

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

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа № 6  
Защита ПО от несанкционированного использования.

Выполнила студент гр. 853503  
Яговдик О.И.

Проверил  
Протько М.И

Минск 2021

## **1. Введение**

На сегодняшний день для любого коммерческого приложения является актуальным решение защиты авторских прав либо пользователя системы. Все авторские права объединяет то, что они заключаются в праве автора (правообладателя) совершать определенные действия и запрещать такие действия другим лицам. При этом предполагается, что разрешение автора отсутствует. Несоблюдение запрета является неправомерным. Таким образом, нарушение авторских прав – действия субъектов права, выражающиеся в несоблюдении личных неимущественных прав автора или исключительного права на использование произведения.

Авторское право – право, позволяющее регулировать правоотношения, связанные с созданием и использованием (изданием, исполнением, показом и т. д.) произведений науки, литературы или искусства, то есть объективных результатов творческой деятельности людей в этих областях.

Одним из основных способов защиты авторских прав разработчика является запутывание (obfuscated) или обфускация программного кода.

## Цель

- I. Реализовать на выбор 3 метода обфуксации программного кода приложения, разработанного в рамках лабораторных работ 4,5, позволяющие защитить ПО от несанкционированного использования в следующих комбинациях:
  - ✓ По одному.
  - ✓ Любые 2 на выбор из трех одновременно.
  - ✓ Все три одновременно.
- II. Протестировать работоспособность приложения с запутанным программным кодом.
- III. Проверить и пояснить следующие свойства, которым должна удовлетворять запутанная программа:
  - ✓ Запутывание должно быть *замаскированным*. То, что к программе были применены запутывающие преобразования, не должно бросаться в глаза.
  - ✓ Запутывание не должно быть регулярным. Регулярная структура запутанной программы или её фрагмента позволяет человеку отделить запутанные части и даже идентифицировать алгоритм запутывания.

## 2. Теоретические сведения

Обфускация – процесс, в результате которого код программы приобретает вид, трудный для анализа. Обфускация, как нетрудно догадаться, осуществляется с целью защиты программного кода и алгоритмов, которые он реализует, от чужих глаз, которые, несмотря на строжайшие запреты разработчиков, любят подсматривать за тем, что и каким именно образом написано.

Существует несколько разных видов обфускации в зависимости от того, какой именно код запутывается при ней. Для тех языков программирования, для которых программа представляется прямо в виде исходных текстов, используются специальные методы, делающие код нечитабельным для человека, но приемлемым для интерпретатора. Среди них – запись программы в одну строку, замена имён переменных и функций, вставка ничего не значащих комментариев.

Для обфускации двоичного кода (как для исполняемых файлов, так и двоичного кода виртуальных машин .NET и Java) используются уже более серьёзные методы, в том числе и шифрование. Однако следует учитывать, что для двоичного кода обфускация способна серьёзно снизить скорость работы защищённых участков программы. Поэтому критичные ко времени выполнения куски кода, как правило, не имеют серьёзной криптографической защиты.

### Цели обфускации

- 1) Затруднение декомпиляции/отладки и изучения программ с целью обнаружения функциональности.
- 2) Затруднение декомпиляции проприетарных программ с целью предотвращения обратной разработки или обхода DRM и систем проверки лицензий.
- 3) Оптимизация программы с целью уменьшения размера работающего кода и (если используется некомпилируемый язык) ускорения работы.
- 4) Демонстрация неочевидных возможностей языка и квалификации программиста (если производится вручную, а не инструментальными средствами).

«Запутывание» кода может осуществляться на уровне алгоритма, исходного текста и/или ассемблерного текста. Для создания запутанного ассемблерного текста могут использоваться специализированные компиляторы, использующие неочевидные или недокументированные возможности среды исполнения программы. Существуют также специальные программы, производящие обфускацию, называемые обфускаторами (англ. obfuscator). Описание запутывающих преобразований приведено в приложении.

## **Что такое обфускация кода с точки зрения эффективного управления разработкой**

При управлении разработкой оценивается целесообразность и эффективность обфускации. Другими словами, на что вам, как руководителю, следует обращать внимание при выборе исполнителя или постановке такой задачи в принципе?

### **1. Скрытность**

Оцените степень сокрытия алгоритмов управления программой. Целесообразно ли проведение control-flow обфускации (применяется, например в Apple FairPlay для библиотек iTunes или iOS), когда отслеживается обновление управляющей переменной и заменяется переход к узлу диспетчера переходами к следующему блоку, что соответствует новому значению управляющей переменной.

### **2. Стоимость**

Рентабельность метода обфускации. Необходимо оценить, насколько оправданы затраты, чтобы выбранный метод можно было широко применять в нескольких аналогичных приложениях.

### **3. Защита**

В какой степени преобразованный код нечитаем в сравнении с исходным. Метрики сложности программного обеспечения определяют различные меры защиты. Например, количество содержащихся в нем предикатов, глубина иерархии, уровни вложенности и так далее. Цель хорошего проектирования программного обеспечения — минимизировать сложность на основе этих параметров, когда как цель обфускации — максимально усложнить.

### **4. Стабильность**

Определяет, насколько хорошо преобразованный код может противостоять автоматическим атакам деобфускации. Наивысшая степень устойчивости — одностороннее преобразование, которое не может быть отменено деобфускатором. Например, обфускация удаляет такую информацию, как форматирование исходного кода.

## Недостатки

- Потеря гибкости кода

Код после обфускации может стать более зависимым от платформы или компилятора.

- Трудности отладки

Обфускатор не даёт постороннему выяснить, что делает код, но также не даёт и разработчику отлаживать его. При отладке приходится отключать обфускатор.

- Недостаточная безопасность

Хотя обфускация помогает сделать распределённую систему более безопасной, не стоит ограничиваться только ею. Обфускация — это безопасность через неясность. Ни один из существующих обфускаторов не гарантирует сложности декомпиляции и не обеспечивает безопасности на уровне современных криптографических схем. Вполне вероятно, что эффективная защита невозможна (по крайней мере в некотором конкретном классе решаемых задач).

- Ошибки в обфускаторах

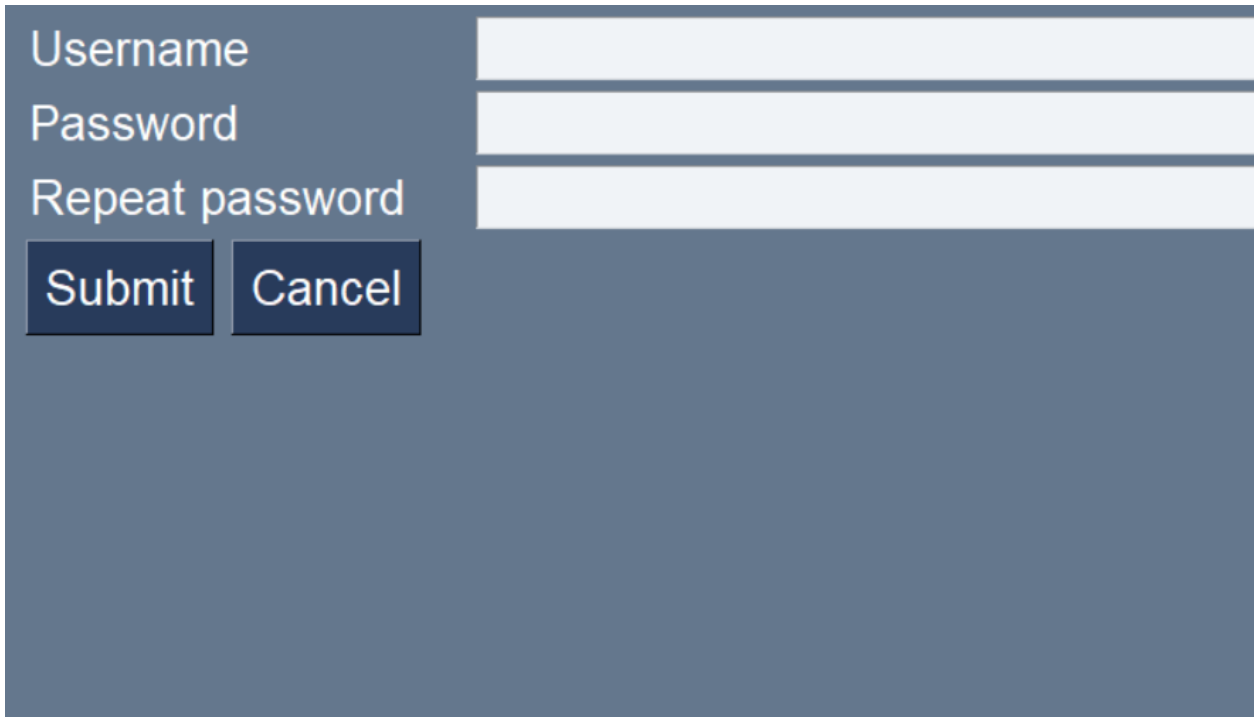
Современный обфускатор — сложный программный комплекс. Зачастую в обфускаторы, несмотря на тщательное проектирование и тестирование, вкрадываются ошибки. Так что есть ненулевая вероятность, что прошедший через обфускатор код вообще не будет работать. И чем сложнее разрабатываемая программа, тем больше эта вероятность.

- Вызов класса по имени

Большинство языков с промежуточным кодом может создавать или вызывать объекты по именам их классов. Современные обфускаторы позволяют сохранить указанные классы от переименования, однако подобные ограничения сокращают гибкость программ.

### 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'.

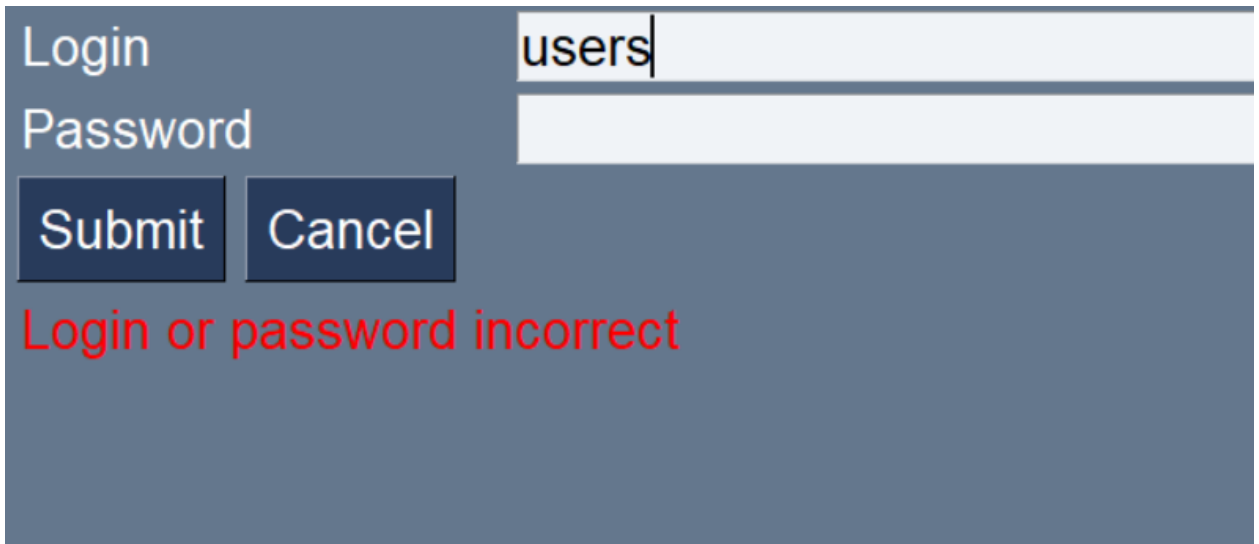
Username

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 мы рассмотрели такое понятие как обфускация. Его цели, задачи, применения и т.д. Также рассмотрели описание запутывающих преобразований: на что нацелено, преобразование форматирования и потока управления.

В ходе работы можно отметить следующие свойства, которым должна удовлетворять запутанная программа:

- Запутывание должно быть замаскированным. То, что к программе были применены запутывающие преобразования, не должно бросаться в глаза.
- Запутывание не должно быть регулярным. Регулярная структура запутанной программы или её фрагмента позволяет человеку отделить запутанные части и даже идентифицировать алгоритм запутывания.
- Применение стандартных синтаксических и статических методов анализа программ на начальном этапе её анализа не должно давать существенных результатов, так как быстрый результат может воодушевлять человека, а его отсутствие, наоборот, угнетать.

## 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()
```