

## COSTRUTTI C – ASSEMBLY x86

La figura seguente mostra un estratto di codice di un Malware

```
.text:00401000      push    ebp |
.text:00401001      mov     ebp, esp
.text:00401003      push    ecx
.text:00401004      push    0          ; dwReserved
.text:00401006      push    0          ; lpdwFlags
.text:00401008      call   ds:InternetGetConnectedState
.text:0040100E      mov     [ebp+var_4], eax
.text:00401011      cmp     [ebp+var_4], 0
.text:00401015      jz      short loc_40102B
.text:00401017      push    offset aSuccessInterne ; "Success: Internet Connection\n"
.text:0040101C      call   sub_40105F
.text:00401021      add     esp, 4
.text:00401024      mov     eax, 1
.text:00401029      jmp     short loc_40103A
.text:0040102B ; -----
.text:0040102B
```

## COSRTUTTI IDENTIFICATI

- IF

```
# Cmp [ebp+var_4], 0 - (IF VAR = 0) - jz --> 40102B (dopo il tratteggio)
# Jz short loc_40102B -
...
...
# 0040102B ; -----
```

- ELSE

```
# Push offset aSuccesInterne ; "Succes: Internet Connection\n" - Else VAR != 0
segue le istruzioni dopo
# Call sub_40105F
# Add esp, 4
# Mov eax, 1
# Jmp shot loc_40103A
```

## IPOSTESI

```
* .text:00401000      push    ebp |
* .text:00401001      mov     ebp, esp
* .text:00401003      push    ecx
* .text:00401004      push    0          ; dwReserved
* .text:00401006      push    0          ; lpdwFlags
* .text:00401008      call   ds:InternetGetConnectedState
* .text:0040100E      mov     [ebp+var_4], eax
* .text:00401011      cmp     [ebp+var_4], 0
* .text:00401015      jz      short loc_40102B
* .text:00401017      push    offset aSuccessInterne ; "Success: Internet Connection\n"
* .text:0040101C      call   sub_40105F
* .text:00401021      add     esp, 4
* .text:00401024      mov     eax, 1
* .text:00401029      jmp     short loc_40103A
* .text:0040102B      ; -----
* .text:0040102B
```

Il codice, estratto da un malware, ha come compito quello di vedere se è presente una connessione ad internet. Questo lo possiamo identificare anche dai commenti presenti.

Dopo “**cmp [ebp+var\_4], 0**” e “**Jz short loc\_40102B**” possiamo identificare la condizione per il quale si può effettuare o meno un **jump zero** (jz). Se questa condizione non si dovesse verificare allora si proseguirebbe con le azioni presenti in seguito.

Push offset aSuccessInterne insieme al comando call (subito di seguito) invocano una sub routine (sub\_40105F) dell’argomento che segue la push offset. In questo caso si identifica la “posizione” della memoria non il valore presente al suo interno.

<https://groups.google.com/g/comp.lang.asm.x86/c/oT3jFGitjvQ>

## BONUS

**Push ebp** - passare i paramentri ebp in cima allo stack

**Mov ebp, esp** - copia il valore della sorgente (esp), nel destinatario (ebp)

**Push ecx** - passare i paramentri ecx in cima allo stack

**Push 0 ; dwReserved** - passare il valore dei paramentri (valore0) in cima allo stack - ; commento

**Push 0; lpdwFlags** - passare il valore dei paramentri (valore0) in cima allo stack - ; commento

**Call ds:InternetGetConnectedState** – chiamata di funzione

**Mov [ebp+var\_4], - eax** - copia il valore della sorgente (eax), nel destinatario (ebp+var\_4)

**Cmp [ebp+var\_4], -** mette a confronto destinatario e sorgente per restituire un valore binario

**Jz short loc\_40102B** - (salta alla locazione di memoria specificata se ZF=1) – salta alla locazione di memoria se ZF = 1 riferito al cmp precedente

**Push offset aSuccessInterne ; “Success: Internet Connection\n”** – passa l’alloggio di memoria aSuccessInterne in cima allo stack

**Add esp, 4** -= somma il valore della sorgente (4) con quello del destinatario (esp)

**Mov eax, 1** - copia il valore della sorgente (1), nel destinatario (eax)

**Jmp shot loc\_40103A** – fa un salto all’istruzione nella locazione data (40103A)

**40102B -----** - istruzione dove il jz “salta”