Министерство образования и науки Российской Федерации

1. Санкт-Петербургский Политехнический Университет Петра Великого
2. —
3. Институткибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 6**

**Защита программного обеспечения от нелегального использования**

по дисциплине «Основы информационной безопасности»

1. Выполнил
2. студент гр. 4851003/10002 Лобов Е.А

1. Ассистент
   1. преподавателя ЗубковЕ.А.
3. Санкт-Петербург
4. 2022

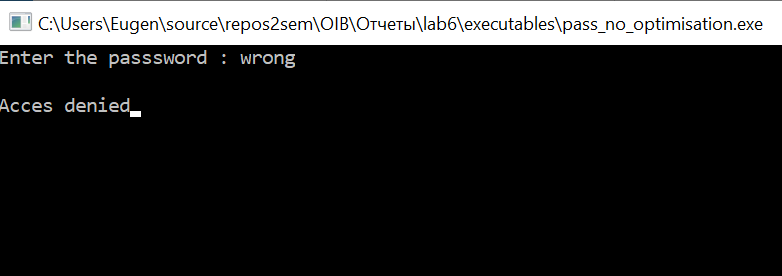
# цель работы

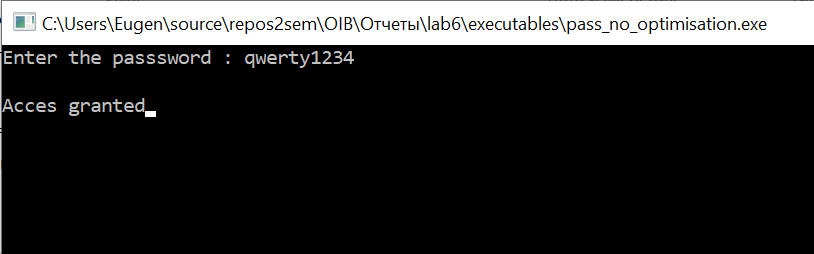
Цель работы ― приобретение навыков по защите приложений от нелегального использования, по анализу исполняемых кодов в отсутствие исходных текстов и по применению способов защиты программы от дизассемблирования и отладки.

# ход работы

## Работа с паролем

Была реализована программа на языке Си, которая запрашивает у пользователя пароль. При ошибочной комбинации всплывает предупреждение об ошибке, при верной появляется уведомление о предоставлении доступа. Код программы находится в папке code(pass.c).



Демонстрация работы программы

С помощью утилиты IDA рассмотрим дизассемблированный код программы. eax сравнивается с единицей, тем самым проверяется правильность введенного пароля. Инструкция jnz если результат сравнения 0 продолжит исполнение кода при неправильном пароле, а если 1 при правильном.

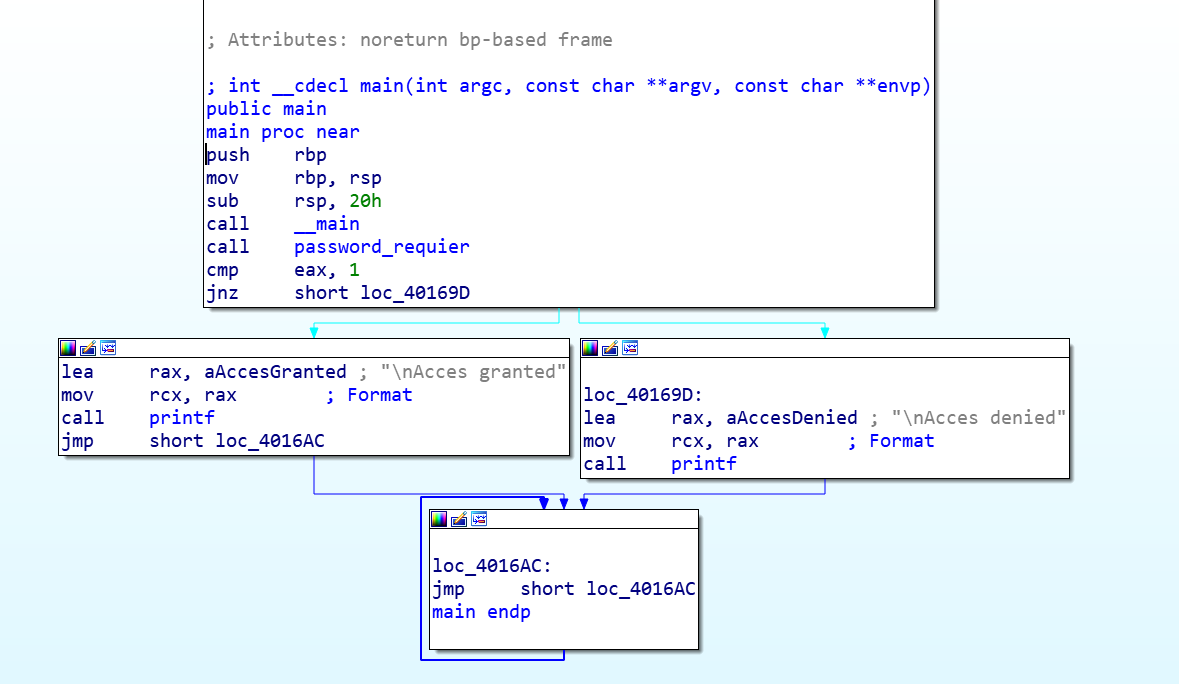


Рисунок 2 ― Дизассемблированный код программы

Заменю инструкцию jnz на jz. Так доступ будет предоставлен даже при неверно введенном пароле.

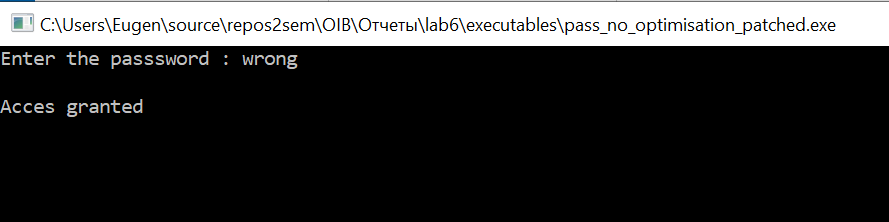


Рисунок 3 ― Демонстрация работы программы после изменения

Для исключения открытого хранения пароля необходимо добавить функцию XOR. Зашифрованный этой функцией пароль находится в папке executables(password\_coded.txt).

## Работа с Nag.exe

При вводе текста в файле каждые 7-8 секунд появляется окно со счётчиком, которое нельзя закрыть, пока счётчик не дойдёт от 100 до 0.

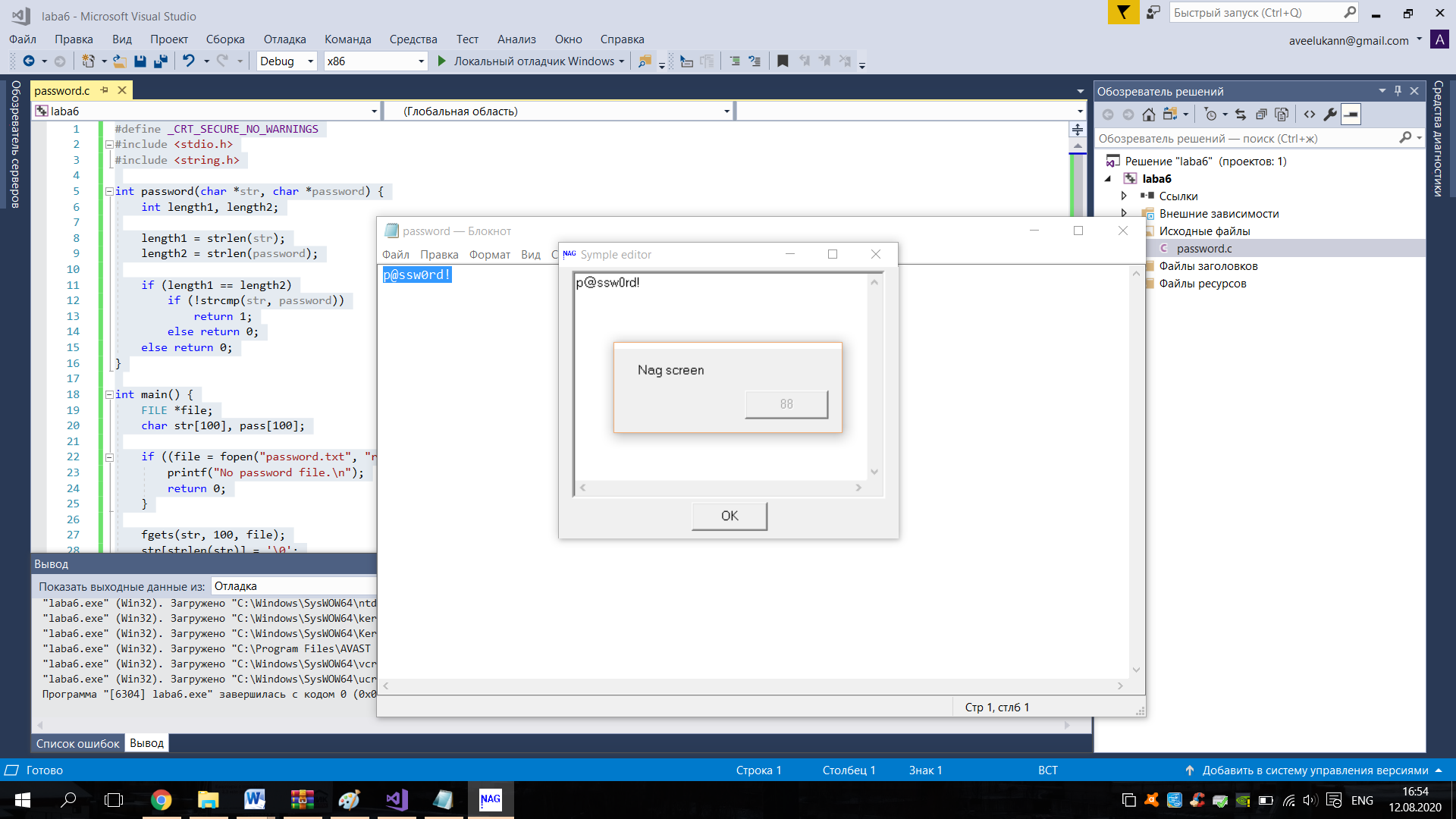


Рисунок 4 ― Демонстрация работы Nag.exe

За счётчик отвечает функция SetTimer. Но если эта функция не будет вызвана, кнопка ОК появится сразу. Для этого меняю инструкцию, сравнивая eax с 0.

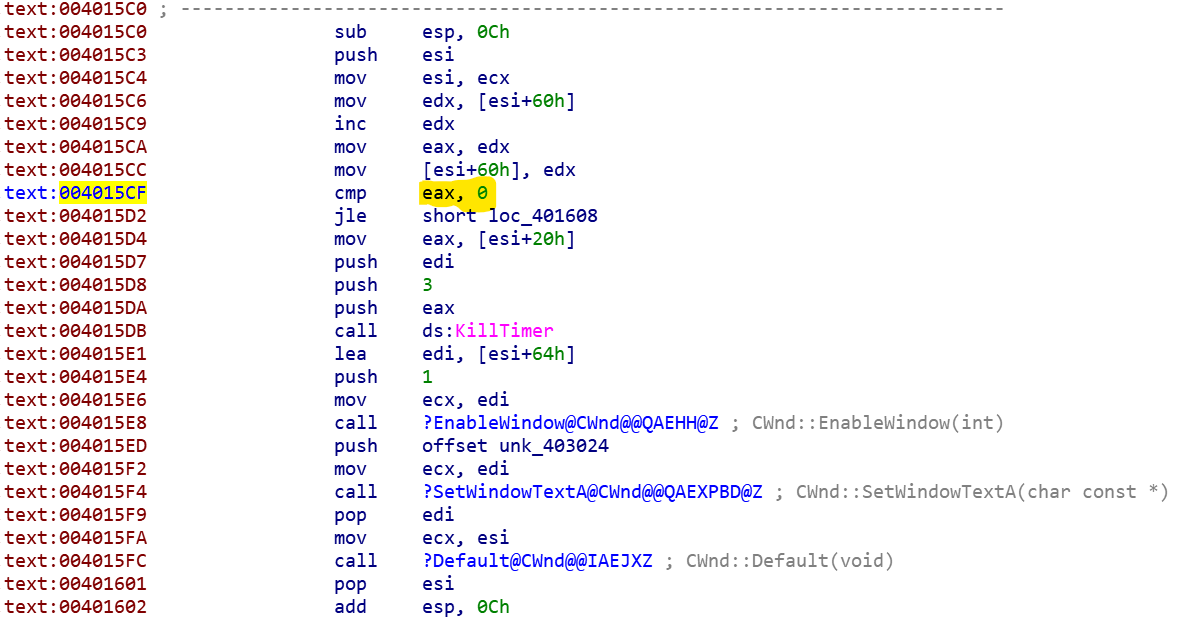


Рисунок 5 ― Дизассемблированный код Nag.exe

После этого программа стала работать без счётчика, то есть окно «ОК» появляется сразу.

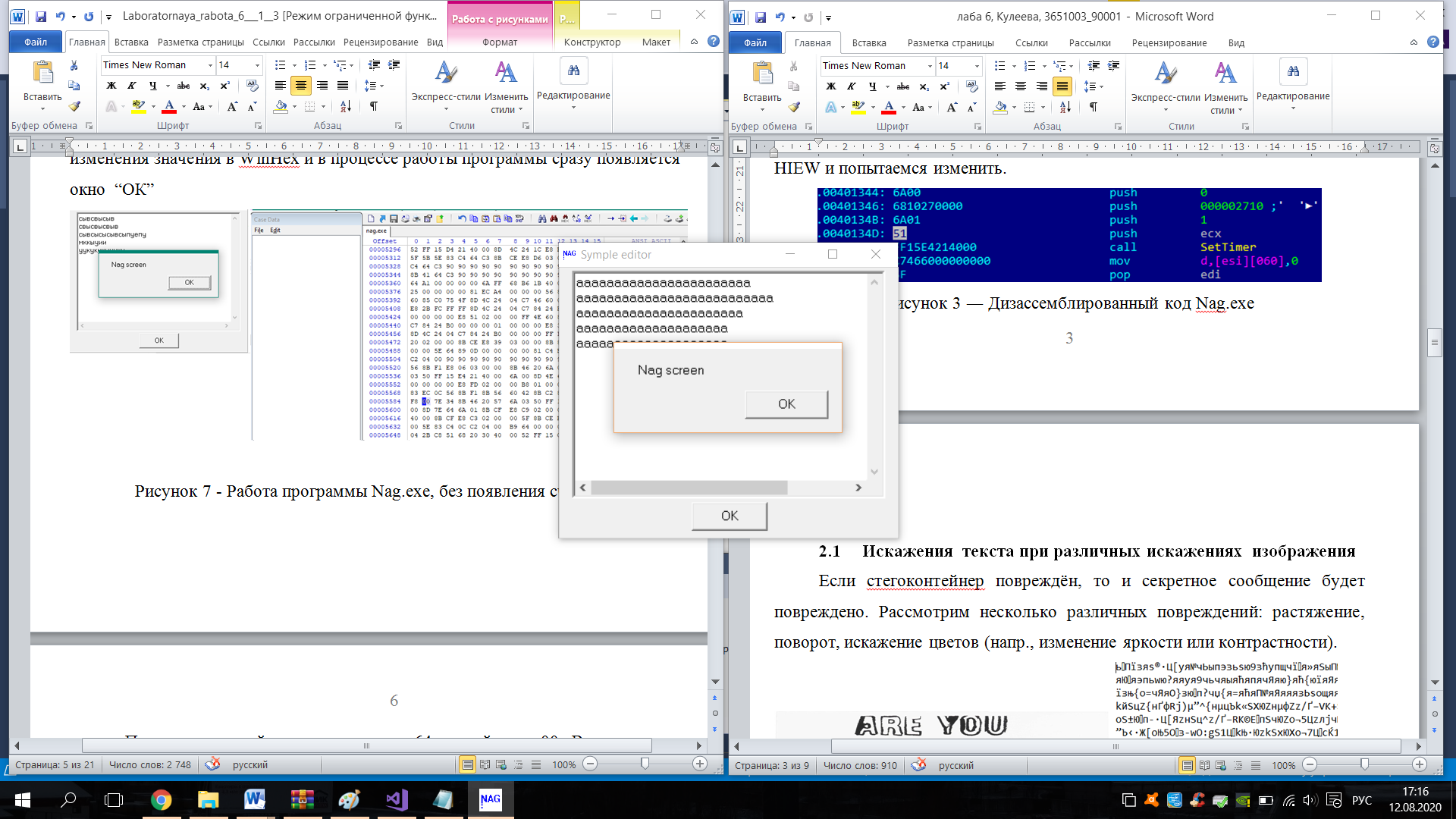


Рисунок 7 ― Демонстрация работы Nag.exe без счётчика

## Работа с Guard.exe

Данная утилита запрашивает пароль длиною не более 6 символов. При вводе неверной комбинации программа завершается.

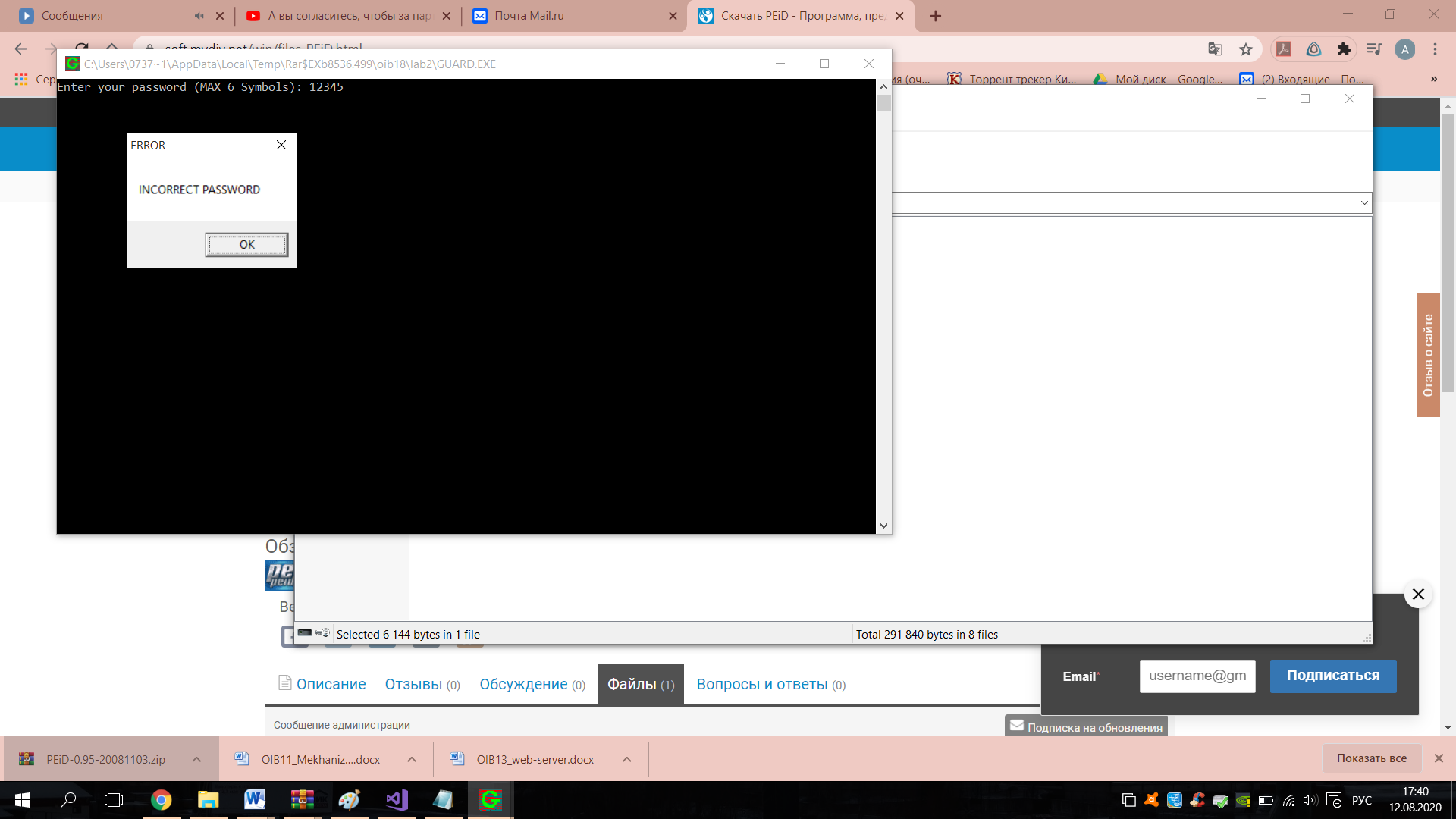


Рисунок 10 ― Демонстрация работы Guard.exe

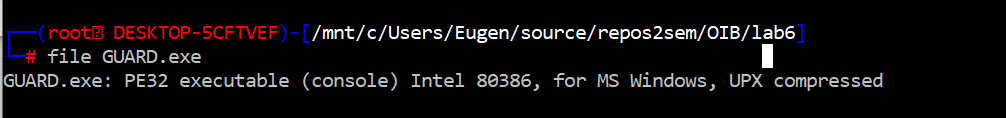
С помощью инструментов kali - linux определим упаковщик(Рисунок 11). 

Рисунок 11.

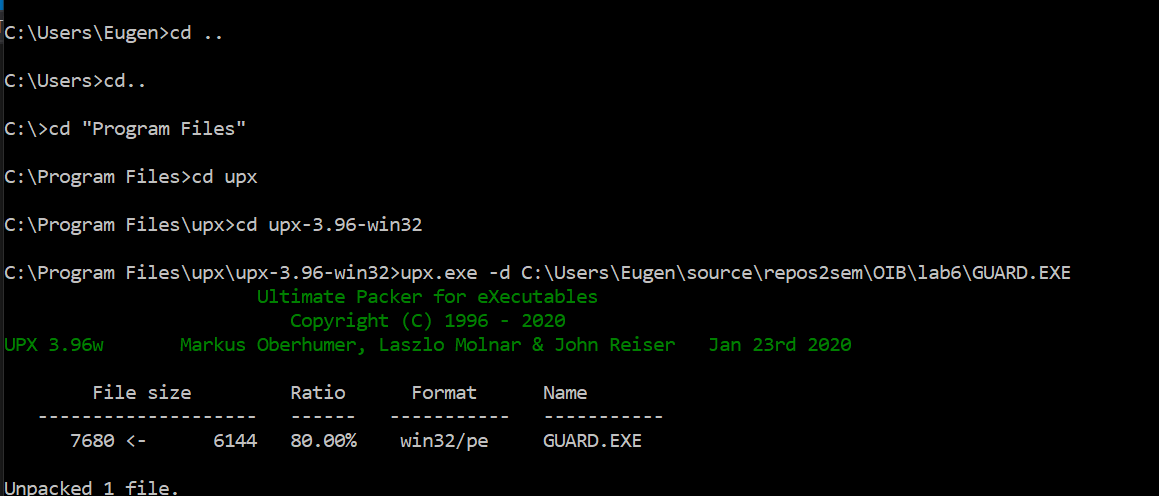
Распаковать файл можно с помощью UPX (см. Рисунок 12).

Рисунок 12.

Теперь код можно дизассемблировать. С помощью утилиты IDA pro рассмотрим структуру программы (см. Рисунок 13).

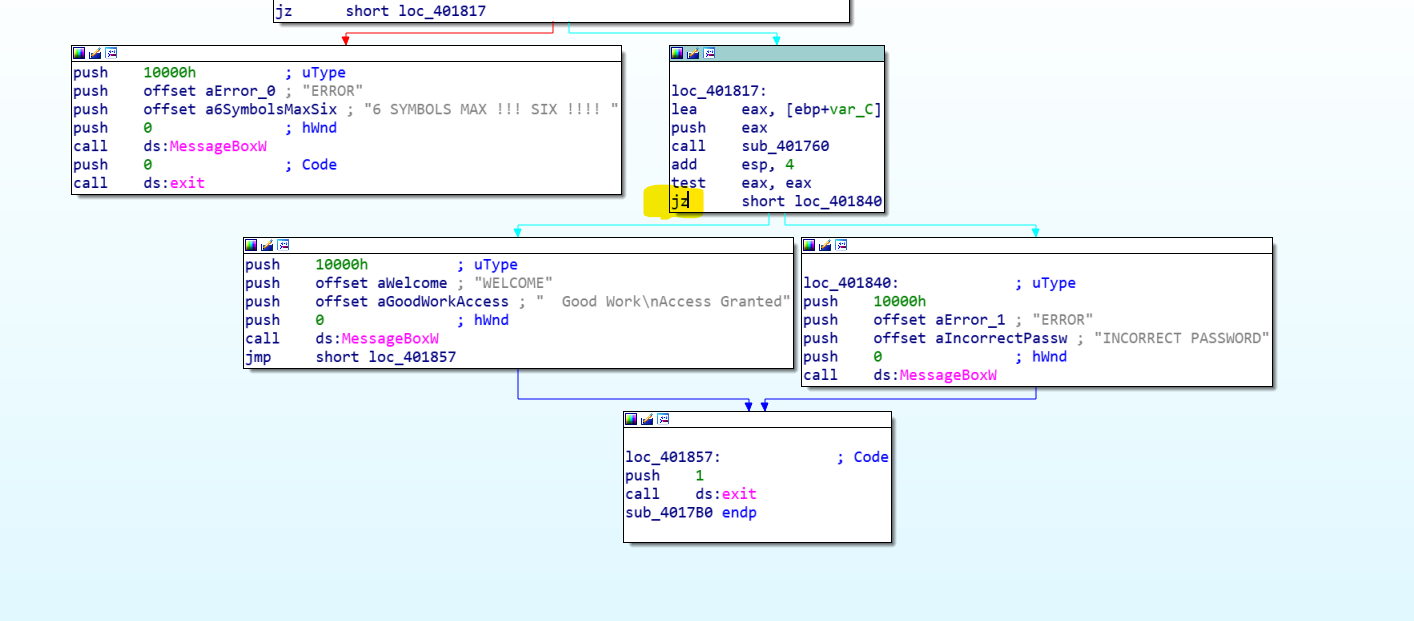


Рисунок 13 ― Структура программы Guard.exe

Также как в случае с программой из пункта 2.1 заменим jz на jnz. После этого программа должна предоставить доступ при вводе абсолютно любого пароля. Но при использовании отладчика доступ по-прежнему не будет предоствален.

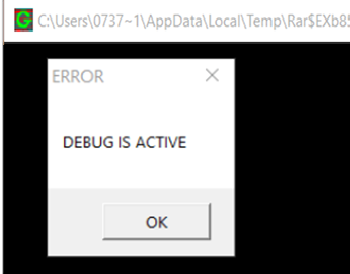


Рисунок 13 ― Работа программы после изменения

Рассмотрим структуру программы, а именно часть с проверкой на наличие отладчика (см. Рисунок 14). Как и в предыдущем пункте, необходимо заменить jz на jnz. Теперь всё получилось: программа предоставляет доступ при вводе абсолютно любого пароля (см. Рисунок 15).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 14 ― Структура проверки использования отладчика

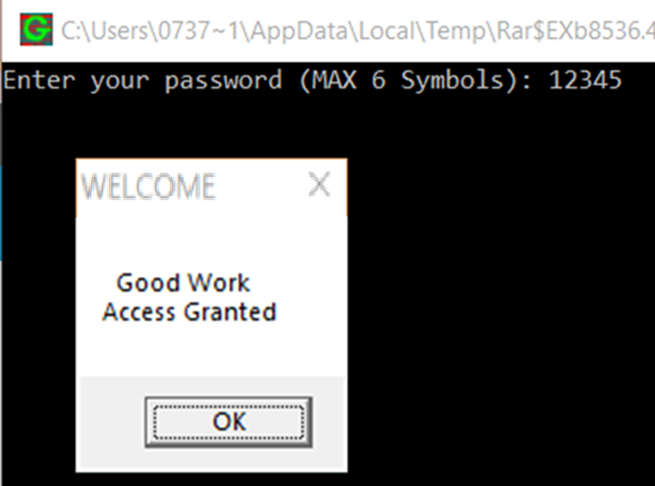


Рисунок 15 ― Демонстрация работы Guard.exe после изменения кода

## Разработка обфускатора

Обфускатор ― это программа, которая изменяет исходный текст программы таким образом, что он не теряет своей функциональности, но при этом становится труднее для понимания. Обфускатор изменяет имена переменных и функций, добавляет мусорные переменные, удаляет перенос строки, табуляцию и комментарии. Код программы обфускатора представлен в папке codes(obfuscator.c). Код программы после обфускации представлен в той же папке(obfuscated.c). Параметры для обфускации хранятся в текстовом файле (pars.txt).

# Вывод

В ходе выполнения данной работы были приобретены навыки по защите приложений от нелегального использования, по анализу исполняемых кодов в отсутствие исходных текстов и по применению способов защиты программы от дизассемблирования и отладки. Обойти защиту было легко, нужно было всего лишь поменять одну ассемблерную команду. Также в рамках данной работы был реализован обфускатор.

# ответы на контрольные вопросы

## Какие методы применяются для защиты коммерческих программ от ввода некорректных данных?

* Запрос пароля у пользователя
* Запрет на открытое хранение пароля во внешнем файле
* Появление всплывающего окна, мешающего использованию программы

## По каким признакам можно найти интересующий код сопоставления с паролем в программе?

Сопоставлению пароля в программе будет соответствовать команда CMP или команда условного перехода.

## Какие существуют основные методы защиты от дизассемблирования и отладки программ?

* Проверка целостности программ;
* Применение упаковки и шифрования распространяемого программного обеспечения;
* Проверка наличия отладчика;
* Применение оптимизирующего компилятора;

## Как реализуется обфускация с помощью виртуальных машин?

Создается случайная виртуальная машина, в которой создается случайный набор инструкций, после чего весь код трансформируется под виртуальную машину.

## Предложите методы усиления механизмов защиты программ от нелегального использования, исследованные в данной работе.

Шифрование данных пользователя, проверка на наличие отладчика, шифрование исходного кода с целью усложнения или полного запрета на его модификацию.