CS0424IT — ESERCITAZIONE S10L4
LINGUAGGIO ASSEMBLY PT.2: COSTRUTTI C

Simone La Porta

GitHub

---

1 agosto 2024

INDICE

1 TRACCIA

Di seguito un estratto del 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 ; ---------------------------------------------------------------
```

Traccia dell'esercizio:

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.

2 SVOLGIMENTO

2.1 *Identificazione dei costrutti noti in Linguaggio C*

Nell'estratto di codice mostrato, possiamo identificare i seguenti costrutti in linguaggio C equivalenti alle operazioni svolte dal codice assembly:

- **Chiamata di funzione**

  L'istruzione `call ds:InternetGetConnectedState` in assembly è equivalente a una chiamata di funzione in C come `InternetGetConnectedState(&lpdwFlags, dwReserved)`.

- **If**

  L'istruzione `cmp [ebp+var_4], 0` seguita dall'istruzione `jz` (jump if zero) è equivalente a un controllo `if` in C: `if (var_4 == 0)`.

- **Assegnazione**

  L'istruzione `mov [ebp+var_4], eax` in assembly è equivalente a un'assegnazione in C: `var_4 = eax`.

- **Push e Pop**

  Le istruzioni `push` e `pop` in assembly vengono usate per salvare e ripristinare i registri. In C, queste operazioni sono gestite automaticamente dal compilatore tramite l'uso dello stack frame.

- **Salto incondizionato**

  L'istruzione `jmp` in assembly è equivalente a un salto incondizionato in C usando `goto`.

2.2 *Ipotizzare la funzionalità*

Questo codice sembra verificare se esiste una connessione internet attiva. Utilizza l'API di Windows `InternetGetConnectedState` per determinare lo stato della connessione.

2.3 *Analisi dettagliata del Codice Assembly*

Di seguito è riportata un'analisi dettagliata di ogni singola riga di codice assembly:

- `push ebp`

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

- `mov ebp, esp`

  Imposta `ebp` per puntare alla base dello stack frame corrente, stabilendo un nuovo frame dello stack.

- `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.

- `push 0`

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

- `push 0`

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

- `call ds:InternetGetConnectedState`

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

- `mov [ebp+var_4], eax`

  Salva il risultato della chiamata a `InternetGetConnectedState` (contenuto in `eax`) nella variabile locale `var_4`.

- `cmp [ebp+var_4], 0`

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

- `jz short loc_40102B`

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

- `push offset aSuccessInterne`

  Pusha l'offset della stringa "Success: Internet Connection\n" sullo stack. aSuccessInterne è una stringa che contiene "Success: Internet Connection\n".

- `call sub_40105F`

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

- `add esp, 4`

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

- `mov eax, 1`

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

- `jmp short loc_40103A`

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

3 CONVERSIONE IN LINGUAGGIO C

Di seguito è riportata una possibile conversione del codice in linguaggio C:

```c
#include <windows.h>
#include <stdio.h>

int main() {
    // Dichiarazione e inizializzazione della variabile dwReserved
    DWORD dwReserved = 0;

    // Dichiarazione e inizializzazione della variabile lpdwFlags
    DWORD lpdwFlags = 0;

    // Dichiarazione della variabile booleana isConnected
    BOOL isConnected;

    // Chiamata per verificare la connessione internet
    isConnected = InternetGetConnectedState(&lpdwFlags, dwReserved);

    // Se connesso, stampa il messaggio di successo
    if (isConnected) {
        printf("Success: Internet Connection\n");
    }

    // Restituisce 1 se connesso, altrimenti 0
    return isConnected ? 1 : 0;
}
```

Descrizione del codice nel dettaglio:

- `#include <windows.h>`

  Include l'header file necessario per utilizzare le API di Windows.

- `#include <stdio.h>`

  Include l'header file per le funzioni di input/output standard.

- `DWORD dwReserved = 0;`

  Dichiara e inizializza la variabile `dwReserved` a 0. La variabile `dwReserved` è utilizzata come parametro nella chiamata alla funzione InternetGetConnectedState. La funzione richiede che questo parametro sia passato, ma per l'attuale implementazione dell'API di Windows, esso deve essere impostato a zero. In sintesi, il suo utilizzo è riservato per futuri sviluppi o funzionalità dell'API, e al momento non ha un impatto diretto sull'esecuzione della funzione. Questo parametro è quindi principalmente un placeholder per eventuali future estensioni della funzionalità.

- `DWORD lpdwFlags = 0;`

  Dichiara e inizializza la variabile `lpdwFlags` a 0. La variabile `lpdwFlags` è un puntatore a una variabile DWORD che riceve vari flag che descrivono lo stato della connessione. Questi flag forniscono informazioni dettagliate sul tipo di connessione Internet attualmente in uso (connessione tramite modem, LAN, proxy, offline, RAS).

- `BOOL isConnected;`

  Dichiara la variabile `isConnected` che verrà utilizzata per memorizzare lo stato della connessione.

- `isConnected = InternetGetConnectedState(&lpdwFlags, dwReserved);`

  Chiama la funzione `InternetGetConnectedState` per verificare lo stato della connessione internet e salva il risultato in `isConnected`.

- `if (isConnected)  printf(\Success: Internet Connection\n");`

  Se `isConnected` è vero, stampa il messaggio "Success: Internet Connection\n".

- `return isConnected ? 1 : 0;`

  Restituisce 1 se `isConnected` è vero, altrimenti restituisce 0.

4 CONCLUSIONI

In conclusione, l'analisi dettagliata del codice assembly e la sua conversione in linguaggio C, ha permesso di comprendere a fondo il funzionamento del malware. Il codice assembly analizzato utilizza vari costrutti comuni come chiamate di funzione, condizioni `if`, assegnazioni e salti incondizionati per verificare lo stato della connessione Internet tramite l'API di Windows `InternetGetConnectedState`. La conversione in linguaggio C ha reso più chiaro come questi costrutti vengano utilizzati per determinare se il sistema è connesso a Internet e per eseguire azioni specifiche in base a tale stato.

Le variabili `dwReserved` e `lpdwFlags` svolgono ruoli importanti nel fornire dettagli sulla connessione Internet. `dwReserved` è un parametro riservato che deve essere impostato a zero, mentre `lpdwFlags` fornisce informazioni dettagliate sul tipo di connessione Internet, permettendo al software di reagire in modo appropriato alle diverse condizioni di rete.

Questo esercizio ha evidenziato l'importanza di comprendere sia il codice assembly che il linguaggio di alto livello per analizzare e interpretare correttamente il comportamento del software, specialmente nel contesto della sicurezza informatica e dell'analisi dei malware.