MALWARE ANALYSIS - LA MEMORIA ED IL LINGUAGGIO ASSEMBLY

TRACCIA

Nella lezione teorica del mattino, abbiamo visto i fondamenti del linguaggio Assembly. Dato il codice in Assembly per la CPU x86 allegato qui di seguito, identificare lo scopo di ogni istruzione, inserendo una descrizione per ogni riga di codice. Ricordate che i numeri nel formato 0xYY sono numeri esadecimali. Per convertirli in numeri decimali utilizzate pure un convertitore online, oppure la calcolatrice del vostro computer (per programmatori).

```
0×00001141 <+8>:     mov    EAX,0×20
0×00001148 <+15>:    mov    EDX,0×38
0×00001155 <+28>:    add    EAX,EDX
0×00001157 <+30>:    mov    EBP, EAX
0×0000115a <+33>:    cmp    EBP,0xa
0×0000115e <+37>:    jge     0×1176 <main+61>
0×0000116a <+49>:    mov    eax,0×0
0×0000116f <+54>:    call    0×1030 <printf@plt>
```

- mov EAX,0x20: Carica il valore esadecimale 0x20 (32 in decimale) nel registro EAX.
- mov EDX,0x38: Carica il valore esadecimale 0x38 (56 in decimale) nel registro EDX.
- add EAX,EDX: Somma i valori nei registri EAX ed EDX e salva il risultato in EAX.
- mov EBP, EAX: Copia il valore presente in EAX nel registro di base EBP.
- cmp EBP,0xa: Confronta il valore in EBP con il valore esadecimale 0xa (10 in decimale).
- jge 0x1176 <main+61>: Salta all'indirizzo 0x1176 (probabilmente una parte successiva del programma) se il confronto precedente è maggiore o uguale a 10.
- mov eax,0x0: Carica il valore 0 nel registro EAX.
- call 0x1030 <printf@plt>: Chiama la funzione printf, presumibilmente con il valore 0 come argomento. La specifica della chiamata a printf dipende dal contesto, e il risultato verrà stampato sulla console.

Quindi, in sintesi, questo codice sembra caricare due costanti, sommarle, confrontare il risultato con 10 e, a seconda del risultato del confronto, effettuare una chiamata alla funzione printf con un argomento di 0.