Report Dettagliato sull'Esercitazione di Analisi del Codice Assembly

Introduzione

Questa esercitazione si concentra sull'analisi di un estratto di codice assembly proveniente da un malware. Il compito principale è identificare i costrutti noti e ipotizzare quale funzionalità viene implementata nel codice. Utilizzando le conoscenze acquisite durante le lezioni teoriche, ho analizzato dettagliatamente il codice assembly per comprenderne il funzionamento e fornire una spiegazione esaustiva.

Analisi del Codice Assembly

Creazione dello Stack

1. **push ebp**
    - Questa istruzione salva il valore corrente del base pointer (ebp) nello stack. È utilizzata per preservare il contesto della funzione chiamante.
2. **mov ebp, esp**
    - Imposta il base pointer (ebp) all'inizio del frame corrente dello stack, stabilendo un nuovo frame di stack per la funzione.
3. **push ecx**
    - Salva il valore del registro ecx sullo stack, conservando il valore del registro per l'uso successivo.

Chiamata di Funzione

4. **push 0** (x2)
    - Due istruzioni consecutive che spingono il valore 0 sullo stack. Questi rappresentano i parametri dwReserved e lpdwFlags per la chiamata alla funzione InternetGetConnectedState.
5. **call ds**
    - Questa istruzione chiama la funzione InternetGetConnectedState, che verifica lo stato della connessione Internet. I parametri sono passati tramite lo stack.

Gestione del Ritorno dalla Funzione

6. **mov [ebp+var_4], eax**
    - Memorizza il valore di ritorno della funzione InternetGetConnectedState in una variabile locale, situata a [ebp+var_4].

Costrutto IF

7. **cmp [ebp+var_4], 0**
    - Confronta il valore memorizzato in [ebp+var_4] con 0, verificando se la connessione a Internet è attiva (valore diverso da 0).
8. **jz short loc_40102B**
    - Se il valore confrontato è zero (nessuna connessione Internet), salta all'etichetta loc_40102B.

Log e Ritorno di Valore

9. **push offset aSuccessInterne**
    - Spinge l'offset di una stringa (probabilmente "Success: Internet Connection\n") sullo stack, preparandosi per la chiamata di log o stampa.
10. **call sub_40105F**
    - Chiama una sottoroutine, probabilmente per stampare o loggare la stringa.
11. **add esp, 4**
    - Ripristina lo stack pointer (esp), aggiungendo 4 per bilanciare il push precedente.
12. **mov eax, 1**
    - Imposta il registro eax a 1, indicando successo.
13. **jmp short loc_40103A**
    - Salta all'etichetta loc_40103A.

Funzionalità Implementata

La funzionalità principale del codice assembly è verificare se la macchina ha accesso a Internet utilizzando la funzione InternetGetConnectedState. Se la connessione è presente, il codice logga un messaggio di successo e imposta il valore di ritorno a 1. Se non c'è connessione, il flusso di esecuzione salta una parte del codice (non mostrata nel frammento fornito).

Pseudocodice C

```
state = InternetGetConnectedState(0, 0, 0);
if (state != 0) {
   printf("Success: Internet Connection\n");
   return 1;
}
```

Costrutti Noti	
Sezione	**Descrizione**
Creazione dello Stack	Preparazione dello stack frame per la funzione.
Chiamata di Funzione	Passaggio dei parametri e chiamata a InternetGetConnectedState.
Gestione del Ritorno	Memorizzazione del valore di ritorno della funzione.
Costrutto IF	Confronto del valore di ritorno con 0 e salto condizionale.
Log e Ritorno di Valore	Log del messaggio di successo e impostazione del valore di ritorno a 1.

Conclusione

In conclusione, l'esercitazione ha permesso di comprendere come il codice assembly fornito verifichi la connessione a Internet e logghi un messaggio di successo se la connessione è attiva. Ho identificato i costrutti noti utilizzati nel codice, tra cui la gestione dello stack, le chiamate a funzioni esterne e i salti condizionali. La funzione InternetGetConnectedState è centrale per questa funzionalità, confermando l'importanza della verifica della connessione Internet in questo estratto di codice di malware.

Questa analisi dettagliata contribuisce alla comprensione dei meccanismi utilizzati dai malware per interagire con le funzionalità di rete, fornendo una base solida per ulteriori studi e attività di sicurezza informatica.