**Содержание**

**[1.](#_Toc65156180)****[Цель работы](#_Toc65156180)** [3](#_Toc65156180)

**[2. Краткие теоретические сведения](#_Toc65156181)** [5](#_Toc65156181)

**[3. Реализация задачи](#_Toc65156182)** [10](#_Toc65156182)

**[Приложение. Текст программ](#_Toc65156183)** [12](#_Toc65156183)

# 

# **Цель работы**

Реализовать программы, выполняющие указанные в задании действия.

**Задание I.**

Реализовать приложение с графическим интерфейсом, удовлетворяющее следующим требованиям:

1. Приложение проводит аутентификацию пользователя.
2. Каждый пользователь программы должен относиться к какой-нибудь группе пользователей (роли), членам которой доступны различные функциональные возможности программы.
3. Программа должна принимать от пользователя некоторые данные и, возможно, после некоторой обработки, отображать их.
4. При этом должна осуществляться защита от как минимум 4-х типов возможных атак на приложение:
   1. Атака «переполнение буфера».
   2. Атака «SQL-инъекции».
   3. Атака, эксплуатирующая ошибки канонизации.
   4. Атака «XSS» (межсайтовое кодирование).
   5. Принцип минимизации привилегий.
   6. DoS-атака (отказ в обслуживании).

При разработке защиты нужно предположить, что приложение работает с базой данных, в которой сохраняет введённые пользователем данные.

**Задание II.**

Реализовать приложение-инсталлятор, позволяющее установить на компьютер пользователя приложение, реализованное в предыдущем пункте задания.

Требования к приложению:

1) Приложение-инсталлятор совместно с устанавливаемым приложением должно обеспечивать защиту программного продукта от несанкционированного тиражирования.

2) Приложение-инсталлятор должно иметь защиту от возможных атак на него.

**Задание III.**

Протестировать правильность работы систем защиты разработанных приложений посредством реализации тестовых атак выбранных 4-х типов.

# **2.** **Краткие теоретические сведения**

**Защищенное программирование. Типичные ошибки**

**Разработка защищенных программ** – процесс, требующий повышенного внимания на всех стадиях и уровнях. Это означает строгие требования безопасности, отличное понимание проблем защищенности и наличие группы оценки качества, которая может обнаруживать проблемы безопасности. Однако, чтобы решить эту проблему, нужно много учиться, и даже при этом, нельзя решить все проблемы безопасности одновременно.

Существует пять основных проблем, которые охватывают 90% уязвимостей программного обеспечения. Если вы учтете их, вы сможете на порядок увеличить защищенность ваших программ.

**Переполнение буфера**

**Переполнение буфера** – пожалуй, наиболее часто используемая хакерами ошибка программирования. Подробности того, как переполнение буфера может быть использовано для запуска произвольного кода, не входят в тему этой статьи, но вы легко найдете множество материала по данной теме (в частности, и на *securitylab.ru*). Достаточно сказать, что если программа удаленно подвержена переполнению буфера, хакер может получить полный контроль над системой. Первичная причина этой проблемы в том, что для вводимой информации используются статичные, фиксированного размера переменные, и злоумышленник может превысить размер отведенного под переменную буфера.

Ключ к решению этой проблемы состоит в проверке размера вводимой информации перед копированием ее в буфер. Если вводимые данные превышают размер буфера, необходимо вывести сообщение об ошибке в лог-файл и изменить работу программы. Это может оказаться длинным и утомительным процессом для достаточно большой программы. К счастью, существует множество инструментов для автоматической проверки этого условия. Однако эти инструменты только обнаружат проблему и не избавят программиста от необходимости анализа результатов и исправления ошибок.

Написание эксплоита для атаки с использованием переполнения буфера часто считается «черной магией», так как требует огромного опыта и знания операционных систем, компиляторов, ассемблера. В итоге, многие люди ошибочно полагают, что если так сложно определить и атаковать эту уязвимость, то существующий риск минимальный. Однако, даже самый беглый поиск по нескольким популярным сайтам, посвященным безопасности, даст ссылки на множество эксплоитов для практически всех коммерческих программ. Самый неопытный хакер, и то может просто скачать программу и совершить взлом, даже не задумываясь, как автор программы-эксплоита обнаружил и взломал переполнение буфера. Действительно, как это происходит?

Одна хакерская группа, например, в настоящее время исследует все расширения ISAPI в Microsoft IIS для определения условий переполнения буфера. Они методично анализируют все расширения ISAPI с помощью дебаггера, дизассемблера и специализированных хакерских программ, которые передают неожиданный вызов на Web сервер. Программы Microsoft – любимая цель хакеров вследствие широкой распространенности этих продуктов и непопулярности компании в определенных хакерских кругах. Однако, все компании – разработчики программ подвержены этому риску.

**Уязвимости форматной строки**

**Уязвимости форматной строки** – новый класс проблем безопасности, обнаруженный в последние пару лет. Форматные строки – это программные конструкции, используемые в языках C и C++ для форматирования ввода-вывода. Они содержат специальные идентификаторы (такие как %s для строк, %d для целых чисел), которые, в случае использования в злонамеренном вводе, могут открыть информацию о содержании стэка и используемых в функции переменных. В частности, опасный идентификатор %n может использоваться для изменения данных в памяти. Поскольку это позволяет хакерам делать почти то же, что и при переполнении буфера, результаты одинаковы: запуск произвольного кода.

Первичная причина уязвимости форматной строки в использовании функций переменного параметра в C/C++. Эту проблему можно решить с помощью проверки на правильность ввода и поиска ошибок в коде. Также можно использовать автоматические утилиты проверки кода для обнаружения ошибок типа: printf (string); будет рекомендовано заменить на printf (“%s”, string).

**Аутентификация**

**Аутентификация** – наиболее важный компонент в любой защищенной системе. Некорректная аутентификация пользователя сделает все остальные функции безопасности, такие как шифрование, аудит и авторизация, просто бесполезными. Наиболее типичная ошибка в аутентификации – использование слабых мандатов (credentials), что позволяет хакеру применить метод грубой силы (brute force), например, к взлому пароля. Кроме того, необходима строгая политика паролей для уменьшения вероятности их взлома. Требования к составу пароля зависят от уровня безопасности, требуемого приложением и его функциями. Обычно рекомендуется, чтобы пароли были длиной минимум в 8 символов и включали буквы, цифры и спецсимволы.

Многие приложения, особенно Web-приложения, используют аутентификаторы для определения зарегистрировавшегося пользователя. Аутентификаторы – это как «билеты», которые выдаются после ввода пользователем аутентификационной информации. Эти «билеты» могут использоваться для проверки подлинности в течение сессии вместо запроса имени пользователя и пароля. В Web-приложениях аутентификаторы часто сохраняются в куки (cookies). При создании приложения важно убедиться, что аутентификаторы не подвержены атакам грубой силы или предсказания.

**Авторизация**

**Авторизация** – это процесс разрешения или запрета доступа к определенному ресурсу, основанный на идентификации аутентификационной информации. Уязвимости авторизации являются типичными проблемами во многих приложениях.

*Наиболее частые ошибки:*

**Авторизация проведена некорректно**. Интерфейс обеспечивает доступ к ресурсу до тех пор, пока обеспечивается аутентификация. Обычно, в этом случае требуется некий идентификатор, и предполагается, что он не может быть угадан или изменен. Например, в одном банковском Web-приложении Web-сервер устанавливал переменную в Javascript в номер аккаунта. Простым изменением этой переменной стало возможно просматривать и изменять другие аккаунты.

**Слишком много доверия к пользовательскому вводу**. Например, HTTP куки – это один из типов пользовательского ввода в Web-приложение, который может быть изменен. Если приложение осуществляет авторизацию с помощью куки, оно может довериться фальсифицированной информации. Это может быть или имя пользователя, или запрашиваемый ресурс. Например, при испытании одного приложения была найдена переменная в куки с именем “admin”. Если она устанавливалась в “true”, приложение предоставляло нелегитимному пользователю администраторские права.

**Ошибки канонизации**. Часто приложения принимают решения об авторизации на основе аутентификационной информации и запрошенных ресурсов. Многие уязвимости безопасности относятся к ошибкам канонизации в имени ресурса. Например, хотя приложение может запретить доступ к \secure\secret.txt, оно может дать доступ к \public\..\secure\secret.txt на основании имени директории. Кодировки в Юникод и в шестнадцатеричном формате относятся к той же категории.

Криптография

Если криптография производится в приложении, чувствительность данных очень высока. Опытных программистов, имеющих полное математическое понимание криптографических алгоритмов, как это ни странно и печально, очень мало. Одна ошибка в разработке криптографического алгоритма или его реализации может полностью подорвать защищенность приложения. Многие программы используют алгоритмы промышленного стандарта, такие как RSA или blowfish, но не некорректно работают с памятью, что оставляет возможным кражу пароля и других данных в чистом виде. Очень рекомендуется использовать CryptoAPI и CAPICOM, встроенные в операционные системы Windows, или купить библиотеки третьих лиц для избежания проблем с использованием криптографии. Очень легко по ошибке попасть в ловушку, считая, что данные «выглядят» зашифровано, и потому защищены.

**Рекомендации для разработки защищенных программ**

К этому моменту вы, если вы являетесь разработчиком программ, уже можете задаться вопросом - при всех этих потенциальных проблемах как мне определить, не имеет ли моя программа какие-то уязвимости? Более того, если имеет, то как мне решить их? Ответ на первый вопрос прост. Проведите аудиты и тесты безопасности и анализ кода для определения уязвимостей безопасности в приложении. Ответ на второй вопрос гораздо сложнее. Создание защищенного программного обеспечения является проблемой, которую пытались разрешить многие люди в течение долгого времени. Вы не найдете здесь ответа, но найдете список рекомендаций, которые, как мы ми говорили, помогут вам избежать 90% типичных уязвимостей безопасности.

**Не доверяйте пользовательскому вводу**

Хакер зачастую атакует приложение через его внешние интерфейсы. Примеры внешних интерфейсов: сокеты, Web-формы, ввод командной строки, файлы и т.п. Многие эксплоиты посылают злонамеренные входные данные, чтобы заставить приложение вести себя непредусмотренным образом, например как при переполнении буфера. Хотя это и кажется легким, но распознавание всего ввода не всегда может оказаться таким уж простым. Например, в Web-приложении наиболее типичный пользовательский ввод – URI и данные для форм. Однако, хакер, который не использует стандартный Web-броузер, может манипулировать HTTP заголовками, куки, и скрытыми данными. Эти типы входных данных никогда не следует считать содержащими безопасные данные.

**Проверка правильности ввода**

Необходимо разработать описание достоверных данных для каждого ввода. Пример для случая аутентификации: имя пользователя должно содержать буквы и цифры, быть длиной 6-20 символов, и пароль должен содержать 6-20 печатаемых символов. Если клиент посылает переполнение буфера более 20 символов, приложение должно отсоединить пользователя. Если производится атака SQL injection, проверка запроса в приложении не должна пропустить его к SQL серверу. Такое недоверие к пользовательскому воду в комбинации с хорошей проверкой корректности ввода в корне предотвратит большинство атак. Ваше приложение может иметь уязвимости SQL injection и переполнения буфера, но при правильной проверке корректности ввода оно может стать невзламываемым для подобных атак.

## **3. Реализация задачи**

Программу реализовал на языке программирования **Python(PyQt)** в **Vscode**. Результат программы:

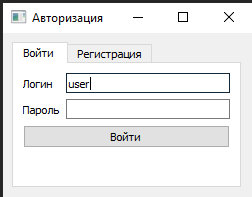


Рис 1. Окно авторизации.

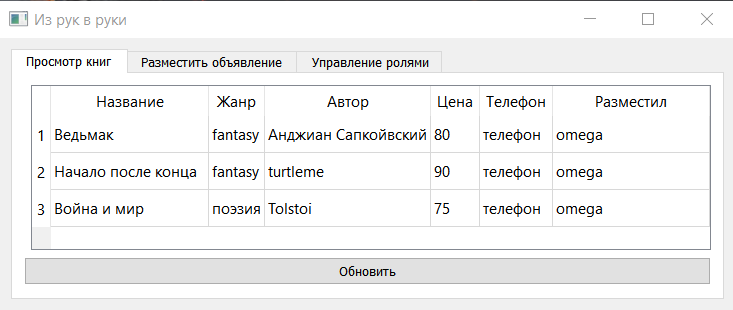


Рис 2. Окно просмотра книг.

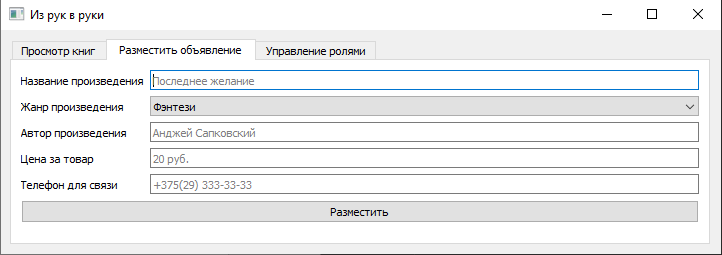


Рис 3. Окно размещения объявлений

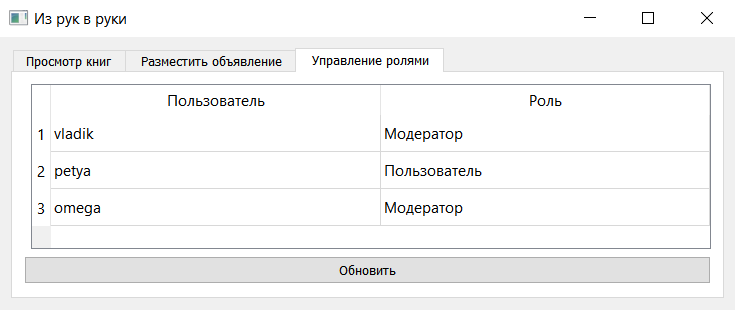


Рис 4. Окно управления ролями

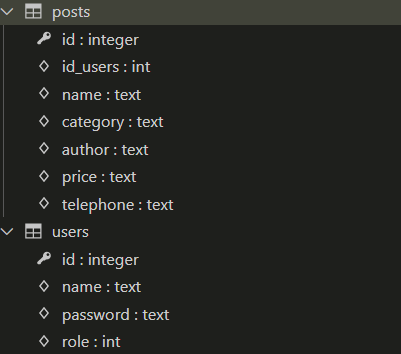


Рис 5. Схема базы данных

# 

# **Приложение. Текст программ**

import sqlite3

import socket

import sys

import threading

import json

import test

conn\_tcp = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_addr = ('127.0.0.1', 8888)

is\_finished = [False]

ban\_list = list()

*def* thread\_tcp():

    conn\_tcp.bind(server\_addr)

    conn\_tcp.listen()

    while not is\_finished[0]:

        try:

            conn, addr = conn\_tcp.accept()

            if addr in ban\_list:

                print('Ban connection', addr)

                conn.close()

            else:

                print('Get connection', addr)

                t = threading.Thread(*target*=thread\_connection, *args*=(conn, addr))

                t.start()

        except:

            ban\_list.append(addr)

            print('Connection lost', addr)

            conn.close()

*def* thread\_connection(*client\_socket*, *client\_addr*):

    try:

        while True:

            data = *client\_socket*.recv(2048).decode()

            print(sys.getsizeof(data))

            if sys.getsizeof(data) > 2048:

                print('Catch buffer overflow from', *client\_addr*)

*client\_socket*.close()

                return

            if not data:

                break

            print('GET FROM', *client\_addr*, data, sys.getsizeof(data), 'bytes')

            data = data\_menu(data, *client\_addr*)

            print('SEND TO', *client\_addr*, data)

*client\_socket*.sendall(data.encode())

    except:

        print('Connection lost', *client\_addr*)

        ban\_list.append(*client\_addr*)

    finally:

*client\_socket*.close()

*def* data\_menu(*data*: str, *addr*) -> str:

    conn\_sql = sqlite3.connect('new.db')

    cursor = conn\_sql.cursor()

    try:

*data* = json.loads(*data*)

    except:

*data* = ['bad request']

    if *data*[0] == 'in':

        res = cursor.execute(' SELECT name, password, role FROM users WHERE name = ? and password = ?', (*data*[1], *data*[2],)).fetchone()

        if res is not None:

*data* = [[res[0], res[1], res[2]], get\_posts(cursor, conn\_sql)]

            if res[2] == 2:

*data*.append(get\_users(cursor))

        else:

*data* = ['wrong input']

    elif *data*[0] == 'up':

        if cursor.execute(' SELECT COUNT(\*) FROM users WHERE *name* = ? ', (*data*[1],)).fetchone()[0] == 0:

            cursor.execute(' INSERT INTO users(*name*, *password*, *role*) VALUES (?, ?, 1)', (*data*[1], *data*[2],))

*data* = [[*data*[1], *data*[2], 1], get\_posts(cursor, conn\_sql)]

        else:

*data* = ['already exist']

    elif *data*[0] == 'books':

*data* = get\_posts(cursor, conn\_sql)

    elif *data*[0] == 'users':

        res = cursor.execute(' SELECT role FROM users WHERE name = ? and password = ? ', (*data*[1], *data*[2],)).fetchone()

        if res is not None and res[0] == 2:

*data* = get\_users(cursor)

        else:

*data* = ['permission denied']

    elif *data*[0] == 'create book':

        res = cursor.execute(' SELECT id FROM users WHERE name = ? and password = ? ', (*data*[1], *data*[2],)).fetchone()

        if res is not None:

            cursor.execute(' INSERT INTO posts(*id\_users*, *name*, *category*, *author*, *price*, *telephone*) VALUES(?, ?, ?, ?, ?, ?) ', (res[0], *data*[3], *data*[4], *data*[5], *data*[6], *data*[7],))

*data* = ['create book', *data*[3]]

        else:

*data* = ['no create book']

    elif *data*[0] == 'delete book':

        if cursor.execute(' SELECT role FROM users WHERE *name* = ? and password = ? ', (*data*[1], *data*[2],)).fetchone()[0] == 2:

            cursor.execute(' DELETE FROM posts WHERE *id* = ? ', (*data*[3], ))

*data* = ['delete book', *data*[3]]

        else:

            res = cursor.execute(' SELECT id\_users FROM posts WHERE id = ? ', (*data*[3],)).fetchone()[0]

            if cursor.execute(' SELECT COUNT(\*) FROM users WHERE *name* = ? and password = ? and id = ? ', (*data*[1], *data*[2], res,)).fetchone()[0] == 1:

                cursor.execute(' DELETE FROM posts WHERE *id* = ? ', (*data*[3],))

*data* = ['delete book', *data*[3]]

            else:

*data* = ['no delete book', *data*[3]]

    elif *data*[0] == 'update role':

        res = cursor.execute(' SELECT id, role FROM users WHERE name = ? and password = ? ', (*data*[1], *data*[2],)).fetchone()

        if res is not None:

            if str(res[0]) != str(*data*[3]) and res[1] == 2:

                role = cursor.execute(' SELECT role FROM users WHERE id = ? ', (*data*[3],)).fetchone()[0]

                role = 1 if role == 2 else 2

                cursor.execute(' UPDATE users SET *role* = ? WHERE id = ? ', (role, *data*[3],))

*data* = ['update role']

            else:

*data* = ['no update role', *data*[3]]

        else:

*data* = ['no update role', *data*[3]]

    cursor.close()

    conn\_sql.commit()

    conn\_sql.close()

    return json.dumps(*data*)

def get\_posts(*cur*, *conn*) -> list:

    posts = list()

    for row in *cur*.execute(" SELECT *id*, *name*, *category*, *author*, *price*, *telephone*, id\_users FROM posts "):

        row = list(row)

        cursor\_user = *conn*.cursor()

        user\_name = cursor\_user.execute(" SELECT name FROM users WHERE id = " + str(row[-1]) + "").fetchone()[0]

        cursor\_user.close()

        row[-1] = user\_name

        posts.append(row)

    return posts

*def* get\_users(*cur*) -> list:

    users = list()

    for row in *cur*.execute(" SELECT *id*, *name*, role FROM users "):

        users.append(list(row))

    return users

if \_\_name\_\_ == '\_\_main\_\_':

    t\_server = threading.Thread(*target*=thread\_tcp)

    t\_server.start()

    t\_server.join()

***Window Client:***

import sys

import socket

import json

from PyQt5.QtCore import \*

from PyQt5.QtWidgets import \*

import logging

# logger = logging.getLogger('logger')

# logger.setLevel(logging.info('INFO'),logger.error('ERROR'))

Log\_Format = "%(levelname)s %(asctime)s - %(message)s"

logging.basicConfig(*filename* = "logfile.log",

*filemode* = "w",

*format* = Log\_Format,

*level* = logging.ERROR)

logger = logging.getLogger()

logger.error("Our First Log Message")

conn\_tcp = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_addr = ('127.0.0.1', 8888)

info = list()

*class* AuthWindow(QMainWindow):

*def* \_\_init\_\_(*self*):

        QMainWindow.\_\_init\_\_(*self*)

*self*.setMinimumSize(QSize(70, 70))

*self*.setWindowTitle("Авторизация")

        layout = QVBoxLayout()

*self*.setLayout(layout)

        tabs = QTabWidget()

        tabs.addTab(*self*.sign\_in\_ui(), "Войти")

        tabs.addTab(*self*.sign\_up\_ui(), "Регистрация")

        layout.addWidget(tabs)

        w = QWidget()

        w.setLayout(layout)

*self*.setCentralWidget(w)

        w.show()

    # GUI для TAB входа

*def* sign\_in\_ui(*self*) -> QWidget:

        sign\_in\_tab = QWidget()

        layout = QFormLayout()

*self*.e\_name\_in = QLineEdit()

*self*.e\_pass\_in = QLineEdit()

        b\_si\_in = QPushButton('Войти', *self*)

        b\_si\_in.clicked.connect(*self*.on\_click\_sign\_in)

        layout.addRow('Логин', *self*.e\_name\_in)

        layout.addRow('Пароль', *self*.e\_pass\_in)

        layout.addRow(b\_si\_in)

        sign\_in\_tab.setLayout(layout)

        return sign\_in\_tab

    def sign\_in\_ui(*self*):

*def* get\_reg\_info():

            info[1] = QFormLayout()

            return QVBoxLayout()

        info = [QWidget(), QFormLayout(), QWidget(), QFormLayout()]

*self*.e\_name\_in = QLineEdit()

*self*.e\_pass\_in = QLineEdit()

*self*.e\_name\_reg = QLineEdit()

*self*.e\_pass\_reg = QLineEdit()

        b\_si\_in = QPushButton('Войти', *self*)

        b\_si\_in.clicked.connect(*self*.on\_click\_sign\_in)

        info[1].addRow('Логин', *self*.e\_name\_in)

        info[3].addRow('Войти без регистрации', *self*.e\_pass\_reg)

        info[1].addRow('Пароль', *self*.e\_pass\_in)

        info[1].addRow(b\_si\_in)

        info[0].setLayout(info[1])

        if info[0] is not info[2]:

            info[2] = get\_reg\_info()

        return info[0]

    # GUI для TAB регистрации

*def* sign\_up\_ui(*self*) -> QWidget:

        sign\_up\_tab = QWidget()

        layout = QFormLayout()

*self*.e\_name\_up = QLineEdit()

*self*.e\_pass\_up = QLineEdit()

*self*.e\_pass\_up\_check = QLineEdit()

        b\_si\_up = QPushButton('Регистрация', *self*)

        b\_si\_up.clicked.connect(*self*.on\_click\_sign\_up)

        layout.addRow('Логин', *self*.e\_name\_up)

        layout.addRow('Пароль', *self*.e\_pass\_up)

        layout.addRow('Повт. пароль', *self*.e\_pass\_up\_check)

        layout.addRow(b\_si\_up)

        sign\_up\_tab.setLayout(layout)

        return sign\_up\_tab

    # GUI Показать окно с ошибкой

    def show\_error\_msg(*self*, *title*: str, *msg*: str):

        msg\_box = QMessageBox()

        msg\_box.setIcon(QMessageBox.Critical)

        msg\_box.setText(*msg*)

        msg\_box.setWindowTitle(*title*)

        msg\_box.exec\_()

    # Вход через существующего пользователя

    @pyqtSlot()

*def* on\_click\_sign\_in(*self*):

        data = json.dumps(['in', *self*.e\_name\_in.text(), *self*.e\_pass\_in.text()]).encode()

        print('Send', data)

        try:

            data = json.loads(tcp\_data(data))

            print('Get', data)

            if data[0] == "wrong input":

*self*.show\_error\_msg('Ошибка', 'Неправильный логин или пароль')

            else:

                global info

                info = data

*self*.close()

*self*.window = WorkWindow()

*self*.window.show()

        except:

*self*.show\_error\_msg('Ошибка', 'Непредвиденный ответ с сервера!')

    # Вход через регистрацию

    @pyqtSlot()

*def* on\_click\_sign\_up(*self*):

        data = json.dumps(['up', *self*.e\_name\_up.text(), *self*.e\_pass\_up.text()]).encode()

        print('Send', data)

        try:

            data = json.loads(tcp\_data(data))

            print('Get', data)

            if data[0] == "already exist":

*self*.show\_error\_msg('Ошибка', 'Данный пользователь уже зарегистрирован')

            else:

                global info

                info = data

*self*.close()

*self*.window = WorkWindow()

*self*.window.show()

        except:

*self*.show\_error\_msg('Ошибка', 'Непредвиденный ответ с сервера!')

*class* WorkWindow(QMainWindow):

*def* \_\_init\_\_(*self*):

        QMainWindow.\_\_init\_\_(*self*)

        print(info)

*self*.setMinimumSize(QSize(70, 70))

*self*.setWindowTitle("Из рук в руки")

        layout = QVBoxLayout()

*self*.setLayout(layout)

        tabs = QTabWidget()

        tabs.addTab(*self*.show\_book\_ui(), "Просмотр книг")

        tabs.addTab(*self*.create\_book\_ui(), "Разместить объявление")

        if info[0][2] == 2:

            tabs.addTab(*self*.manage\_role\_ui(), "Управление ролями")

        layout.addWidget(tabs)

        w = QWidget()

        w.setLayout(layout)

*self*.setCentralWidget(w)

*def* show\_book\_ui(*self*) -> QWidget:

        show\_book\_tab = QWidget()

        layout = QFormLayout()

*self*.t\_book = QTableWidget()

*self*.t\_book.setColumnCount(6)

*self*.t\_book.setHorizontalHeaderLabels(["Название", "Жанр", "Автор", "Цена", "Телефон", "Разместил"])

*self*.update\_books\_ui()

*self*.t\_book.setEditTriggers(QAbstractItemView.*NoEditTriggers*)

*self*.t\_book.setSelectionBehavior(QAbstractItemView.*SelectRows*)

*self*.t\_book.itemDoubleClicked.connect(*self*.delete\_book)

*self*.t\_book.setSizeAdjustPolicy(QAbstractScrollArea.*AdjustToContentsOnFirstShow*)

        header = *self*.t\_book.horizontalHeader()

        header.setSectionResizeMode(0, QHeaderView.*Stretch*)

        for i in range(1, 6):

            header.setSectionResizeMode(i, QHeaderView.ResizeToContents)

*self*.t\_book.horizontalHeader().setStretchLastSection(True)

        layout.addWidget(*self*.t\_book)

        b\_update = QPushButton('Обновить', *self*)

        b\_update.clicked.connect(*self*.update\_books)

        layout.addRow(b\_update)

        show\_book\_tab.setLayout(layout)

        return show\_book\_tab

*def* create\_book\_ui(*self*) -> QWidget:

        create\_book\_tab = QWidget()

        layout = QFormLayout()

*self*.e\_name = QLineEdit()

*self*.e\_name.setPlaceholderText("Последнее желание")

        layout.addRow('Название произведения', *self*.e\_name)

*self*.c\_catg = QComboBox()

*self*.c\_catg.addItem("Фэнтези")

*self*.c\_catg.addItem("Детективы")

*self*.c\_catg.addItem("Ужасы")

*self*.c\_catg.addItem("Поэзия")

*self*.c\_catg.addItem("Фантастика")

*self*.c\_catg.addItem("Любовные романы")

*self*.c\_catg.addItem("Триллеры")

*self*.c\_catg.addItem("Комиксы и манга")

*self*.c\_catg.addItem("Проза")

        layout.addRow("Жанр произведения", *self*.c\_catg)

*self*.e\_auth = QLineEdit()

*self*.e\_auth.setPlaceholderText("Анджей Сапковский")

        layout.addRow('Автор произведения', *self*.e\_auth)

*self*.e\_pric = QLineEdit()

*self*.e\_pric.setPlaceholderText("20 руб.")

        layout.addRow('Цена за товар', *self*.e\_pric)

*self*.e\_phone = QLineEdit()

*self*.e\_phone.setPlaceholderText("+375(29) 333-33-33")

        layout.addRow('Телефон для связи', *self*.e\_phone)

        b\_create = QPushButton('Разместить', *self*)

        b\_create.clicked.connect(*self*.create\_book)

        layout.addRow(b\_create)

        create\_book\_tab.setLayout(layout)

        return create\_book\_tab

    def manage\_role\_ui(*self*) -> QWidget:

        manage\_role\_tab = QWidget()

        layout = QFormLayout()

*self*.t\_user = QTableWidget()

*self*.t\_user.setColumnCount(2)

*self*.t\_user.setHorizontalHeaderLabels(["Пользователь", "Роль"])

*self*.update\_users\_ui()

*self*.t\_user.setEditTriggers(QAbstractItemView.*NoEditTriggers*)

*self*.t\_user.setSelectionBehavior(QAbstractItemView.*SelectRows*)

*self*.t\_user.itemDoubleClicked.connect(*self*.change\_role)

*self*.t\_user.setSizeAdjustPolicy(QAbstractScrollArea.*AdjustToContentsOnFirstShow*)

        header = *self*.t\_user.horizontalHeader()

        header.setSectionResizeMode(0, QHeaderView.*Stretch*)

        for i in range(1, 2):

            header.setSectionResizeMode(i, QHeaderView.ResizeToContents)

*self*.t\_user.horizontalHeader().setStretchLastSection(True)

        layout.addWidget(*self*.t\_user)

        b\_update = QPushButton('Обновить', *self*)

        b\_update.clicked.connect(*self*.update\_users)

        layout.addRow(b\_update)

        manage\_role\_tab.setLayout(layout)

        return manage\_role\_tab

    # GUI Показать окно с ошибкой

*def* show\_error\_msg(*self*, *title*: str, *msg*: str):

        msg\_box = QMessageBox()

        msg\_box.setIcon(QMessageBox.Critical)

        msg\_box.setText(*msg*)

        msg\_box.setWindowTitle(*title*)

        msg\_box.exec\_()

    @pyqtSlot()

*def* update\_books\_ui(*self*):

        books = info[1]

*self*.t\_book.setRowCount(len(books))

        for i in range(len(books)):

*self*.t\_book.setItem(i, 0, QTableWidgetItem(books[i][1]))

*self*.t\_book.setItem(i, 1, QTableWidgetItem(books[i][2]))

*self*.t\_book.setItem(i, 2, QTableWidgetItem(books[i][3]))

*self*.t\_book.setItem(i, 3, QTableWidgetItem(books[i][4]))

*self*.t\_book.setItem(i, 4, QTableWidgetItem(books[i][5]))

*self*.t\_book.setItem(i, 5, QTableWidgetItem(books[i][6]))

    @pyqtSlot()

*def* update\_books(*self*):

        try:

            data = json.dumps(['books', info[0][0], info[0][1]]).encode()

            print('Send', data)

            data = json.loads(tcp\_data(data))

            print('Get', data)

            info[1] = data

*self*.update\_books\_ui()

        except:

*self*.show\_error\_msg('Ошибка', 'Ошибка при обновлении книг со стороны сервера')

    @pyqtSlot()

*def* update\_users\_ui(*self*):

        users = info[2]

*self*.t\_user.setRowCount(len(users))

        for i in range(len(users)):

            role\_name = 'Модератор' if users[i][2] == 2 else 'Пользователь'

*self*.t\_user.setItem(i, 0, QTableWidgetItem(users[i][1]))

*self*.t\_user.setItem(i, 1, QTableWidgetItem(role\_name))

    @pyqtSlot()

*def* update\_users(*self*):

        try:

            data = json.dumps(['users', info[0][0], info[0][1]]).encode()

            print('Send', data)

            data = json.loads(tcp\_data(data))

            print('Get', data)

            if data[0] == 'permission denied':

*self*.show\_error\_msg('Ошибка', 'Доступ запрещён')

            else:

                info[2] = data

*self*.update\_users\_ui()

        except:

*self*.show\_error\_msg('Ошибка', 'Ошибка при обновлении пользователей со стороны сервера')

    @pyqtSlot()

*def* delete\_book(*self*):

        if *self*.t\_book.item(*self*.t\_book.currentRow(), 5).text() == info[0][0] or info[0][2] == 2:

            try:

                data = json.dumps(['delete book', info[0][0], info[0][1], info[1][*self*.t\_book.currentRow()][0]]).encode()

                print('Send', data)

                data = json.loads(tcp\_data(data))

                print('Get', data)

                if data[0] == 'no delete book':

*self*.show\_error\_msg('Ошибка', 'Удалить не получилось!')

*self*.update\_books()

            except:

*self*.show\_error\_msg('Ошибка', 'Ошибка при удалении со стороны сервера')

    @pyqtSlot()

*def* create\_book(*self*):

        try:

            data = json.dumps(['create book', info[0][0], info[0][1], *self*.e\_name.text(), *self*.c\_catg.currentText(), *self*.e\_auth.text(), *self*.e\_pric.text(), *self*.e\_phone.text()]).encode()

            print('Send', data)

            data = json.loads(tcp\_data(data))

            print('Get', data)

            if data[0] == 'no create book':

*self*.show\_error\_msg('Ошибка', 'Разместить объявление не получилось!')

*self*.update\_books()

        except:

*self*.show\_error\_msg('Ошибка', 'Ошибка при размещении со стороны сервера')

    @pyqtSlot()

*def* change\_role(*self*):

        if info[0][2] == 2:

            try:

                data = json.dumps(['update role', info[0][0], info[0][1], info[2][*self*.t\_user.currentRow()][0]]).encode()

                print('Send', data)

                data = json.loads(tcp\_data(data))

                print('Get', data)

                if data[0] == 'no update role':

*self*.show\_error\_msg('Ошибка', 'Изменить роль не вышло!')

*self*.update\_users()

            except:

*self*.show\_error\_msg('Ошибка', 'Ошибка при обновлении роли со стороны сервера')

*def* tcp\_init():

    try:

        conn\_tcp.connect(server\_addr)

    except socket.error:

        print('Server not responding')

*def* tcp\_data(*data*: str) -> str:

    try:

        conn\_tcp.send(*data*)

        return conn\_tcp.recv(2048).decode()

    except socket.error:

        return 'bad request'

if \_\_name\_\_ == "\_\_main\_\_":

    app = QApplication(sys.argv)

    window = AuthWindow()

    window.show()

    tcp\_init()

    app.exec\_()

***Console Attack.py:***

import sys

import socket

import json

conn\_tcp = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_addr = ('127.0.0.1', 8888)

*def* tcp\_init():

    try:

        conn\_tcp.connect(server\_addr)

    except socket.error:

        print('Server not responding')

*def* tcp\_data(*data*: str) -> str:

    try:

        conn\_tcp.send(*data*)

        return conn\_tcp.recv(10000).decode()

    except socket.error:

        return 'bad request'

if \_\_name\_\_ == "\_\_main\_\_":

    tcp\_init()

    # SQL-*Attack*

    data = json.dumps(["users", "vladik", "'random' OR 1 = 1;--"]).encode()

    data = json.loads(tcp\_data(data))

    print('Get', data)

    data = json.dumps(['up', "'test'; DELETE FROM users;", '']).encode()

    data = json.loads(tcp\_data(data))

    print('Get', data, '\n')

    # Privilege-*Attack*

    data = json.dumps(['users', 'vlad', 'password']).encode()

    data = json.loads(tcp\_data(data))

    print('Get', data, '\n')

    # Buffer-*Attack*

    try:

        data = json.dumps(['a' \* (4096 - 37)]).encode()

        print('Size: ', sys.getsizeof(data), ' bytes')

        data = json.loads(tcp\_data(data))

        print('Get', data, '\n')

    except:

        print('Connection closed!')