EPICODE-CS0124
S10/L4 - Pratica

Flaviano Sedici

Indice

Riferimenti e versioni

Responsabile del documento: Flaviano Sedici

Versionamento

Versione	Descrizione	Riferimento	Data
1.0	Redazione documento	Responsabile	28/03/2024

1. Traccia

Costrutti C -Assembly x86

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:

- Identificare i costrutti noti (es. while, for, if, switch, creazione/distruzione stack ecc.)
- Ipotizzare la funzionalità – esecuzione ad alto livello
- BONUS: studiare e spiegare ogni singola riga di codice

1.1. Definizioni

Assembly

Assembly è un linguaggio di programmazione a basso livello che rappresenta le istruzioni macchina utilizzando mnemonici leggibili dall'uomo. È specifico dell'architettura del processore e fornisce un controllo preciso sulle operazioni di un computer, ma richiede una conoscenza approfondita dell'architettura del processore.

1.2. Identificazione dei costrutti noti

1.2.1. Creazione di Stack

Il codice:

```
push    ebp
mov     ebp, esp
```

Creazione di uno stack

rappresenta la creazione di uno stack. Il comando **push** inserisce un nuovo stack sopra il pointer **EBP**, poi con il comando **mov** il valore del pointer **ESP** viene copiato nel pointer **EBP**. Non viene però creato un spazio preallocato tramite il comando **sub ESP, 0xYY**, che non consente la corretta creazione di ulteriori variabili locali.

1.2.2. Costrutto Call

Il codice:

```
push    ecx
push    0               ; dwReserved
push    0               ; lpdwFlags
call    ds:InternetGetConnectedState
```

Costrutto Call

Nelle prime tre righe, tramite il comando **push**, vengono caricati 3 valori che probabilmente serviranno per il corretto funzionamento della chiamata di funzione call.

La funzione sembra verificare lo **stato della connessione** ad internet della macchina target.

1.2.3. Costrutto IF/Else

Il codice:

```
cmp     [ebp+var_4], 0
jz      short loc_40102B
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40105F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

Costrutto IF

si riferisce ad un costrutto **IF/Else**, notiamo infatti come a seguito della funzione **cmp**, ci sia una funzione **jz** che salta alla riga indicata se e solo se ZF = 1 e quindi i due parametri confrontati siano uguali tra loro.

La riga a cui il programma salta è indicata all'interno del codice fornito ed è immediatamente successiva al codice riportato nell'immagine.

Ipotizziamo che per quanto riguarda le righe successive, le stesse facciano parte di un **else** che termina in con un **jump** ad una riga di codice successiva al testo della traccia e presumibilmente al codice che viene eseguito qualora il costrutto **IF** sia verificato.

1.2.4. Costrutto Call

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40105F
```

Costrutto IF

Il codice:

effettua un **push** di un valore stringa e richiama la **subroutine/funzione** fornendo il valore **stringa** alla riga precedente.

1.2.5. Modifica ESP

```
add     esp, 4
mov     eax, 1
```

Rimozione Stack

Il codice:

Il codice modifica il valore del puntatore **ESP** in modo che venga retrocesso di **4 byte** al fine di liberare spazio dopo che è stato utilizzato per memorizzare dati temporanei.

Al registro **EAX** viene poi assegnato il valore 1 con il comando **mov**.

1.2.6. Ipotesi di codice C

```
if (InternetGetConnectedState (x,dwReserved,lpdwFlags)==0) {
        esecuzione codice tra le linee 40102B e 401039
} else {
        esecuzione della subroutine_40105F
}
```

1.3. Ipotizzare la funzionalità

Il codice come ipotizzato nel paragrafo precedente, potrebbe effettuare delle operazioni sulla base della presenza o meno di connessione ad una rete del terminale target.