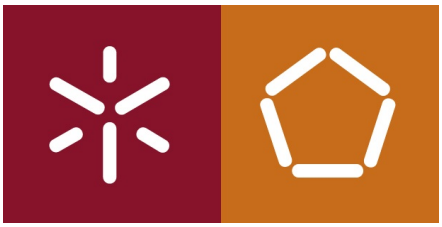

Engenharia de Segurança
Grupo 7
Aula 11

Bruno Machado - A74941
Diogo Gomes - A73825
Francisco Mendes - A75097

Maio 2018

1

1.1

A vulnerabilidade encontrada deve-se ao facto de se tentar alcançar com um *int* o valor do *size_t*, que é impossível, caso o valor do *size_t* seja muito grande. O que impede o programa de escrever o valor no lugar pretendido

1.2

```c
#include <stdio.h>
#include <stdlib.h>

void vulneravel (char *matriz, size_t x, size_t y, char valor) {
        int i, j;
        matriz = (char *) malloc(x*y);
        for (i = 0; i < x; i++) {
                for (j = 0; j < y; j++) {
                        matriz[i*y+j] = valor;
                }
        }
        // testar para ver se escreveu em toda a matriz
        for (long z = 0; z < x; z++){
          for (long q = 0; q < y; q++){
            if(matriz[z*y+q] != valor)
              printf("Deu erro no z: %ld\tq:%ld\n",z,q );
          }
        }
}

int main() {
  char* matriz;
  size_t x = 4294967295;
  // valor máximo de um int
  int y = 2147483647;

  printf("size_t : %zu\n",x );
  printf("int : %d\n",y );

  vulneravel(matriz, 2, 2147483650, 'c');
  printf("finish\n" );
}
```

1.3

```
MacBook-Pro-de-Diogo:aula11 diogogomes$ gcc 1.c -o 1
MacBook-Pro-de-Diogo:aula11 diogogomes$ ./1
size_t : 4294967295
int : 2147483647
Deu erro no z: 0        q:2147483648
Deu erro no z: 0        q:2147483649
Deu erro no z: 1        q:2147483648
Deu erro no z: 1        q:2147483649
finish
```

Como é possível ver na imagem de cima, não foi possível escrever quando se ultrapassou o valor máximo dos *Int's*.

2

A função vulnerável já trata de erros de *overflow* ao especificar o tamanho `MAX_SIZE` e apenas fazer cópias da origem para o destino caso o tamanho da origem seja inferior a esse tamanho. A vulnerabilidade acontece quando se passa como segundo argumento à função `vulneravel` o tamanho `0`, fazendo com que a variável `tamanho_real` toma o valor `-1`. Contudo, esta é do tipo `size_t` e apenas pode tomar valores inteiros positivos. Ao lhe ser atribuído o valor `-1` ela fica armazenada com o valor `4294967295`, causando o overflow e o `Segmentation Fault`.

```
int main(int argc, char** argv){
    char* origem = strdup(argv[1]);
    vulneravel(origem, 0);
    return 0;
}
```

3

3.1

A vulnerabilidade existente na função "vulnerável()"é o facto de se fazer uma subtração de tipos diferentes com tamanhos máximos diferentes. Enquanto que o tamanho máximo de um inteiro é 2,147,483,647 o tamanho máximo de um size_t é 4,294,967,295. Isto leva a que quando se atribui um valor ao "tamanho"maior que o range dos inteiros ao subtrair 1 dá como resultado um número negativo. Assim o programa irá alocar para o apontador "destino"um tamanho negativo.

3.2

```
int main(){
    size_t tamanho = 2147483649;
    char * origem = NULL;
    vulneravel(&origem,tamanho);
}
```

3.3

Sim. Ao executar o programa dá um erro de `Segmentation Fault` devido ao facto de na função `memcpy` o apontador "destino"ter um tamanho negativo.