

## Esercitazione W21D1 - Pratica 2

# Malware Analysis: Analisi dinamica basica

Fabio Benevento - 28/03/2024

## 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).

```
0x00001141 <+8>:  mov EAX,0x20
0x00001148 <+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>
```

## Implementazione

0x00001141 <+8>:  mov EAX,0x20	sposta il valore intero 32 nel registro EAX
0x00001148 <+15>:  mov EDX,0x38	sposta il valore 56 nel registro EDX
0x00001155 <+28>:  add EAX,EDX	somma EDX con EAX dove viene memorizzato il risultato (88)
0x00001157 <+30>:  mov EBP, EAX	sposta il contenuto di EAX (88) in EBP
0x0000115a <+33>:  cmp EBP,0xa	confronta EBP (88 = 0x58) con 0xa (10 decimale)
0x0000115e <+37>:  jge 0x1176 <main+61>	salta alla posizione (main + 61) se destinazione nella cmp precedente è maggiore della variabile di controllo. Il salto viene eseguito dato che la condizione risulta vera (88 > 10)

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

mette il valore 0 in eax

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

chiama la funzione printf presente