Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЁТ

к лабораторной работе №5

на тему

**ЗАЩИТА ОТ АТАКИ НА ПЕРЕПОЛНЕНИЕ БУФЕРА**

|  |
| --- |
| Выполнил: студент гр. 253503  Кудош А.С. |
| Проверил: ассистент кафедры информатики Герчик А.В. |

Минск 2025

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc189743996)

[2 Реализация программного средства 4](#_Toc189743997)

[Заключение 6](#_Toc189743998)

[Список использованных источников 7](#_Toc189743999)

[Приложение А (обязательное) Исходный код программы 8](#_Toc189744000)

**1 ПОСТАНОВКА ЗАДАЧИ**

В данной лабораторной работе будет рассмотрена проблема безопасности, связанная с атаками на переполнение буфера. Переполнение буфера является одной из наиболее распространенных уязвимостей, которая может использоваться злоумышленниками для выполнения произвольного кода, получения несанкционированного доступа к системе или изменения ее поведения [1]. Целью данной работы является изучение механизмов защиты от таких атак и реализация их на языке *C++*.

Для практической части работы будет разработан пример программы на *C++*, демонстрирующий уязвимость переполнения буфера, а затем реализованы механизмы защиты. Будет создано две версии программы: первую – с уязвимостями, которая позволит продемонстрировать последствия переполнения буфера, и вторую – с применением защитных технологий. Это позволит наглядно увидеть, как использование правильных практик программирования и современных методов защиты может предотвратить атаки.

Также будет проведен тестирование обеих версий программы для оценки их устойчивости к атакам. В результате выполнения лабораторной работы можно будет сделать выводы о том, какие меры наиболее эффективны для защиты от переполнения буфера и как их можно внедрить в процесс разработки программного обеспечения.

Таким образом, данная лабораторная работа направлена не только на практическое освоение языка *C++*, но и на формирование навыков анализа и реализации методов защиты в контексте актуальных угроз безопасности в программировании.

**2 РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА**

В данной главе мы представим реализацию программного средства, направленного на демонстрацию уязвимости переполнения буфера и применение методов защиты от нее. Разработанное приложение на языке *C++*, код которого предоставлен в приложении А предназначено для обработки пользовательского ввода и предотвращения возможных атак, связанных с переполнением буфера.

Программа начинается с подключения необходимых заголовочных файлов и определения размера буфера. Используется *#define BUFFER\_SIZE 10*, что устанавливает максимальный размер буфера, используемого для хранения пользовательского ввода.

В функции *main()* устанавливается локализация для корректного отображения русских символов. Далее создаётся два массива: *buffer* размером *BUFFER*\_*SIZE* для хранения безопасного ввода и *input* размером сто для временного хранения пользовательского ввода.

Для получения данных от пользователя используется функция *fgets()*, которая позволяет избежать переполнения буфера, ограничивая количество считываемых символов до размера массива. В случае, если ввод не удался, программа выводит сообщение об ошибке и завершает выполнение.

После ввода удаляем символ новой строки, который может быть добавлен в конце строки, с помощью функции *strcspn()*, которая находит позицию символа новой строки и заменяет его на нуль-терминатор.

Перед копированием строки в буфер программа проверяет, превышает ли длина введенных данных размер буфера. Если длина введенной строки больше или равна *BUFFER\_SIZE*, выводится сообщение об ошибке, и программа завершает выполнение, что предотвращает переполнение буфера.

Перед копированием строки в буфер программа проверяет, превышает ли длина введенных данных размер буфера. Если длина введенной строки больше или равна BUFFER\_SIZE, выводится сообщение об ошибке, и программа завершает выполнение, что предотвращает переполнение буфера.

В завершение, программа выводит содержимое буфера на экран, демонстрируя успешный ввод данных (рисунок 2.1).

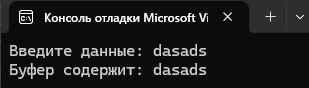


Рисунок 2.1 – Успешное выполнение программы

Если был введен текст, превышающий размер буфера, то программа сообщит об этом и не добавит ничего в буфер (рисунок 2.2).

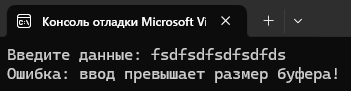


Рисунок 2.2 – Сообщение о превышении размера буфера

Если отключить защиту и попытаться ввести текст, который больше размера буфера, то программа выдаст ошибку (рисунок 2.3).

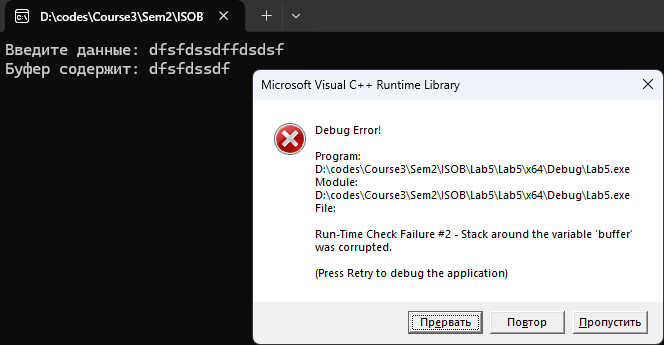


Рисунок 2.3 – Ошибка переполнения буфера

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы была проведена реализация программного средства, направленного на изучение проблемы переполнения буфера и методов защиты от этой уязвимости. Переполнение буфера является одной из самых распространенных и опасных уязвимостей в современных программных системах, что может приводить к выполнению произвольного кода, получению несанкционированного доступа к данным и другим критически важным последствиям. Поэтому понимание механизмов защиты и умение их применять являются ключевыми навыками для разработчиков.

В процессе работы были подробно рассмотрены основные принципы безопасного программирования на языке *C++*. Использование функции *fgets()* для получения пользовательского ввода стало первым шагом, направленным на снижение рисков. Эта функция позволяет ограничивать количество считываемых символов, что существенно снижает вероятность переполнения буфера. Это решение делает код более предсказуемым и безопасным.

Также была реализована проверка длины строки перед её копированием в буфер. Применение функции *strcspn()* для удаления символа новой строки и проверка длины введенных данных обеспечивают корректное завершение строки и предотвращают переполнение. Эти меры значительно снижают риск возникновения уязвимостей и поддерживают безопасность программного обеспечения.

В результате работы сделан вывод о необходимости интеграции методов защиты в процесс разработки. Безопасность программного обеспечения не должна рассматриваться как второстепенная задача; она должна быть неотъемлемой частью всего жизненного цикла разработки. Это требует от разработчиков постоянного обучения и повышения осведомленности о современных угрозах, а также знакомства с новыми технологиями и методами защиты.

Лабораторная работа не только углубила знания о переполнении буфера, но и подчеркнула значимость применения практик безопасного программирования для защиты от современных угроз. Осознание механизмов атаки и защиты является важным аспектом в создании надежных и безопасных программных продуктов. Это знание будет полезно в будущей практике, позволяя создавать более защищенные приложения и системы.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Как устроены дыры в безопасности: переполнение буфера [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/266591/.

ПРИЛОЖЕНИЕ А  
(обязательное)  
Исходный код программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <string.h>

#define BUFFER\_SIZE 10

int main() {

setlocale(LC\_ALL, "Russian");

char buffer[BUFFER\_SIZE];

char input[100];

printf("Введите данные: ");

if (fgets(input, sizeof(input), stdin) == NULL) {

printf("Ошибка ввода!\n");

return 1;

}

input[strcspn(input, "\n")] = '\0';

if (strlen(input) >= BUFFER\_SIZE) {

printf("Ошибка: ввод превышает размер буфера!\n");

return 1;

}

strcpy(buffer, input);

buffer[BUFFER\_SIZE - 1] = '\0';

printf("Буфер содержит: %s\n", buffer);

return 0;

}