Analisi Codice Assembly

Analizzeremo di seguito un semplice estratto di codice Assembly, determinandone la funzione.

Precisiamo innanzitutto che la prima parte della stringa si riferisce all'indirizzo dell'allocazione della memoria sulla RAM, mentre il numero tra < e > indica il numero di bytes presenti dall'inizio della funzione.

1. 0x00001141 <+8>: mov EAX,0x20
   Assegna al registro EAX il valore 32, espresso in esadecimale.

2. 0x00001148 <+15>: mov EDX,0x38
   Assegna al registro EDX il valore 56, espresso in esadecimale.

3. 0x00001155 <+28>: add EAX,EDX
   Somma i valori contenuti nei due registri precedentemente visti (quindi dando come risultato 88), salvando il risultato nel registro EAX.

4. 0x00001157 <+30>: mov EBP, EAX
   Copia il valore del registro EAX nel registro EBP (Extended Base Pointer), indicante che da qui potrebbe partire un nuovo stack.

5. 0x0000115a <+33>: cmp EBP,0xa
   Effettua un paragone tra il valore contenuto nel registro EBP e il valore 10 (espresso qui in esadecimale); nelle istruzioni successive sarà specificato come avviene il paragone (se uguale, diverso, maggiore, ecc) e come si comporta il codice di conseguenza.

6. 0x0000115e <+37>: jge 0x1176 <main+61>
   Fornisce le istruzioni dopo il paragone: se il valore del registro EBP è maggiore (jge) di 10, viene richiesto di saltare alle istruzioni all'indirizzo 0x1176. In questo caso quindi, essendo EBP 88 e quindi maggiore di 10, ignoriamo tutte le istruzioni prima di quella indicata.

7. 0x0000116a <+49>: mov eax,0x0
   Sostituisce il valore nel registro eax con 0, espresso qui in esadecimale; di fatto, è come se andasse a cancellare i valori presenti nel registro. A causa dell'istruzione al punto 6, questa non viene eseguita.

8. 0x0000116f <+54>: call 0x1030 <printf@plt>
   Viene qui richiesto di richiamare una funzione printf all'indirizzo di memoria 0x1030, in particolare una voce del Procedure Linkage Table (PLT). A causa dell'istruzione al punto 6, questa non viene eseguita.