

# Exercise 05

## Relatório

AEV

Telmo Sauce - Ricardo Covelo - João Almeida



# Exercise 05

AEV

DETI

Universidade de Aveiro

Telmo Sauce - Ricardo Covelo - João Almeida

telmobelasauce@ua.pt - ricardocovelo11@ua.pt - boia13@ua.pt

09/12/2023

# Índice

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Team 3 code review</b>	<b>3</b>
2.1	Overflow . . . . .	3
2.2	Weak Memory Management . . . . .	4
2.3	Improper Input Validation . . . . .	5
2.4	Improper Null Termination . . . . .	5

# Chapter 1

## Introduction

In this report, our team, consisting of three individuals (Ricardo Covelo, Telmo Sauce, and João Almeida) from group number 7, will present our findings after reviewing the code written by team number 1.

After carefully analyzing the code and addressing all the potential issues related to its robustness, we have determined that it deserves a score of 8, indicating that the code was well structured and the problems weren't easy to find.

In addition, our team exchanged code with team number 3 and received their findings. After reviewing their findings, we have assigned a score of 5 to their discoveries. They were able to identify two easy vulnerabilities and one vulnerability that is not very critical. The team discovered several other errors that we were unable to reproduce, 5.2. Furthermore, we do not agree with error 5.5 on password attempts since the existing implementation already safeguards against brute force attacks.

In the next sections, we will delve deeper into our findings and explain the rationale behind them.

# Chapter 2

# Team 3 code review

## 2.1 Overflow

In the displayed image 2.1, an error can be observed which is related to Stack Smashing Protection (SSP), a protocol used in C programming to prevent buffer overflows. On the image 2.2 the same problem occurs, but the "SSP" wasn't triggered.

This issue arises due to the developer failing to implement a mechanism to restrict the size of user inputs. The usage of the "fgets" function also intensifies the problem as it does not possess any built-in protection against buffer overflow, it simply discards any input exceeding the defined maximum length.

This problem can be associated with the "CWE-190: Integer Overflow or Wraparound vulnerability". According to the "OWASP Secure Coding Practices," this error falls into the category of "Memory Management" - "Check buffer boundaries if calling the function in a loop and protect against overflow".

[illegible]

Figure 2.1: Overflow Error

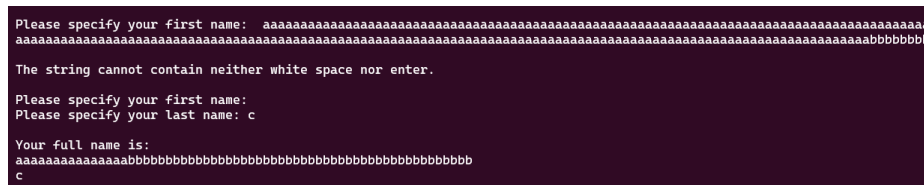


Figure 2.2: Overflow Error

## 2.2 Weak Memory Management

The Memory Management is also not robust since the developer failed to properly release and use the memory as we can see in image 2.3, that a buffer was initialized but its memory wasn't freed which can occur in some problems in the future.

This Problem comes in the "CWE-401: Missing Release of Memory after Effective Lifetime". According to the "OWASP Secure Coding Practices," this error falls into the category of "Memory Management" - "Properly free allocated memory upon the completion of functions and at all exit points".

```
char* getBlock(int fd) {
    char* buf = (char*) malloc(BLOCK_SIZE);

    if (!buf) {
        return NULL;
    }

    do{
        printf("\nWrite anything with 15 characters:\n");
    }
    while (read(fd, buf, BLOCK_SIZE) != BLOCK_SIZE);

    buf[BLOCK_SIZE] = '\0';
    return buf;
}
```

Figure 2.3: Memory Management Error

## 2.3 Improper Input Validation

Based on the 2 images shown below, image 2.4 and image 2.5, it becomes apparent that the input provided is invalid and inadequately handled by the developer.

In image 2.4, the user manages to input a value larger than expected, and the developer fails to properly handle this situation. Instead of discarding or reporting the invalid input, the developer simply places a String Terminator, "\0", character at the last position to display the "15 values". In the second image, image 2.5, it is evident that the input is not properly sanitized, as the size of the input is not taken into consideration by the developer.

This Problem comes in the "CWE-20: Improper Input Validation". According to the "OWASP Secure Coding Practices," this error falls into the category of "Input validation" - "Validate data length".

[illegible]

Figure 2.4: Input Validation Error

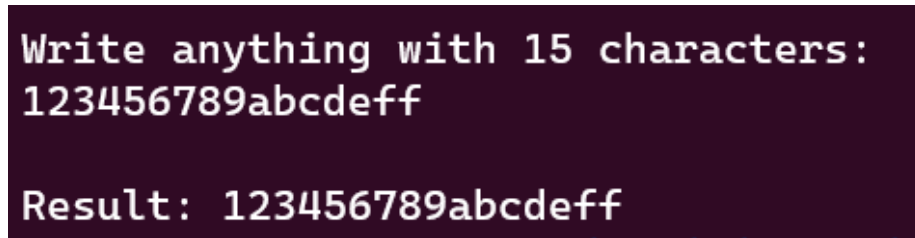
[illegible]

Figure 2.5: Input Validation Error

## 2.4 Improper Null Termination

As evident from the image 2.6, there seems to be a discrepancy in the total number of numbers displayed on the output. Instead of the expected 15 numbers, the output shows 16 numbers. This issue arises due to an error made by the developer, who incorrectly placed the null terminator in the 16th position instead of the maximum minus one. This mistake can be observed in image 2.3.

This Problem comes in the "CWE-170: Improper Null Termination". According to the "OWASP Secure Coding Practices," this error falls into the category of "Memory Management" - "When using functions that accept a number of bytes ensure that NULL termination is handled correctly"



```
Write anything with 15 characters:  
123456789abcdeff  
  
Result: 123456789abcdeff
```

Figure 2.6: Null Termination Error