# S10_L4
## Unit 3 - CS0424

MATTEO BELTRAMI MARZOLINI
CYBEREAGLES

## Giorno 4 – Costrutti C - Assembly x86

### TRACCIA

La figura seguente mostra un estratto del codice di un malware. Identificare i costrutti noti Esercizio Linguaggio Assembly visti durante la lezione teorica.

```
.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
```

Provate ad ipotizzare che funzionalità è implementata nel codice assembly.

Hint :

La funzione ***internetgetconnectedstate*** prende in input 3 parametri e permette di controllare se una macchina ha accesso ad Internet.

Consegna:

1. Identificare i costrutti noti (e s. while, for, if, switch, ecc.)

2. Ipotizzare la funzionalità – esecuzione ad alto livello

3. BONUS: studiare e spiegare ogni singola riga di codice

# SVOLGIMENTO

## 1. Identificare i costrutti noti

Possiamo identificare 4 costrutti:

- ***call*** (*chiamata di funzione*)

  E' utilizzata per chiamare una funzione. Durante questa operazione, l'indirizzo di ritorno (cioè, l'indirizzo dell'istruzione successiva alla chiamata) viene salvato sullo stack, e l'esecuzione del programma continua dalla funzione chiamata.

  Esempio:

  ***.text:00401008 call ds:InternetGetConnectedState***

  Qui, `InternetGetConnectedState` è una funzione di sistema che verifica lo stato della connessione a Internet. L'istruzione `call` trasferisce il controllo a questa funzione.

- ***cmp*** (*confronto*)

  Confronta due operandi senza modificarli, impostando i flag appropriati nel registro di stato in base al risultato. I flag settati includono, tra gli altri, lo Zero Flag (ZF), che è particolarmente importante per determinare se gli operandi confrontati sono uguali.

  Esempio:

***.text:00401011 cmp [ebp+var_4], 0***

Questa istruzione confronta il valore memorizzato in `[ebp+var_4]` con `0`. Se i due valori sono uguali, il flag ZF sarà impostato a 1.

- ***jz*** (salto condizionale, jump if zero)

  E' un'istruzione di salto condizionale che trasferisce il controllo a un'altra parte del programma se il flag ZF (Zero Flag) è impostato a 1, indicante che il risultato dell'ultima operazione era zero. In altre parole, `jz` effettua un salto se l'ultimo confronto (`cmp`) ha trovato gli operandi uguali.

  Esempio:

  ***.text:00401015 jz short loc_40102B***

  Se il confronto tra `[ebp+var_4]` e `0` risulta uguale (cioè ZF = 1), il controllo viene trasferito all'etichetta `loc_40102B`.

- ***jmp*** (salto incondizionato)

  E' un'istruzione di salto incondizionato che trasferisce il controllo del flusso del programma all'indirizzo specificato, senza verificare alcuna condizione.

  Esempio:

  ***.text:00401029 jmp short loc_40103A***

  Questo `jmp` trasferisce il controllo all'etichetta `loc_40103A` indipendentemente dallo stato dei flag o di altre condizioni. È spesso

utilizzato per evitare l'esecuzione di istruzioni indesiderate o per organizzare il flusso del programma in modo chiaro.

Questi quattro costrutti insieme formano il meccanismo di controllo del flusso del programma. **`Call`** permette di eseguire funzioni, **`cmp`** confronta i valori, **`jz`** effettua salti condizionali in base al risultato del confronto, e **`jmp`** gestisce i salti incondizionati. In questo modo, il programma può prendere decisioni e dirigere il flusso dell'esecuzione in base a condizioni logiche e risultati operativi.

## 2. Ipotizzare la funzionalità

La funzione `InternetGetConnectedState` viene chiamata per verificare lo stato della connessione Internet. Il codice sembra verificare se la macchina ha accesso a Internet e, in caso positivo, esegue alcune istruzioni. Quindi, possiamo ipotizzare che il codice verifichi la connessione a Internet e, se la connessione è presente, stampa un messaggio di successo.

Se è connesso, stampa un messaggio di successo e imposta il valore di ritorno a 1. Se non è connesso, imposta il valore di ritorno a `0`.

1. **Inizio del frame dello stack e salvataggio dei registri:**

*.text:00401000 push ebp ;*

*.text:00401001 mov ebp, esp ;*

*.text:00401003 push ecx ;*

Queste istruzioni salvano il contesto corrente del processo (ebp ed ecx) sullo stack e inizializza un nuovo frame dello stack.

2. **Preparazione dei parametri per `InternetGetConnectedState`:**

*.text:00401004 push 0 ;*

*.text:00401006 push 0 ;*

Vengono passati due parametri a `InternetGetConnectedState`, entrambi con valore `0`.

3. **Chiamata a `InternetGetConnectedState`:**

*.text:00401008 call ds:InternetGetConnectedState ;*

Questa chiamata verifica lo stato della connessione a Internet.

4. **Gestione del risultato della chiamata:**

*.text:0040100E mov [ebp+var_4], eax ;*

*.text:00401011 cmp [ebp+var_4], 0 ;*

*.text:00401015 jz short loc_40102B ;*

Il risultato della chiamata (ritornato in `eax`) viene memorizzato in `[ebp+var_4]`. Se il risultato è zero, significa che non c'è connessione a Internet, e il codice salta all'etichetta `loc_40102B`.

5. **Caso di connessione Internet presente:**

*.text:00401017 push offset aSuccessInterne ;*

*.text:0040101C call sub_40105F ;*

*.text:00401021 add esp, 4 ;*

*.text:00401024 mov eax, 1 ;*

*.text:00401029 jmp short loc_40103A ;*

Se c'è connessione a Internet, viene chiamata una subroutine (`sub_40105F`) per stampare il messaggio "Success: Internet Connection\n" e `eax` viene impostato a `1` per indicare successo.

6. **Caso di assenza di connessione Internet:**

*.text:0040102B loc_40102B: ;*

*.text:0040102B mov eax, 0 ;*

*.text:00401030 jmp short loc_40103A ;*

Se non c'è connessione, `eax` viene impostato a `0` per indicare fallimento.

## BONUS

### 3. Studiare e spiegare ogni singola riga di codice

.text:00401000 push ebp; **Salva il valore del base pointer (ebp) sullo stack**

.text:00401001 mov ebp, esp; **Copia il valore di esp (stack pointer) nel base pointer (ebp)**

.text:00401003 push ecx; **Salva il valore di ecx sullo stack**

.text:00401004 push 0; **Push del parametro dwReserved (0) sullo stack**

.text:00401006 push 0; **Push del parametro lpdwFlags (0) sullo stack**

.text:00401008 call ds:InternetGetConnectedState; **Chiamata alla funzione InternetGetConnectedState**

.text:0040100E mov [ebp+var_4], eax; **Memorizza il valore di ritorno di InternetGetConnectedState in [ebp+var_4]**

.text:00401011 cmp [ebp+var_4], 0; **Confronta il valore memorizzato in [ebp+var_4] con 0**

.text:00401015 jz short loc_40102B; **Se il confronto è zero, salta a loc_40102B**

.text:00401017 push offset aSuccessInterne; **Push dell'offset della stringa "Success: Internet Connection\n" sullo stack**

.text:0040101C call sub_40105F; **Chiamata alla subroutine per stampare la stringa**

.text:00401021 add esp, 4; **Ripristina lo stack (bilancia la push)**

.text:00401024 mov eax, 1; **Imposta il valore di ritorno di eax a 1 (successo)**

.text:00401029 jmp short loc_40103A; **Salta a loc_40103A per uscire**

.text:0040102B loc_40102B: ; **Etichetta per il salto condizionale**

.text:0040102B mov eax, 0 ; **Imposta il valore di ritorno di eax a 0 (fallimento)**

.text:00401030 jmp short loc_40103A ; **Salta a loc_40103A per uscire**