SEC 522-001: Cybersecurity Lab Spring 2021, KSU 2021 – Buffer Overflow

Sumaya Altamimi – 442203026

This document is organized as follow:

- Introduction
- How to exploit
- Code Vulnerability explanation

Introduction

I develop an ATM C program with one intentional buffer overflow. The Program allows legitimate user to check balance, withdraw or deposit cash. The program resources are protected by password (PIN), However the buffer overflow vulnerability can be exploitable to give access to all of these resources (balance and withdraw/deposit cash) even for illegitimate users who can perform the exploit.

How to exploit?

In terminal/command line, open the 'src' directory, which contains the source code files. For example:

cd Desktop/src

Execute the program by typing:

./main

Now, you are asked for your password, the correct password is 1520, if you enter anything else it will not be accepted.

However, because of the intended vulnerability, you can write any **longer password** and it will be accepted. Simply try this one: 12345678901234567890 .

```
ENTER YOUR SECRET PIN NUMBER:1529
INVALID PASSWORD!!
ENTER YOUR SECRET PIN NUMBER:1234567890
INVALID PASSWORD!!
ENTER YOUR SECRET PIN NUMBER:12345678901234
INVALID PASSWORD!!
ENTER YOUR SECRET PIN NUMBER:12345678901234567890
INVALID PASSWORD!!

1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit
***********************************************
Enter your choice: 
```

Now, you can do whatever you want as if you are the actual user.

Vulnerability in code

If we go back to the code, these two lines are the source of the
problem:

```c
char pin[16];
int checkPIN =0;
```

PIN is the user password, and **checkPIN** is the password status (either
true or false) and zero means it is false by default. It can be true if it is
assigned to any other value, say one.

Normal scenario

If the user enters the valid PIN, I change the value of checkPIN to 1.
Otherwise, it is unchanged.

```c
if(strcmp(pin, "1520")){

    printf("INVALID PASSWORD!!\n");

}
else{
    printf("\n********Welcome to our ATM **************\n");
    checkPIN=1;
}
```

Alternative scenario

As shown in the first screenshot, PIN has a size of 16 only, so if user
pass anything more than 16 to it, the remaining value will be assigned
to the next variable, In this case, checkPIN.

The user will not be asked for any password again if the checkPIN is true/not zero.

```c
while (!checkPIN)
{
    printf("ENTER YOUR SECRET PIN NUMBER:");
    gets(pin);

    if(strcmp(pin, "1520")){

        printf("INVALID PASSWORD!!\n");

    }
    else{
        printf("\n********Welcome to our ATM *************\n");
        checkPIN=1;
    }

}
```

The next piece of code will be executed, and he/she will gain access to all resources.