

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа № 4, 5
Разработка защищенных приложений.

Выполнила студент гр. 853503
Яговдик О.И.

Проверил
Протьюко М.И

Минск 2021

1. Введение

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

Существует пять основных проблем, которые охватывают большую часть уязвимостей программного обеспечения. Благодаря их учету, возможно на порядок увеличить защищенность программ.

В англоязычной литературе существует два термина, которые могут быть переведены, как безопасное программирование.

Defensive programming (защитное, безопасное программирование) — принцип разработки ПО, при котором разработчики пытаются учесть все возможные ошибки и сбои, максимально изолировать их и при возможности восстановить работоспособность программы в случае неполадок. Это должно делать программное обеспечение более стабильным и менее уязвимым. Например, аппаратной реализацией данного принципа является сторожевой таймер, вычисление контрольной суммы — для выявления ошибок при пакетной передаче данных.

Secure coding (безопасное программирование) — методика написания программ, устойчивых к атакам со стороны вредоносных программ и злоумышленников. Безопасное программирование помогает защитить данные пользователя от кражи или порчи. Кроме того, небезопасная программа может предоставить злоумышленнику доступ к управлению сервером или компьютером пользователя; последствия могут быть различны: от отказа в обслуживании одному пользователю до компрометации секретной информации, потери обслуживания или повреждения систем тысяч пользователей.

Цель

Познакомиться с концепцией ролевого управления доступом и способами защиты программного обеспечения от существующих угроз.

Научиться разрабатывать приложения, которые используют ролевое управление доступом для разграничения полномочий пользователей.

Получить навыки защиты разработанной программы от несанкционированного копирования и других угроз, которым может подвергаться программное обеспечение.

2. Теоретические сведения

Типичные ошибки защищенного программирования

Существует пять основных проблем уязвимостей программного обеспечения:

- Переполнение буфера
- Уязвимости форматной строки
- Аутентификация
- Авторизация
- Криптография

Рассмотрим каждый из этих типов более подробно:

Переполнение буфера – пожалуй, наиболее часто используемая хакерами ошибка программирования. Достаточно сказать, что если программа удаленно подвержена переполнению буфера, хакер может получить полный контроль над системой. Первичная причина этой проблемы в том, что для вводимой информации используются статичные, фиксированного размера переменные, и злоумышленник может превысить размер отведенного под переменную буфера.

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

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

эксплоита обнаружил и взломал переполнение буфера. Действительно, как это происходит?

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

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

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

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

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

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

- Авторизация проведена некорректно. Интерфейс обеспечивает доступ к ресурсу до тех пор, пока обеспечивается аутентификация. Обычно, в этом случае требуется некий идентификатор, и предполагается, что он не может быть угадан или изменен. Например, в одном банковском Web-приложении Web-сервер устанавливал переменную в Javascript в номер аккаунта. Простым изменением этой переменной стало возможно просматривать и изменять другие аккаунты.
- Слишком много доверия к пользовательскому вводу. Например, HTTP куки – это один из типов пользовательского ввода в Web-приложение, который может быть изменен. Если приложение осуществляет авторизацию с помощью куки, оно может довериться фальсифицированной информации. Это может быть или имя пользователя, или запрашиваемый ресурс. Например, при испытании одного приложения была найдена переменная в куки с именем “admin”. Если она устанавливалась в “true”, приложение предоставляло нелегитимному пользователю администраторские права.
- Ошибки канонизации. Часто приложения принимают решения об авторизации на основе аутентификационной информации и запрошенных ресурсов. Многие уязвимости безопасности относятся к ошибкам канонизации в имени ресурса. Например, хотя приложение может запретить доступ к \secure\secret.txt, оно может дать доступ к \public\..\secure\secret.txt на основании имени директории. Кодировки в Юникод и в шестнадцатеричном формате относятся к той же категории.

Криптография

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

Основные факторы, определяющие безопасность программных средств

Если брать в расчет не только приложения, но и в целом программные средства, то любые программные средства, прежде всего, должны иметь экономическую, техническую, научную или социальную эффективность применения. Эта системная эффективность может быть описана количественно или качественно, их отличий от имеющихся у других программ, а также факторов и источников возможной эффективности.

В результате должна быть формализована цель использования и набор требований заказчика и пользователей при создании или приобретении ПС, а также его предполагаемое назначение и сфера применения.

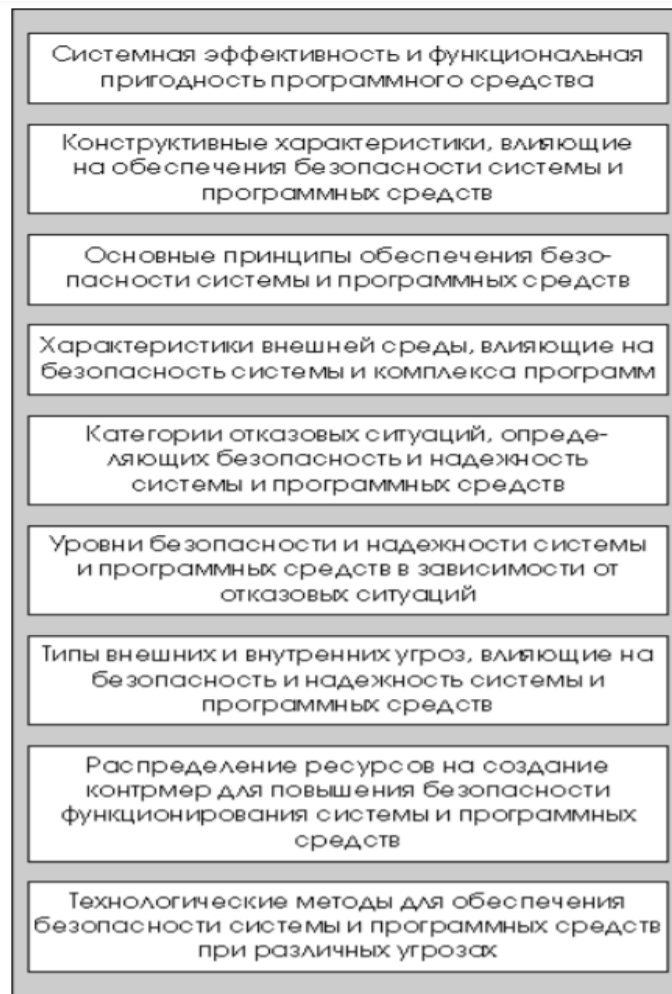
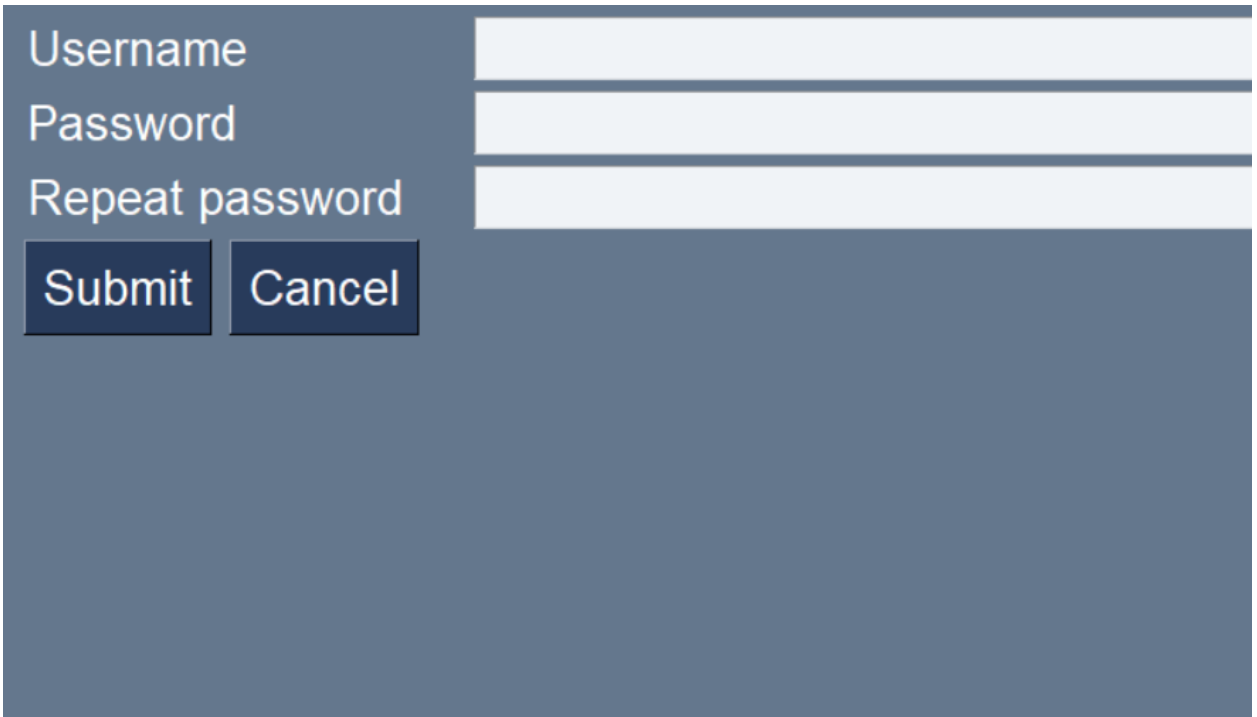


Схема 1. Основная совокупность факторов и действий по обеспечению функциональной безопасности систем и ПС

3. Результаты выполнения программы

Страничка регистрации

A screenshot of a registration form on a dark blue background. The form contains three input fields for 'Username', 'Password', and 'Repeat password'. Below the fields are two buttons: 'Submit' and 'Cancel'.

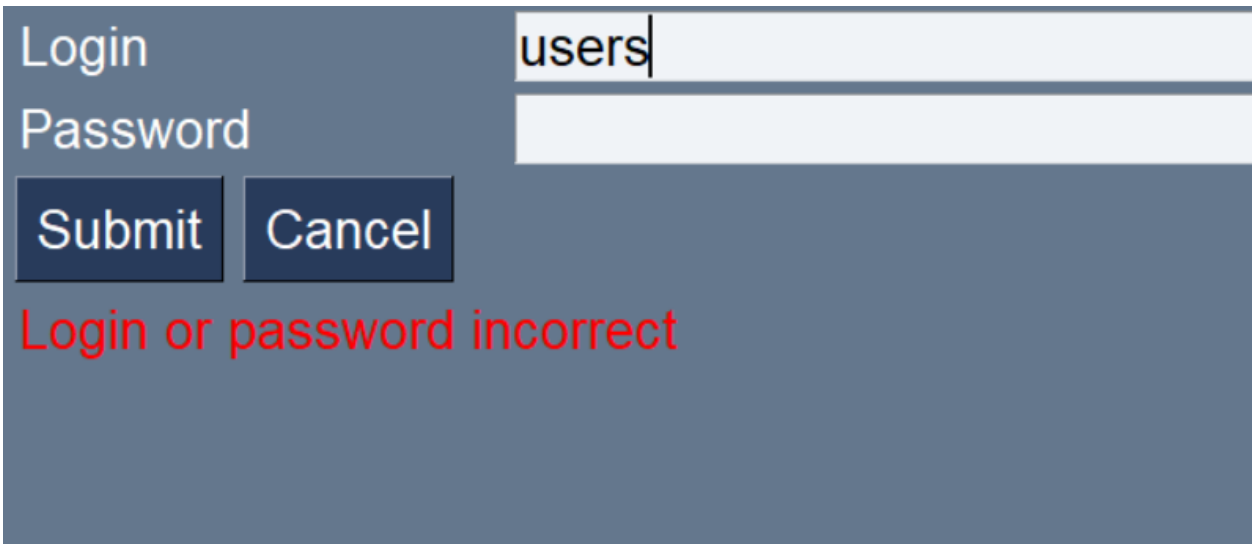
Username

Password

Repeat password

Submit Cancel

Страничка входа, показывающее если заполнить не все поля

A screenshot of a login form on a dark blue background. The 'Login' field contains the text 'users' and the 'Password' field is empty. Below the fields are 'Submit' and 'Cancel' buttons. A red error message 'Login or password incorrect' is displayed at the bottom.

Login users

Password

Submit Cancel

Login or password incorrect

Страничка, что отображает что будет видеть пользователь после входа

Reload

Update my status

Logout

username: qwe role: User status: 2

Страничка, что отображает что будет видеть админ после входа

Reload

Update my status

Logout

username: admin1 role: Admin status:

username: asd role: User status: 22

username: qwe role: User status: 12

username: zxc role: Moderator status:

username: rfv role: Admin status: φsdf

username: tyui role: User status: jjj

username: admin11 role: Admin status: 11

4.ВЫВОД

В данной лабораторной работе реализовали приложение с графическим интерфейсом, которое может проводить аутентификацию пользователя, у каждого могут быть роли, членам которой доступны различные функциональные возможности программы. Также осуществляется защита от 4 типов атак на приложение, что дает нам понять, что безопасность будет всегда актуальной проблемой.

И в настоящее время безопасность является значимой частью для разработки программ. Во многих компаниях именно безопасность ставят приоритетом номер 1(примером может служить заявление Билла Гейтса в Microsoft). Стремление улучшить безопасность в своих продуктах является гарантом и возможностью улучшить ситуацию для людей и корпораций во всем мире.

И если рассматривать Microsoft как крупнейшую компанию с баснословными ресурсами и многочисленным штатом сотрудников, если у них есть проблемы с безопасностью что уж говорить о маленьких предприятиях.

Как скажет вам любой хороший профессионал по безопасности, невозможно создать полностью безошибочное и неуязвимое программное обеспечение. Ресурсы и финансы, требуемые для создания такого программного обеспечения, будут бесконечны. Но даже несмотря на это стоит озаботиться об уменьшениях риска и учесть основные проблемы и ошибки, что охватывают 90% уязвимостей ПО.

5. КОД ПРОГРАММЫ

```
import PySimpleGUI as sg
from db import *
from app import *

SIZE = (800, 600)

def start(in_data=None):
    layout = [
        [sg.Button('Login',size=(10, 5)), sg.Button('Register',size=(10, 5))]
    ]

    window = sg.Window(
        'App',
        layout,
        default_element_size=(90, 10),
        auto_size_text=True,
        font=('Helvetica', 20),
        size=SIZE,
        default_button_element_size=(90, 10)
    )

    next_page = None

    while True:
        event, values = window.read()
        if event == 'Login':
            next_page = login
            break
        if event == 'Register':
            next_page = register
            break
        if event in (None, 'Exit', 'Cancel'):
            break
    window.close()

    if next_page:
        next_page()

def login(in_data=None):
    layout = [
        [sg.Text('Login', size=(15, 1)), sg.InputText(key='login')],
        [sg.Text('Password', size=(15, 1)), sg.InputText(key='password',
password_char='*')],
        [sg.Submit(), sg.Cancel()],
        [sg.Text('', text_color='red', key='explanation', size=(40, 1))]
    ]

    window = sg.Window(
        'App',
        layout,
        default_element_size=(90, 10),
        auto_size_text=True,
        font=('Helvetica', 20),
        size=SIZE
    )

    next_page = None
    data = None

    while True:
```

```

        event, values = window.read()
        if event == 'Submit':
            user, comment = get_user(values)
            if user:
                next_page = application
                data = (user, '')
                break
            else:
                window['explanation'].update(comment)
        if event in (None, 'Exit', 'Cancel'):
            next_page = start
            break
    window.close()

    if next_page:
        next_page(data)

def register(in_data=None):
    layout = [
        [sg.Text('Username', size=(15, 1)), sg.InputText(key='login')],
        [sg.Text('Password', size=(15, 1)), sg.InputText(key='password1',
password_char='*')],
        [sg.Text('Repeat password', size=(15, 1)),
sg.InputText(key='password2', password_char='*')],
        [sg.Submit(), sg.Cancel()],
        [sg.Text('', text_color='red', key='explanation', size=(40, 1))]
    ]
    window = sg.Window(
        'App',
        layout,
        default_element_size=(90, 10),
        auto_size_text=True,
        font=('Helvetica', 20),
        size=SIZE
    )

    next_page = None
    data = None

    while True:
        event, values = window.read()
        if event == 'Submit':
            user, comment = create_user(values)
            if user:
                next_page = application
                data = (user, '')
                break
            else:
                window['explanation'].update(comment)
        if event in (None, 'Exit', 'Cancel'):
            next_page = start
            break
    window.close()

    if next_page:
        next_page(data)

def prepare_user_layout(user: UserModel, comment: str):
    layout = [
        [
            sg.Button('Reload'),
            sg.Button('Update my status'),

```

```

        sg.Button('Logout', key='Cancel')
    ],
    [sg.Text(comment, text_color='red')],
    [
        sg.Text('username: ', size=(10, 1)),
        sg.Text(user.username, size=(10, 1), text_color='Yellow'),
        sg.Text('role: ', size=(5, 1)),
        sg.Text(ROLES[user.role], size=(10, 1), text_color='Yellow'),
        sg.Text('status: ', size=(5, 1)),
        sg.InputText(user.status, size=(15, 1), key='status',
text_color='Black'),
    ],
]

all_users = get_all_users(user.role)
for index, u in enumerate(all_users):
    if u.username != user.username :
        if check_role(user.role, Action.ModifyUsers):
            layout += [[
                sg.Text('username: ', size=(10, 1)),
                sg.InputText(u.username, size=(10, 1),
text_color='Black', key=str(index) + '-username'),
                sg.Text('role: ', size=(5, 1)),
                sg.InputCombo(ROLES, default_value=ROLES[u.role],
text_color='Black', key=str(index) + '-role'),
                sg.Text('status: ', size=(5, 1)),
                sg.InputText(u.status, size=(15, 1), text_color='Black',
key=str(index) + '-status'),
                sg.Button('U', button_color=('black', 'green'),
key=str(index) + '-update')
            ]]
        elif check_role(user.role, Action.ModifyStatuses):
            layout += [[
                sg.Text('username: ', size=(10, 1)),
                sg.Text(u.username, size=(10, 1), text_color='Yellow'),
                sg.Text('role: ', size=(5, 1)),
                sg.Text(ROLES[u.role], size=(10, 1),
text_color='Yellow'),
                sg.Text('status: ', size=(5, 1)),
                sg.InputText(u.status, size=(15, 1), text_color='Black',
key=str(index) + '-status'),
                sg.Button('U', button_color=('black', 'green'),
key=str(index) + '-' + 'update')
            ]]
        else:
            layout += [[
                # sg.Text('username: ', size=(10, 1)),
                # sg.Text(u.username, size=(10, 1), text_color='Yellow'),
                # sg.Text('role: ', size=(5, 1)),
                # sg.Text(ROLES[u.role], size=(10, 1),
text_color='Yellow'),
                # sg.Text('status: ', size=(5, 1)),
                # sg.Text(u.status, size=(15, 1), text_color='Yellow')
            ]]

    return layout

def application(in_data=None):
    user, comment = in_data
    assert user
    assert type(user) == UserModel

    layout = prepare_user_layout(user, comment)

```

```

window = sg.Window('App', layout, default_element_size=(90, 10),
auto_size_text=True, font=('Helvetica', 20),
                    size=SIZE)

next_page = None
new_user = None
comment = ''

while True:
    event, values = window.read()
    print(event, values)

    if event and event.endswith('-update'):
        id = event.split('-')[0]
        username = values.get(id + '-username')
        role = values.get(id + '-role')
        status = values.get(id + '-status')

        if username:
            resp = update_user_username(int(id), username)
            if resp:
                _, comment = resp
        if role:
            resp = update_user_role(int(id), role)
            if resp:
                _, comment = resp
        if status:
            resp = update_user_status(int(id), status)
            if resp:
                _, comment = resp
        event = 'Reload'

    if event == 'Update my status':
        resp = update_user_status(user.username, values['status'])
        if resp:
            _, comment = resp
        event = 'Reload'
    if event == 'Reload':
        next_page = application
        new_user = get_user_by_username(user.username)
        break
    if event in (None, 'Exit', 'Cancel'):
        next_page = start
        break
window.close()

if next_page:
    next_page((new_user, comment))

def check_installation():
    try:
        with open('appdata.txt') as f:
            appdata = f.read()
            if is_lisense_key(appdata):
                return True
    except Exception as e:
        pass
    return False

def install():
    key = input('Enter your key\n')
    if is_lisense_key(key):

```

```
        print('Ok')
        with open('appdata.txt', 'w') as f:
            f.write(key)
        start()
    else:
        print('Wrong key, bye bye')

if __name__ == '__main__':
    if check_installation():
        start()
    else:
        install()
```