МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ »

Кафедра «Информационная безопасность систем и технологий»

Политехнический институт Факультет приборостроения, информационных технологий и электроники	Кафедра «Информационная безопасность систем и технологий»
Специальность 10.05.03 –	Информационная безопасность автоматизированных систем
Специализация –	Защищенные автоматизированные системы управления

ДИПЛОМНАЯ РАБОТА

на тему

УЧЕБНЫЙ АППАРАТНО-ПРОГРАММНЫЙ КОМПЛЕКС ПО ИССЛЕДОВАНИЮ МЕХАНИЗМОВ БЕЗОПАСНОСТИ БЕСПРОВОДНЫХ СЕТЕЙ НА ОСНОВЕ ОДНОПЛАТНЫХ КОМПЬЮТЕРОВ

Студент Ревнивцев Кирилл Викторович
Руководитель Мали В.А.
Нормоконтролёр Фатеев А.Г.
Рецензент ведущий инженер АО "ПНИЭИ" Горячев Д.В.

Работа допущена к защите (протокол заседания кафедры от 15.06.20 № 13)
Заведующий кафедрой С.Л. Зефиров

Работа защищена с отметкой отлично (протокол заседания ГЭК от 26.06.20 № 15)
Секретарь ГЭК А.П. Иванов

Пенза 2020

Оглавление

Введение.....…….5
1 Проектирование аппаратной части учебного комплекса……...6
2 Проектирование программной части учебного комплекса……13
3 Разработка аппаратной части учебного комплекса……..17
4 Разработка программной части учебного комплекса…….…..21
5 Тестирования учебного комплекса……..……...38
6 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие……...54
6.1 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации……54
6.2 Методы и средства физической культуры, снижающие воздействие вредных психофизиологических факторов ……57
Заключение.....….....60
Список использованных источников…..…..61
Приложение А Исходный код программы «Студент»……62
Приложение А Исходный код программы «Администратор»……...70

Термины и определения

В настоящей работе применяются следующие термины с соответствующими определениями:

Коммутатор – устройство, предназначенное для соединения нескольких узлов компьютерной сети в пределах одного или нескольких сегментов сети;

WPA/WPA2 – программа сертификации устройств беспроводной связи;

WPS – стандарт (и одноимённый протокол) полуавтоматического создания беспроводной сети Wi-Fi;

Wi-Fi – технология беспроводной локальной сети с устройствами на основе стандартов IEEE 802.11;

DoS – атака на вычислительную систему с целью довести её до отказа, то есть создание таких условий, при которых добросовестные пользователи системы не смогут получить доступ к предоставляемым системным ресурсам (серверам), либо этот доступ будет затруднён.

Сокращения

В работе применяются следующие сокращения:

— ТД — точка доступа;

— ЗИ — защита информации;

— ИБ — информационная безопасность;

— WPS – Wi-Fi protected setup;

— DoS – Denial of Service;

— VLAN – Virtual Local Area Network.

Введение

Задача подготовки специалистов по защите информации и компьютерной безопасности, обладающих достаточным уровнем компетенций как по администрированию современных систем безопасности, так и по организационной поддержке принимаемых решений по безопасности на сегодняшний момент является особенно актуальной. Широкое применение беспроводных сетей в информационных системах требует от специалистов дополнительных практических навыков и умений по обеспечению информационной безопасности.

В настоящее время обеспечению безопасности беспроводных сетей уделяется недостаточное внимание. Используемые в беспроводных сетях алгоритмах шифрования и аутентификации часто подвержены различным атакам. Беспроводные сети используются практически везде и поэтому для обеспечения общего уровня информационной безопасности сети необходимо изучать механизмы и способы защиты от действий злоумышленников. Поэтому необходимо создавать учебные стенды для изучения механизмов атак на беспроводные сети и способов защиты от них.

1 Проектирование аппаратной части учебного комплекса

Самой частой используемой технологией беспроводной передачи данных является Wi-Fi. Для создания непрерывной и независимой информационной сети данная технология используется повсеместно: общественные места, различные организации и предприятия, учебные заведения и т. д. В последнее время большинство используемых человеком носимых устройств также поддерживают использование Wi-Fi. Использование беспроводных сетей включает новые риски для информационной сети, например автоматическое подключение к открытым Wi-Fi сетям на мобильных телефонах и других устройств поддерживающих использование Wi-Fi, что может привезти к утечке конфиденциальной информации. В связи с этим изучение безопасности беспроводных сетей очень актуально. Малозащищенные беспроводные сети дают злоумышленнику возможность для проведения своих атак.

В настоящее время наиболее актуальными сценариями атак на беспроводные сети являются:

— избыточное покрытые сети;

— использование словарных ключей безопасности;

— использование механизма WPS. [2]

Избыточное покрытие сети может привести к тому, что радиус действия беспроводной точки доступа будет больше размеров контролируемой зоны организации. Злоумышленник, используя мощные Wi-Fi-адаптеры для работы в различных частотных диапазонах, все направленные антенны, микрокомпьютеры для создания поддельной точки доступа, оборудование для скрытного анализа беспроводных сетей и всевозможное ПО, позволяющее проводить активный анализ безопасности Wi-Fi-сетей сможет реализовать большой спектр несанкционированных воздействий на беспроводную сети, целями которых могут быть получение информации об используемых алгоритмах шифрования и механизмах аутентификации. Вся полученная

информация в дальнейшем может быть использована непосредственно для проведения атак на инфраструктуру организации. Например, атака вида «отказ в обслуживании» DoS Wi-Fi. Реализация данной атаки происходит за счёт постоянной отправки пакетов деаутентификации, что приводит к тому, что клиенты не могут подключиться к беспроводной сети и взаимодействовать с сетью в штатном режиме. Данная атака не приводит к утечки конфиденциальной информации.

Взлом WPA/WPA2 словарного пароля методом грубой силы является универсальной атакой на беспроводные сети. К её плюсам можно отнести то, что данная атака применима ко всем точкам доступа, которые используют программу сертификации устройств беспроводной сети WPA/WPA2. Минусами данной угрозы является то, что для реализации атаки необходимо, чтобы к точке доступа беспроводной сети были подключены клиенты. Так же стоит учесть, что расшифровка пароля ведётся методом грубой силы (метод перебора), что значит, что при достаточно сложном пароле обойти защиту Wi-Fi точки доступа за приемлемое время не получится.

Взлом WPS пин-кода является менее универсальной атакой так как количество точек доступа к беспроводной сети очень мало, но является отличным примером для изучения в образовательных целях. WPS - стандарт и одноимённый протокол полуавтоматического создания беспроводной сети Wi-Fi. Если в точке доступа активирован WPS с PIN, то подобрать PIN-код для подключения возможно за приемлемое время. PIN-код состоит из восьми цифр — следовательно, существует 10^8 (100 000 000) вариантов PIN-кода для подбора. Однако количество вариантов можно существенно сократить. Последняя цифра PIN-кода представляет собой контрольную сумму, которая можно вычислить на основании первых семи цифр. Таким образом количество вариантов сокращается до 10^7 (10 000 000). Недостаток протокола позволяет разделить PIN-код на две части, 4 начальные цифры и 3 последующие и затем

необходимо проверять каждую часть на корректность. Если PIN-код разбить на две части:

— 10^4 (10 000) вариантов для первой половины;

— 10^3 (1000) для второй.

В итоге это составляет всего лишь 11 000 вариантов для полного перебора, что в более 9000 раз меньше исходного числа вариантов 10^8.[3]

Для создания учебного стенда по исследованию механизмов безопасности беспроводных сетей необходимо, чтобы в каждое автоматизированное рабочее место компьютерного класса было оснащено следующим комплектом оборудования и программного обеспечения: сетевой адаптер с поддержкой технологии Wi-Fi, программное обеспечение по исследованию механизмов безопасности беспроводных сетей, точка доступа Wi-Fi в качестве объекта исследования.

Реализация учебного стенда в таком составе имеет ряд недостатков:

— избыточность использованного оборудования;

— сложность централизованного управления;

— сложность организации работы нескольких рабочих мест одновременно при реализации модели «нарушитель-защищающийся».

В настоящее время самым популярным одноплатным компьютером является Raspberry Pi. Большая часть моделей одноплатных компьютеров Raspberry Pi распространяется полностью собранным на четырёхслойной печатной плате. Мини-компьютер Raspberry Pi выпускался или выпускается в следующих версиях: «A», «A+», «B», «B+», «2B», «Zero», «Zero W», «3B», «3B+», «3A+» и «4B». Версии мини-компьютера «3B», «3B+» и «3A+» оснащены 4-х ядерным процессором Cortex-A53 (ARM v8), с частотой процессора 1,2 ГГц у «3B» и 1,4 ГГц у «3B+» и «3A+», с оперативной памятью 1 ГБ у «3B» и «3B+» и 512 МБ у «3A+».

Raspberry Pi работает в основном на операционных системах, основанных на Linux ядре. Также возможна установка Windows 10 IOT. Официально поддерживаемые следующие операционные системы:

— Raspbian;
— Pidora – Fedora для Raspberry Pi;
— OpenELEC;
— OSMC;
— RISC OS;
— Windows 10 IOT.

Модель Raspberry Pi 3B содержит Wi-Fi модуль, что позволяет подключаться к точкам доступа или использовать одноплатный компьютер как Wi-Fi точку доступа.

Встроенный Wi-Fi модель поддерживает стандарт 802.11n на модели «3B». В версии «3B+» адаптер поддерживает стандарты IEEE 802.11.b/g/n/ac, а так же имеет возможность работать с 2.4 ГГц и 5 ГГц сетями.

Поддержка различных операционных систем позволяет использовать одноплатный компьютер в различных целях.

Так же данный одноплатный компьютер имеет разъём 8P8C, что дает возможность включения учебного комплекса в локальную сеть.

Учебный комплекс по исследованию механизмов безопасности беспроводных сетей будет основана на описанном выше одноплатном компьютере.

Учебный комплекс должен находиться в отдельной VLAN сети для того, чтобы изолировать трафик учебного комплекса от трафика учебной лаборатории.

Реализация учебного стенда в таком составе имеет ряд недостатков:

— избыточность использованного оборудования;
— сложность централизованного управления;

— сложность организации работы нескольких рабочих мест одновременно при реализации модели «нарушитель-защищающийся».

Предлагаемая модель архитектуры учебного стенда предусматривает возможность его реализации в рамках компьютерного класса с проводными автоматизированными рабочими местами, объединенными в локальную сеть с помощью коммутатора. Архитектура учебного комплекса приведена на рисунке 1.

Учебный стенд содержит общую аппаратную часть в виде коммутатора учебного стенда с поддержкой стандарта 802.1Q и двух точек доступа. Точки доступа реализованы на базе одноплатного компьютера Raspberry Pi. На каждой из них установлено специализированное программное обеспечение, которое позволяет подключаться к ним удаленно с помощью протокола SSH и производить необходимые административные действия.

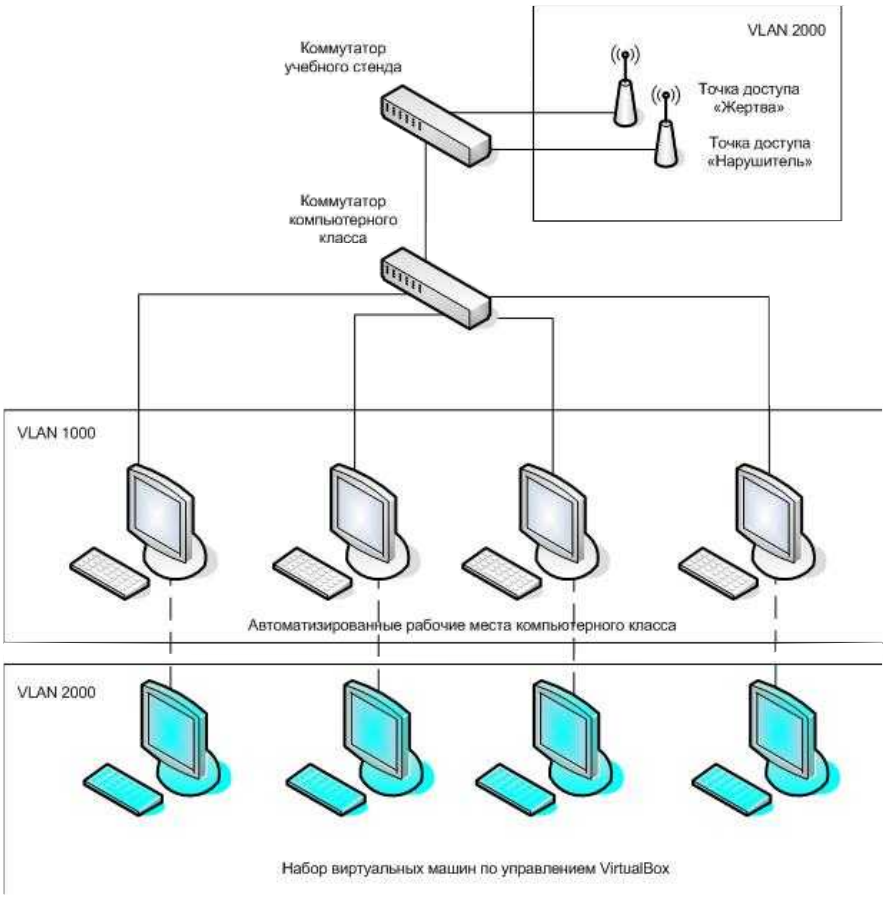

Рисунок 1 — Архитектура учебного комплекса

Точка доступа «Жертва» выступает в роли уязвимой точки доступа беспроводной сети, на которой возможны реализации различных сценариев обеспечения информационной безопасности беспроводной сети, в том числе с ошибками администрирования, уязвимостями. Точка доступа «Нарушитель» содержит программное обеспечение по исследованию механизмов безопасности беспроводных сетей, с помощью которого можно проводить воздействия на точку доступа «Жертва».

На автоматизированные рабочие места компьютерного класса установлено программное обеспечение виртуализации для операционных систем Oracle Virtual Box. С помощью данного программного обеспечения создается набор виртуальных машин с установленным клиентским программным обеспечением для удаленного подключения к точкам доступа. Особенностью программного средства виртуализации VirtualBox является то, что в режиме настройки сетевого интерфейса возможно подключение к виртуальным локальным сетям (VLAN) компьютерного класса. Для организации доступа на коммутаторе компьютерного класса и коммутаторе учебного стенда создаются две VLAN: VLAN1000 для работы проводных клиентов и VLAN2000 для доступа к компонентам учебного стенда. Такая архитектура исключает доступ к точкам доступа с автоматизированных рабочих мест компьютерного класса.

На основе приведенной выше архитектуры учебного стенда была разработана диаграмма развертывания. Результат представлен на рисунке 2.

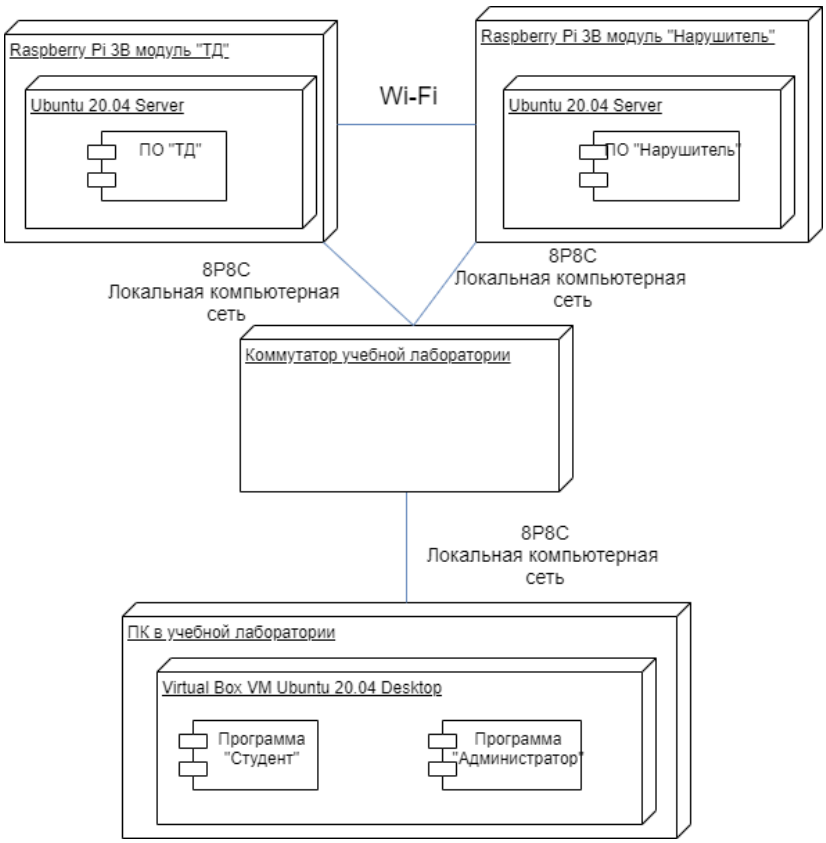

Рисунок 2 — Диаграмма развертывания учебного комплекса

2 Проектирование программной части учебного комплекса

На основе приведенной выше архитектуры учебного стенда была разработана диаграмма вариантов использования учебного комплекса, диаграмма последовательности действий пользователей при взаимодействии с программной частью учебного комплекса, алгоритм функционирования механизмов обеспечения информационной безопасности учебного ресурса, алгоритм взаимодействия программного модуля подключения к учебному комплексу с модулем клиентской части. Результат приведён на рисунках с 3 по 6.

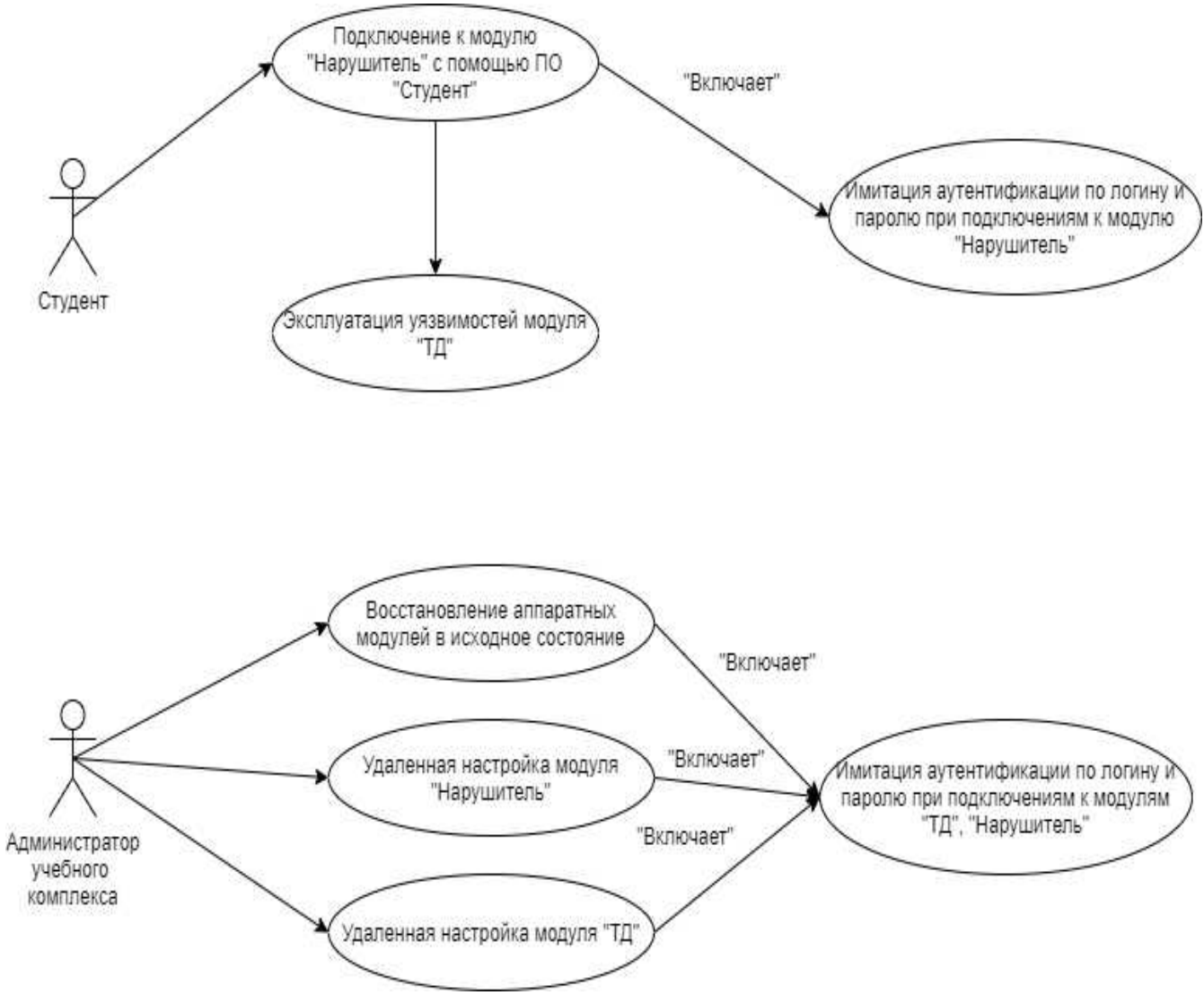

Рисунок 3 — Диаграмма вариантов использования учебного комплекса

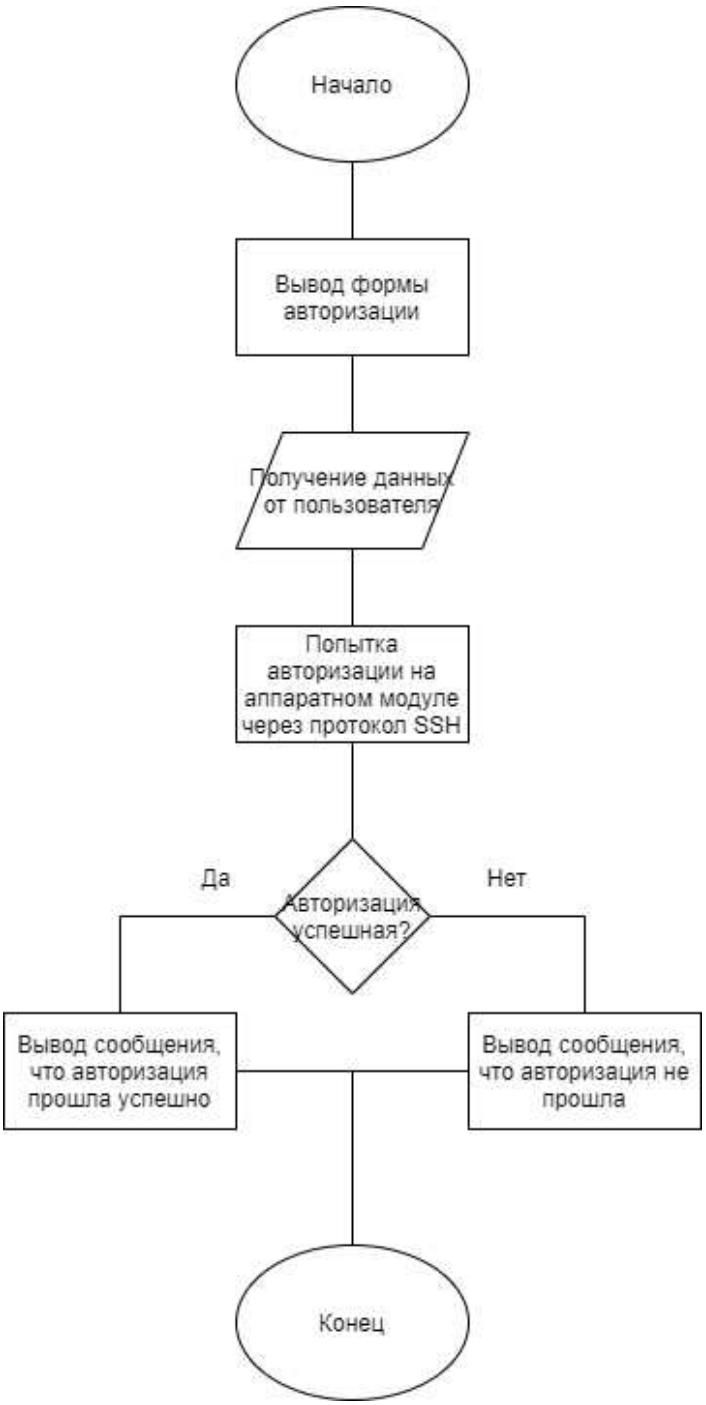

Рисунок 4 — Алгоритм функционирования механизмов обеспечения информационной безопасности учебного ресурса

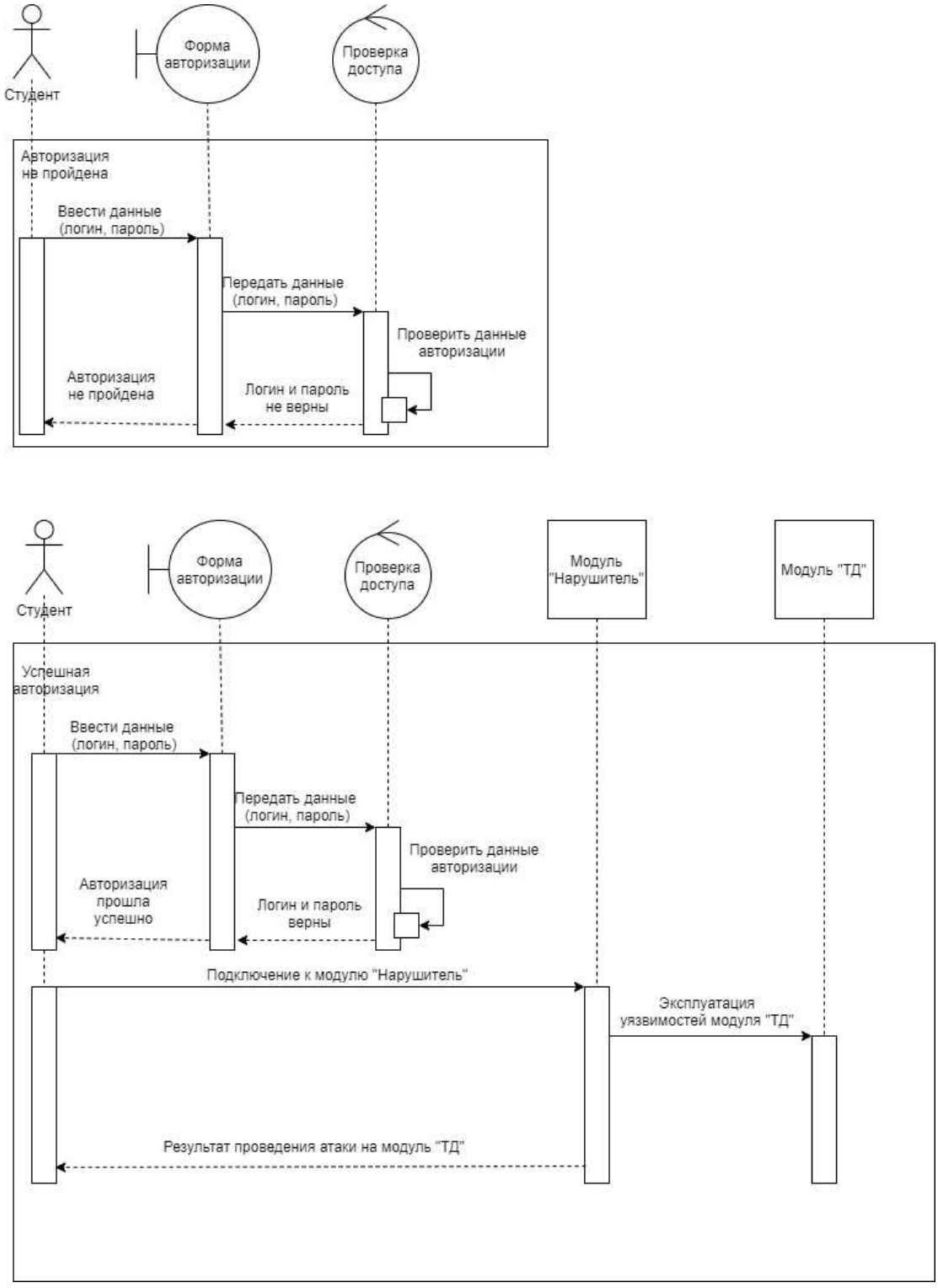

Рисунок 5 — Диаграмма последовательности действий пользователей при взаимодействии с программной частью учебного комплекса

Рисунок 6 — Алгоритм взаимодействия программного модуля подключения к учебному комплексу с модулем клиентской части

3 Разработка аппаратной части учебного комплекса

В ходе выполнения выпускной квалификационной работы было использовано два одноплатных компьютера «Raspberry Pi 3B». Для создания Wi-Fi ТД было использовано программное обеспечение «hostapd». Данное программное обеспечение позволяет создать программную точку доступа Wi-Fi, превращая пользовательские сетевые карты в точки доступа и сервера аутентификации.

Данное программное обеспечение имеет следующие возможности:

— создавать точку доступа;

— создавать несколько точек доступа на одной сетевой карте, если есть аппаратная поддержка этого;

— создавать две точки доступа на двух различных сетевых картах;

— создавать точки доступа с поддержкой 2.4 ГГц и 5 ГГц одновременно на одной сетевой карте.

Данное программное обеспечение не может:

— создавать несколько точек доступа на разных каналах связи на одной сетевой карте. Несколько точек доступа на одной карте будут использовать один и тот же канал;

— назначать IP-адреса устройствам, подключающимся к точке доступа, для этого необходим dhcp-сервер;

— присваивать IP-адрес самой точке доступа.

Для назначение IP-адресов устройствам, подключающимся к точке доступе необходимо запустить dhcp-server на том же интерфейсе, на котором находиться точка доступа.

В рамках выполнения выпускной квалификационной работы было разработано два конфигурационных файла программного обеспечения hostapd.

Исходный код конфигурационного файла настройки точки доступа с поддержкой WPS:

```
interface=wlan0
ssid=Vuln_AP
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_passphrase=12345678
#wpa_psk_file=/etc/hostapd.psk
wps_state=2
ap_setup_locked=0
wps_pin_requests=/var/run/hostapd_wps_pin_req
device_name=USB2.0 Hub
manufacturer=Company
model_name=WAP
model_number=123
serial_number=12345
device_type=6-0050F204-1
os_version=01020300
config_methods=label display push_button keypad
hw_mode=g
wmm_enabled=0
#macaddr_acl=0
auth_algs=3
ignore_broadcast_ssid=0
ieee80211n=1
channel=6
ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
#driver=rt2800usb
ap_pin=12345670
```

```
eap_server=1
ap_max_inactivity=300
```

Исходный код конфигурационного файла настройки точки доступа без поддержкой WPS:

```
interface=wlan0
ssid=Vuln_AP
hw_mode=g
channel=6
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
ieee80211n=1
```

Для того, чтобы у пользователей точки доступа был доступ в интернет был настроен NAT. Для этого был выполнено перенаправление сетевых портов Raspberry Pi. Был настроен межсетевой экран модуля «ТД».

Были выполнены следующие команды для настройки NAT:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --
state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
sudo sh -c "iptables-save > /etc/iptables.rules"
sudo nano /etc/rc.local
```

Далее был настроен DHCP-сервер для раздачи IP-адресов, подключающимся клиентам точки доступа.

Конфигурационный файл DHCP-сервера:

```
subnet 192.168.100.0 netmask 255.255.255.0 {
range 192.168.100.101 192.168.100.199;
option broadcast-address 192.168.100.255;
option routers 192.168.100.1;
default-lease-time 600;
max-lease-time 7200;
option domain-name "local";
option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

Так же был настроен беспроводной интерфейс на котором будет находиться будущая точка доступа Wi-Fi.

Конфигурационный файл беспроводного интерфейса:

```
iface wlan0 inet static
address 192.168.100.1
netmask 255.255.255.0
```

4 Разработка программной части учебного комплекса

В процессе выполнения выпускной квалификационной работы были реализованы следующие программы:

— программа «Студент»;

— программа «Администратор».

Программа «Студент» состоит из следующих основных логических частей:

— функциональная часть;

— графическая реализация окна авторизации;

— графическая реализация основного окна программы.

Графическая реализация представляет собой программный код, отвечающий за создание графического интерфейса программы.

Функциональная часть программы представляет собой программный код, отвечающий за обработку событий графического интерфейса и вызов соответствующих функций.

Код функциональной части программы «Студент»:

```
import PyQt5
from auth_student import Ui_Dialog
import sys
from main_client import Ui_MainWindow
import time
import paramiko
import os
client=paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy
())
class Main(PyQt5.QtWidgets.QMainWindow):
    def __init__(self,login,password,parent=None,):
        self.login=login
        self.password=password
        client.connect(hostname="192.168.1.78", username=
self.login, password=self.password, port="22")
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_MainWindow()
```

```
        self.ui.setupUi(self)
        self.changeButtonState(False)
        self.ui.textEdit.setReadOnly(True)
        self.ui.checkButton.clicked.connect(self.checkCon
nect)
        self.ui.crackWPAButton.clicked.connect(self.crack
WPA)
        self.ui.dosButton.clicked.connect(self.dos)
        self.ui.crackWPSButton.clicked.connect(self.crack
WPS)
        self.timer = PyQt5.QtCore.QTimer()
        self.timer.setInterval(1000)
        self.timer.timeout.connect(self.test_update)
        self.timer.start()
    def test_update(self):
        with client.invoke_shell() as ssh:
            time.sleep(2)
            result=ssh.recv(50000).decode("ascii")
            self.refresh_text(result)
        self.timer.stop()
        self.changeButtonState(True)
    def refresh_text(self,mystring):
        self.ui.textEdit.append(mystring)
        PyQt5.QtGui.QGuiApplication.processEvents()
    def checkConnect(self):
        self.changeButtonState(False)
        response = os.system("ping -c 1 192.168.1.78")
        if response == 0:
            self.mbox("Connected","Success")
        else:
            self.mbox("Failed to connect","Fail")
        self.changeButtonState(True)
    def crackWPA(self):
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('./
crack_wpa2.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            if "(100%)" in  line:
                break
            self.refresh_text(line)
        self.changeButtonState(True)
```

```
    def crackWPS(self):
        print("WPS")
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('./
crack_wps.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            if "Time taken" in  line:
                break
            self.refresh_text(line)
        self.changeButtonState(True)
    def dos(self):
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('./dos.sh')
        timeout = time.time() + 15
        while(True):
            line = sout.readline().strip('\n')
            if not line:
                continue
            if time.time()>timeout:
                break
            self.refresh_text(line)
        self.changeButtonState(True)

    def changeButtonState(self,state):
        self.ui.checkButton.setEnabled(state)
        self.ui.crackWPAButton.setEnabled(state)
        self.ui.crackWPSButton.setEnabled(state)
        self.ui.dosButton.setEnabled(state)
    def mbox(self,body, title):
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()
class AuthWin(PyQt5.QtWidgets.QDialog):
    def __init__(self,parent=None):
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.password_input.setEchoMode(PyQt5.QtWidget
s.QLineEdit.Password)
        self.ui.cancel_Button.clicked.connect(self.cancel
Clicked)
```

```
        self.ui.connect_Button.clicked.connect(self.conne
ctClicked)
    def connectClicked(self):
        loginText=self.ui.login_input.text()
        passwordText=self.ui.password_input.text()
        try:
            client.connect(hostname="192.168.1.78", usern
ame=loginText, password=passwordText, port="22")
            self.mbox("Succesfull","Success")
            client.close()
            self.accept()
        except paramiko.ssh_exception.AuthenticationExcep
tion:
            self.mbox("Fail","Invalid data")
    def cancelClicked(self):
        sys.exit(app.exec_())
    def mbox(self,body, title):
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()

if __name__ == "__main__":
    app=PyQt5.QtWidgets.QApplication(sys.argv)
    authAppExec= AuthWin()
    if authAppExec.exec_() == PyQt5.QtWidgets.QDialog.Acc
epted:
        login=authAppExec.ui.login_input.text()
        password=authAppExec.ui.password_input.text()
        mainAppExec= Main(login,password)
        mainAppExec.show()
    sys.exit(app.exec_())
```

Функциональная часть программы вызывает функции графической реализации окна авторизации и основного окна программы.

В ходе выполнения программы пользователю показывается окно авторизации и представляется возможность ввести данные для авторизации в модуле «Нарушитель». После успешной авторизации открывается основное окно программы и выполняется подключение к модулю «Нарушитель». Пользователю предоставляется окно вывода команд и 4 функциональной кнопки. Для выполнения проверки доступности модуля «Нарушитель»

пользователю необходимо нажать на кнопку «Check connect». Программа проверяет доступность модуля с помощью отправки ICMP пакетов.

Код функции кнопки «Check connect»:

```
def checkConnect(self):
        self.changeButtonState(False)
        response = os.system("ping -c 1 192.168.1.78")
        if response == 0:
            self.mbox("Connected","Success")
        else:
            self.mbox("Failed to connect","Fail")
        self.changeButtonState(True)
```

После выполнения данной функции пользователю выведется окно с результатом проверки.

Для эксплуатации уязвимости аутентификации при использования словарных паролей в Wi-Fi сети с технологией WPA/WPA2 пользователю необходимо нажать на кнопку «Crack WPA2 Password». Программа выполнит атаку «грубой силы» и выведет пароль в окно вывода.

Код функции кнопки «Crack WPA2 Password»:

```
def crackWPA(self):
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('./
crack_wpa2.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            if "(100%)" in  line:
                break
            self.refresh_text(line)
        self.changeButtonState(True)
```

После выполнения данной функции пользователь может увидеть подобранный пароль в окне вывода программы.

Для эксплуатации уязвимости стандартов семейства 802.11 к атаке "отказ в обслуживании" пользователю необходимо нажать на кнопку «DoS». Программа выполнит атаку вида «отказ в обслуживании».

Код функции кнопки «DoS»:

```
    def dos(self):
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('./dos.sh')
        timeout = time.time() + 15
        while(True):
            line = sout.readline().strip('\n')
            if not line:
                continue
            if time.time()>timeout:
                break
            self.refresh_text(line)
        self.changeButtonState(True)
```

После выполнения данной функции пользователь может увидеть результат выполнения в окне вывода программы.

Для эксплуатации уязвимости стандарта WPS к подбору PIN-кода пользователю необходимо нажать на кнопку «Crack WPS». Программа выполнит атаку вида «Pixie Dust».

Код функции кнопки «Crack WPS»:

```
def crackWPS(self):
        print("WPS")
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('./
crack_wps.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            if "Time taken" in  line:
                break
            self.refresh_text(line)
        self.changeButtonState(True)
```

После выполнения данной функции пользователь может увидеть подобранный WPS код в окне вывода программы.

Полный код программы «Студент» приведен в приложении А.

Программа «Администратор» состоит из следующих основных логических частей:

— функциональная часть;

— графическая реализация окна авторизации;

— графическая реализация основного окна программы.

Графическая реализация представляет собой программный код, отвечающий за создание графического интерфейса программы.

Функциональная часть программы представляет собой программный код, отвечающий за обработку событий графического интерфейса и вызов соответствующих функций.

Код функциональной части программы «Студент»:

```
import PyQt5
from auth_admin import Ui_Dialog_AP
from auth_student import Ui_Dialog
import sys
from main_admin import Ui_MainWindow
import time
import paramiko
import os

client=paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy
())
class Main(PyQt5.QtWidgets.QMainWindow):
    def __init__(self,loginAP,passwordAP,loginClient,pass
wordClient,parent=None,):
        self.loginAP=loginAP
        self.passwordAP=passwordAP
        self.loginClient=loginClient
        self.passwordClient=passwordClient
        client.connect(hostname="192.168.1.78", username=
self.loginClient, password=self.passwordClient, port="22"
)
        self.AP=paramiko.SSHClient()
        self.AP.set_missing_host_key_policy(paramiko.Auto
AddPolicy())
        self.AP.connect(hostname="192.168.1.161", usernam
e=self.loginAP, password=self.passwordAP, port="22")
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.changeButtonState(False)
```

```
        self.ui.textEdit.setReadOnly(True)
        self.timer = PyQt5.QtCore.QTimer()
        self.timer.setInterval(1000)
        self.timer.timeout.connect(self.test_update)
        self.timer.start()
        self.ui.checkConnecClientButton.clicked.connect(s
elf.checkConnecClient)
        self.ui.checkConnectAPButton.clicked.connect(self
.checkConnectAP)
        self.ui.rebootAPButton.clicked.connect(self.reboo
tAP)
        self.ui.rebootClientButton.clicked.connect(self.r
ebootClient)
        self.ui.resetConfigClientButton.clicked.connect(s
elf.resetConfigClient)
        self.ui.resetConfigAPButton.clicked.connect(self.
resetConfigAP)
        self.ui.disableWPSButton.clicked.connect(self.dis
ableWPS)
        self.ui.enableWPSButton.clicked.connect(self.enab
leWPS)
        self.ui.reloadAPSoftButton.clicked.connect(self.r
eloadAPSoft)
    def checkConnecClient(self):
        self.changeButtonState(False)
        response = os.system("ping -c 1 192.168.1.78")
        if response == 0:
            self.mbox("Connected","Success")
        else:
            self.mbox("Failed to connect","Fail")
        self.changeButtonState(True)
    def checkConnectAP(self):
        self.changeButtonState(False)
        response = os.system("ping -c 1 192.168.1.161")
        if response == 0:
            self.mbox("Connected","Success")
        else:
            self.mbox("Failed to connect","Fail")
        self.changeButtonState(True)
    def rebootAP(self):
        self.changeButtonState(False)
        self.refresh_text("Точка доступа будет перезагруж
ена. Перезайдите в приложение.")
```

```
        time.sleep(5)
        sin,sout,serr = self.AP.exec_command('sudo reboot
now')
        sys.exit(app.exec_())
    def rebootClient(self):
        self.changeButtonState(False)
        self.refresh_text("Клиент будет перерагружен. Пер
езайдите в приложение.")
        time.sleep(5)
        sin,sout,serr = client.exec_command('sudo reboot
now')
        sys.exit(app.exec_())
    def resetConfigClient(self):
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('bash resetCl
ient.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def resetConfigAP(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('bash resetC
onfigAP.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def disableWPS(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('./
disableWPS.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
```

```
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def enableWPS(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('./
enableWPS.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def reloadAPSoft(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('bash reload
AP.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def test_update(self):
        with self.AP.invoke_shell() as ssh:
            time.sleep(2)
            result=ssh.recv(50000).decode("ascii")
            self.refresh_text(result)
        self.timer.stop()
        self.changeButtonState(True)
    def refresh_text(self,mystring):
        self.ui.textEdit.append(mystring)
        PyQt5.QtGui.QGuiApplication.processEvents()

    def changeButtonState(self,state):
        self.ui.checkConnecClientButton.setEnabled(state)
        self.ui.checkConnectAPButton.setEnabled(state)
        self.ui.rebootAPButton.setEnabled(state)
        self.ui.rebootClientButton.setEnabled(state)
```

```
        self.ui.resetConfigClientButton.setEnabled(state)
        self.ui.resetConfigAPButton.setEnabled(state)
        self.ui.disableWPSButton.setEnabled(state)
        self.ui.enableWPSButton.setEnabled(state)
        self.ui.reloadAPSoftButton.setEnabled(state)
    def mbox(self,body, title):
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()
class AuthWinAP(PyQt5.QtWidgets.QDialog):
    def __init__(self,parent=None):
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_Dialog_AP()
        self.ui.setupUi(self)
        self.ui.password_input.setEchoMode(PyQt5.QtWidget
s.QLineEdit.Password)
        self.ui.cancel_Button.clicked.connect(self.cancel
Clicked)
        self.ui.connect_Button.clicked.connect(self.conne
ctClicked)
    def connectClicked(self):
        loginText=self.ui.login_input.text()
        passwordText=self.ui.password_input.text()
        try:
            client.connect(hostname="192.168.1.161", user
name=loginText, password=passwordText, port="22")
            self.mbox("Succesfull","Success")
            client.close()
            self.accept()
        except paramiko.ssh_exception.AuthenticationExcep
tion:
            self.mbox("Fail","Invalid data")
    def cancelClicked(self):
        sys.exit(app.exec_())
    def mbox(self,body, title):
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()
class AuthWin(PyQt5.QtWidgets.QDialog):
    def __init__(self,parent=None):
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.password_input.setEchoMode(PyQt5.QtWidget
```

```
s.QLineEdit.Password)
        self.ui.cancel_Button.clicked.connect(self.cancel
Clicked)
        self.ui.connect_Button.clicked.connect(self.conne
ctClicked)
    def connectClicked(self):
        loginText=self.ui.login_input.text()
        passwordText=self.ui.password_input.text()
        try:
            client.connect(hostname="192.168.1.78", usern
ame=loginText, password=passwordText, port="22")
            self.mbox("Succesfull","Success")
            client.close()
            self.accept()
        except paramiko.ssh_exception.AuthenticationExcep
tion:
            self.mbox("Fail","Invalid data")
    def cancelClicked(self):
        sys.exit(app.exec_())
    def mbox(self,body, title):
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()
if __name__ == "__main__":
    app=PyQt5.QtWidgets.QApplication(sys.argv)
    authAppExecAP=AuthWinAP()
    authAppExecClient=AuthWin()
    if authAppExecAP.exec_() == PyQt5.QtWidgets.QDialog.A
ccepted:
        loginAP=authAppExecAP.ui.login_input.text()
        passwordAP=authAppExecAP.ui.password_input.text()
        if authAppExecClient.exec_() == PyQt5.QtWidgets.Q
Dialog.Accepted:
            loginClient=authAppExecClient.ui.login_input.
text()
            passwordClient=authAppExecClient.ui.password_
input.text()
            mainAppExec= Main(loginAP,passwordAP,loginCli
ent,passwordClient)
            mainAppExec.show()
    sys.exit(app.exec_())
```

Функциональная часть программы вызывает функции графической реализации окна авторизации и основного окна программы.

В ходе выполнения программы пользователю показывается окно авторизации и представляется возможность ввести данные для авторизации в модуле «Нарушитель», затем появляется вторая форма авторизации для подключения к модулю «ТД» После успешной авторизации открывается основное окно программы и выполняется подключение к модулю «Нарушитель» и «ТД». Пользователю предоставляется окно вывода команд и 9 функциональных кнопок. Для выполнения проверки доступности модуля «Нарушитель» пользователю необходимо нажать на кнопку «Check connect client». Программа проверяет доступность модуля с помощью отправки ICMP пакетов.

Код функции кнопки «Check connect client»

```
def checkConnecClient(self):
        self.changeButtonState(False)
        response = os.system("ping -c 1 192.168.1.78")
        if response == 0:
            self.mbox("Connected","Success")
        else:
            self.mbox("Failed to connect","Fail")
        self.changeButtonState(True)
```

После выполнения данной функции пользователю выведется окно с результатом проверки.

Для выполнения проверки доступности модуля «ТД» пользователю необходимо нажать на кнопку «Check connect AP». Программа проверяет доступность модуля с помощью отправки ICMP пакетов.

Код функции кнопки «Check connect AP»

```
def checkConnectAP(self):
        self.changeButtonState(False)
        response = os.system("ping -c 1 192.168.1.161")
        if response == 0:
            self.mbox("Connected","Success")
        else:
            self.mbox("Failed to connect","Fail")
        self.changeButtonState(True)
```

После выполнения данной функции пользователю выведется окно с результатом проверки.

Для перезагрузки модуля «ТД» пользователю необходимо нажать на кнопку «Reboot AP». Программа отошлёт команду перезагрузки системы.

Код функции кнопки «Reboot AP»:

```
 def rebootAP(self):
        self.changeButtonState(False)
        self.refresh_text("Точка доступа будет перезагруж
ена. Перезайдите в приложение.")
        time.sleep(5)
        sin,sout,serr = self.AP.exec_command('sudo reboot
now')
        sys.exit(app.exec_())
```

Результат работы функции будет выведено в окно программы. Пользователю будет необходимо будет перезагрузить программу для повторной авторизации модуля «ТД».

Для перезагрузки модуля «Нарушитель» пользователю необходимо нажать на кнопку «Reboot Client». Программа отошлёт команду перезагрузки системы.

Код функции кнопки «Reboot Client»:

```
def rebootClient(self):
        self.changeButtonState(False)
        self.refresh_text("Клиент будет перераГружен. Пер
езайдите в приложение.")
        time.sleep(5)
        sin,sout,serr = client.exec_command('sudo reboot
now')
        sys.exit(app.exec_())
```

Результат работы функции будет выведено в окно программы. Пользователю будет необходимо будет перезагрузить программу для повторной авторизации модуля «Нарушитель».

Для возвращения в исходное состояние модуля «Нарушитель» пользователю необходимо нажать кнопку «Reset config client». Программа запустит программу по восстановлению системы в исходное состояние.

Код функции кнопки «Reset config client»:

```
def resetConfigClient(self):
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('bash resetCl
ient.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
```

Результат работы функции будет приведен в окне программы.

Для возвращения в исходное состояние модуля «ТД» пользователю необходимо нажать кнопку «Reset config AP». Программа запустит программу по восстановлению системы в исходное состояние.

Код функции кнопки «Reset config AP»:

```
def resetConfigAP(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('bash resetC
onfigAP.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
```

Результат работы функции будет приведен в окне программы.

Для выключение стандарт WPS на модуле «ТД» пользователю необходимо нажать кнопку «Disable WPS». Программа перезагрузит программное обеспечение модуля «ТД» и настроит Wi-Fi точку с новыми параметрами.

Код функции «Disable WPS»:

```
 def disableWPS(self):
```

```
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('./
disableWPS.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
```

Результат работы функции будет приведен в окне программы.

Для включения стандарта WPS на модуле «ТД» пользователю необходимо нажать кнопку «Enable WPS». Программа перезагрузит программное обеспечение модуля «ТД» и настроит Wi-Fi точку с новыми параметрами.

Код функции «Enable WPS»:

```
def enableWPS(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('./
enableWPS.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
```

Результат работы функции будет приведен в окне программы.

Для перезагрузки программного обеспечения модуля «ТД» пользователю необходимо нажать кнопку «Reload AP soft». Программа перезагрузит программное обеспечения модуля «ТД».

Код функции «Reload AP soft»:

```
def reloadAPSoft(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('bash reload
AP.sh')
        while True:
```

```
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
```

Результат работы функции будет приведен в окне программы.

Полный код программы «Администратор» приведен в приложении Б.

5 Проведение тестирования учебного комплекса

Было проведено тестирование программного обеспечения «Студент». Тестирование проводилось в рабочей среде на виртуальной машине с установленной операционной системой Ubuntu 20.04 Desktop.

Было проведено тестирование формы авторизации для подключения к модулю «Нарушитель». Для прохождения авторизации необходимо было ввести логин и пароль.

Ожидаемый результат при введение верных данных: сообщение об успешной авторизации и открытие основного окна программы.

Результат приведен на рисунке 7 и 8.

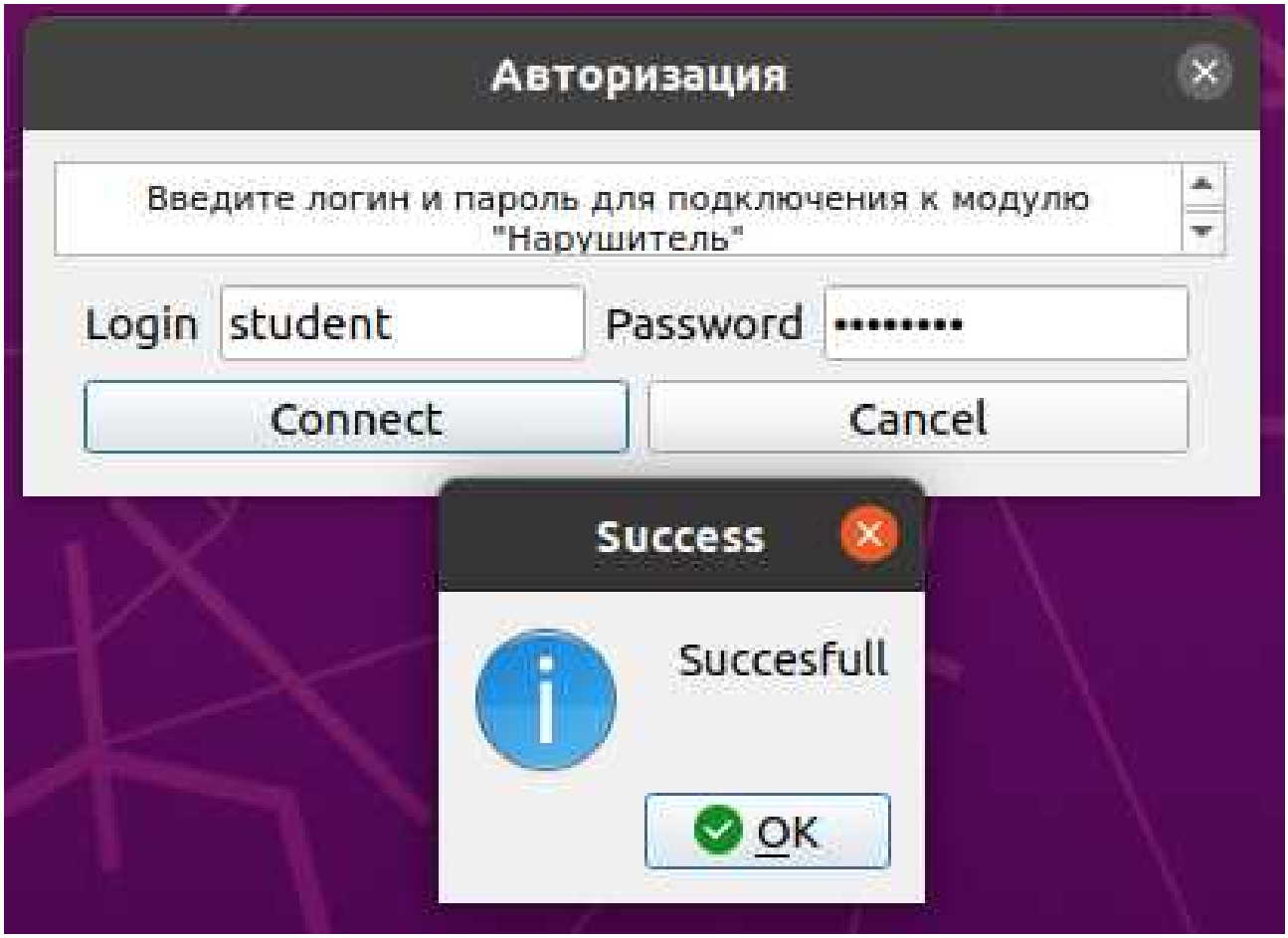

Рисунок 7- Вывод сообщения об успешной авторизации при подключении к модулю «Нарушитель»

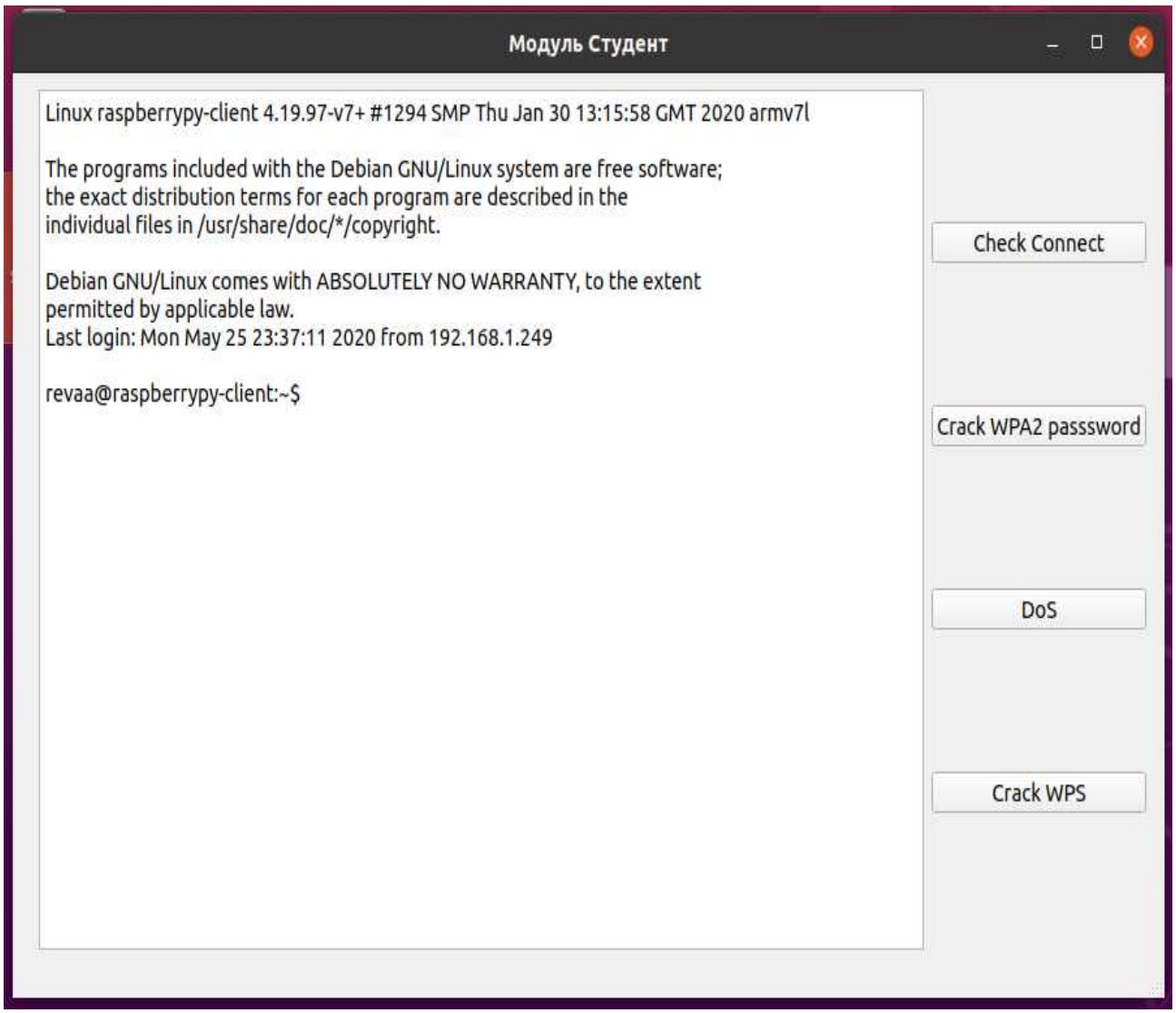

Рисунок 8 — Вывод основного окна программы

Фактический результат, представленный на рисунках 7,8, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что форма авторизации реализована корректно.

Было проведено тестирование формы авторизации при вводе не верных данных.

Ожидаемый результат при вводе не верных данных: сообщение о не успешной авторизации, вывод окна авторизации.

Результат приведён на рисунке 9.

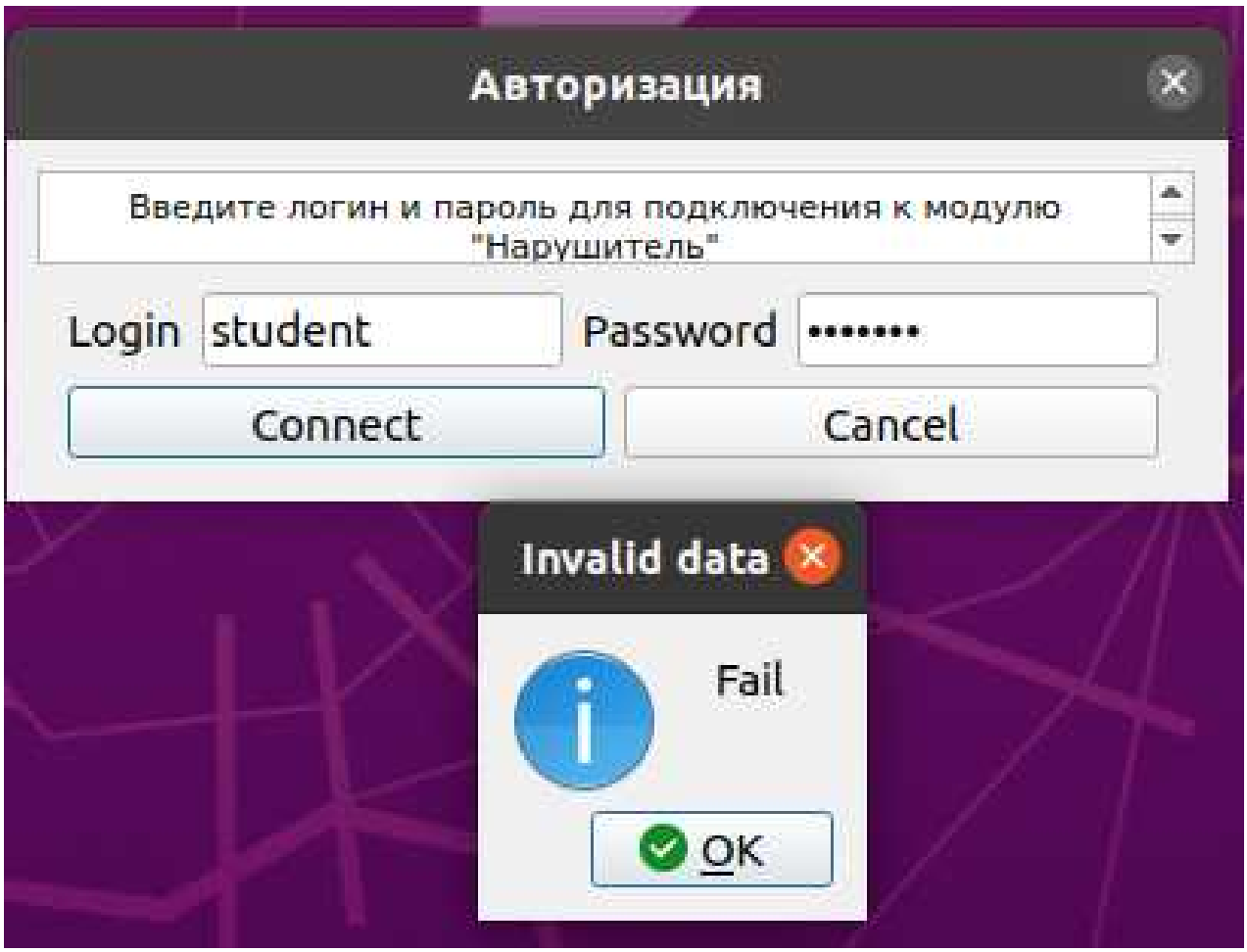

Рисунок 9 — Вывод окна об ошибке при авторизации в модуле «Нарушитель»

Фактический результат, представленный на рисунке 9, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что форма авторизации реализована корректно.

Было проведено тестирование функции проверки соединения модуля «Нарушитель». Для проверки доступности модуля необходимо нажать кнопку «Check connect».

Ожидаемые результат: вывод сообщения о доступности модуля «Нарушитель».

Результат приведён на рисунке 10.

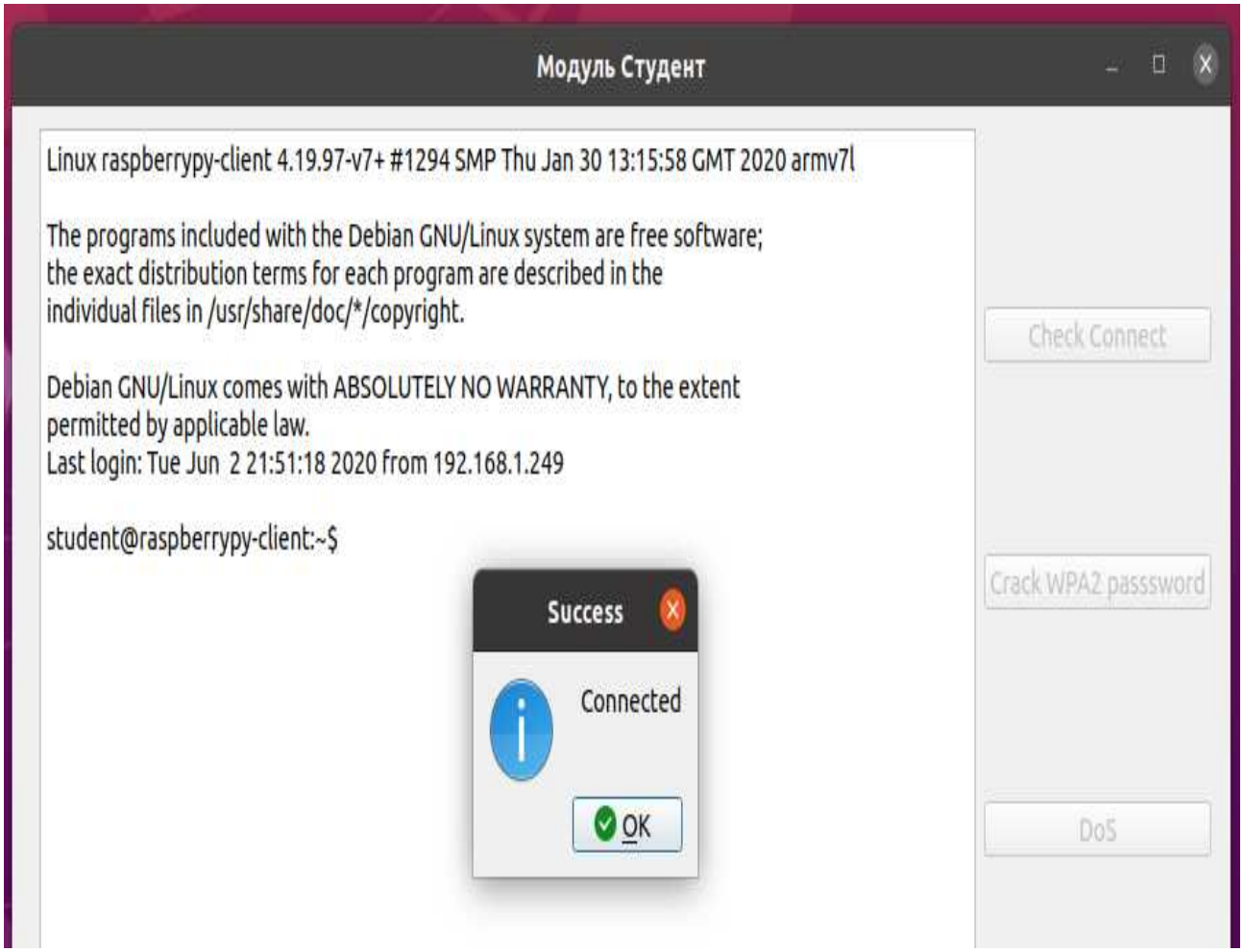

Рисунок 10 — Вывод сообщения о доступности модуля «Нарушитель»

Фактический результат, представленный на рисунке 10, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция проверки соединения реализована корректно.

Было проведено тестирование функции эксплуатации уязвимости аутентификации при использования словарных паролей в Wi-Fi сети с технологией WPA/WPA2. Для проверки доступности модуля необходимо нажать кнопку «Crack WPA2 password».

Ожидаемый результат: вывод в окно программы пароля Wi-Fi уязвимого модуля «ТД».

Результат представлен на рисунке 11.

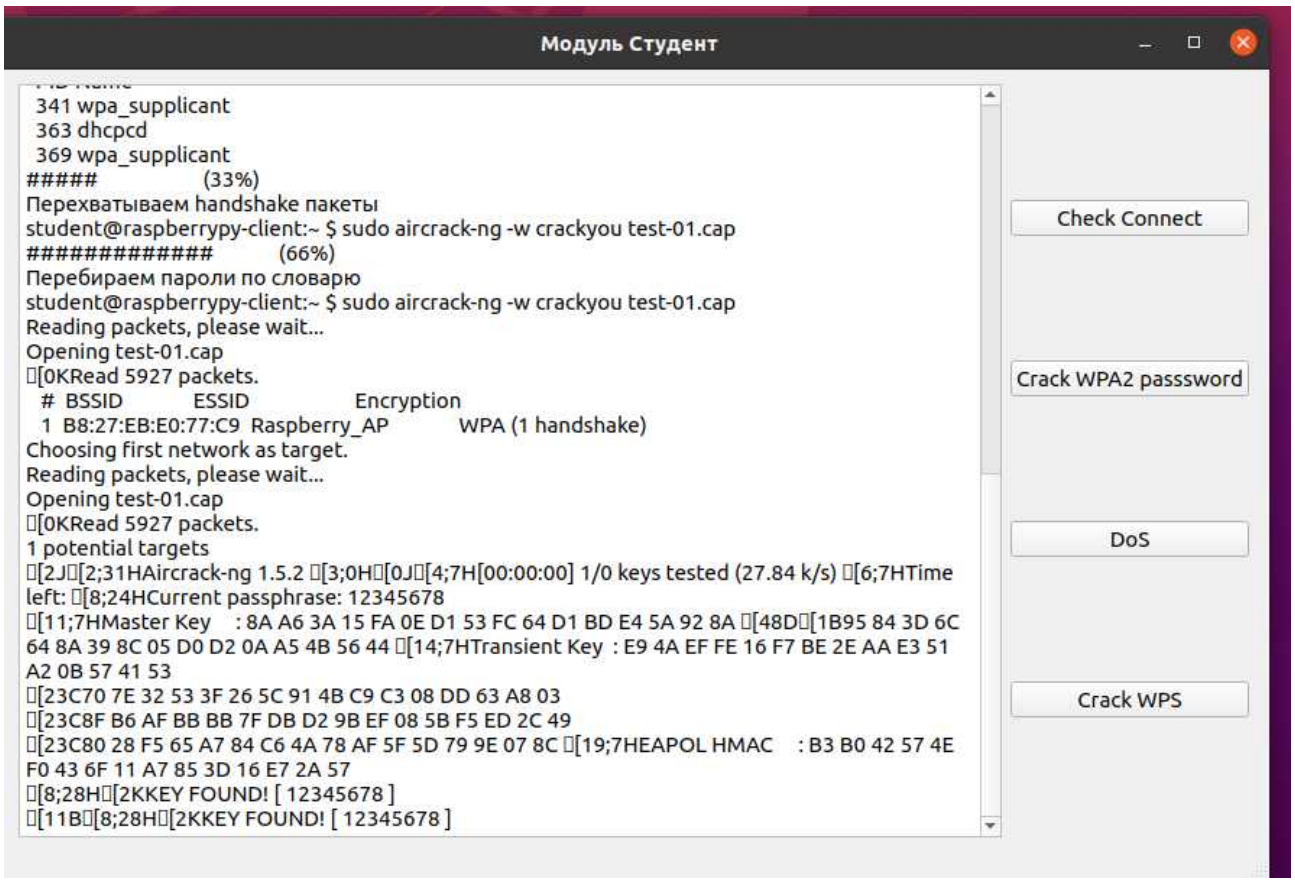

Рисунок 11 — Результат работы функции эксплуатации уязвимости аутентификации при использования словарных паролей в Wi-Fi сети с технологией WPA/WPA2

Фактический результат, представленный на рисунке 11, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция эксплуатации уязвимости аутентификации при использования словарных паролей в Wi-Fi сети с технологией WPA/WPA2 реализована корректно.

Было проведено тестирование функции эксплуатации уязвимости стандартов семейства 802.11 к атаке "отказ в обслуживании". Для проверки доступности модуля необходимо нажать кнопку «DoS».

Ожидаемый результат: вывод результата выполнения функции

Результат представлен на рисунке 12.

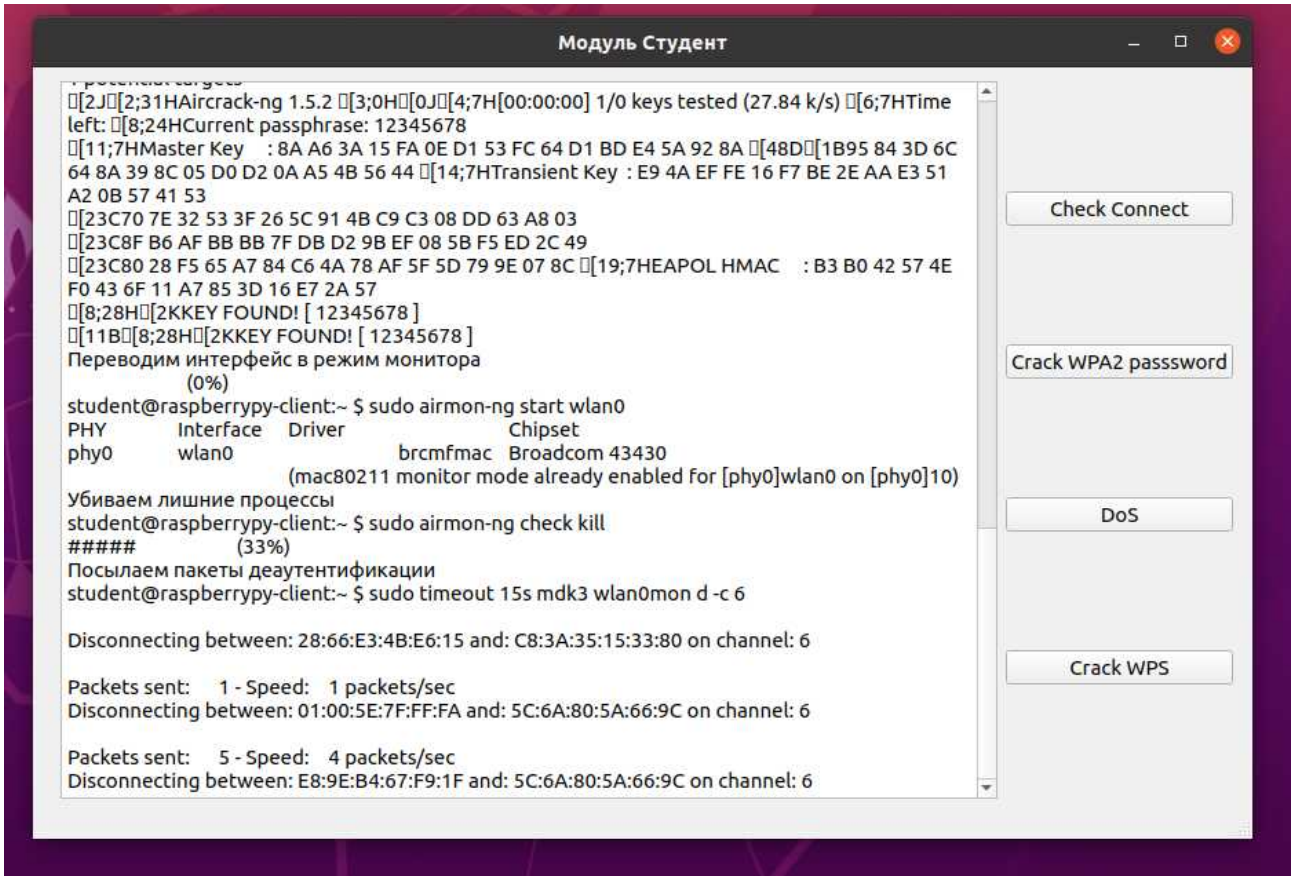

Рисунок 12 — Результат выполнения функции эксплуатации уязвимости стандартов семейства 802.11 к атаке "отказ в обслуживании"

Фактический результат, представленный на рисунке 12, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция эксплуатации уязвимости стандартов семейства 802.11 к атаке "отказ в обслуживании" реализована корректно.

Было проведено тестирование функции эксплуатации уязвимости стандарта WPS к подбору PIN-кода. Для проверки доступности модуля необходимо нажать кнопку «Crack WPS».

Ожидаемый результат: вывод пин-кода WPS уязвимой точки доступа модуля «ТД».

Результат представлен на рисунке 13

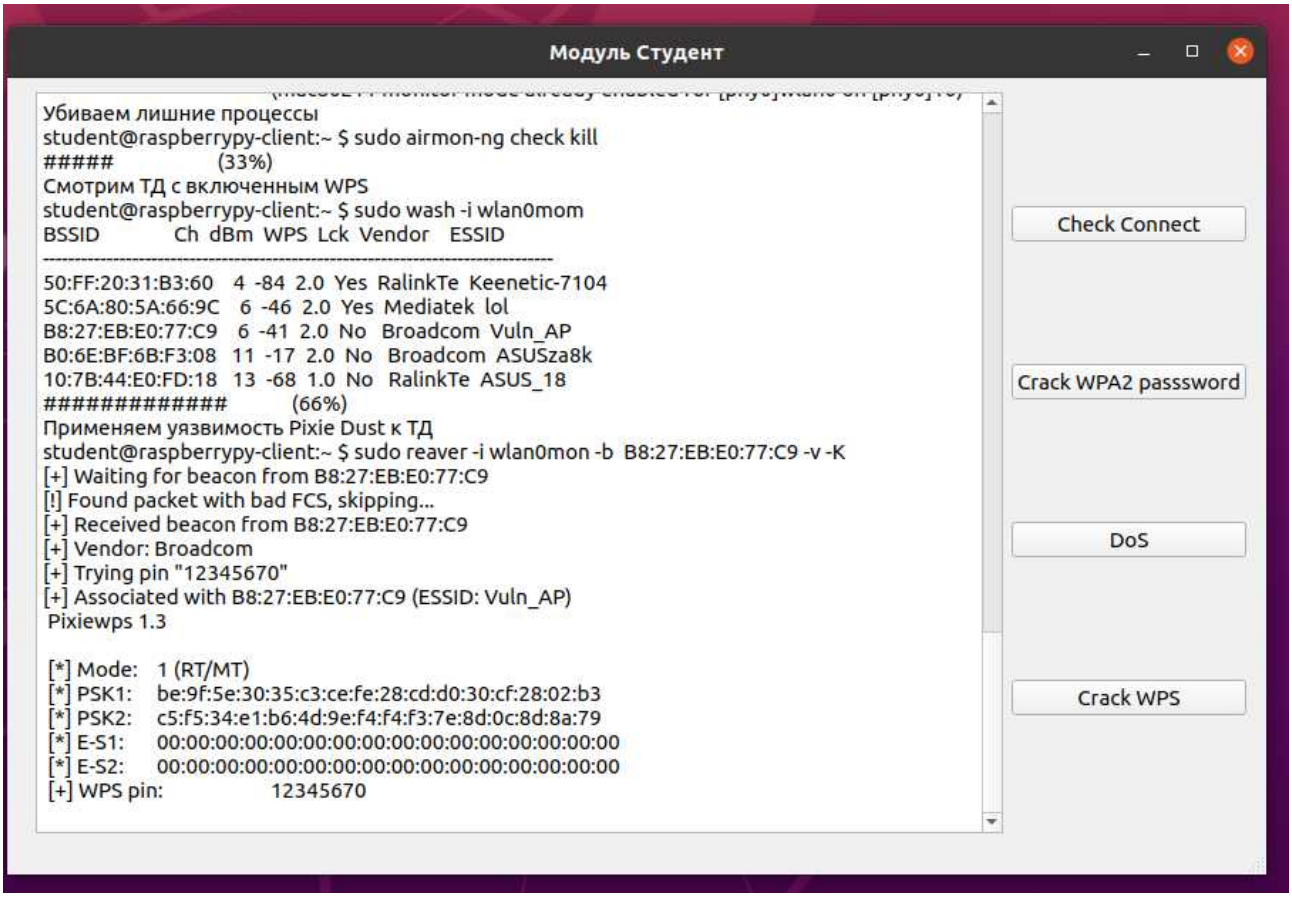

Рисунок 13 - Результат выполнения функции эксплуатации уязвимости стандарта WPS к подбору PIN-кода

Фактический результат, представленный на рисунке 13, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция эксплуатации уязвимости стандарта WPS к подбору PIN-кода реализована корректно.

Было проведено тестирование программного обеспечения «Администратор». Тестирование проводилось в рабочей среде на виртуальной машине с установленной операционной системой Ubuntu 20.04 Desktop.

Было проведено тестирование формы авторизации для подключения к модулю «Нарушитель» и модулю «ТД». Для прохождения авторизации необходимо было ввести логин и пароль от аппаратных модулей.

Ожидаемый результат: сообщения об успешных авторизациях и открытие основного окна программы.

Результат приведен на рисунке с 13 по 15.

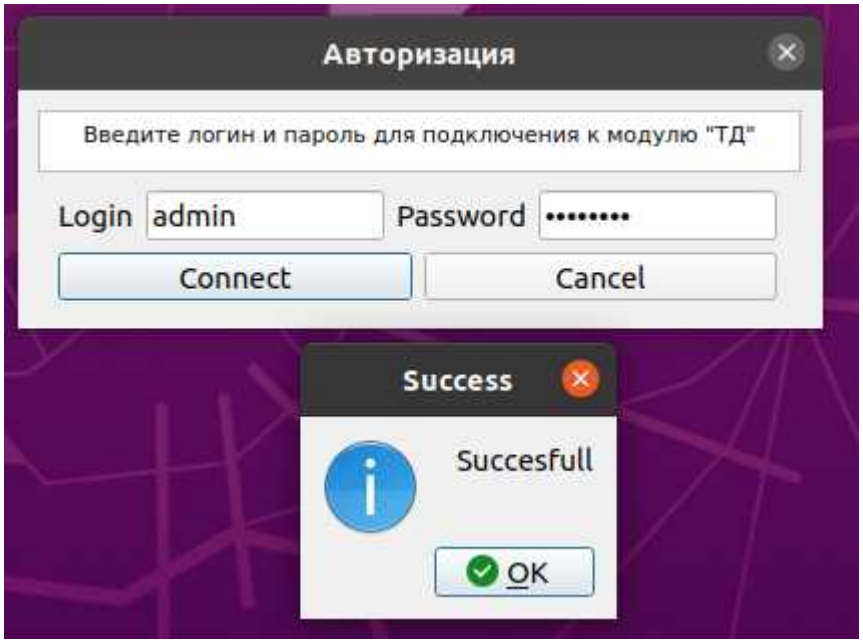

Рисунок 13 — Вывод сообщения об успешной авторизации при подключении к модулю «ТД»

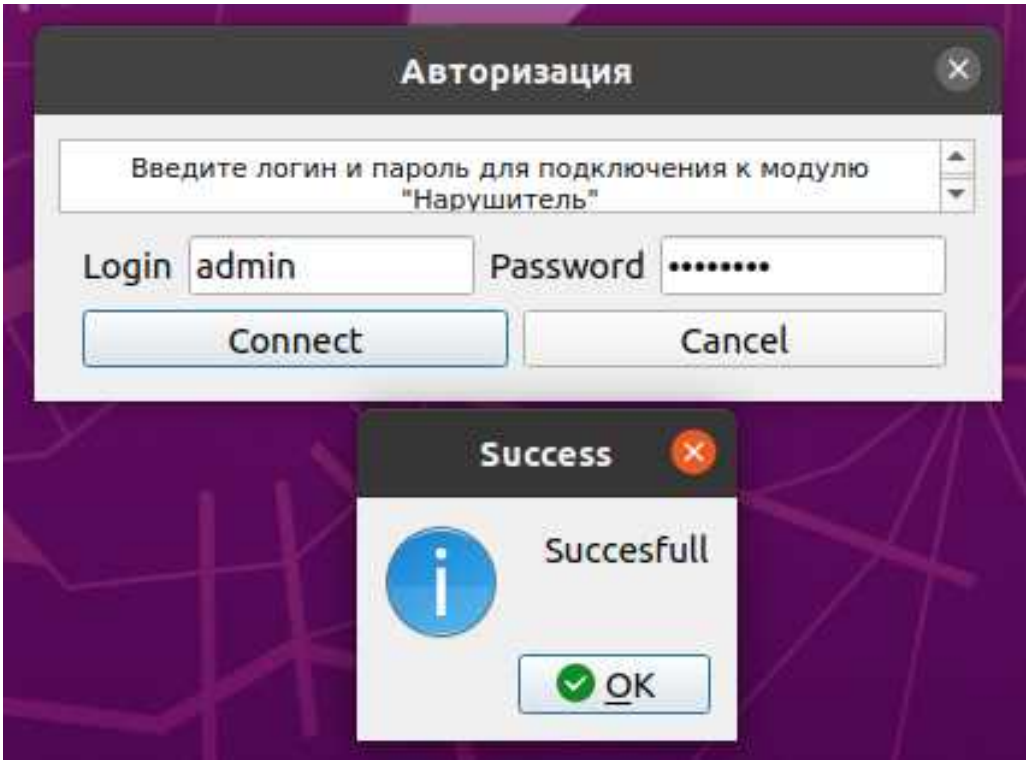

Рисунок 14 — Вывод сообщения об успешной авторизации при подключении к модулю «Нарушитель»

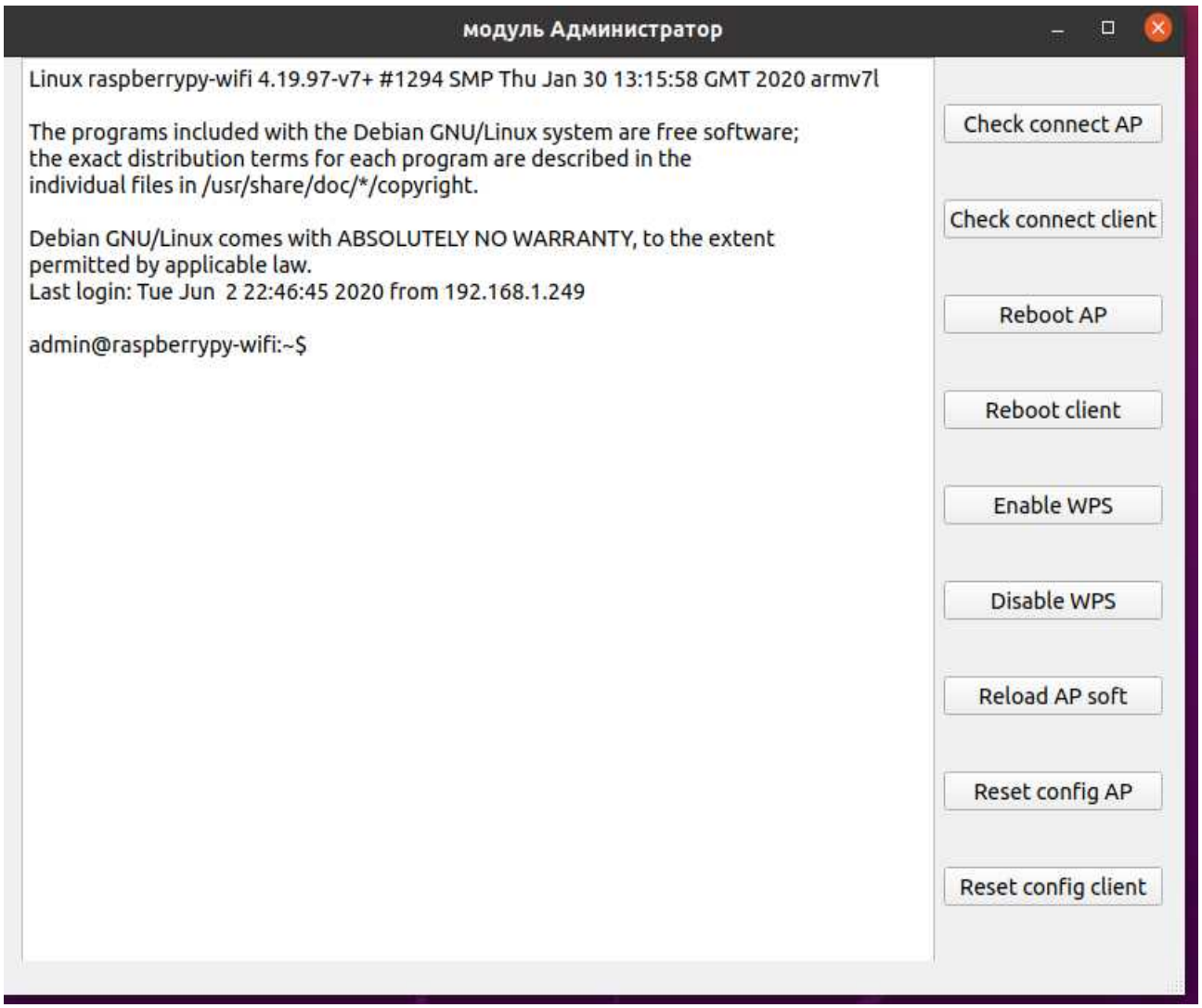

Рисунок 15 — Вывод основного окна программы

Фактический результат, представленный на рисунках 13,14,15, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что форма авторизации реализована корректно.

Было проведено тестирование функции проверки соединения модуля «ТД». Для проверки доступности модуля необходимо нажать кнопку «Check connect AP».

Ожидаемые результат: вывод сообщения о доступности модуля «ТД».

Результат приведён на рисунке 16

Рисунок 16 — Вывод сообщения о доступности модуля «ТД»

Фактический результат, представленный на рисунке 16, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция проверки соединения с модулем «ТД» реализована корректно.

Было проведено тестирование функции проверки соединения модуля «Нарушитель». Для проверки доступности модуля необходимо нажать кнопку «Check connect client».

Ожидаемые результат: вывод сообщения о доступности модуля «Нарушитель».

Результат приведён на рисунке 17

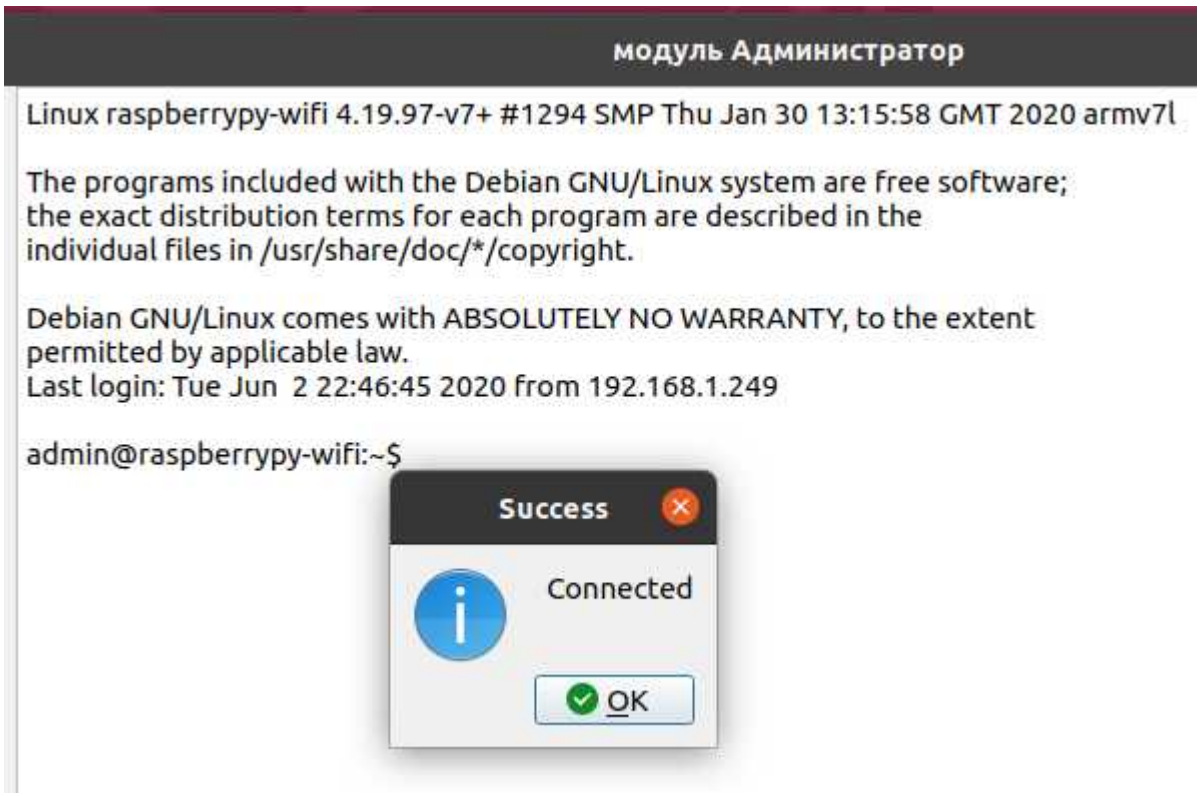

Рисунок 17 — Вывод сообщения о доступности модуля «ТД»

Фактический результат, представленный на рисунке 17, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция проверки соединения с модулем «Нарушитель» реализована корректно.

Было проведено тестирование функции перезагрузки модуля «ТД». Для проверки доступности модуля необходимо нажать кнопку «Reboot AP».

Ожидаемый результат: вывод сообщения о перезагрузке модуля «ТД», перезапуск программы.

Результат представлен на рисунке 18.

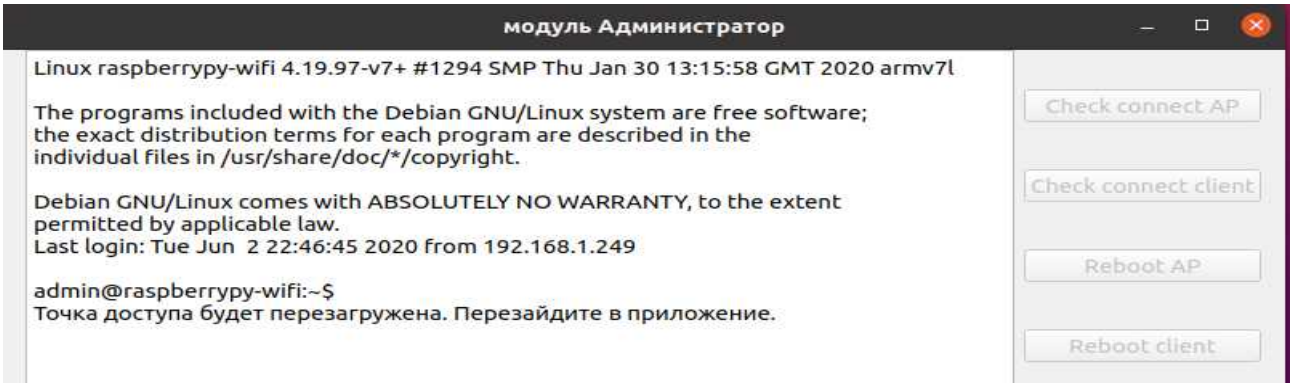

Рисунок 18 — Результат работы функции перезагрузки модуля «ТД»

Фактический результат, представленный на рисунке 18, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция перезагрузки модуля«ТД» реализована корректно.

Было проведено тестирование функции перезагрузки модуля «Нарушитель». Для проверки доступности модуля необходимо нажать кнопку «Reboot client».

Ожидаемый результат: вывод сообщения о перезагрузке модуля «Нарушитель», перезапуск программы.

Результат представлен на рисунке 19.

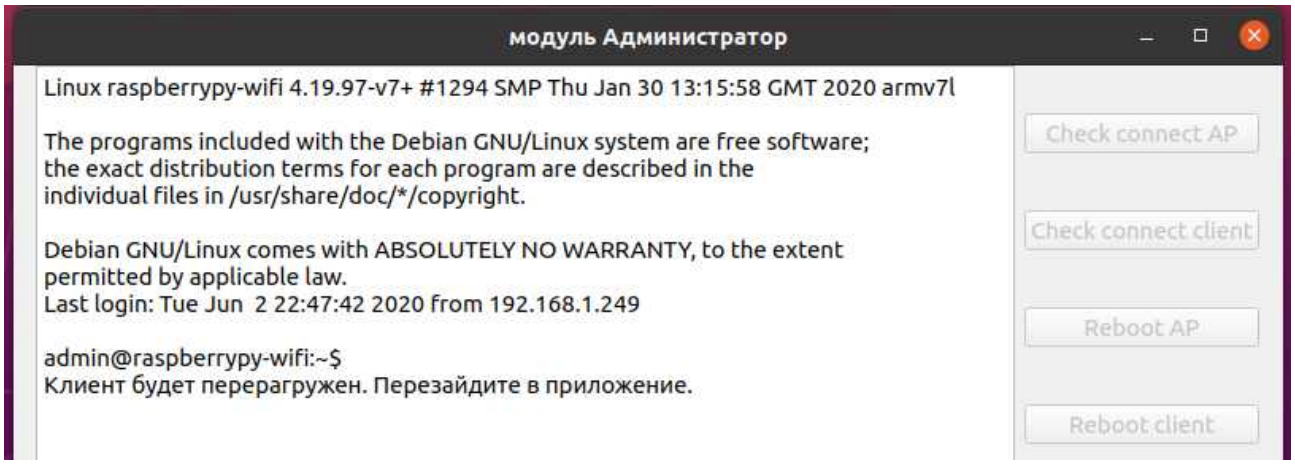

Рисунок 19 — Результат работы функции перезагрузки модуля «Нарушитель»

Фактический результат, представленный на рисунке 19, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция перезагрузки модуля«Нарушитель» реализована корректно.

Было проведено тестирование функции включения стандарта WPS на модуле «ТД». Для того чтобы включить поддержку WPS стандарта на точке доступа модуля «ТД» необходимо нажать кнопку «Enable WPS».

Ожидаемый результат: вывод сообщения об успешной выполнения функции и включения стандарта WPS на точке доступа модуля «ТД».

Результат представлен на рисунке 20.

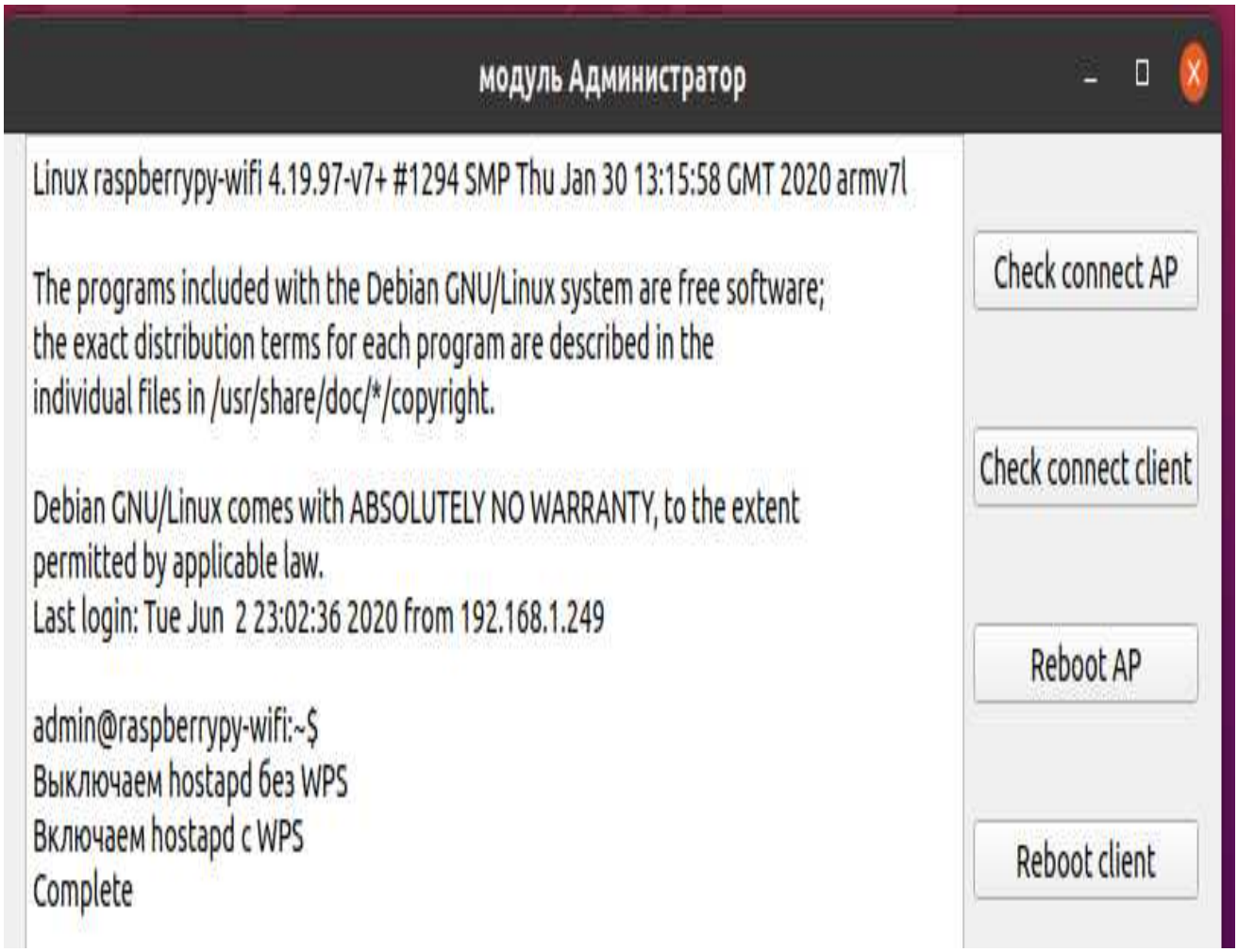

Рисунок 20 — Результат работы функции включения стандарта WPS на модуле «ТД»

Фактический результат, представленный на рисунке 20, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция включения стандарта WPS на модуле «ТД» реализована корректно.

Было проведено тестирование функции выключения стандарта WPS на модуле «ТД». Для того чтобы выключить поддержку WPS стандарта на точке доступа модуля «ТД» необходимо нажать кнопку «Disable WPS».

Ожидаемый результат: вывод сообщения об успешной выполнения функции и выключения стандарта WPS на точке доступа модуля «ТД».

Результат представлен на рисунке 21.

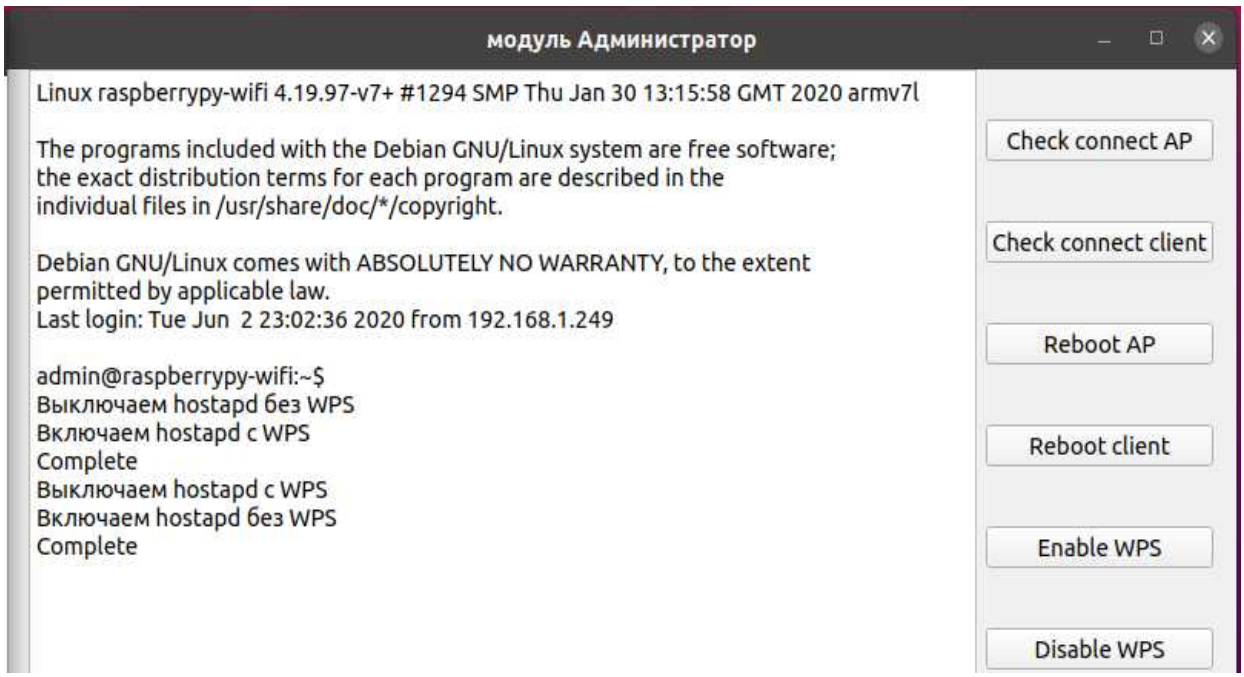

Рисунок 21 — Результат работы функции выключения стандарта WPS на модуле «ТД»

Фактический результат, представленный на рисунке 21, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция выключения стандарта WPS на модуле «ТД» реализована корректно.

Было проведено тестирование функции возвращения модуля «ТД» к исходным настройкам. Для того чтобы вернуть модуль «ТД» к исходным настройкам необходимо нажать кнопку «Reset config AP».

Ожидаемый результат: вывод сообщения об успешной выполнения функции и восстановление исходных настроек модуля «ТД».

Результат представлен на рисунке 22.

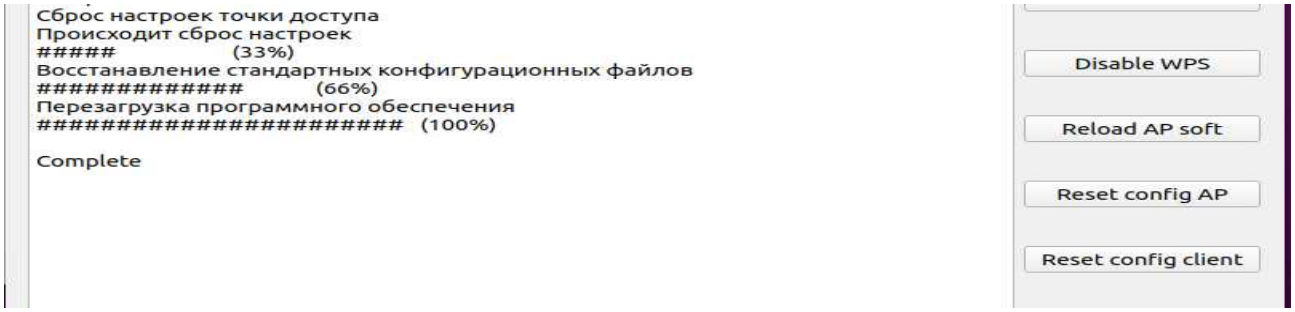

Рисунок 22 — Результат работы функции возвращения к исходным настройкам модуля «ТД»

Фактический результат, представленный на рисунке 22, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция возвращения к исходным настройкам модуля «ТД» реализована корректно.

Было проведено тестирование функции возвращения модуля «Нарушитель» к исходным настройкам. Для того чтобы вернуть модуль «Нарушитель» к исходным настройкам необходимо нажать кнопку «Reset config client».

Ожидаемый результат: вывод сообщения об успешной выполнения функции и восстановление исходных настроек модуля «Нарушитель».

Результат представлен на рисунке 23.

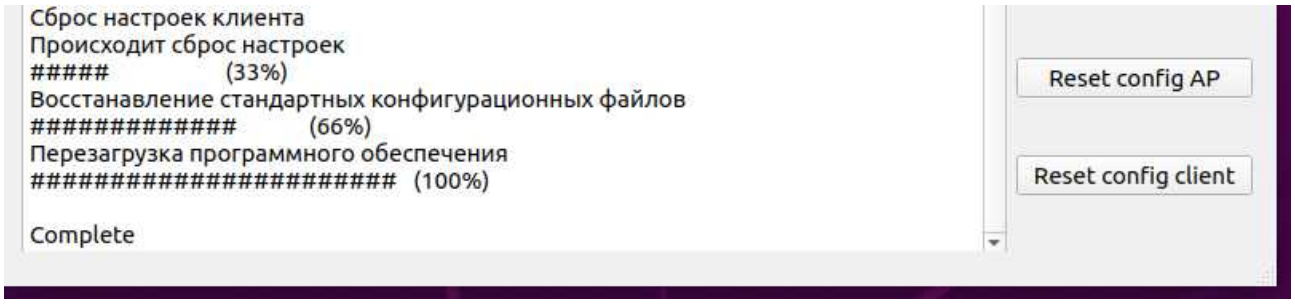

Рисунок 23 — Результат работы функции возвращения к исходным настройкам модуля «Нарушитель»

Фактический результат, представленный на рисунке 23, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция возвращения к исходным настройкам модуля «Нарушитель» реализована корректно.

Было проведено тестирование функции перезагрузки программного обеспечения модуля «ТД» Для того чтобы перезагрузить программное обеспечение модуля «ТД» необходимо нажать кнопку «Reload AP soft».

Ожидаемый результат: вывод сообщения об успешной выполнения функции и перезагрузка программного обеспечения модуля «ТД».

Результат представлен на рисунке 24.

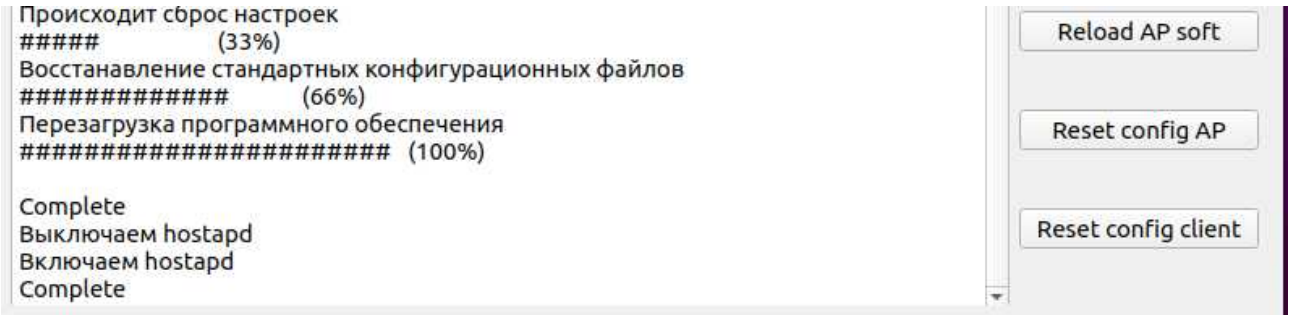

Рисунок 24 — Результат работы функции перезагрузки программного обеспечения модуля«ТД»

Фактический результат, представленный на рисунке 24, совпадает с ожидаемым результатом. Таким образом можно сделать вывод, что функция перезагрузки программного обеспечения модуля«ТД» реализована корректно.

6 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

6.1 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации

Согласно ГОСТ 12.0.003-2015 среди производственных факторов, оказывающих отрицательное психофизиологическое воздействие на организм человека, выделяют: [10]

— физические перегрузки, определяемые тяжестью трудового процесса;

— нервно-психические перегрузки, определяемые напряженностью трудового процесса. [4]

К физическим перегрузкам относятся:

— статические нагрузки, определяемые рабочей позой;

— динамические нагрузки, определяемые массой поднимаемого и перемещаемого вручную груза;

— – динамические нагрузки, определяемые повторением монотонных рабочих движений [5].

В своей работе специалист по защите информации связан только с одним видом вредных психофизиологических факторов - статическими перегрузками, определяемые рабочей позой, так как проводит большую часть рабочего времени за компьютером в одном положении.

Среди нервно-психических перегрузок выделяют:

— умственное перенапряжение, в том числе связанное с информационной нагрузкой;

— чрезмерное напряжение анализаторов, также вызванное информационной перегрузкой;

— монотонность труда, способствующая формированию монотонии, то есть снижению работоспособности в результате выполнения однообразной работы, связанной с частым повторением стереотипных действий;

— эмоциональные перегрузки [5].

Специалист по защите информации подвержен всем видам нервно-психических перегрузок, так как его деятельность включает работу с большим количеством информации и высокую степень ответственности. Ухудшению психоэмоционального состояния в этом случае способствует повышенное зрительное перенапряжение, умственные и нервно-эмоциональные нагрузки, продолжительная концентрация внимания, монотонность трудового процесса, большой объем информации, нерациональная организация рабочего места.

Таким образом, нервно-психические перегрузки в работе специалиста по защите информации - это следствие информационного взаимодействия в системе «пользователь – ПЭВМ». Они определяются некачественными условиями передачи изображения, несоответствием некоторых параметров информационных систем психофизиологическими особенностями человека, необходимостью внимательного наблюдения за информационными знаками, быстрого анализа постоянно изменяющейся информации, принятия продуктивных решений и выполнения определённых корректирующих действий.

Как правило, операторы ПЭВМ жалуются к концу рабочего дня на зрительное переутомление, головную боль, ноющие боли в мышцах шеи, плечевого пояса и спины, снижение способности к концентрации внимания.

Ухудшение самочувствия пользователя ПЭВМ при длительной работе связано также повышением пульсации потоков света от источников искусственного освещения и монитора. Пульсации потоков света способствуют повышению общего и зрительного переутомления, ухудшению зрительной и общей работоспособности. При уменьшении частоты световых пульсаций вредное воздействие усиливается. К нервно-психическим факторам относится

также низкочастотное мерцание экрана (до 200 Гц), то есть кажущееся изменение яркости свечения [6]. Оно возникает из-за взаимодействия пульсирующих потоков света экрана и источников света. Даже на начальном этапе компьютеризации общества было обнаружено специфическое зрительное утомление при работе за дисплеем, которое было названо «компьютерный зрительный синдром. Это объясняется тем, что у человека зрительная система по ходу эволюции приспособилась для восприятия изображения в отраженном свете (напечатанная информация, тексты, рисунки), а не для работы за монитором компьютера. Изображение на дисплее сильно отличается от привычного глазу объектов внешнего мира — оно светится, мерцает, состоит из дискретных точек, кроме этого цвета компьютерного изображения отличаются от естественных цветов. Однако зрительное переутомление вызывают не только особенности изображения на экране. Сильная нагрузка на глаза оказывается при введении информации, так как пользователю необходимо переводить часто взгляд с экрана на набираемый текст и клавиатуру, которые находятся на разном расстоянии от монитора и по-разному освещены. Зрительное переутомление обычно проявляется при затуманивании зрения, сопровождается трудностями при переносе взгляда с близких предметов на дальние и с дальних на близкие, кажущееся изменение цвета предметов, двоение предметов, чувством жжения, «песка» в глазах, болезненными ощущениями при движении глаз, покраснением век.

Специфическое воздействие электромагнитного поля выражается в биохимических изменениях, происходящие в клетках и тканях. Наиболее подвержены данным воздействиям центральная и сердечно-сосудистая системы. Могут наблюдаться отклонения в работе эндокринной системы. В начале воздействия отмечается повышение возбудимости нервной системы, проявляющееся повышением раздражительности, нарушением сна, эмоциональной неуравновешенностью. Затем развиваются астенические проявления, то есть физическая и нервнопсихическая слабость. В целом

хроническое воздействие ЭМП проявляется следующими симптомами: головная боль, повышенная утомляемость, ухудшение самочувствия, брадикардия (урежение пульса), а также гипотония (понижение артериального давления), боли в сердце. Данные негативные симптомы могут выражаться в разной степени [7].

6.2 Методы и средства физической культуры, снижающие воздействие вредных психофизиологических факторов

Для нормального функционирования мозга необходимо, чтобы к нему поступали нервные импульсы от всех систем организма, в том числе и от мышц, которые составляют половину от массы всего организма. Движения мышц создают огромный поток нервных импульсов, способствующих обогащению мозга разнообразными ощущениями, поддерживающих его в хорошем рабочем состоянии. Именно поэтому умственная работоспособность непосредственно связана с общим состоянием здоровья, в укреплении которого важная роль отводится физической культуре.

При умственном труде, который лежит в основе деятельности специалиста по защите информации, в коре головного мозга формируются замкнутые циклы возбуждения, которые характеризуются большой устойчивостью и инертностью. Поэтому, если после прекращения физической нагрузки человек практически сразу может переключиться на новый вид деятельности, то при умственном труде интенсивная работа мозга не заканчивается продолжительное время и после завершения интеллектуальной нагрузки [8].

В основе организации отдыха при умственных перегрузках лежит принцип активного отдыха, который предусматривает систему организованных движений до, в процессе и по окончании умственного деятельности, что оказывает благоприятный эффект не только в сохранении, но и повышении

умственной работоспособности. Ежедневные и регулярные физические упражнения способствуют формированию в коре больших полушарий «доминанты движения», которая оказывает положительное воздействие на состояние дыхательной, мышечной, сердечно-сосудистой систем организма, активизирует зону сенсомоторной активности в коре головного мозга, способствует повышению тонуса всего организма.

Перед началом интенсивной умственной работы, для сокращения периода вырабатывания, рекомендуются в течении 5 минут произвольные движения умеренной или средней интенсивности, направленные на дополнительное напряжение мышц конечностей. При этом чем ниже исходное нервно-психическое и мышечное напряжение человека, и чем быстрее необходимо включиться в работу, тем интенсивнее должно быть упражнения на напряжение скелетных мышц. В процессе напряженной продолжительной умственной деятельности, сопровождаемой к тому же эмоциональным стрессом, рекомендуется проделать произвольные упражнения на общее расслабление и снижение тонуса скелетных мышц, желательно в сочетании с ритмичным сокращением небольших мышечных групп (разгибателей и сгибателей пальцев кисти, мимических мышц лица и т. п.).

Для повышения работоспособности мозга на фоне мышечного расслабления, улучшения функций дыхания и кровообращения, необходимо регулярно выполнять кратковременные физические упражнения. Например, раз в час при работе за компьютером проделать познотонические упражнения в течении нескольких минут. а через каждые 2 часа желательно проводить минутные динамические упражнения, такие как бег на месте, с ритмичным, глубоким дыханием [9].

Для снижения напряжения глаз каждые 2 часа рекомендуется выполнять упражнения для глаз. Гимнастику для глаз за компьютером лучше выполнять в расслабленном состоянии. При наличии боли в области шеи или спины, необходимо сначала выполнить упражнения для этих зон. При выполнении

упражнений для глаз надо глубоко дышать, так как при работе глазные мышцы должны активно насыщаться кислородом.

Правильно подобранный комплекс упражнений повышает общий эмоциональный тонус работника, создавая устойчивое настроение, которое станет благоприятным фоном для активной умственной деятельности и необходимо для предупреждения раннего развития утомления. Физическая нагрузка выстроенная на основе принципа активного отдыха, оказывает благоприятное воздействие на повышение умственной работоспособности специалиста по защите информации.

Заключение

В ходе работы была создана архитектура учебного стенда по изучению механизмов безопасности беспроводных сетей на основе одноплатных компьютеров. Было произведено проектирование аппаратной и программной части учебного комплекса.

Была разработана диаграмма вариантов использования и алгоритмы работы программной части учебного комплекса.

Была выполнена программная реализация учебного комплекса.

Были изучены вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации, а так же методы и средств физической культуры, снижающие их воздействие.

Разработанный учебный стенд обладает следующими преимуществами: изолированная сетевая среда с контролируемым доступом виртуальных рабочих мест, минимальное количество беспроводного оборудования, возможность реализации любых сценариев взаимодействия между нарушителем и жертвой, возможность исследования эффективности мер по обеспечению безопасности беспроводных сетей. Использование учебного стенда позволит обучающимся получить достаточный уровень практических навыков по обеспечению информационной безопасности беспроводных сетей.

Таким образом, все пункты технического задания были выполнены в полном объёме.

Список используемых источников

1 Wi-Fi [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Wi-Fi

2 Атаки на корпоративный Wi-Fi. [Электронный ресурс]. Режим доступа: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/Wi-Fi-Corporate-Attacks-rus.pdf

3 Атака на Wi-Fi с WPS, используя Reaver [Электронный ресурс]. Режим доступа: https://hackware.ru/?p=4213

4 Федеральный закон «Об информации, информационных технологиях и о защите информации» от 27.07.2006 № 149-ФЗ // «Консультант Плюс» - законодательство РФ. с изм. и допол. в ред. от 1.01.2018

5 ГОСТ 12.0.003-2015. Система стандартов по безопасности труда. Опасные и вредные производственные факторы. Классификация. Введ. 01.03.2013 // Сайт юридической фирмы «Интернет и Право». 1998-2017. URL: http://www.internet-law.ru/gosts/gost/62075/ (дата обращения: 14.04.2020).

6 Опасные и вредные факторы при работе с ПЭВМ // РГРТУ Рязанский Государственный Радиотехнический Университет Группа 640 URL: http://rgrtu640.narod.ru/bezopasnost-zhiznedeyatelnosti/45.html (дата обращения: 14.04.2020).

7 Вредные и опасные факторы при работе с компьютером // Энциклопедия Экономиста URL: http://www.grandars.ru/shkola/bezopasnost-zhiznedeyatelnosti/vrednye-faktory-pri-rabote-na-pk.html (дата обращения: 14.04.2020)

8 Экзамен по физической культуре в вузе / Коваль Л.Н., Богданов О.Г., Ярошенко Е.В., Алексеева Е.Н., М.-Берлин: Директ-Медиа, 2015.

9 Физическая культура студента / Под ред. Ильинича В.И. М.: Гардарики, 2000.

10 ГОСТ 12.0.003-2015 Система стандартов безопасности труда

(ССБТ). Опасные и вредные производственные факторы. Классификация

Приложение А

(обязательное)

Исходный код программы «Студент»

Исходный код файла «student.py»:

```
#!/usr/bin/env python3
import PyQt5
from auth_student import Ui_Dialog
import sys
from main_client import Ui_MainWindow
import time
import paramiko
import os

client=paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy
())
class Main(PyQt5.QtWidgets.QMainWindow):
    def __init__(self,login,password,parent=None,):
        self.login=login
        self.password=password
        client.connect(hostname="192.168.1.78", username=
self.login, password=self.password, port="22")
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.changeButtonState(False)
        self.ui.textEdit.setReadOnly(True)
        self.ui.checkButton.clicked.connect(self.checkCon
nect)
        self.ui.crackWPAButton.clicked.connect(self.crack
WPA)
        self.ui.dosButton.clicked.connect(self.dos)
        self.ui.crackWPSButton.clicked.connect(self.crack
WPS)
        self.timer = PyQt5.QtCore.QTimer()
        self.timer.setInterval(1000)
        self.timer.timeout.connect(self.test_update)
        self.timer.start()
    def test_update(self):
        with client.invoke_shell() as ssh:
            time.sleep(2)
```

```
                result=ssh.recv(50000).decode("ascii")
                self.refresh_text(result)
            self.timer.stop()
            self.changeButtonState(True)
        def refresh_text(self,mystring):
            self.ui.textEdit.append(mystring)
            PyQt5.QtGui.QGuiApplication.processEvents()
        def checkConnect(self):
            self.changeButtonState(False)
            response = os.system("ping -c 1 192.168.1.78")
            if response == 0:
                self.mbox("Connected","Success")
            else:
                self.mbox("Failed to connect","Fail")
            self.changeButtonState(True)
        def crackWPA(self):
            self.changeButtonState(False)
            sin,sout,serr = client.exec_command('./
crack_wpa2.sh')
            while True:
                line = sout.readline().strip('\n')
                if not line:
                    continue
                if "(100%)" in  line:
                    break
                self.refresh_text(line)
            self.changeButtonState(True)
        def crackWPS(self):
            print("WPS")
            self.changeButtonState(False)
            sin,sout,serr = client.exec_command('./
crack_wps.sh')
            while True:
                line = sout.readline().strip('\n')
                if not line:
                    continue
                if "Time taken" in  line:
                    break
                self.refresh_text(line)
            self.changeButtonState(True)
        def dos(self):
            self.changeButtonState(False)
            sin,sout,serr = client.exec_command('./dos.sh')
            timeout = time.time() + 15
```

```
        while(True):
            line = sout.readline().strip('\n')
            if not line:
                continue
            if time.time()>timeout:
                break
            self.refresh_text(line)
        self.changeButtonState(True)

    def changeButtonState(self,state):
        self.ui.checkButton.setEnabled(state)
        self.ui.crackWPAButton.setEnabled(state)
        self.ui.crackWPSButton.setEnabled(state)
        self.ui.dosButton.setEnabled(state)
    def mbox(self,body, title):
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()
class AuthWin(PyQt5.QtWidgets.QDialog):
    def __init__(self,parent=None):
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.password_input.setEchoMode(PyQt5.QtWidget
s.QLineEdit.Password)
        self.ui.cancel_Button.clicked.connect(self.cancel
Clicked)
        self.ui.connect_Button.clicked.connect(self.conne
ctClicked)
    def connectClicked(self):
        loginText=self.ui.login_input.text()
        passwordText=self.ui.password_input.text()
        try:
            client.connect(hostname="192.168.1.78", usern
ame=loginText, password=passwordText, port="22")
            self.mbox("Succesfull","Success")
            client.close()
            self.accept()
        except paramiko.ssh_exception.AuthenticationExcep
tion:
            self.mbox("Fail","Invalid data")
    def cancelClicked(self):
        sys.exit(app.exec_())
    def mbox(self,body, title):
```

```
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()

if __name__ == "__main__":
    app=PyQt5.QtWidgets.QApplication(sys.argv)
    authAppExec= AuthWin()
    if authAppExec.exec_() == PyQt5.QtWidgets.QDialog.Acc
epted:
        login=authAppExec.ui.login_input.text()
        password=authAppExec.ui.password_input.text()
        mainAppExec= Main(login,password)
        mainAppExec.show()
    sys.exit(app.exec_())
```

Исходный код файла «auth_student.py»:

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'aut
h_student.ui'
#
# Created by: PyQt5 UI code generator 5.14.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(402, 119)
        self.textBrowser = QtWidgets.QTextBrowser(Dialog)
        self.textBrowser.setGeometry(QtCore.QRect(10, 10,
381, 31))
        self.textBrowser.setObjectName("textBrowser")
        self.widget = QtWidgets.QWidget(Dialog)
        self.widget.setGeometry(QtCore.QRect(20, 50, 359,
55))
        self.widget.setObjectName("widget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.
widget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0
)
        self.verticalLayout.setObjectName("verticalLayout
```

```
")
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLa
yout")
        self.label = QtWidgets.QLabel(self.widget)
        self.label.setObjectName("label")
        self.horizontalLayout.addWidget(self.label)
        self.login_input = QtWidgets.QLineEdit(self.widge
t)
        self.login_input.setObjectName("login_input")
        self.horizontalLayout.addWidget(self.login_input)
        self.label_2 = QtWidgets.QLabel(self.widget)
        self.label_2.setObjectName("label_2")
        self.horizontalLayout.addWidget(self.label_2)
        self.password_input = QtWidgets.QLineEdit(self.wi
dget)
        self.password_input.setObjectName("password_input
")
        self.horizontalLayout.addWidget(self.password_inp
ut)
        self.verticalLayout.addLayout(self.horizontalLayo
ut)
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontal
Layout_2")
        self.connect_Button = QtWidgets.QPushButton(self.
widget)
        self.connect_Button.setObjectName("connect_Button
")
        self.horizontalLayout_2.addWidget(self.connect_Bu
tton)
        self.cancel_Button = QtWidgets.QPushButton(self.w
idget)
        self.cancel_Button.setObjectName("cancel_Button")
        self.horizontalLayout_2.addWidget(self.cancel_But
ton)
        self.verticalLayout.addLayout(self.horizontalLayo
ut_2)

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
```

```
        Dialog.setWindowTitle(_translate("Dialog", "Автор
изация"))
        self.textBrowser.setHtml(_translate("Dialog", "<!
DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http:/
/www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /
><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg
2\'; font-size:8.25pt; font-weight:400; font-
style:normal;\">\n"
"<p align=\"center\" style=\" margin-top:0px; margin-
bottom:0px; margin-left:0px; margin-right:0px; -qt-block-
indent:0; text-indent:0px;\"><span style=\" font-
family:\'MS Shell Dlg 2\';\">Введите логин и пароль для п
одключения к модулю &amp;quot;Нарушитель&amp;quot;</span></p></
body></html>"))
        self.label.setText(_translate("Dialog", "Login"))
        self.label_2.setText(_translate("Dialog", "Passwo
rd"))
        self.connect_Button.setText(_translate("Dialog",
"Connect"))
        self.cancel_Button.setText(_translate("Dialog", "
Cancel"))
```

Исходный код файла «main_student.py»:

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'mai
n_student.ui'
#
# Created by: PyQt5 UI code generator 5.14.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(885, 560)
        self.centralwidget = QtWidgets.QWidget(MainWindow
)
```

```
        self.centralwidget.setObjectName("centralwidget")
        self.layoutWidget = QtWidgets.QWidget(self.centra
lwidget)
        self.layoutWidget.setGeometry(QtCore.QRect(20, 10
, 851, 521))
        self.layoutWidget.setObjectName("layoutWidget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(sel
f.layoutWidget)
        self.horizontalLayout.setContentsMargins(0, 0, 0,
0)
        self.horizontalLayout.setObjectName("horizontalLa
yout")
        self.textEdit = QtWidgets.QTextEdit(self.layoutWi
dget)
        self.textEdit.setObjectName("textEdit")
        self.horizontalLayout.addWidget(self.textEdit)
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout
")
        self.checkButton = QtWidgets.QPushButton(self.lay
outWidget)
        self.checkButton.setObjectName("checkButton")
        self.verticalLayout.addWidget(self.checkButton)
        self.crackWPAButton = QtWidgets.QPushButton(self.
layoutWidget)
        self.crackWPAButton.setObjectName("crackWPAButton
")
        self.verticalLayout.addWidget(self.crackWPAButton
)
        self.dosButton = QtWidgets.QPushButton(self.layou
tWidget)
        self.dosButton.setObjectName("dosButton")
        self.verticalLayout.addWidget(self.dosButton)
        self.crackWPSButton = QtWidgets.QPushButton(self.
layoutWidget)
        self.crackWPSButton.setObjectName("crackWPSButton
")
        self.verticalLayout.addWidget(self.crackWPSButton
)
        self.horizontalLayout.addLayout(self.verticalLayo
ut)
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
```

```
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow"
, "Модуль Студент"))
        self.checkButton.setText(_translate("MainWindow",
"Check Connect"))
        self.crackWPAButton.setText(_translate("MainWindo
w", "Crack WPA2 passsword"))
        self.dosButton.setText(_translate("MainWindow", "
DoS"))
        self.crackWPSButton.setText(_translate("MainWindo
w", "Crack WPS"))
```

Приложение Б

(обязательное)

Исходный код программы «Администратор»

Исходный код файла «admin.py»:

```
#!/usr/bin/env python3
import PyQt5
from auth_admin import Ui_Dialog_AP
from auth_student import Ui_Dialog
import sys
from main_admin import Ui_MainWindow
import time
import paramiko
import os

client=paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy
())
class Main(PyQt5.QtWidgets.QMainWindow):
    def __init__(self,loginAP,passwordAP,loginClient,pass
wordClient,parent=None,):
        self.loginAP=loginAP
        self.passwordAP=passwordAP
        self.loginClient=loginClient
        self.passwordClient=passwordClient
        client.connect(hostname="192.168.1.78", username=
self.loginClient, password=self.passwordClient, port="22"
)
        self.AP=paramiko.SSHClient()
        self.AP.set_missing_host_key_policy(paramiko.Auto
AddPolicy())
        self.AP.connect(hostname="192.168.1.161", usernam
e=self.loginAP, password=self.passwordAP, port="22")
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.changeButtonState(False)
        self.ui.textEdit.setReadOnly(True)
        self.timer = PyQt5.QtCore.QTimer()
        self.timer.setInterval(1000)
        self.timer.timeout.connect(self.test_update)
        self.timer.start()
```

```
        self.ui.checkConnecClientButton.clicked.connect(s
elf.checkConnecClient)
        self.ui.checkConnectAPButton.clicked.connect(self
.checkConnectAP)
        self.ui.rebootAPButton.clicked.connect(self.reboo
tAP)
        self.ui.rebootClientButton.clicked.connect(self.r
ebootClient)
        self.ui.resetConfigClientButton.clicked.connect(s
elf.resetConfigClient)
        self.ui.resetConfigAPButton.clicked.connect(self.
resetConfigAP)
        self.ui.disableWPSButton.clicked.connect(self.dis
ableWPS)
        self.ui.enableWPSButton.clicked.connect(self.enab
leWPS)
        self.ui.reloadAPSoftButton.clicked.connect(self.r
eloadAPSoft)
    def checkConnecClient(self):
        self.changeButtonState(False)
        response = os.system("ping -c 1 192.168.1.78")
        if response == 0:
            self.mbox("Connected","Success")
        else:
            self.mbox("Failed to connect","Fail")
        self.changeButtonState(True)
    def checkConnectAP(self):
        self.changeButtonState(False)
        response = os.system("ping -c 1 192.168.1.161")
        if response == 0:
            self.mbox("Connected","Success")
        else:
            self.mbox("Failed to connect","Fail")
        self.changeButtonState(True)
    def rebootAP(self):
        self.changeButtonState(False)
        self.refresh_text("Точка доступа будет перезагруж
ена. Перезайдите в приложение.")
        time.sleep(5)
        sin,sout,serr = self.AP.exec_command('sudo reboot
now')
        sys.exit(app.exec_())
    def rebootClient(self):
        self.changeButtonState(False)
```

```
        self.refresh_text("Клиент будет перерагружен. Пер
езайдите в приложение.")
        time.sleep(5)
        sin,sout,serr = client.exec_command('sudo reboot
now')
        sys.exit(app.exec_())
    def resetConfigClient(self):
        self.changeButtonState(False)
        sin,sout,serr = client.exec_command('bash resetCl
ient.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def resetConfigAP(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('bash resetC
onfigAP.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def disableWPS(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('./
disableWPS.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def enableWPS(self):
        self.changeButtonState(False)
```

```
        sin,sout,serr = self.AP.exec_command('./
enableWPS.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def reloadAPSoft(self):
        self.changeButtonState(False)
        sin,sout,serr = self.AP.exec_command('bash reload
AP.sh')
        while True:
            line = sout.readline().strip('\n')
            if not line:
                continue
            self.refresh_text(line)
            if "Complete" in  line:
                break
        self.changeButtonState(True)
    def test_update(self):
        with self.AP.invoke_shell() as ssh:
            time.sleep(2)
            result=ssh.recv(50000).decode("ascii")
            self.refresh_text(result)
        self.timer.stop()
        self.changeButtonState(True)
    def refresh_text(self,mystring):
        self.ui.textEdit.append(mystring)
        PyQt5.QtGui.QGuiApplication.processEvents()

    def changeButtonState(self,state):
        self.ui.checkConnecClientButton.setEnabled(state)
        self.ui.checkConnectAPButton.setEnabled(state)
        self.ui.rebootAPButton.setEnabled(state)
        self.ui.rebootClientButton.setEnabled(state)
        self.ui.resetConfigClientButton.setEnabled(state)
        self.ui.resetConfigAPButton.setEnabled(state)
        self.ui.disableWPSButton.setEnabled(state)
        self.ui.enableWPSButton.setEnabled(state)
        self.ui.reloadAPSoftButton.setEnabled(state)
    def mbox(self,body, title):
```

```
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()
class AuthWinAP(PyQt5.QtWidgets.QDialog):
    def __init__(self,parent=None):
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_Dialog_AP()
        self.ui.setupUi(self)
        self.ui.password_input.setEchoMode(PyQt5.QtWidget
s.QLineEdit.Password)
        self.ui.cancel_Button.clicked.connect(self.cancel
Clicked)
        self.ui.connect_Button.clicked.connect(self.conne
ctClicked)
    def connectClicked(self):
        loginText=self.ui.login_input.text()
        passwordText=self.ui.password_input.text()
        try:
            client.connect(hostname="192.168.1.161", user
name=loginText, password=passwordText, port="22")
            self.mbox("Succesfull","Success")
            client.close()
            self.accept()
        except paramiko.ssh_exception.AuthenticationExcep
tion:
            self.mbox("Fail","Invalid data")
    def cancelClicked(self):
        sys.exit(app.exec_())
    def mbox(self,body, title):
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()
class AuthWin(PyQt5.QtWidgets.QDialog):
    def __init__(self,parent=None):
        PyQt5.QtWidgets.QWidget.__init__(self,parent)
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.password_input.setEchoMode(PyQt5.QtWidget
s.QLineEdit.Password)
        self.ui.cancel_Button.clicked.connect(self.cancel
Clicked)
        self.ui.connect_Button.clicked.connect(self.conne
ctClicked)
    def connectClicked(self):
```

```
        loginText=self.ui.login_input.text()
        passwordText=self.ui.password_input.text()
        try:
            client.connect(hostname="192.168.1.78", usern
ame=loginText, password=passwordText, port="22")
            self.mbox("Succesfull","Success")
            client.close()
            self.accept()
        except paramiko.ssh_exception.AuthenticationExcep
tion:
            self.mbox("Fail","Invalid data")
    def cancelClicked(self):
        sys.exit(app.exec_())
    def mbox(self,body, title):
        dialog = PyQt5.QtWidgets.QMessageBox(PyQt5.QtWidg
ets.QMessageBox.Information,title,body)
        dialog.exec_()
if __name__ == "__main__":
    app=PyQt5.QtWidgets.QApplication(sys.argv)
    authAppExecAP=AuthWinAP()
    authAppExecClient=AuthWin()
    if authAppExecAP.exec_() == PyQt5.QtWidgets.QDialog.A
ccepted:
        loginAP=authAppExecAP.ui.login_input.text()
        passwordAP=authAppExecAP.ui.password_input.text()
        if authAppExecClient.exec_() == PyQt5.QtWidgets.Q
Dialog.Accepted:
            loginClient=authAppExecClient.ui.login_input.
text()
            passwordClient=authAppExecClient.ui.password_
input.text()
            mainAppExec= Main(loginAP,passwordAP,loginCli
ent,passwordClient)
            mainAppExec.show()
    sys.exit(app.exec_())
```

Исходный код файла «auth_admin.py»:

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'aut
h_student.ui'
#
# Created by: PyQt5 UI code generator 5.14.2
```

```
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Dialog_AP(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(402, 119)
        self.textBrowser = QtWidgets.QTextBrowser(Dialog)
        self.textBrowser.setGeometry(QtCore.QRect(10, 10,
381, 31))
        self.textBrowser.setObjectName("textBrowser")
        self.widget = QtWidgets.QWidget(Dialog)
        self.widget.setGeometry(QtCore.QRect(20, 50, 359,
55))
        self.widget.setObjectName("widget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.
widget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0
)
        self.verticalLayout.setObjectName("verticalLayout
")
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLa
yout")
        self.label = QtWidgets.QLabel(self.widget)
        self.label.setObjectName("label")
        self.horizontalLayout.addWidget(self.label)
        self.login_input = QtWidgets.QLineEdit(self.widge
t)
        self.login_input.setObjectName("login_input")
        self.horizontalLayout.addWidget(self.login_input)
        self.label_2 = QtWidgets.QLabel(self.widget)
        self.label_2.setObjectName("label_2")
        self.horizontalLayout.addWidget(self.label_2)
        self.password_input = QtWidgets.QLineEdit(self.wi
dget)
        self.password_input.setObjectName("password_input
")
        self.horizontalLayout.addWidget(self.password_inp
ut)
        self.verticalLayout.addLayout(self.horizontalLayo
ut)
```

```
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontal
Layout_2")
        self.connect_Button = QtWidgets.QPushButton(self.
widget)
        self.connect_Button.setObjectName("connect_Button
")
        self.horizontalLayout_2.addWidget(self.connect_Bu
tton)
        self.cancel_Button = QtWidgets.QPushButton(self.w
idget)
        self.cancel_Button.setObjectName("cancel_Button")
        self.horizontalLayout_2.addWidget(self.cancel_But
ton)
        self.verticalLayout.addLayout(self.horizontalLayo
ut_2)

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Автор
изация"))
        self.textBrowser.setHtml(_translate("Dialog", "<!
DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http:/
/www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" />
><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg
2\'; font-size:8.25pt; font-weight:400; font-
style:normal;\">\n"
"<p align=\"center\" style=\" margin-top:0px; margin-
bottom:0px; margin-left:0px; margin-right:0px; -qt-block-
indent:0; text-indent:0px;\"><span style=\" font-
family:\'MS Shell Dlg 2\';\">Введите логин и пароль для п
одключения к модулю &amp;quot;ТД&amp;quot;</span></p></body></
html>"))
        self.label.setText(_translate("Dialog", "Login"))
        self.label_2.setText(_translate("Dialog", "Passwo
rd"))
        self.connect_Button.setText(_translate("Dialog",
"Connect"))
```

```
        self.cancel_Button.setText(_translate("Dialog", "
Cancel"))
```

Исходный код файла «main_admin.py»:

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'mai
n_admin.ui'
#
# Created by: PyQt5 UI code generator 5.14.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(800, 600)
        self.centralwidget = QtWidgets.QWidget(MainWindow
)
        self.centralwidget.setObjectName("centralwidget")
        self.widget = QtWidgets.QWidget(self.centralwidge
t)
        self.widget.setGeometry(QtCore.QRect(14, 0, 771,
581))
        self.widget.setObjectName("widget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(sel
f.widget)
        self.horizontalLayout.setContentsMargins(0, 0, 0,
0)
        self.horizontalLayout.setObjectName("horizontalLa
yout")
        self.textEdit = QtWidgets.QTextEdit(self.widget)
        self.textEdit.setObjectName("textEdit")
        self.horizontalLayout.addWidget(self.textEdit)
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout
")
        self.checkConnectAPButton = QtWidgets.QPushButton
(self.widget)
        self.checkConnectAPButton.setObjectName("checkCon
nectAPButton")
```

```
        self.verticalLayout.addWidget(self.checkConnectAP
Button)
        self.checkConnecClientButton = QtWidgets.QPushBut
ton(self.widget)
        self.checkConnecClientButton.setObjectName("check
ConnecClientButton")
        self.verticalLayout.addWidget(self.checkConnecCli
entButton)
        self.rebootAPButton = QtWidgets.QPushButton(self.
widget)
        self.rebootAPButton.setObjectName("rebootAPButton
")
        self.verticalLayout.addWidget(self.rebootAPButton
)
        self.rebootClientButton = QtWidgets.QPushButton(s
elf.widget)
        self.rebootClientButton.setObjectName("rebootClie
ntButton")
        self.verticalLayout.addWidget(self.rebootClientBu
tton)
        self.enableWPSButton = QtWidgets.QPushButton(self
.widget)
        self.enableWPSButton.setObjectName("enableWPSButt
on")
        self.verticalLayout.addWidget(self.enableWPSButto
n)
        self.disableWPSButton = QtWidgets.QPushButton(sel
f.widget)
        self.disableWPSButton.setObjectName("disableWPSBu
tton")
        self.verticalLayout.addWidget(self.disableWPSButt
on)
        self.reloadAPSoftButton = QtWidgets.QPushButton(s
elf.widget)
        self.reloadAPSoftButton.setObjectName("reloadAPSo
ftButton")
        self.verticalLayout.addWidget(self.reloadAPSoftBu
tton)
        self.resetConfigAPButton = QtWidgets.QPushButton(
self.widget)
        self.resetConfigAPButton.setObjectName("resetConf
igAPButton")
        self.verticalLayout.addWidget(self.resetConfigAPB
utton)
```

```
        self.resetConfigClientButton = QtWidgets.QPushBut
ton(self.widget)
        self.resetConfigClientButton.setObjectName("reset
ConfigClientButton")
        self.verticalLayout.addWidget(self.resetConfigCli
entButton)
        self.horizontalLayout.addLayout(self.verticalLayo
ut)
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow"
, "модуль Администратор"))
        self.checkConnectAPButton.setText(_translate("Mai
nWindow", "Check connect AP"))
        self.checkConnecClientButton.setText(_translate("
MainWindow", "Check connect client"))
        self.rebootAPButton.setText(_translate("MainWindo
w", "Reboot AP"))
        self.rebootClientButton.setText(_translate("MainW
indow", "Reboot client"))
        self.enableWPSButton.setText(_translate("MainWind
ow", "Enable WPS"))
        self.disableWPSButton.setText(_translate("MainWin
dow", "Disable WPS"))
        self.reloadAPSoftButton.setText(_translate("MainW
indow", "Reload AP soft"))
        self.resetConfigAPButton.setText(_translate("Main
Window", "Reset config AP"))
        self.resetConfigClientButton.setText(_translate("
MainWindow", "Reset config client"))
```