Выполнил

студент КТбо2-4 \_\_\_\_\_\_\_\_\_\_\_\_\_ В. С. Якунин

Выполнил

студент КТбо2-4 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Воронов

Принял

ст. пр. каф. ИБТКС, к. т. н. \_\_\_\_\_\_\_\_\_\_\_\_\_ Я. К. Миронов

ЛАБОРАТОРНАЯ РАБОТА №5

**на тему: «ИССЛЕДОВАНИЕ УЯЗВИМОСТИ «ПЕРЕПОЛНЕНИЕ**

**БУФЕРА»**

по дисциплине «Безопасность информационных технологий»

**Вариант 2**

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Инженерно-технологическая Академия

Институт компьютерных технологий и информационной безопасности

Кафедра Систем Автоматизированного Проектирования

им. В. М. Курейчика

Таганрог 2024

Содержание

[Введение 3](#_Toc184656500)

[1 Практическая часть 4](#_Toc184656501)

[1.1 Настройка и запуск программы 4](#_Toc184656502)

[1.2 Тестирование программы 5](#_Toc184656503)

[1.3 Отладка программы 6](#_Toc184656504)

[Заключение 11](#_Toc184656505)

Введение

Одной из ключевых задач информационной безопасности является выявление и предотвращение уязвимостей в программном обеспечении. Одной из наиболее известных и часто встречающихся уязвимостей является «переполнение буфера». Это явление возникает, когда данные, записываемые в буфер фиксированного размера, превышают его емкость, что приводит к записи данных в смежные области памяти. Такое поведение может быть использовано злоумышленниками для изменения поведения программы, выполнения вредоносного кода или получения несанкционированного доступа к данным.

Исторически уязвимость «переполнение буфера» являлась причиной многих инцидентов информационной безопасности, начиная от атак на серверные системы до заражения пользовательских устройств вредоносными программами. Понимание механизмов возникновения этой уязвимости, а также способов её эксплуатации и защиты от неё имеет фундаментальное значение для разработчиков программного обеспечения и специалистов по безопасности.

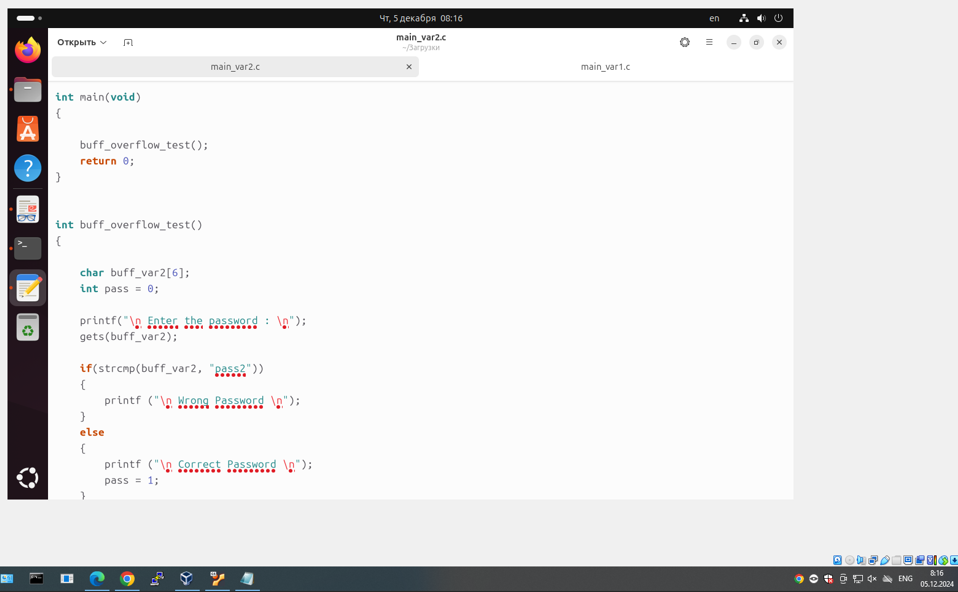
Целью данной лабораторной работы является изучение природы уязвимости «переполнение буфера», механизмов её возникновения и последствий эксплуатации. В рамках работы будут рассмотрены примеры программного кода с данной уязвимостью, способы её обнаружения, а также методы защиты и предотвращения.

Результаты лабораторной работы помогут сформировать базовые навыки анализа кода с точки зрения безопасности, что является важным элементом профессиональной подготовки специалистов в области разработки программного обеспечения и информационной безопасности.

# Практическая часть

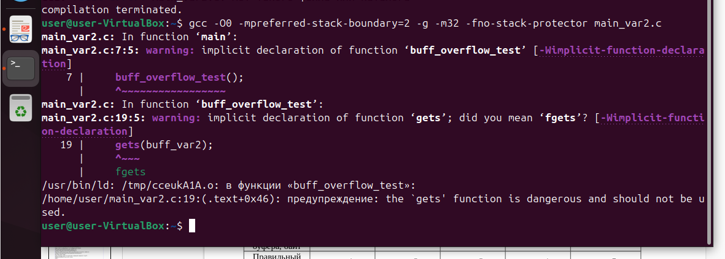
## Настройка и запуск программы

Запустили виртуальную машину с установленным на ней дистрибутивом ОС Linux — Ubuntu. Открыли файл main.c и изменили его согласно данному варианту задания (рисунок 1). Название файла изменили на main\_var2.c, буфер назвали buff\_var2, размер буфера указали равным 6 байтам, правильный пароль установили pass2.



1. — Корректировка файла main.c

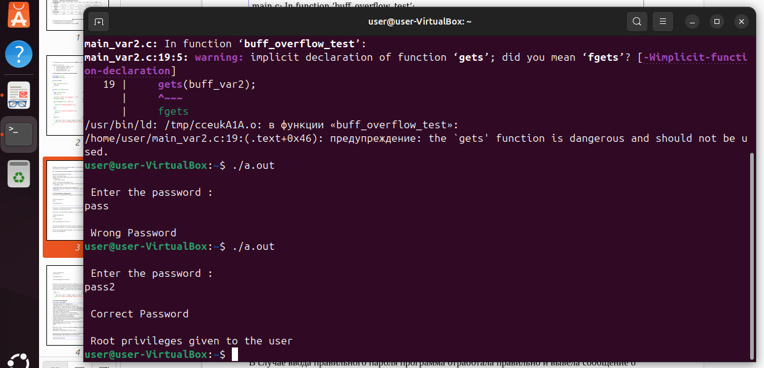
Скомпилировали программу командой «gcc -O0 -mpreferred-stack-boundary=2 -g -m32 –fno-stack-protector main\_var2.c». Это команда для создания исполняемого файла с отключением механизмов защиты адресного пространства и оптимизаций (рисунок 2).



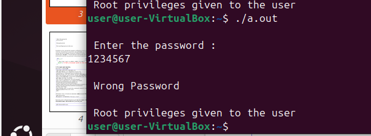
1. — Результат работы компилятора

## Тестирование программы

Далее протестировали программу. Запустили скомпилированный файл a.out командой ./a.out. Ввели правильный и неправильный пароли. Как видно, программа правильно их распознала — на неверный пароль ответила Wrong Password, а на корректный пароль ответила Correct Password и «дала доступ» — Root privileges given to user (рисунок 3). После этого запустили программу снова и протестировали ее на данных, которые больше размера буфера — ввели пароль 1234567 (рисунок 4).



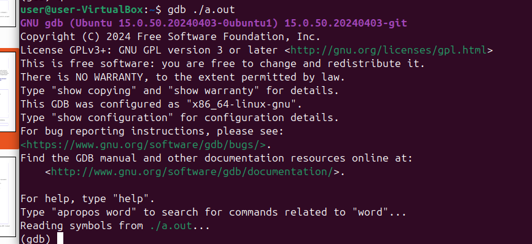
1. — Тестирование программы



1. — Тестирование программы на пароле, большем, чем буфер

## Отладка программы

Чтобы провести отладку программы, запустили отладчик gbd. Использовали команду gbd ./a.out (рисунок 5). Далее добавили точку останова на функции main командой break main (рисунок 6).

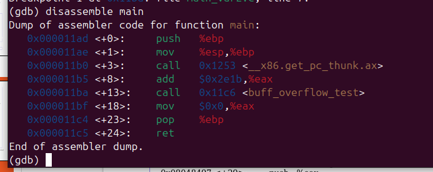


1. — Запуск отладчика



1. — Добавление точки останова

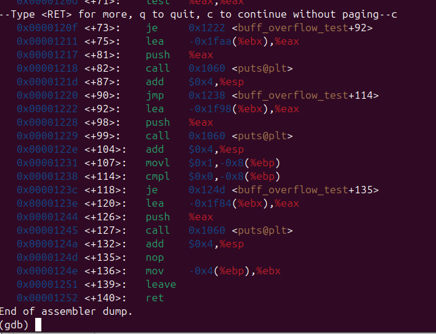
Затем выводим дизассемблированный код для функции main и buff\_overflow\_test командами disassemble main и disassemble buff\_overflow\_test (рисунок 7-9). Обратим внимание на регистры указателей стека: ESP – текущий указатель стека, EBP – базовый указатель стека.



1. — Дизассемблирование кода функции main

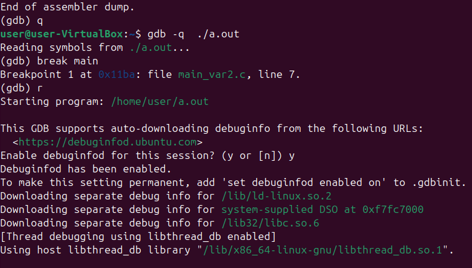


1. — Дизассемблирование кода функции buff\_overflow\_test, часть 1

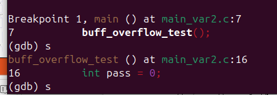


1. — Дизассемблирование кода функции buff\_overflow\_test, часть 2

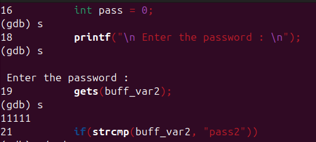
После дизассемблирования функций вышли из отладчика gdb введя команду q и перезапустили его командой gdb -q a.out. Заново установили точку останова командой break main, после чего запустили программу командой r (run) (рисунок 10). После осуществили пошаговое выполнение командой s (step) (рисунок 11). Ввели для проверки пароль 11111 (рисунок 12).



1. — Запуск отладчика и программы

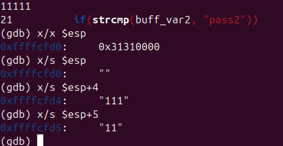


1. — Пошаговое выполнение программы



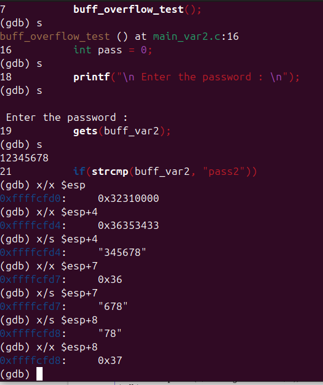
1. — Ввод пароля, меньшего, чем размер буфера

На 21 строке, в которой осуществляется сравнение введённого пароля с «pass2», посмотрели командами x/x и x/s содержимое памяти в стеке по адресу, указанному в регистре ESP.

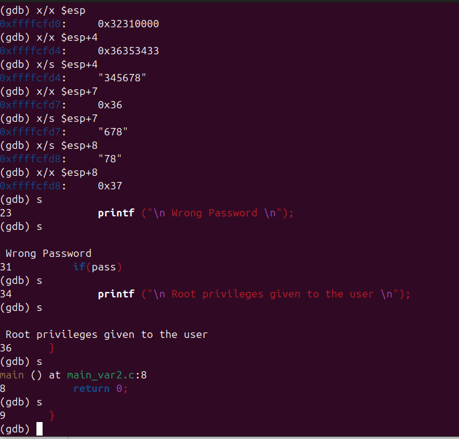


1. — Просмотр содержимого памяти

Перезапустили отладчик и повторили предыдущие шаги. В качестве пароля ввели 12345678, что на 2 байта превышает размер буфера (рисунок 14). При продолжении выполнения программы видно, что пароль определен как неправильный, но доступ был все равно предоставлен (рисунок 15).



1. — Переполнение буфера



1. — Выполнение программы

Заключение

В ходе выполнения данной лабораторной работы были изучены основные аспекты уязвимости «переполнение буфера», включая её природу, причины возникновения и последствия эксплуатации. Рассмотренные примеры позволили на практике убедиться в возможностях злоупотребления данной уязвимостью для выполнения произвольного кода и нарушения работы приложений.

Результаты лабораторной работы подчеркивают необходимость постоянного мониторинга и обновления программного обеспечения для снижения рисков, связанных с эксплуатацией уязвимостей. Эти навыки и знания являются важным шагом на пути к обеспечению информационной безопасности и защите современных компьютерных систем.