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 № 3

|  |  |
| --- | --- |
|  | Выполнил студент группы 053505  Слуцкий Никита Сергеевич |
|  | Проверил ассистент каф.информатики  Протько Мирослав Игоревич |

Минск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc132111711)

[1 Теоретические сведения 5](#_Toc132111712)

[2 Выполнение работы 7](#_Toc132111713)

[3 Тестирование программного продукта 8](#_Toc132111714)

[Заключение 9](#_Toc132111715)

[ПРИЛОЖЕНИЕ А Листинг кода 10](#_Toc132111716)

# ВВЕДЕНИЕ

Целью данной лабораторной работы являются следующие пункты:

* Изучить теоретические сведения.
* Создать приложение, реализующее атаки на протокол при установке TCP-соединения и в рамках заданного протокола прикладного уровня.
  + В интерфейсе приложения должны быть наглядно представлены:
    - Исходные данные протокола (модули, ключи, флаги, иные данные);
    - Данные, передаваемые по сети каждой из сторон;
    - Проверки, выполняемые каждым из участников.

# 1 Теоретические сведения

Компьютер в сети TCP/IP может иметь адреса трех уровней (но не менее двух):

* Локальный адрес компьютера. Для узлов, входящих в локальные сети - это МАС-адрес сетевого адаптера. Эти адреса назначаются производителями оборудования и являются уникальными адресами.
* IP-адрес, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов
* Символьный идентификатор-имя (DNS), например, [www.kstu.ru](http://www.kstu.ru/).

**IP-адреса**

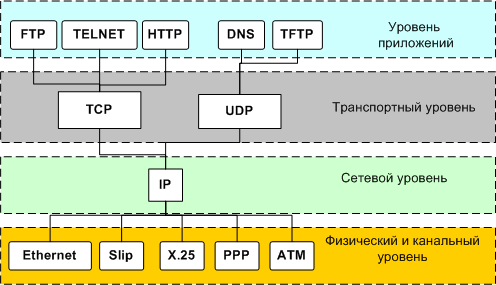
**IPv4** - адрес является уникальным 32-битным идентификатором IP-интерфейса в Интернет.

**IPv6** - адрес является уникальным 128-битным идентификатором IP-интерфейса в Интернет, иногда называют **Internet-2,** адресного пространства IPv4 уже стало не хватать, поэтому постепенно вводят новый стандарт.

IP-адреса принято записывать разбивкой всего адреса по октетам (8), каждый октет записывается в виде десятичного числа, числа разделяются точками.

TCP/IP - собирательное название для набора (стека) сетевых протоколов разных уровней, используемых в Интернет. Особенности TCP/IP:

* Открытые стандарты протоколов, разрабатываемые независимо от программного и аппаратного обеспечения;
* Независимость от физической среды передачи;
* Система уникальной адресации;
* Стандартизованные протоколы высокого уровня для распространенных пользовательских сервисов.



Стек протоколов TCP/IP

Стек протоколов TCP/IP делится на 4 уровня:

* Прикладной,
* Транспортный,
* Межсетевой,
* Физический и канальный.

Позже была принята 7-ми уровневая модель ISO.

Данные передаются в пакетах. Пакеты имеют заголовок и окончание, которые содержат служебную информацию. Данные, более верхних уровней вставляются, в пакеты нижних уровней.

# 2 Выполнение работы

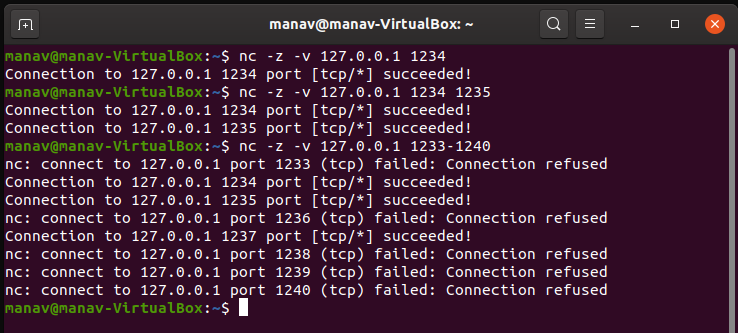
Для выполнения данной лабораторной работы была использована ОС Linux Ubuntu, язык программирования C++ (с применением синтаксиса современного C++ 17), компилятор GNU g++, а также системные вызовы Linux для создания сокетов, соединений.

За теоретический источник информации взята книга «Computer Security: A Hands-on Approach» .

Для успешного выполнения работы также необходимы утилиты Linux, описанные в файле Readme в репозитории с лабораторной работой. Дальнейшее описание опускается во избежание дублирования уже написанной информации с ходом выполнения.

# 3 тестирование программного продукта

Программный продукт разработан на языке программирования C++ и исполнен в ОС Linux компиляции GNU.



# Заключение

В результате выполнения лабораторной работы на практике была реализована связка клиента TCP и сервера TCP, а также продемонстрирована работа сканирующих утилит Linux для наблюдения за этой связкой.

Была использована утилита Linux для атаки на работающий сервер, а также написан код, который выполняет то же самое.

Цели лабораторной работы можно считать достигнутыми. Работа выполнена.

# ПРИЛОЖЕНИЕ А

**Листинг кода**

#include <cstring>  
#include <iostream>  
#include <netinet/ip.h>  
#include <unistd.h>  
#include <sys/socket.h>  
#include <arpa/inet.h>  
#include "./shared.hpp"  
  
  
int main()  
{  
 sockaddr\_in my\_addr{};  
 sockaddr\_in client\_addr{};  
 char buffer[kBufferMaxSize];  
  
 // socket itself  
 const auto socket\_file\_descriptor{socket(AF\_INET, SOCK\_STREAM, kTcpProtocolValue)};  
  
 // bind to a port number  
 memset(&my\_addr, 0, sizeof(struct sockaddr\_in));  
 my\_addr.sin\_family = AF\_INET;  
 my\_addr.sin\_port = htons(kPort);  
 bind(socket\_file\_descriptor, (struct sockaddr \*)&my\_addr, sizeof(struct sockaddr\_in));  
  
 // listening for connections  
 listen(socket\_file\_descriptor, 5);  
  
 const auto client\_len{sizeof(client\_addr)};  
  
 while (true) {  
 const auto new\_socket\_file\_descriptor{  
 accept(  
 socket\_file\_descriptor,  
 (struct sockaddr \*)&client\_addr,  
 (socklen\_t \* ) & client\_len  
 )  
 };  
  
 if (const auto process\_id{fork()}; process\_id == 0) {  
 // 0 <==> child (new) process  
 close(socket\_file\_descriptor);  
  
 // Read data.  
 memset(buffer, 0, sizeof(buffer));  
 const auto received\_data\_length{read(new\_socket\_file\_descriptor, buffer, kBufferMaxSize)};  
  
 std::cout << "Received " << received\_data\_length << " bytes" << '\n';  
 std::cout << "Message received : " << '\n' << "{" << '\n' << buffer << "}" << '\n' << '\n';  
  
 close(new\_socket\_file\_descriptor);  
  
 return 0;  
 } else {  
 // not 0 <==> parent (source) process  
 close(new\_socket\_file\_descriptor);  
 }  
 }  
  
 return 0;  
}

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <netinet/in.h>  
#include <sys/socket.h>  
#include <arpa/inet.h>  
#include <unistd.h>  
#include "./shared.hpp"  
  
  
int main()  
{  
 *// Step 1: Create a socket* const auto socket\_file\_descriptor{socket(AF\_INET, SOCK\_STREAM, kTcpProtocolValue)};  
  
 *// Step 2: Set the destination information* struct sockaddr\_in dest;  
 memset(&dest, 0, sizeof(struct sockaddr\_in));  
 dest.sin\_family = AF\_INET;  
 dest.sin\_addr.s\_addr = inet\_addr(kLocalHostIpOnCurrentMachine);  
 dest.sin\_port = htons(kPort);  
  
 *// Step 3: Connect to the server* connect(socket\_file\_descriptor, (struct sockaddr \*)&dest, sizeof(struct sockaddr\_in));  
  
 *// Step 4: Send data to the server* const char \*buffer1 = "Hello Server!\n";  
 const char \*buffer2 = "Hello Again!\n";  
 write(socket\_file\_descriptor, buffer1, strlen(buffer1));  
 write(socket\_file\_descriptor, buffer2, strlen(buffer2));  
  
 *// Step 5: Close the connection* close(socket\_file\_descriptor);  
  
 return 0;  
}