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.253504 Давыдовский Д.В.

Проверил: ассистент кафедры информатики Герчик А.В.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc188986632)

[2 Ход работы 4](#_Toc188986633)

[Заключение 7](#_Toc188986634)

# 1 ЦЕЛЬ РАБОТЫ

Цель работы – разработка программных средств для защиты от атак, основанных на переполнении буфера. В ходе исследования будут рассмотрены механизмы возникновения данной уязвимости, методы ее эксплуатации злоумышленниками, а также существующие способы защиты и предотвращения подобных атак.

На первом этапе будет проведен анализ принципов работы буферов в памяти, механизмов их управления и причин, по которым возможны атаки на переполнение буфера. Рассматриваемые угрозы включают стековые и кучевые переполнения, перезапись памяти, внедрение вредоносного кода и обход защитных механизмов (например, *ASLR*, *DEP*, *Stack Canaries*). Особое внимание будет уделено примерам реальных атак и их последствиям.

Далее будет разработана архитектура защитной системы, включающая современные методы предотвращения атак на переполнение буфера. Важными аспектами станут применение безопасных функций работы с памятью, реализация механизма контроля границ буфера, использование компиляторных защит и выявление попыток эксплуатации уязвимостей. Также будет рассмотрена возможность динамического анализа поведения программ для обнаружения аномальных операций с памятью.

На следующем этапе будет реализовано программное решение, направленное на выявление и предотвращение атак на переполнение буфера. Оно должно корректно обрабатывать входные данные, обеспечивать безопасное управление памятью и предотвращать попытки перезаписи критических участков памяти, тем самым исключая возможность выполнения вредоносного кода.

Ожидаемыми результатами работы являются разработка эффективных методов защиты от атак на переполнение буфера и создание надежного программного инструмента, который может быть интегрирован в системы с повышенными требованиями к безопасности. Это обеспечит защиту от одной из наиболее распространенных категорий уязвимостей, повышая общую устойчивость программного обеспечения к потенциальным угрозам.

# 2 ХОД РАБОТЫ

В ходе работы было разработано два программных продукта. Один с защитой, второй без защиты от переполнения буфера. Данную уязвимость можно отлично проследить на языках низкого уровня, в частности *С/С++*. В результате переполнения буфера может случится всё, что угодно, вплоть до выполнения вредоносного кода, без ведома разработчика. На рисунке 1 представлен исходный код беззащитной программы.



Рисунок 1 – Исходный код беззащитной программы

Казалось бы, что может пойти не так, при выполнении этого кода. Разберёмся немного в функциональности программы. При запуске программного продукта, пользователь получает сообщение с требование ввода строки. После окончания ввода задействуется фиктивная переменная *isAdmin*, которая используется в условном операторе *if* для вывода сообщения. Изначально она является *false*(0) и в коде нигде не изменяется. Значит сообщение “Доступ предоставлен” не выведется никогда. Проверим данное утверждение на практике. Запустив программу и введя строку длинной больше 16 символов.

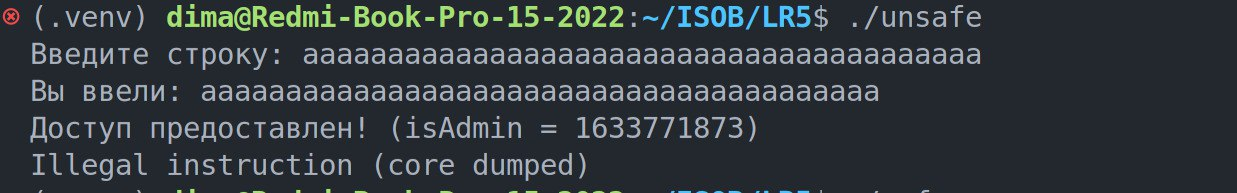


Рисунок 2 – Результат работы беззащитной программы

Как видно из практического примера, мы получаем сообщение “Доступ предоставлен”, что никак не может быть, т.к мы нигде не меняли изначально ложную переменную. А происходит это по следующей причине: при вводе строки длинной больше 16 происходит переполнение буфера, что в свою очередь ведет к изменению переменной *isAdmin*, что в свою очередь приводит к выводу сообщения “Доступ предоставлен”.

Рассмотрим теперь код, который является защищенным от подобной атаки. На рисунке 3 можно заметить, что данный код от предыдущего отличается только одной функцией: вместе функции *gets* используется функция *fgets.*



Рисунок 3 – Исходный код защищенной программы

В данной программе присутствует защита от переполнения буффера, что гарантирует поведение программы, которое задумал автор. Результат работы представлен на рисунке 4

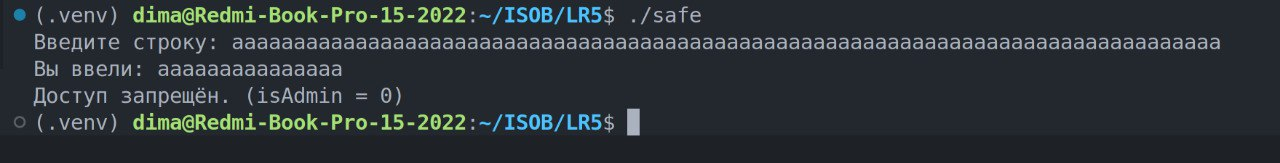


Рисунок 4 – Результат работы программы с защитой

Как видно из рисунка 4 несмотря на то, что мы ввели строку даже длиннее, чем в предыдущем примере, доступ остался закрытым. А происходит это за счет того, что функция *fgets* принимает одним из параметров размер буфера, если строка вылазит за этот размер, результат усекается. Данный факт гарантирует нам полную сохранность наших данных.

Существует ещё один вид атаки, который в связи с повышенной сложностью реализовать не получится. Все современные компиляторы языков низкого уровня, при компиляции кода защищают программный продукт от приведённых видов атак, что почти сводит на нет попытки атак связанные с переполнение буфера.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была разработана и протестирована система защиты от атак, основанных на переполнении буфера. Для наглядной демонстрации были реализованы два программных продукта: один без защиты, позволяющий проиллюстрировать уязвимость, и второй с механизмами предотвращения атак.

Практический эксперимент показал, что использование небезопасных функций работы с памятью, таких как *gets*, может привести к неконтролируемому поведению программы, включая изменение значений переменных, нарушение структуры памяти и даже выполнение произвольного кода злоумышленником. В защищённой версии программы применение безопасных функций, таких как *fgets*, позволяет ограничить размер вводимых данных и предотвратить выход за границы выделенной памяти, что исключает возможность эксплуатации уязвимости.

Кроме того, было рассмотрено влияние современных компиляторов и операционных систем на безопасность программ, а также дополнительные защитные механизмы, такие как *ASLR*, *DEP* и *Stack Canaries*, которые существенно усложняют проведение атак.

Таким образом, данная работа наглядно демонстрирует опасность атак на переполнение буфера и методы их предотвращения. Реализованные защитные механизмы могут быть интегрированы в программные продукты, повышая их безопасность и устойчивость к возможным угрозам. Работа также подчёркивает важность использования безопасных методов работы с памятью и необходимости внедрения современных защитных технологий при разработке программного обеспечения.