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
```

Consegna:

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

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

Nel codice assembly mostrato, i seguenti costrutti possono essere identificati:

if: rappresentato dalla sequenza `cmp` e `jz`, che condiziona l'esecuzione del codice.

```
cmp [ebp+var_4], 0
jz short loc_40102B
```

-

chiamata di funzione: rappresentata dall'istruzione `call`.

```
call ds:InternetGetConnectedState
call sub_40105F
```

2. Ipotizzare la funzionalità – esecuzione ad alto livello

Il codice sembra essere una parte di un malware che verifica la connessione Internet e stampa un messaggio di successo se la connessione è attiva. A livello alto, la funzionalità può essere descritta come:

1. Configura lo stack frame per la funzione.
2. Salva i registri utilizzati (`ecx`).
3. Chiama la funzione API `InternetGetConnectedState` per verificare se il computer è connesso a Internet.
4. Se la connessione è attiva (`eax` != 0), stampa il messaggio "Success: Internet Connection\n".
5. Pulire lo stack e preparare il valore di ritorno.

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

Ho già fornito una spiegazione riga per riga nel mio messaggio precedente. Qui lo ripropongo in dettaglio:

```
.text:00401000 push ebp
```

- Salva il valore corrente di `ebp` sullo stack per preservare il contesto del chiamante.

```
.text:00401001 mov ebp, esp
```

- Imposta `ebp` per puntare alla base dello stack frame corrente. Questo è il setup standard di prologo di funzione.

```
.text:00401003 push ecx
```

- Salva il valore corrente di `ecx` sullo stack. `ecx` è un registro volatile e il suo valore potrebbe essere cambiato durante la chiamata di funzione.

```
.text:00401004 push 0
```

- Spinge 0 sullo stack come primo argomento (`dwReserved`) per `InternetGetConnectedState`.

```
.text:00401006 push 0
```

- Spinge 0 sullo stack come secondo argomento (`lpdwFlags`) per `InternetGetConnectedState`.

```
.text:00401008 call ds:InternetGetConnectedState
```

- Chiama la funzione `InternetGetConnectedState` tramite la tabella di indirizzi delle funzioni (import address table).

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

- Sposta il risultato della chiamata a `InternetGetConnectedState` (contenuto in `eax`) nella variabile locale `var_4` (situata all'indirizzo `ebp+var_4`).

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

- Confronta il valore di `var_4` con 0 per verificare se la connessione Internet è attiva.

```
.text:00401015 jz short loc_40102B
```

- Se `var_4` è 0 (nessuna connessione Internet), salta alla posizione `loc_40102B`, che probabilmente gestisce il caso di errore o di connessione assente.

```
.text:00401017 push offset aSuccessInterne ; "Success: Internet
Connection\n"
```

- Spinge l'indirizzo del messaggio di successo sullo stack. `aSuccessInterne` è una stringa che contiene "Success: Internet Connection\n".

```
.text:0040101C call sub_40105F
```

- Chiama una funzione (`sub_40105F`) per gestire il messaggio di successo. Questa funzione probabilmente visualizza o registra il messaggio.

```
.text:00401021 add esp, 4
```

- Pulisce lo stack aumentando `esp` di 4, rimuovendo l'argomento del messaggio spinto in precedenza.

```
.text:00401024 mov eax, 1
```

- Imposta `eax` a 1, probabilmente per indicare un esito positivo.

```
.text:00401029 jmp short loc_40103A
```

- Salta alla posizione `loc_40103A`, continuando l'esecuzione del codice successivo, che potrebbe gestire ulteriori operazioni o terminare la funzione.

```
.text:0040102B ;
-------------------------------------------------------
```

- Questa sezione rappresenta un punto di salto per la gestione del caso di connessione assente, anche se il codice specifico per questo caso non è incluso nell'estratto.