Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЕТ

к лабораторным работам №4,5

на тему

**РАЗРАБОТКА ЗАЩИЩЁННЫХ ПРИЛОЖЕНИЙ**

Студент А. В. Скворцов

Преподаватель Е. А. Лещенко

Минск 2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc160192830)

[1 Результат выполнения 4](#_Toc160192831)

[Заключение 4](#_Toc160192832)

[Приложение А (обязательное) Листинг кода 7](#_Toc160192833)

**ВВЕДЕНИЕ**

В ходе лабораторной работы нужно разработать программное средство, которое представляет собой десктопное приложение, позволяющее пользователю вводить логин и пароль и в зависимости от своей роли видеть определенные данные из базы данных. Приложение должно быть защищено от разного рода атак, таких SQL-инъекции, XSS-атаки и т.п.

Также должен быть разработан инсталлятор данного приложения, который запрещает несанкционированное распространение продукта.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате работы было создано приложение, позволяющее пользователю войти под своим логином и паролем (рисунок 1).

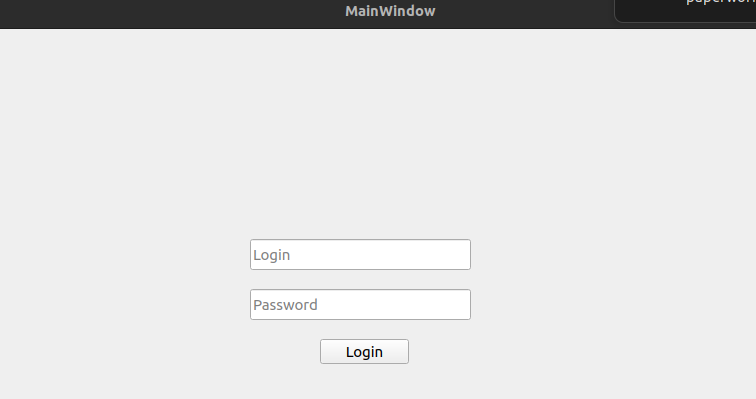
****

Рисунок 1 – Окно входа

Далее, в зависимости от своей роли пользователь может увидеть данные из базы данных и роль решит, сколько данных пользователю покажет программа (рисунок 2, 3).

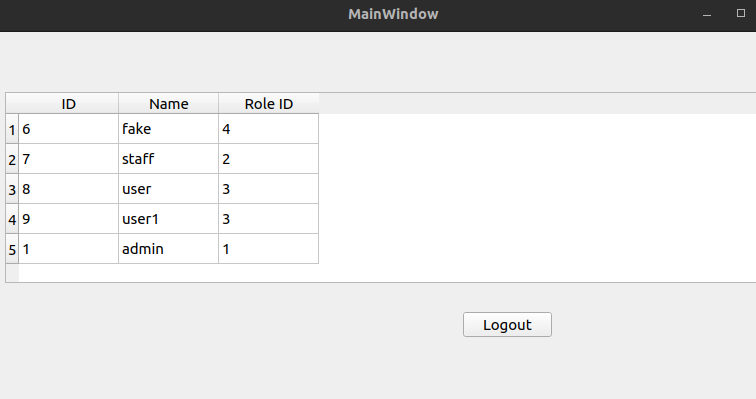


Рисунок 2 – Интерфейс сотрудника

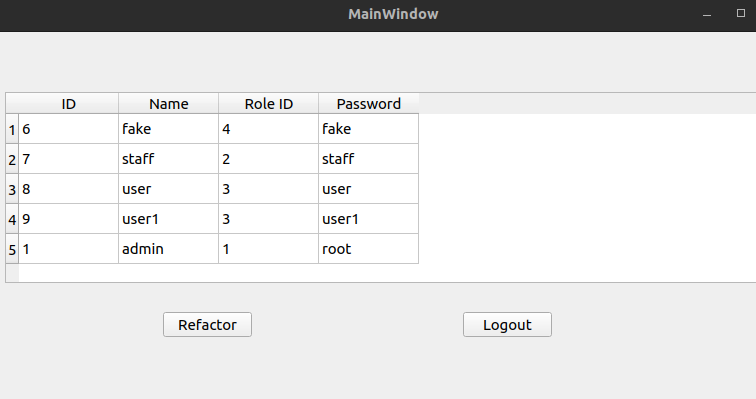


Рисунок 3 – Интерфейс админа

Также на приложении были протестированы различные виды атак, такие как SQL-инъекция, XSS-атака, 4.1) , переполнение буфера и принцип минимизации привилегий.

В рамках лабораторной работы был разработан инсталлятор, который невозможно установить без ключа продукта, что подтверждает его защищенность от несанкционированного тиражирования (рисунок 4).

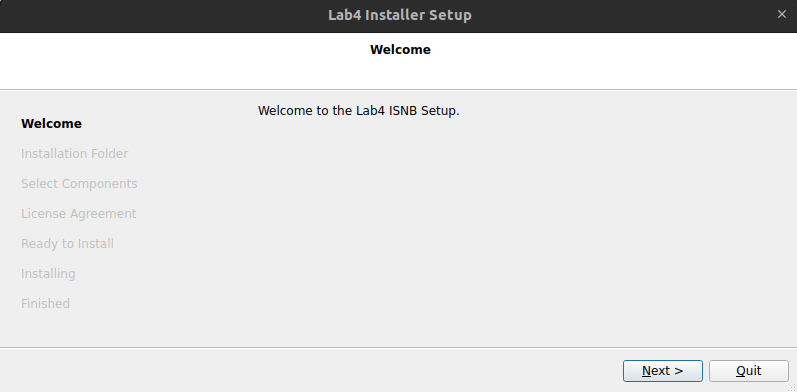


Рисунок 4 – Инсталлятор приложения

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной лабораторной работы было разработано приложение, защищенное от различных видов атак на языке программирования *С++*, также для был предусмотрен защищенный инсталлятор.

Также было проведено изучение видов атак, представленных в лабораторной работе, и было проведено тестирование приложения на работоспособность при нанесении этих атак.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл *mainwindow.cpp*

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent),

ui(new Ui::MainWindow),

dbConnection(PSQLDBHelper("QPSQL")),

login(LoginWindow(&dbConnection)) {

ui->setupUi(this);

if (!dbConnection.connectToDatabase(DB\_HOST, DB\_NAME, DB\_USER, DB\_PASSWORD)) {

qDebug() << "Cannot connect to database!";

}

qDebug() << "Connection with database established!";

login.show();

centralWidget()->setParent(nullptr);

this->setCentralWidget(&login);

this->ui->usersTable->setEditTriggers(QAbstractItemView::NoEditTriggers);

connect(&this->login, SIGNAL(success(int)), this, SLOT(successLogin(int)));

connect(this->ui->logOut, &QPushButton::clicked, this, &MainWindow::logout);

connect(this->ui->refactorButton, &QPushButton::clicked, this, &MainWindow::refactor);

}

void MainWindow::fillTableView(QStandardItemModel \*model, Priviligies role) {

model->setHeaderData(0, Qt::Horizontal, "ID");

model->setHeaderData(1, Qt::Horizontal, "Name");

if (role == Priviligies::ADMIN || role == Priviligies::STAFF) {

model->setHeaderData(2, Qt::Horizontal, "Role ID");

}

if (role == Priviligies::ADMIN) {

model->setHeaderData(3, Qt::Horizontal, "Password");

}

auto users = this->dbConnection.getAllUsers();

for (size\_t i = 0; i < users.size(); ++i) {

model->setData(model->index(i, 0), users[i].id);

model->setData(model->index(i, 1), users[i].name);

if (role == Priviligies::ADMIN || role == Priviligies::STAFF) {

model->setData(model->index(i, 2), users[i].role\_id);

}

if (role == Priviligies::ADMIN) {

model->setData(model->index(i, 3), users[i].password);

}

}

}

void MainWindow::successLogin(int lvl) {

qDebug() << "Priviledge level: " << lvl;

this->login.clearInputs();

this->login.hide();

centralWidget()->setParent(nullptr);

this->setCentralWidget(this->ui->centralwidget);

qDebug() << &this->login;

auto users = this->dbConnection.getAllUsers();

QStandardItemModel \*model = nullptr;

switch (lvl) {

case Priviligies::USER: {

model = new QStandardItemModel(users.size(), 2);

this->ui->refactorButton->hide();

this->ui->logOut->show();

break;

}

case Priviligies::STAFF: {

model = new QStandardItemModel(users.size(), 3);

this->ui->refactorButton->hide();

this->ui->logOut->show();

break;

}

case Priviligies::ADMIN: {

model = new QStandardItemModel(users.size(), 4);

this->ui->refactorButton->show();

this->ui->logOut->show();

break;

}

default: {

QMessageBox::critical(this, "Access denied", "You have no needed role!");

this->ui->refactorButton->hide();

this->ui->usersTable->hide();

this->ui->logOut->show();

return;

}

}

fillTableView(model, Priviligies(lvl));

this->ui->usersTable->setModel(model);

this->ui->usersTable->show();

}

void MainWindow::refactor() {

refactorDialog dialog;

if (dialog.exec() == QDialog::Accepted) {

auto users = this->dbConnection.getAllUsers();

auto res = dialog.getValues();

int row = std::get<0>(res), column = std::get<1>(res);

QString value = std::get<2>(res);

if (value.isEmpty()) {

QMessageBox::warning(this, "Error", "Fill the value field!");

return;

}

if (value.size() > 25) {

QMessageBox::warning(this, "Error", "The value is too long!");

return;

}

if (row >= users.size() || row < 0 || column >= 4 || column < 1) {

QMessageBox::warning(this, "Error", "Invalid row or column number!");

return;

}

if (column == 2 && value.contains(QRegularExpression("[a-zA-Z]"))) {

QMessageBox::warning(this, "Error", "In this column only numbers are available!");

return;

}

QString columnName;

switch (column) {

case 1:

columnName = "login";

break;

case 2:

columnName = "role\_id";

break;

case 3:

columnName = "password";

break;

}

auto index = this->ui->usersTable->model()->index(row, 0);

if (dbConnection.updateRow(this->ui->usersTable->model()->data(index).toInt(), columnName, value)) {

QMessageBox::information(this, "Success!", "Data successfully updated!");

return;

}

QMessageBox::critical(this, "Error!", "Something went wrong in update");

}

}

void MainWindow::logout() {

qDebug() << &this->login;

centralWidget()->setParent(nullptr);

this->setCentralWidget(&this->login);

this->login.show();

}

MainWindow::~MainWindow() {

qDebug() << "MainWindow destructor called!";

dbConnection.disconnect();

// delete login;

delete ui;

}

# ПРИЛОЖЕНИЕ Б (обязательное) Блок-схема алгоритма

