

Universidad Politécnica de Puerto Rico

Departamento de Ingeniería Eléctrica

Hato Rey

AUTH OVEFLOW EXAMPLE FROM TEXT BOOK

CECS 4210 Ethical Hacking

17-Nov-16 (FECHA EN QUE SE TERMINO EL REPORTE)

Schmidt, Coral # 148830

Profesor: Luis Vicente Ph.D.

Tabla de Contenido

[Hand Out 3](#_Toc152257354)

[Introduction 3](#_Toc152257355)

[Capitulo 01: Flowchart 4](#_Toc152257356)

[Capitulo 02: Codigo fuente 5](#_Toc152257357)

[Capitulo 03: Resultado de la Ejecución 5](#_Toc152257358)

[Conclusión 6](#_Toc152257359)

[Referencias 6](#_Toc152257360)

# Hand Out

CECS 4210 ETHICAL HACKING HANDOUT PROYECTO 2:

In the auth\_overflow.c\_example below,

a) Explain every line of the following code.

b) try to execute an example to check it works with the correct passwords. Use the example in the textbook in page 123. auth\_overflow test , auth\_overflow brillig , auth\_overflow outgrabe

c) Perform the exploit with the book example auth\_overflow AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA . Find when the exploit starts to work by using less A’s. Explain how the exploit works.

Please use Reporte Patrón to submit your work. You have to include all the explanations and figures so any reader could repeat the experiment.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int check\_authentication(char \*password) {

int auth\_flag = 0;

char password\_buffer[16];

strcpy(password\_buffer, password);

if(strcmp(password\_buffer, "brillig") == 0)

auth\_flag = 1;

if(strcmp(password\_buffer, "outgrabe") == 0)

auth\_flag = 1;

return auth\_flag;

}

int main(int argc, char \*argv[]) {

if(argc < 2) {

printf("Usage: %s <password>\n", argv[0]);

exit(0);

}

if(check\_authentication(argv[1])) {

printf("\n-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");

printf(" Access Granted.\n");

printf("-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");

} else {

printf("\nAccess Denied.\n");

}

}-

En las conclusiones analice y comente que hizo en el documento y que aprendió.

- Use reporte patrón, todas las referencias IEEE format. Incluya imágenes de todo el proceso, y si encontró dificultades.

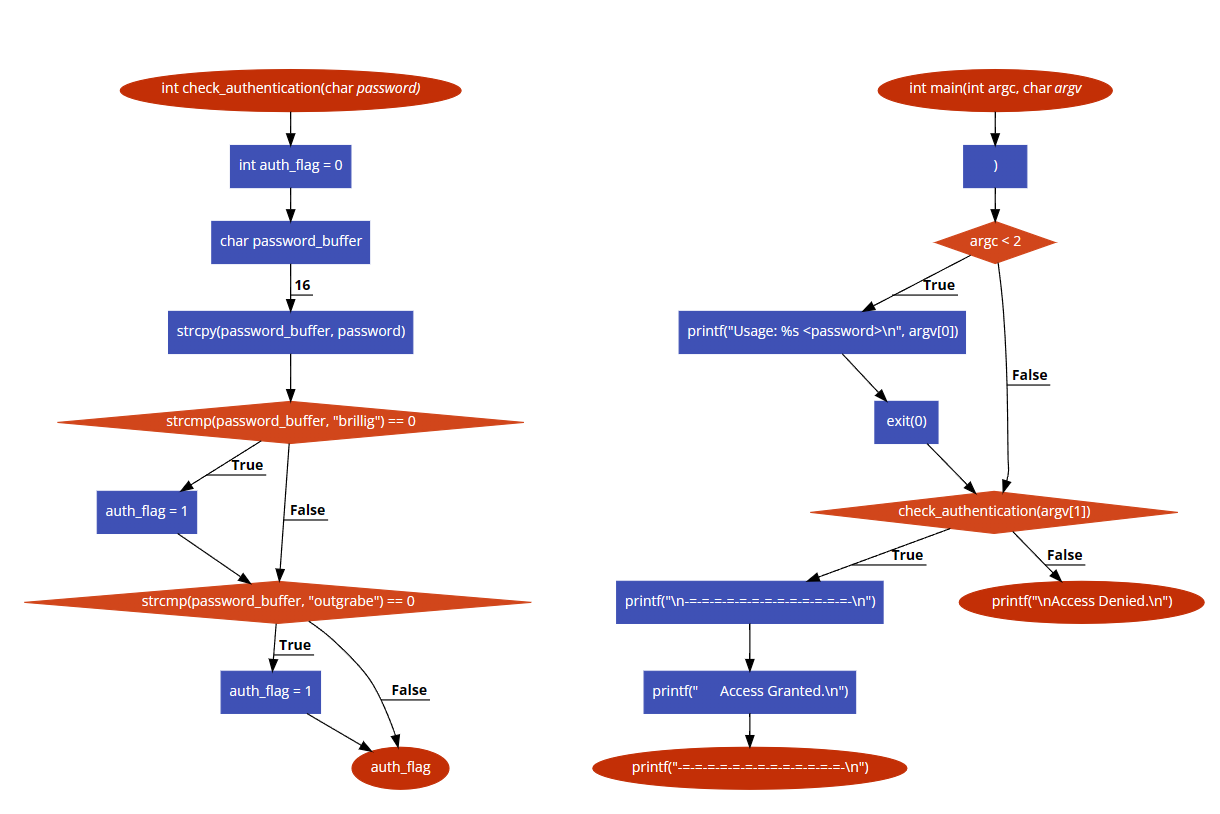
END OF HANDOUT

# Introduction

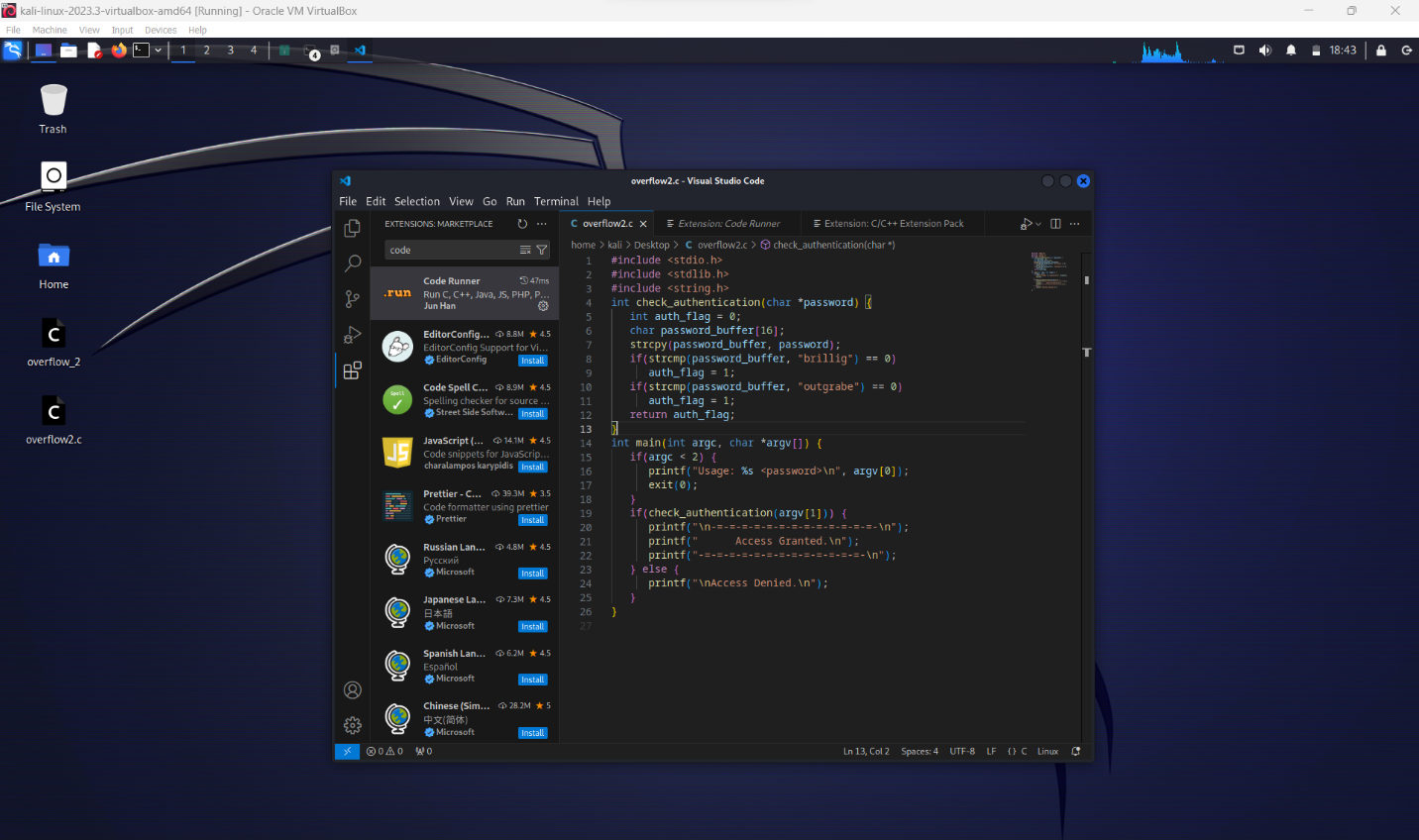
The project illustrates what a "buffer overflow" is, what it does, and how it can be used to uncover vulnerabilities. The task entails identifying and exploiting a buffer overflow vulnerability in the C program auth\_overflow.c. The code begins by defining two variables, the first of which inputs to the second "buffer" and vice versa. This process is repeated until either value reaches its maximum.

First, the code in the handout is a simple authentication program written in C. Two-character buffers of size eight, buffer\_one and buffer\_two, are declared, together with an integer value. The strings “brillig” and “outgrabe” are then copied into buffer\_one and buffer\_two, respectively. The check\_authentication function checks if the input password matches either buffer\_one or buffer\_two. If it does, the auth\_flag is set to 1, granting access. The program outputs buffer\_two, buffer\_one, value, and their respective memory addresses and contents. The main function checks if a password argument is provided. If not, it prints a usage message and exits. If a password is provided, it calls check\_authentication with the password. If access is granted, it prints an “Access Granted” message; otherwise, it prints “Access Denied”. The first command-line parameter is then copied into buffer\_two, which may result in a buffer overflow if the argument contains more characters than eight. Ultimately, following the copy operation, it prints the memory locations and contents of buffer\_one, buffer\_two, and value. The contents of buffer\_one and value might have been overwritten if there was a buffer overflow. However, this code has a buffer overflow vulnerability because it copies the input password into a buffer of fixed size without checking the length of the input, which can be exploited to gain unauthorized access. This application acts as a simple example of how buffer overflows can result in unexpected behavior and possible security flaws.

# Capitulo 01: Flow Chart



# Capitulo 02: Codigo fuente



# Capitulo 03: Resultado de la Ejecución

A screenshot of a computer

Description automatically generated

# Conclusión

This project delves into a buffer overflow vulnerability in the C program auth\_overflow.c. It includes a line-by-line explanation of the code, a demonstration of its functionality with valid passwords, and an exploit with a long string of 'A's. The study also investigates the exact character count at which the exploit begins to function, revealing how buffer overflow vulnerabilities can cause unexpected behavior and potential security issues. This project, which acts as a valuable resource, allows any reader to replicate the experiment or gain a better understanding of buffer overflow, which is a critical component of cybersecurity. The practical information provided by this project demonstrates how buffer overflows can cause unexpected behavior and potential security risks, making it a useful resource for anybody interested in cybersecurity.

# Referencias

[1]

M. Cobb, “buffer overflow,” *Security*, 2021. https://www.techtarget.com/searchsecurity/definition/buffer-overflow (accessed Dec. 12, 2023).