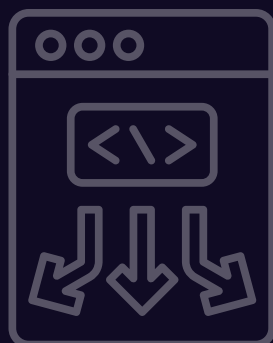




S10-L4



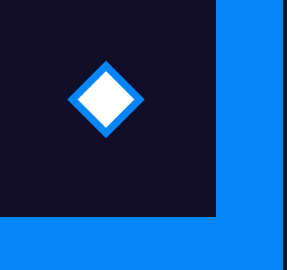
# Traccia:

La figura seguente mostra un estratto del codice di un malware. Identificare i costrutti noti 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.

1. Identificare i costrutti noti (es. while, for, if, switch, ecc.)
2. Ipotizzare la funzionalità –esecuzione ad alto livello
3. BONUS: studiare e spiegare ogni singola riga di codice



```
push    ebp
```


in questa parte di codice il valore contenuto nel registro ebp viene salvato temporaneamente nello stack

```
mov     ebp, esp
```

in questa parte di codice il valore contenuto nel registro esp viene copiato nel registro ebp

```
push    ecx
```

in questa parte di codice il valore contenuto nel registro ecx viene salvato temporaneamente nello stack





```
push 0
push 0
```


in questa parte di codice i valori, in questo caso 0 e 0, vengono salvati temporaneamente nello stack, in modo da essere argomenti per una chiamata a funzione successiva

```
call ds:InternetGetConnectedState
```

in questa parte di codice l'istruzione effettua una chiamata di funzione a InternetGetConnectedState, i precedenti valori 0 potrebbero essere gli argomenti per questa chiamata di funzione

```
mov [ebp+var_4], eax
```

in questa parte di codice il valore contenuto nel registro eax viene copiato alla variabile locale ebp+var\_4



```
cmp [ebp+var_4], 0
```

in questa parte di codice il valore della variabile locale viene confrontato con 0



```
jz short loc_40102B
```

in questa parte di codice se ebp+var\_4 è uguale a zero viene effettuato un salto condizionale all'indirizzo loc\_40102B

```
push offset aSuccessInterne ; "Success: Internet Connection\n"  
call sub_40105F
```

in questa parte di codice viene inserito nello stack un puntatore a una stringa "Success: Internet Connection\n", quindi viene effettuata una chiamata a una funzione sub\_40105F



```
add     esp, 4
```


in questa parte di codice il valore 4 viene aggiunto al valore del registro esp

```
mov     eax, 1
```

in questa parte di codice il valore del registro eax viene impostato a 1

```
jmp     short loc_40103A
```

in questa parte di codice viene eseguito un salto non condizionale all'indirizzo loc\_40103A



Potremmo ipotizzare che questo codice in C serva a effettuare un check della connessione Internet, se la risposta è 0 stamperà successo.