Assembly - ingegneria inversa

Traccia
La figura seguente mostra un estratto del codice di un malware. Identificare i costrutti noti visti durante la lezione teorica. Provate ad ipotizzare che funzionalità è implementata nel codice assembly.

La figura mostrata rappresenta una sequenza di istruzioni in linguaggio Assembly per un'architettura x86. Questo codice sembra essere parte di un malware e mostra il flusso per verificare una connessione Internet e possibilmente eseguire azioni basate su quella verifica. Ecco una descrizione delle istruzioni e una ipotetica funzionalità implementata:

- **push ebp** e **mov ebp, esp**: Queste istruzioni sono tipicamente utilizzate all'inizio di una funzione per impostare il frame dello stack. **ebp** è il base pointer che stabilisce un punto di riferimento fisso nello stack per accedere ai parametri e alle variabili locali della funzione.
- **push ecx**: Questa istruzione salva il valore corrente del registro **ecx** nello stack. **ecx** è spesso usato come contatore nei loop o come registro generico.
- **push 0**: Viene messo lo zero nello stack, potrebbe essere un argomento per la funzione che verrà chiamata successivamente.
- **push 0**: Un altro zero viene messo nello stack, probabilmente un altro argomento per la funzione imminente.
- **call ds:InternetGetConnectedState**: Chiamata alla funzione Windows API **InternetGetConnectedState**, che verifica lo stato della connessione Internet. I due zeri precedenti sono probabilmente passati come parametri a questa funzione, che corrispondono a **lpdwFlags** e **dwReserved** secondo la documentazione della funzione.
- **mov [ebp+var_4], eax**: Salva il risultato della funzione **InternetGetConnectedState** in una variabile locale sullo stack (**var_4** è un offset rispetto a **ebp**).
- **cmp [ebp+var_4], 0**: Confronta il valore appena salvato con zero per determinare se c'è una connessione Internet.
- **jz short loc_40102B**: Salta a un'altra locazione nel codice se il risultato del confronto è zero (cioè, se non c'è connessione Internet).
- **push offset aSuccessInterne**: Impila l'indirizzo di una stringa, presumibilmente per usarlo come argomento per una chiamata di funzione successiva, probabilmente per stampare o loggare il messaggio "Success: Internet Connection\n".

- **call sub_40105F**: Chiama un'altra subroutine, che potrebbe essere responsabile della stampa o della gestione del messaggio di successo della connessione Internet.
- **add esp, 4**: Pulisce lo stack dopo la chiamata alla funzione, rimuovendo l'argomento passato.
- **mov eax, 1**: Sposta 1 nel registro **eax**, che potrebbe essere utilizzato per impostare uno stato di successo o per indicare che l'azione successiva deve essere eseguita poiché la connessione Internet è attiva.
- **jnp short loc_40103A**: Salta a un'altra locazione se il flag di parità (PF) non è impostato. Questo potrebbe essere legato al controllo dello stato di errore o alla logica di flusso del programma.

In sintesi, questo frammento di codice sembra essere una funzione che verifica la presenza di una connessione Internet attiva e, in base al risultato, esegue una serie di azioni: se c'è connessione, procede con l'esecuzione di una subroutine e imposta uno stato di successo; altrimenti, il flusso del programma salta a un'altra sezione del codice, presumibilmente per gestire lo stato di "nessuna connessione". Questo tipo di controllo è comune nei malware per determinare se possono raggiungere il loro server di comando e controllo o eseguire azioni basate sulla rete.