Adam Składanek g. 11c s16787

SKJ project 2
Port knocking – Server and client

1. General description

In folders *PortKnockingServer* and *PortKnockingClient* there are source files for client and server applications. The server works by simply compiling and starting the application. After the server starts, it generates a sequence from 10 to 100 ports with unique values from 1025 to 2024, prints the sequence on the console, exports it to a file *port_sequence.txt* (mainly to make tests on one machine more convenient) and opens UDP sockets on all ports generated. The server will store any attempts to any of the ports by mapping clients IP addresses to the ports they have "knocked". If the following sequence matches ports generated before, server creates a TCP socket and sends its port to the client. Then if a client sends data to the socket it will check if the client ip address mathes the ip address of the client that has knocked UDP ports properly and respond.

The client app requires the IP address of the server. After entering it will search for a file *port_sequence.txt.* If such file exists, it will read the port sequence from it, otherwise ports will need to be entered manually in the console. Then it will send a UDP packet to every port in the sequence and listen for an answer. If it receives a port number it will open a TCP connection and ask the server about the weather. If it does not, it will time out after 10 seconds.

2. Protocol description

The protocol requires a client to "knock" on a proper sequence of UDP ports opened by the server. If it does so, server opens a TCP connection sends a UDP packet to the client containing the port number of the TCP connection opened. After that the client may connect via a TCP connection to the port specified and communication may continue as usual.

3. Difficulties and errors

The main difficulty encountered was that the UDP packets may not always arrive or be processed in the sequence they were sent. I have managed to solve it by adding a delay between sending the packets.