COSTRUTTI C – ASSEMBLY X86

S10/L4

ELEONORA VIOLA

TABLE OF CONTENT

INTRODUCTION

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

Consegna:
1. Id e ntificare 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

PROCEDURE

1. Identificare i costrutti noti:
Nel codice assembly possiamo identificare i seguenti costrutti:

- **Funzioni**: Vediamo chiaramente l'uso delle funzioni. La funzione InternetGetConnectedState viene chiamata per verificare se la macchina è connessa a Internet. In generale, il codice utilizza la chiamata di funzione (call) per invocare queste funzioni.
- **Condizionale (if)**: Il codice utilizza un'istruzione condizionale per determinare il flusso del programma. Dopo aver ottenuto il risultato dalla funzione InternetGetConnectedState, il valore risultante viene confrontato con zero usando l'istruzione cmp. Se il risultato è zero, il programma salta a una parte specifica del codice usando l'istruzione jz (salto se zero). Questo comportamento è simile a un'istruzione "if" nei linguaggi di programmazione ad alto livello.
- **Nessun ciclo (loop):** In questo codice particolare, non vediamo alcun ciclo esplicito come while o for. Il flusso è lineare con salti condizionali basati sui risultati delle funzioni.

Quindi, il codice è principalmente composto da chiamate a funzioni e da istruzioni condizionali per determinare cosa fare in base ai risultati di queste funzioni. Questi sono i costrutti chiave che possiamo riconoscere.

2. Ipotizzare la funzionalità:
Il malware invoca la funzione InternetGetConnectedState per verificare lo stato della connessione Internet. Utilizzando una condizione "if", controlla il valore restituito da questa funzione. Se il valore restituito è diverso da 0, significa che esiste una connessione attiva.

.

PROCEDURE

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

Indirizzo	Codice Assembly	Descrizione
.text:00401000	`push ebp`	Salva il valore di `ebp` sullo stack.
.text:00401001	`mov ebp, esp`	Imposta il registro base `ebp` al valore corrente del puntatore di stack `esp`.
.text:00401003	`push ecx`	Salva il registro `ecx` sullo stack.
.text:00401004	`push 0`	Passa 0 come parametro `dwReserved` alla funzione `InternetGetConnectedState`.
.text:00401006	`push 0`	Passa 0 come parametro `lpdwFlags` alla funzione `InternetGetConnectedState`.
.text:00401008	`call ds:InternetGetConnectedState`	Chiama la funzione per verificare la connessione a Internet.
.text:0040100E	`mov [ebp+var_4], eax`	Salva il risultato della funzione (connesso o no) in `var_4`.
.text:00401011	`cmp [ebp+var_4], 0`	Confronta il risultato con 0 (non connesso).
.text:00401015	`jz short loc_40102B`	Se il risultato è zero (non connesso), salta all'indirizzo `loc_40102B`.
.text:00401017	`push offset aSuccessInterne`	Passa l'offset del messaggio di successo come parametro alla funzione successiva.
.text:0040101C	`call sub_40105F`	Chiama una funzione (probabilmente per stampare il messaggio di successo).
.text:00401021	`add esp, 4`	Ripulisce lo stack dai parametri passati alla funzione.
.text:00401024	`mov eax, 1`	Imposta il registro `eax` a 1 (indicando successo).
.text:00401029	`jmp short loc_40103A`	Salta all'indirizzo `loc_40103A` (fine del codice).
.text:0040102B	`xor eax, eax`	Se non connesso, imposta il registro `eax` a 0 (indicando fallimento).
.text:0040102D	`jmp short loc_40103A`	Salta all'indirizzo `loc_40103A` (fine del codice).