

# Aula 11 - 7 de maio de 2018

## Grupo 8

### Pergunta P1.1

Neste programa, a vulnerabilidade encontra-se no tipo `int` nas dimensões da matriz, que pode levar a um *integer overflow*. Basta que uma das dimensões seja maior que o limite superior do `int` para dar *segmentation fault*. Para vermos isto, basta passar como argumentos à função *vulnerable* o seguinte, por exemplo: `x=2147483650, y=20` e `valor=5`.

```
user@CSI:~/Downloads/1718-EngSeg-master/TPraticas/Aula11$ gcc overflow.c -o overflow
user@CSI:~/Downloads/1718-EngSeg-master/TPraticas/Aula11$ ./overflow
Segmentation fault
```

Nota: no nosso caso, os limites para os diferentes tipos de inteiros são os seguintes

```
user@CSI:~/Downloads/1718-EngSeg-master/TPraticas/Aula11$ java IntegerCheck2
Int válido   entre -2147483648 e 2147483647
Byte válido  entre -128 e 127
Short válido entre -32768 e 32767
Long válido  entre -9223372036854775808 e 9223372036854775807
```

### Pergunta P1.2

Este programa pode dar origem a um *integer underflow*, sendo para isso necessário que a variável *tamanho* seja igual a 0. Neste caso, a variável *tamanho\_real*, também um `size_t`, passará a ter valor -1 que será convertido para um valor positivo. Isto faz com que a função *memcpy* não copie o conteúdo de uma string para outra da forma desejada.

```
user@CSI:~/Downloads/1718-EngSeg-master/TPraticas/Aula11$ gcc underflow.c -o underflow
user@CSI:~/Downloads/1718-EngSeg-master/TPraticas/Aula11$ ./underflow
Segmentation fault
```

### Pergunta P1.3

Neste caso, a variável *tamanho* é um `size_t` mas a variável *tamanho\_real* é um `int`. Assim, ao passarmos um valor negativo a *tamanho*, *tamanho\_real* terá um valor negativo, o que impossibilita a alocação de memória e a cópia da string, dando origem a um *segmentation fault*

```
user@CSI:~/Downloads/1718-EngSeg-master/TPraticas/Aula11$ gcc erro_sinal.c -o erro_sinal
user@CSI:~/Downloads/1718-EngSeg-master/TPraticas/Aula11$ ./erro_sinal
Segmentation fault
```