Entregable 3. Programación de sockets

Redes y sistemas distribuidos

Oscar Guerra Rodríguez

alu0101560342@ull.edu.es

20 de abril de 2024

Description of the developed application:

For this assignment, I have developed an FTP protocol server in C++, which allows users to perform basic file transfer operations over a network in accordance with the standard. As mentioned, this server is designed to be compatible with any FTP-based application by adhering to the internet standard RFC 959. This goal is accomplished through the use of the POSIX socket API, which enables the establishment of TCP connections and the transfer

of data over them. While not every TCP control line command is implemented, the essential ones required for basic operations are included, both in passive and active mode.

The server operates on TCP port 2121 and is capable of handling multiple connections simultaneously. Signal handling and other security measures for working with C file descriptors and FILE* are implemented.

Description of the developed protocol:

FTP, or File Transfer Protocol, is a standard network protocol described in the RFC 959 document. It is meant for transferring files between a client and a server on a computer network. It provides a simple way to work with files remotely. The most common underlying communication protocol is TCP/IP.

Its typical operation consists of two connections: a control connection and a data connection. The control one is established for interchanging commands and its corresponding replies. As its name suggests, the data connection is used to transfer actual data from the files we are working with.

First of all, the control connection is established by the client, and then he can use it to send commands to perform various file operations, such as listing directories, uploading and downloading files, etc.

When the moment comes for establishing the data connection, the operation varies depending on the operation mode that is being used. The two operation modes supported by FTP are "active" and "passive." In active mode, the client initiates the data connection, while in passive mode, the server is the one who opens a data connection to the client. Passive mode is often used in scenarios where the client is behind a firewall or NAT.

Other interesting features provided by FTP are user authentication and access control, allowing administrators to restrict the access to certain resources according to the credentials given by the user.

Compilation process

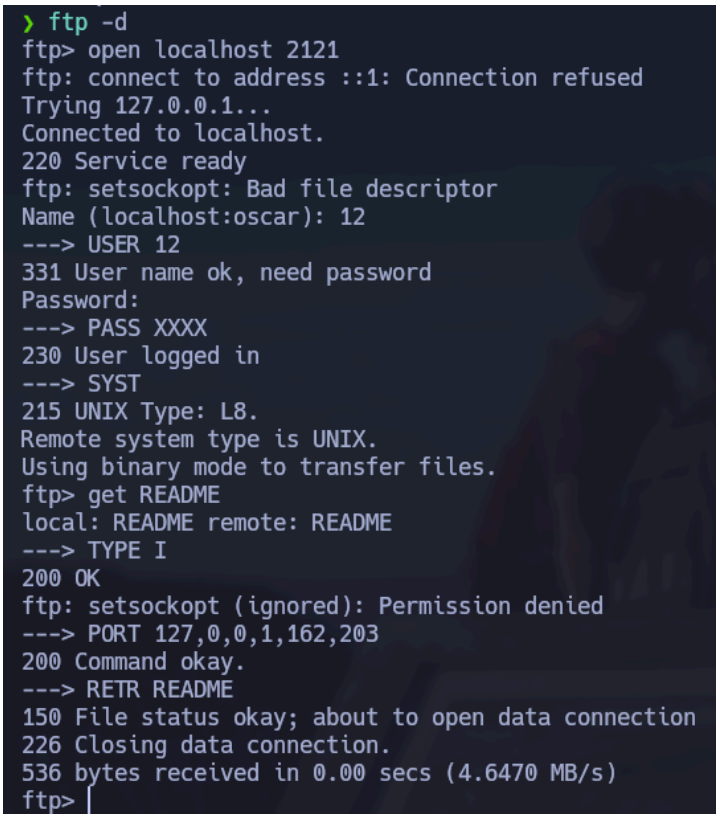
This program is meant to be used on UNIX-like operating systems (as Linux), so it's very likely it won't work on Windows systems. Having the Make and g++ compilation tools installed is necessary for this process. For compiling and running this program we first have to be located at the src directory. Then we can compile it using the “make” command. For

cleaning the object files and executables when we don't want to use the application anymore the “make clean” command can be used.

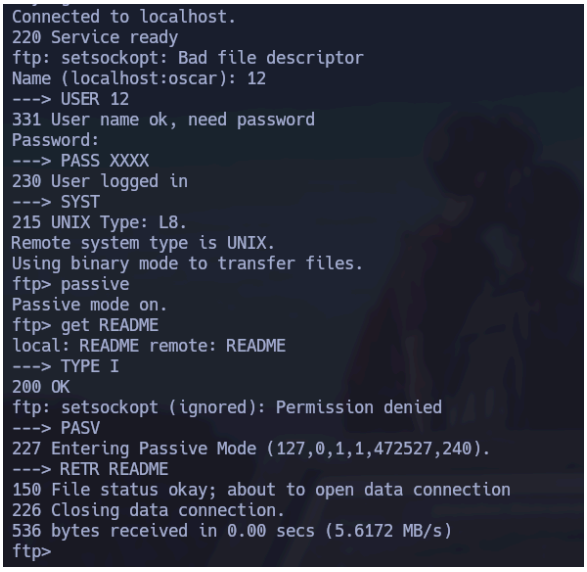
Once the program is compiled the server can be started by using the “./ftp_server” command, and it will be immediately ready to accept requests. If you don't want to run it on the src directory you can replace “./” with the path to the executable.

Test cases:

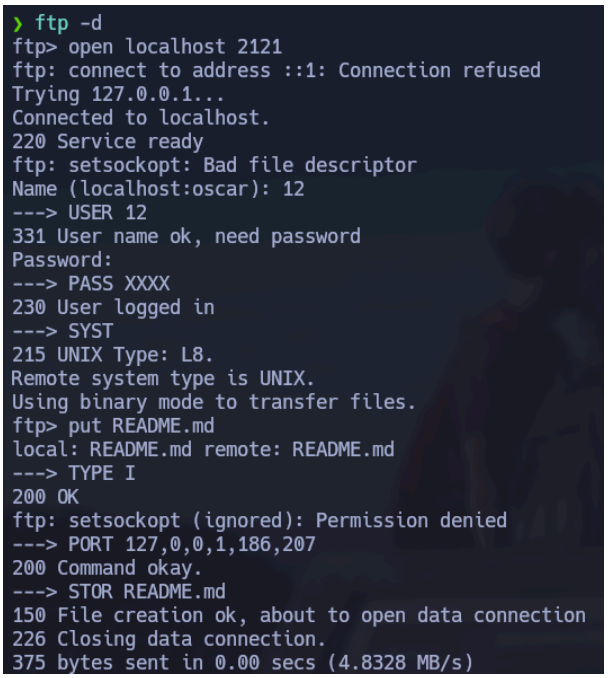
RETR Command on Active mode:



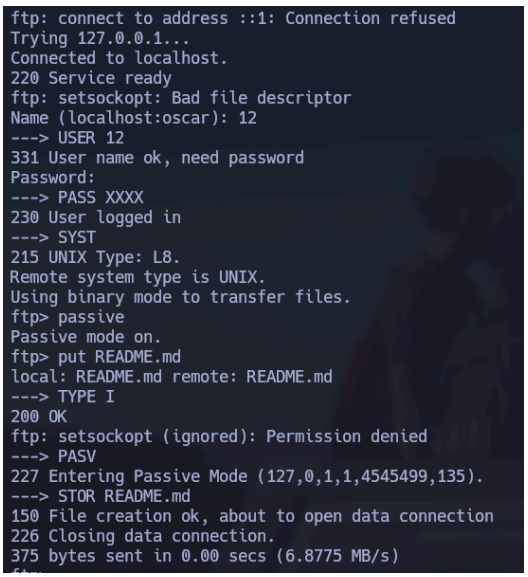
RETR Command on Passive mode:



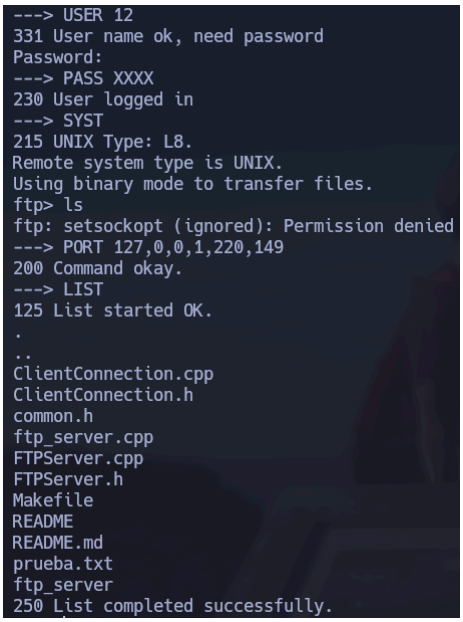
STOR Command on Active mode:



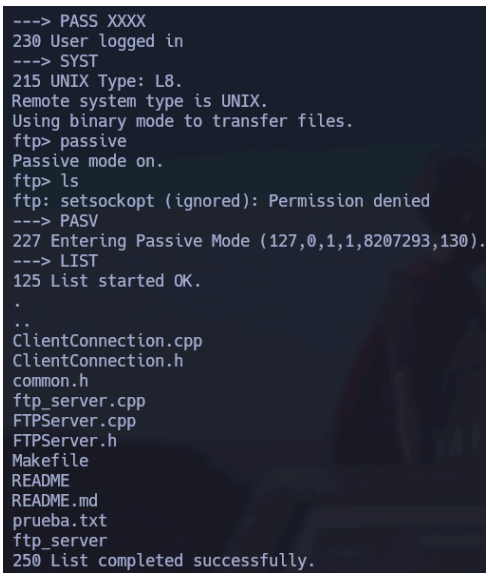
STOR Command on Passive mode:



LIST Command on Active mode:



LIST Command on Passive mode:



References:

Code repository: https://github.com/OscarGuerraRodriguez/FTPServer.git

RFC 959: https://datatracker.ietf.org/doc/html/rfc959