Pratica S11/L4
Giaimo Rosario

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. 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

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();
```

Svolgimento

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

L'analisi del codice malware rivela un comportamento dannoso che include l'intercettazione dei clic del mouse e la copia del file malware nella cartella di avvio del sistema per ottenere la persistenza. Il malware utilizza diverse tecniche per nascondere la sua presenza e rendere difficile la sua rimozione.
L'utilizzo della funzione **`SetWindowsHook`** con il parametro **`WH_MOUSE`** nell'ultima riga di codice indica che questo malware non è un keylogger tradizionale che registra i tasti premuti sulla tastiera, ma bensì un **mouse logger.** Un mouse logger è un tipo di malware che monitora e registra i movimenti e i clic del mouse dell'utente. Questo tipo di malware può essere utilizzato per scopi dannosi come **Rubare informazioni sensibili, Tracciare l'attività dell'utente e Prendere il controllo del computer.**

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

```
.text: 00401010        push eax
.text: 00401014        push ebx
.text: 00401018        push ecx
.text: 0040101C        push WH_Mouse          ; hook to Mouse
.text: 0040101F        call SetWindowsHook()
```

- **`push eax`, `push ebx`, `push ecx`**: Queste istruzioni salvano i registri EAX, EBX ed ECX nello stack. I registri sono aree di memoria temporanea utilizzate per memorizzare valori durante l'esecuzione del programma.
- **`push WH_MOUSE`**: Questa istruzione spinge il valore **`WH_MOUSE`** nello stack. **`WH_MOUSE`** è una costante definita da Windows che identifica l'hook della tastiera.
- **`call SetWindowsHook()`**: Questa istruzione chiama la funzione **`SetWindowsHook()`**. La funzione **`SetWindowsHook()`** installa un hook sul sistema, specificando il tipo di hook da installare e la funzione di callback da eseguire quando si verifica l'evento.
- **`XOR ECX, ECX`**: Questa istruzione imposta il registro ECX a zero. L'istruzione `XOR` esegue un'operazione tra due operandi. In questo caso, l'operando di destra è lo stesso registro ECX, il che significa che il risultato dell'operazione è sempre zero.
- **`movecx, [EDI]`**: Questa istruzione copia il valore contenuto all'indirizzo puntato da EDI nel registro ECX. EDI è un registro che punta alla memoria allocata per memorizzare il percorso della cartella di avvio del sistema.
- **`movedx, [ESI]`**: Questa istruzione copia il valore contenuto nell'indirizzo puntato da ESI nel registro EDX. ESI è un registro che punta alla memoria allocata per memorizzare il percorso del file malware.
- **`push ecx`**, **`push edx`**: Queste istruzioni spingono i registri ECX e EDX nello stack.
- **`Call CopyFile()`**: La funzione `CopyFile()` copia un file da una posizione all'altra. In questo caso, la funzione `CopyFile()` viene utilizzata per copiare il file malware nella cartella di avvio del sistema.

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

L'analisi del codice dimostra che il malware utilizza un metodo efficace per ottenere la persistenza sul sistema operativo. Copiando se stesso nella cartella di avvio, il malware garantisce la sua esecuzione automatica e aumenta la sua probabilità di raggiungere i suoi obiettivi dannosi. Oltre a questo meccanismo di persistenza, il codice analizzato presenta anche altri comportamenti dannosi, come la registrazione dei movimenti del mouse e la crittografia dei file.

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

```
.text: 00401010    push eax
```

Questa istruzione salva il valore del registro EAX nello stack

```
.text: 00401014    push ebx
```

Questa istruzione salva il valore del registro EBX nello stack

```
.text: 00401018    push ecx
```

Questa istruzione salva il valore del registro ECX nello stack

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

Questa istruzione spinge il valore costante `WH_MOUSE` nello stack

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

Questa istruzione chiama la funzione `SetWindowsHook() ed installa un hook sul sistema, specificando il tipo di hook da installare e la funzione di callback da eseguire quando si verifica l'evento.`

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

Questa istruzione esegue un'operazione di XOR tra il registro ECX e se stesso

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

Questa istruzione copia il valore contenuto all'indirizzo puntato da EDI nel registro ECX

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

Questa istruzione copia il valore contenuto nell'indirizzo puntato da ESI nel registro EDX

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

Questa istruzione salva il valore del registro ECX nello stack

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

Questa istruzione salva il valore del registro EDX nello stack

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

Questa istruzione chiama la funzione `CopyFile()` e copia un file da una posizione all'altra