funzione SetWindowsHook di fatto serve a gestire l'hook in questione (WH_MOUSE)

```
.text: 0040101C    push WH_Mouse          ; hook to Mouse
.text: 0040101F    call SetWindowsHook()
```

Per quanto riguarda le chiamate di funzioni, possiamo inoltre notare come, tramite l'istruzione **call CopyFile(),** il malware va a chiamare una funzione delle API di Windows (CopyFile).

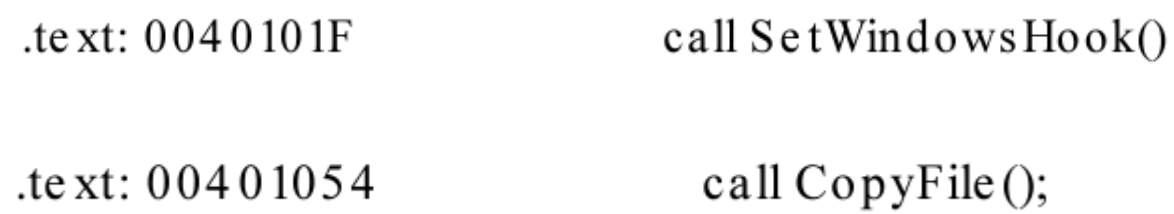
I parametri passati alla funzione sono rispettivamente:

```
push ecx          ; destination folder
push edx          ; file to be copied
call CopyFile();
```

Ovvero *destination folder* (push ecx), quindi la cartella di destinazione dove il file deve essere copiato e *file to be copied* (push edx), il file da copiare, facilitandone in questo modo la replicazione.

Questo tipo di malware può essere identificato come **spyware** o **keylogger**, progettati per raccogliere informazioni sensibili tramite il monitoring dell'input utente.

2.



SetWindowsHook: Come già accennato precedentemente, questa funzione viene utilizzata in Windows per inserire una hook routine che osserva e manipola eventi di sistema. Questa sarà eseguita ogni volta che l'evento si verifica (in questo caso *WH_MOUSE*, quindi installa una hook routine relativa agli eventi del mouse, come movimenti, click ecc).

CopyFile: Questa funzione infine viene utilizzata per permettere la copia di un file, in questo caso fa in modo che il malware ottenga persistenza all'interno del sistema anche dopo il riavvio.

3.

Come ci suggerisce la slide, con i comandi **push ecx** e **push edx** va a definire rispettivamente nei registri corrispondenti, sia la cartella di destinazione (dove il file deve essere copiato) e il file da copiare.

Infatti, questi corrispondono esattamente ai parametri della funzione già menzionata CopyFile.

```
push ecx                ; destination folder
push edx                ; file to be copied
```

I **mov** non fanno altro che inserire:

- in ecx il path della cartella di avvio del sistema
- in edx il path dove è presente il malware

```
mov ecx, [EDI]          EDI = «path to
                        startup_folder_system»
mov edx, [ESI]          ESI = path_to_Malware
```

Con la chiamata alla funzione CopyFile avviene l'effettiva copia del malware (dal path dove è presente il malware al path della cartella di avvio del sistema).

4.

Analisi basso livello delle singole istruzioni:

```
.text: 00401010          push eax
.text: 00401014          push ebx
.text: 00401018          push ecx
```

I push salvano i valori dei registri (eax,ebx,ecx) nello stack.

```
.text: 0040101C          push WH_Mouse          ; hook to Mouse
```

Viene salvato il valore dell'hook WH_Mouse nello stack.

```
.text: 0040101F          call SetWindowsHook()
```

call chiama la funzione SetWindowsHook per la gestione dell'hook.

```
.text: 00401040          XOR ECX,ECX
```

E' un'operazione logica XOR tra il registro ECX e sé stesso, azzerando di fatto il registro.

```
.text: 00401044          mov ecx, [EDI]          EDI = «path to
                                                 startup_folder_system»
```

Copia il valore dell'indirizzo di memoria puntato da EDI nel registro ecx. EDI = il path della cartella di avvio del sistema.

```
.text: 00401048        mov edx, [ESI]        ESI = path_to_Malware
```

Copia il valore dell'indirizzo di memoria puntato da ESI nel registro edx. ESI = il path del file del malware.

```
.text: 0040104C        push ecx              ; destination folder
```

Pusha il valore di ecx nello stack, ovvero la cartella di destinazione.

```
.text: 0040104F        push edx              ; file to be copied
```

Pusha il valore di edx nello stack, ovvero il file da copiare.

```
.text: 00401054        call CopyFile();
```

Chiamata alla funzione CopyFile, che effettivamente copia il file del malware nella cartella di destinazione.