Отчет к проекту

На тему ” Локальный чат с использованием технологии TCP/IP”

По предмету “Интерпретируемый язык программирования высокого уровня”

Выполнили студенты группы БСБО-08-19

Харитонов Алексей Александрович

Сорокин Тимофей Сергеевич

Оглавление

Задание………………………………………………………………..3

Программный код…………………………………………………….4-7

Пояснительная записка………………………………………………8

**Задание**

Создать программу, которая с помощью технологии TCI/IP позволяет организовать локальный чат. Сообщения между пользователями должны быть зашифрованы.

**Программный код**

client.py

import socket  
import threading  
import time  
  
key = 8194 # ключь для шифровки данных  
  
shutdown = False  
join = False  
  
  
def receving(name, sock):  
 while not shutdown:  
 try:  
 while True:  
 data, addr = sock.recvfrom(1024)  
  
 # получение сообщений от других пользователей, расшифровка сообщения от других пользователей  
 decrypt = ""  
 k = False  
 for i in data.decode("utf-8"):  
 if i == ":":  
 k = True  
 decrypt += i  
 elif k == False or i == " ":  
 decrypt += i  
 else:  
 decrypt += chr(ord(i) ^ key)  
 print(decrypt)  
  
 time.sleep(0.2)  
 except:  
 pass  
  
  
host = socket.gethostbyname(socket.gethostname()) # ip локальной сети  
port = 0 # т.к клиент подключается к сети, а не создает ее  
  
server = ("192.168.220.1", 90)# переменная меняется в зависимости от ip сервера  
  
s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) # TCP/IP  
s.bind((host, port))  
s.setblocking(0)  
  
alias = input("Name: ")  
  
rT = threading.Thread(target=receving, args=("RecvThread", s)) # многопоточность которая обробатывается TCP/IP  
rT.start()  
  
while not shutdown:  
  
 # отправка сообщения на сервер,когда пользователь пресоеденился к чату  
  
 if not join:  
 s.sendto(("[" + alias + "] => join chat ").encode("utf-8"), server)  
 join = True  
 else:  
  
 # если пользователь пресоеденился к чату, то он отправляет сообщения  
  
 try:  
 message = input("Enter -> ")  
  
 # шифровка сообщения  
 crypt = ""  
 for i in message:  
 crypt += chr(ord(i) ^ key)  
 message = crypt  
  
 # если отпровляеться пустое сообщение, то на сервер ничего не приходит  
  
 if message != "":  
 s.sendto(("[" + alias + "] :: " + message).encode("utf-8"), server)  
  
 time.sleep(0.2)  
 except:  
 s.sendto(("[" + alias + "] <= left chat ").encode("utf-8"), server)  
 shutdown = True  
  
rT.join()  
s.close()

**server.py**

import socket  
import sys  
import time  
import PyQt5  
from PyQt5 import QtCore, QtGui, QtWidgets  
from PyQt5.QtWidgets import QApplication, QMainWindow  
import time  
  
import serverQt  
import clientQt  
from serverQt import \*  
  
str\_ = ''  
  
  
class ExampleApp(QMainWindow, serverQt.Ui\_Server):  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setupUi(self)  
  
 def ref(self):  
 self.close()  
  
 def print(self, str):  
 self.textBrowser.append(str)  
  
  
def main():  
 app = QApplication(sys.argv) # Новый экземпляр QApplication  
 window = ExampleApp() # Создаём объект класса ExampleApp  
  
 host = socket.gethostbyname(socket.gethostname()) # содержиит в себе ip на котором поднят сервер  
 port = 90 # порт на которм сейчас ничего не запущено  
  
 clients = [] # список клиентов подключенных к серверу  
  
 s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) # содержит в себе сокеты, TCP/IP. Протокол TCP/IP  
 s.bind((host, port)) # создание хоста и порта  
  
 quit = False # для того чтоб цикл прекратился  
 window.print("Server start\n" + "Start at ip: " + host)  
 window.show() # Показываем окно  
 app.exec\_()  
  
 while not quit:  
 try:  
  
 data, addr = s.recvfrom(1024) # 1024байт - сервер может приинять.  
 # data - сообщение, которое отпровляет пользователь. addr - личный адрес пользователя  
  
 if addr not in clients: # если адрес не находится в списке, то его добовляют в список  
 clients.append(addr) #  
  
 ItsaTime = time.strftime("%Y-%m-%d %H:%M", time.localtime()) # для отслеживания  
  
 print("[" + addr[0] + "] = [" + str(addr[1]) + "] = [" + ItsaTime + "]" + "\nLogin:",  
 end="") # пишет информацию  
 print(data.decode('utf-8')) # декодируем сообщение, отправленное клиентом  
 # app.print(data.decode('utf-8'))  
  
 for client in clients:  
 if addr != client: # проверка на дублирование сообщения у отправителя  
 s.sendto(data, client) # отправка сообщений  
 window.textBrowser.reload()  
 except:  
  
 print("\n Server stopped")  
 quit = True  
 window.close()  
  
 s.close()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**serverQt.py**

from PyQt5 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_Server(object):  
 def setupUi(self, Server):  
 Server.setObjectName("Server")  
 Server.resize(400, 300)  
 self.textBrowser = QtWidgets.QTextBrowser(Server)  
 self.textBrowser.setGeometry(QtCore.QRect(10, 10, 381, 100))  
 self.textBrowser.setObjectName("textBrowser")  
  
 self.retranslateUi(Server)  
 QtCore.QMetaObject.connectSlotsByName(Server)  
  
 def retranslateUi(self, Server):  
 \_translate = QtCore.QCoreApplication.translate  
 Server.setWindowTitle(\_translate("Server", "Server"))  
  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
 app = QtWidgets.QApplication(sys.argv)  
 Server = QtWidgets.QDialog()  
 ui = Ui\_Server()  
 ui.setupUi(Server)  
 Server.show()  
 sys.exit(app.exec\_())

**Пояснительная записка**

Программа реализует локальный чат с сервером. Все сообщения зашифрованы методом XOR. Метод XOR - это алгоритм шифрования данных с использованием исключительной дизъюнкции. Алгоритм XOR шифрования заключается в “наложении” последовательности случайных чисел на текст, который необходимо зашифровать. Последовательность случайных чисел называется гамма-последовательность, и используется для шифрования и расшифровки данных.

Сообщения приходят на сервер в зашифрованном виде, после чего отправляются на другие клиенты, где они расшифровываются.

На стороне клиента отображаются сообщения от других пользователей и информация о подключение новых участников. На стороне сервера отображается информация об его IP-адресе, а также информация о сообщениях, передаваемых через него.