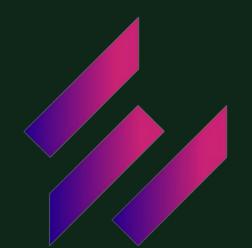


Malware analysis

510

Team





GIORNO 3

L3

La memoria ed il linguaggio Assembly



Traccia giorno 3

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.

01 00001141 <+8>: mov EAX,0x20

x00001148 <+15>: mov EDX,0x38

0x00001155 <+28>: add EAX,EDX

0x00001157 <+30>: mov EBP, EAX

0x0000115a <+33>: cmp EBP,0xa

0x0000115e <+37>: jge 0x1176 <main+61>

0x0000116a <+49>: mov eax,0x0

0x0000116f <+54>: call 0x1030 <printf@plt>

Linguaggio Assembly

La base del linguaggio Assembly sono le istruzioni. Nel linguaggio Assembly le istruzioni sono costituite da due parti:

- Un codice mnemonico, ovvero una parola che identifica l'istruzione da eseguire
- Uno o più operandi (per alcuni codici mnemonici, non è necessario l'operando come vedremo a breve), che identificano le variabili o la memoria oggetto dell'istruzione.

Un linguaggio assembly (detto anche linguaggio assemblativo o linguaggio assemblatore o semplicemente assembly) è un <u>linguaggio di programmazione</u> molto simile ai <u>linguaggi macchina</u>.

Si differenzia da questi ultimi principalmente per l'utilizzo di identificatori mnemonici, valori simbolici e altre caratteristiche che lo rendono più agevole da scrivere e leggere per gli esseri umani. Erroneamente viene spesso chiamato <u>assembler</u>, ma quest'ultimo termine identifica solo l'<u>applicativo</u> che converte i <u>programmi</u> scritti in assembly nell'equivalente in linguaggio macchina.

Conversione esadecimale -> decimale

Il sistema numerico esadecimale utilizza le cifre da 0 a 9 e le lettere da A a F (10-15). Per convertire un numero da base esadecimale a base decimale, è sufficiente moltiplicare la cifra o il carattere per 16 elevato alla posizione in cui si trova la cifra o il carattere proseguendo da destra verso sinistra.

Ecco un esempio pratico per chiarire il concetto:

3B ---->numero in base esadecimale $B = B \times 16^0 = 11 \times 16^0 = 11$

 $3 = 3 \times 16^{1} = 48$

11 + 48 = 59 ----> numero corrispondente in base decimale

Conversione decimale -> esadecimale

Il sistema numerico decimale utilizza le cifre da 0 a 9. Per convertire un numero da base decimale a base esadecimale, è sufficiente dividere il numero per 16, se il quoziente è diverso da zero va diviso nuovamente per 16 finché non sarà uguale a 0. A questo punto bisogna prendere il resto di ogni divisione in ordine opposto di come li abbiamo ottenuti.

Ecco un esempio pratico per chiarire il concetto:

59 ----->numero in base decimale

59:16 = 3 con resto 11

3:16=0 con resto 3

In riga scriviamo tutti i resti ottenuti, dall'ultimo al primo:

3 11

Sappiamo che in base esadecimale B = 11 e quindi il risultato finale sarà

3B ----> numero in base esadecimale

Descrizione codice

```
#Sposta il valore 0x20 (32 in decimale) nel registro EAX. Quindi EAX = 32
1.0x00001141 <+8>: mov EAX, 0x20
2.0x00001148 <+15>: mov EDX, 0x38
                                          #Sposta il valore 0x38 (56 in decimale) nel registro EDX. Quindi EDX = 56
3.0x00001155 <+28>: add EAX, EDX
                                          #Aggiunge il valore in EDX a EAX. EAX = EAX + EDX = 32 + 56 = 88 (0x58)
                                          #Sposta il contenuto di EAX in EBP. Quindi 88, che è il nuovo valore di EAX viene
4.0x00001157 <+30>: mov EBP, EAX
                                            spostato in EBP. EBP = 88
                                          #Confronta il valore in EBP con 0xA (10). EBP>0xA ---> 88 > 10
5.0x0000115a <+33>: cmp EBP, 0xA
6.0x0000115e <+37>: jge 0x1176 <main+61>
                                                 # jge, jump if greater or qual, Salta all'indirizzo 0x1176 se la relazione tra gli
                                                  operatori valutati risulta essere maggiore o uguale. Questo prende il nome
                                                  di salto condizionato. EBP > 0xA e quindi il salto viene effettuato
7.0x0000116a <+49>: mov EAX, 0x0
                                            #Sposta il valore 0 in EAX. Quindi EAX = 0
8.0x0000116f <+54>: call 0x1030 <printf@plt> #Chiama la funzione printf allocata all'indirizzo 0x1030, e in questo caso non
                                                  verrà eseguita perché è stato effettuato il salto direttamente all'indirizzo
                                                  0x1176
```

Zhongshi Liu



Mara Dello Russo

TEAM ALBA



Mario Marsicano



Luca Lenzi



Giovanni Sannino



Andre Vinicius

THANKYOU