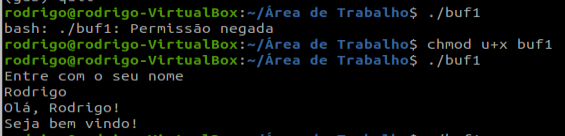
**Lab 3 - Computadores e Programação**





## Desafio 1)

1. **Qual o tamanho do buffer que lê o nome do usuário?**

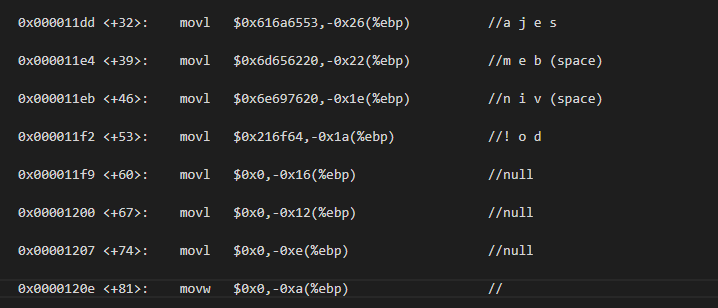
R: 100 bytes para ler o nome do usuário sem ter buffer overflow.

1. **Qual o tamanho do buffer de “Seja bem vindo!”?**

R: 29 bytes, chegando a 30, o programa aponta falha de segmentação.

1. **Por que o primeiro print só para depois de “Tchau, mundo cruel”?**

R: O print só para depois de “Tchau, mundo cruel”, pois apesar desse trecho ter ultrapassado o limite do buffer, a função gets() não conhece tal limite, então continua lendo até encontrar uma nova linha ou EOF.

****

## 

## Desafio 2)

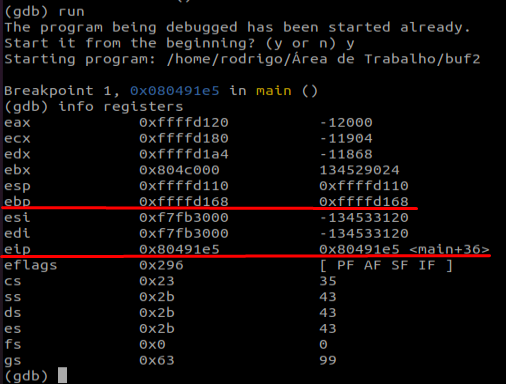
1. **Qual o endereço do código morto?**

R: 0x08049196

1. **Qual o tamanho do buffer de leitura?**

R: 63 bytes até dar falha de segmentação.

1. **Para onde aponta o ebp? E o eip de retorno da função que chama main?**

R:

1. **Quantos bytes deve-se escrever até conseguir sobrescrever eip?**

R: 80 bytes

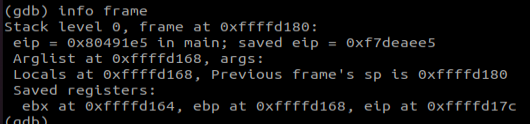
1. **Qual valor deverá receber para que eip aponte agora para codigo\_morto?**

R: python -c 'print "A"\*80 + "\x96\x91\x04\x08"' > attack.txt

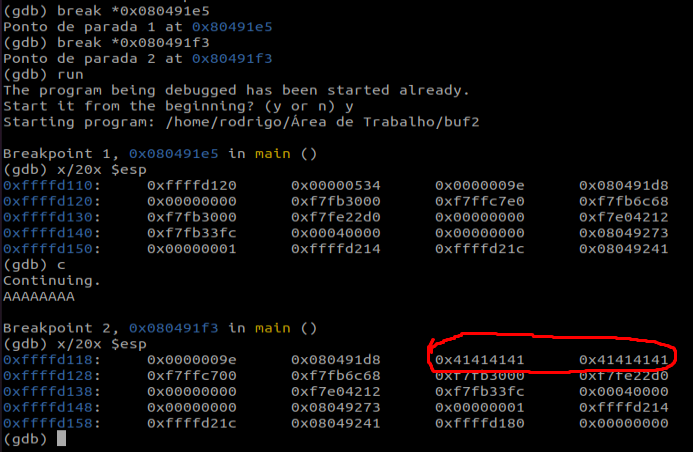
1. **Por que o programa não termina normalmente após ser explorado?**

R:

Encontrando o eip no stack antes de chamar gets():



Testando overwrite nos endereços:



Substituindo endereço em eip pelo endereço da função codigo\_morto():

