МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

Кафедра «Информационные системы»

**ОТЧЕТ**

о выполнении лабораторной работы №7

по дисциплине “Технические средства информационных систем”

Выполнил: ст. гр. ИС/б-20-2-о

Белик Г. М.

«Информационные системы»

Минкин С. И.

Севастополь

2022

**ЛАБОРАТОРНАЯ РАБОТА №7**

**«ИССЛЕДОВАНИЕ БЕЗОПАСНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ В СРЕДЕ ОТЛАДЧИКА OLLYDBG»**

**Цель работы**

Углубление знаний архитектуры 32-разрядных процессоров и системы команд языка ассемблера. Исследование методов защиты программного обеспечения информационных систем и ее нейтрализации, приобретение практических навыков исследования и отладки программ с помощью пакета OllyDbg.

**Постановка задачи**

1. Повторить теоретический материал, касающийся архитектуры 32-разрядных микропроцессоров, программно-доступных регистров и системы команд языка ассемблера (выполняется при домашней подготовке).

2 Исследовать способы парольной защиты в программе CRAСKME1.EXE. Для этого выполнить последовательность действий, описных в п. 2.2 настоящих методических указаний. Изменить программу таким образом, чтобы принимался любой вводимый пароль, независимо от того, верный он или неверный.

3. С помощью отладчика OllyDbg исследовать способы парольной защиты программ CRAСKME2.EXE, CRAСKME3.EXE и CRAСKME4.EXE, которые расположены в папке лабораторных работ. Определить на каких языках написаны программы. Изменить программы таким образом, чтобы

принимался любой вводимый пароль, независимо от того, верный он или неверный.

4. С помощью отладчика OllyDbg исследовать способ защиты программы CRAСKME5.EXE. Определите на каком языке написана программа. В данной программе ключ генерируется по введенному в первом поле имени.

5. Разработать рекомендации по усилению защиты вскрытия пароля.

**Ход работы**

CRAСKME1.EXE

Для начала требовалось узнать язык, на котором написана программа, и зашифрована ли она. Для этого была использована утилита PEiD:

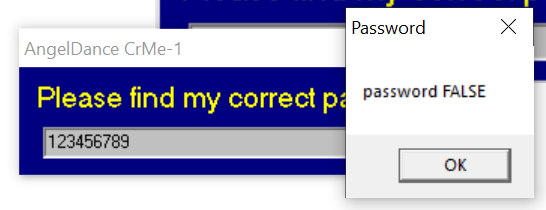


Рисунок 1 – Реакция программы на ввод пароля

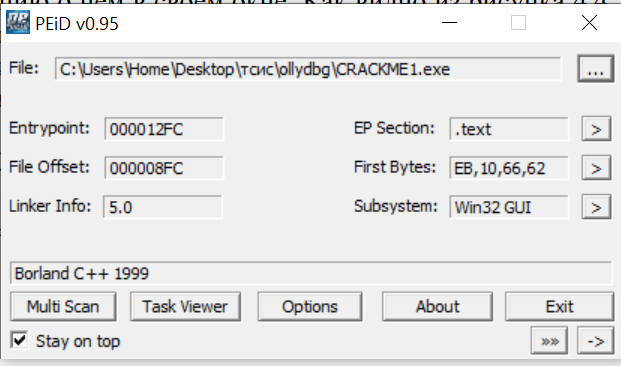


Рисунок 2 – Приложение CRACKME1.EXE в PEiD

Как видно, исследуемая программа написана на языке программирования C++ фирмы Borland и ничем не зашифрована и не запакована.

Далее программа была открыта в OllyDbg. Необходимо было найти в программе выводимые при вводе пароля строки (например,«password FALSE») с помощью последовательности ПКМ –> Search for –> All referenced strings.

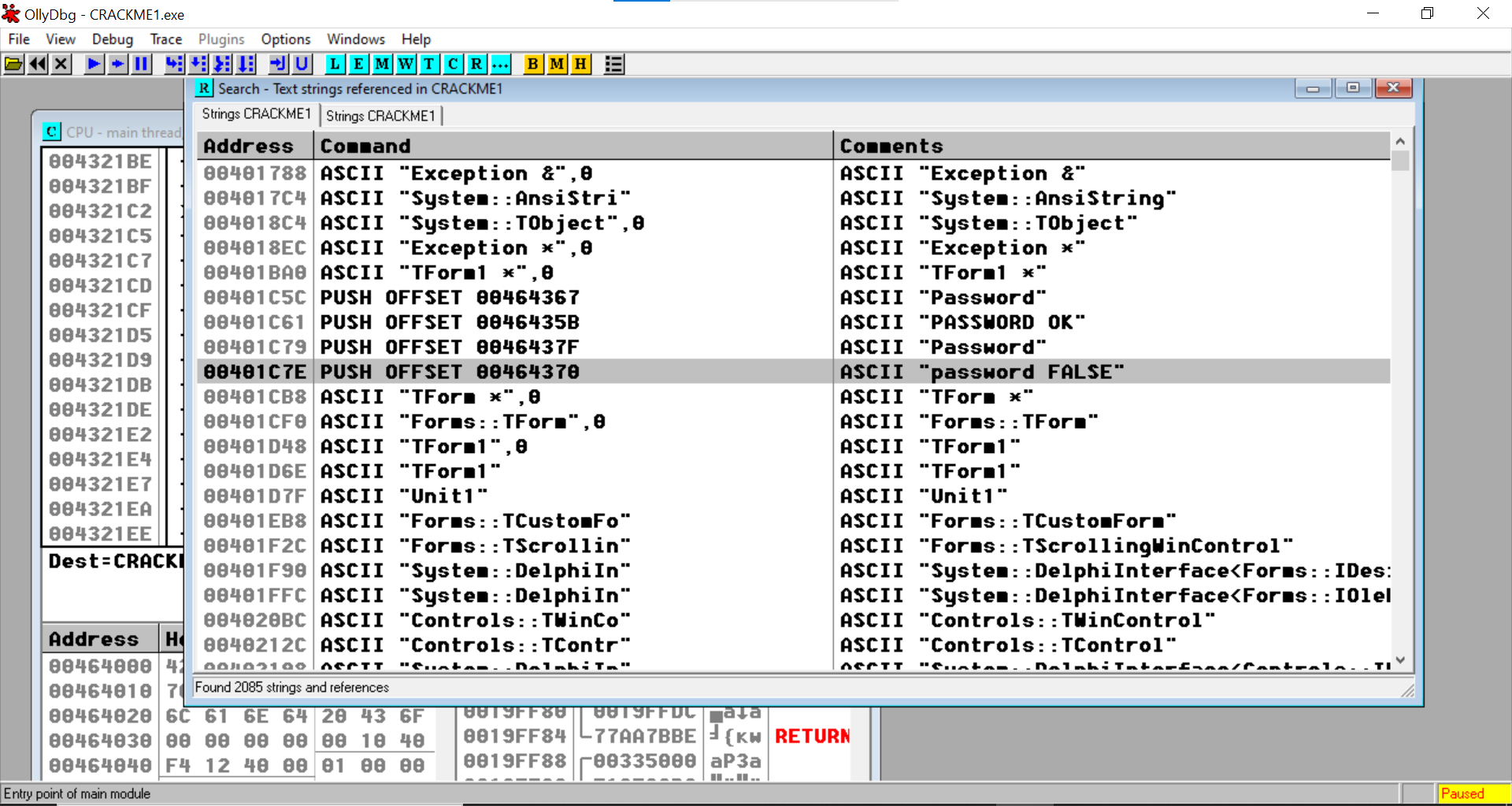


Рисунок 3 – Найденные строки

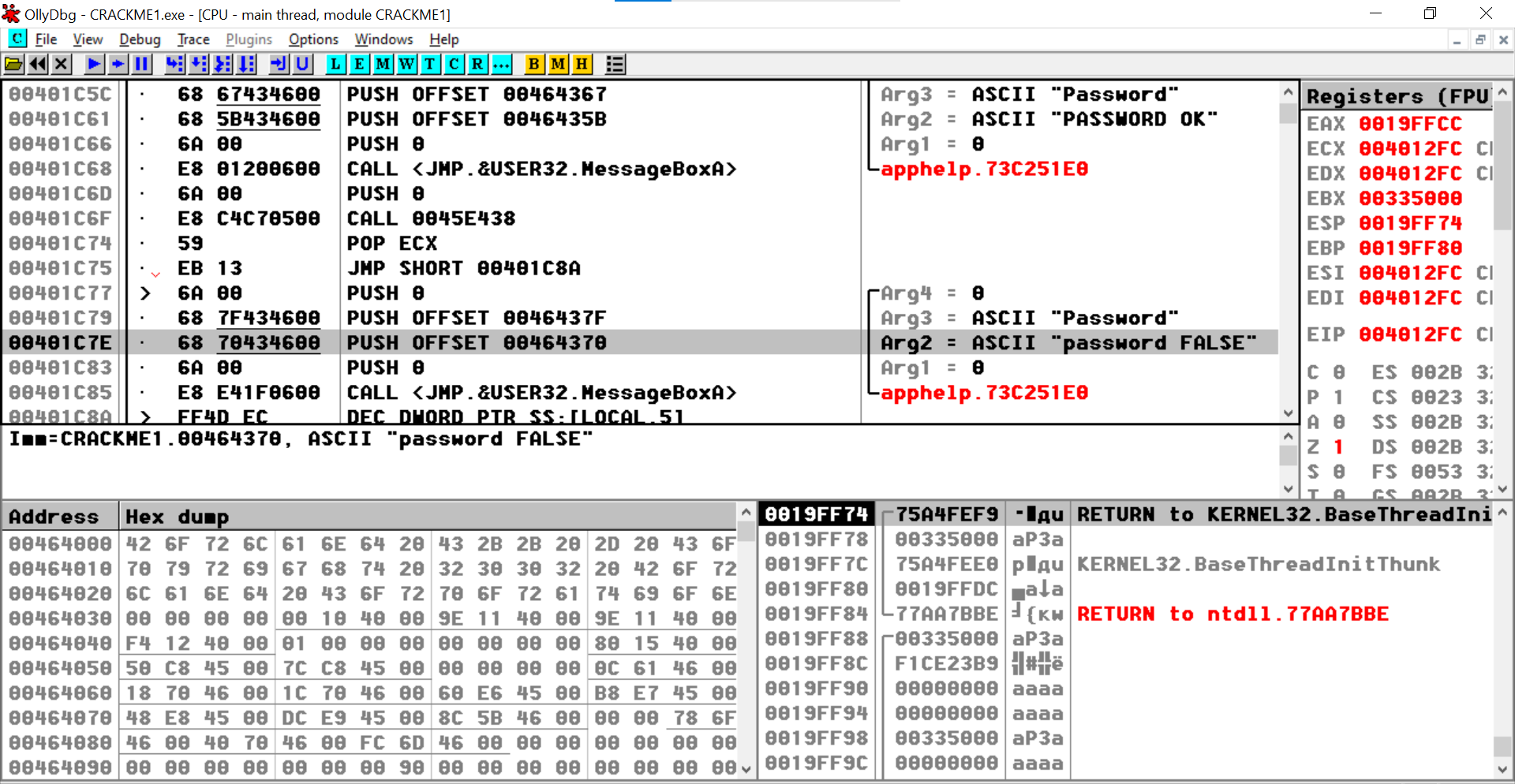


Рисунок 4 – Участки кода с проверкой пароля

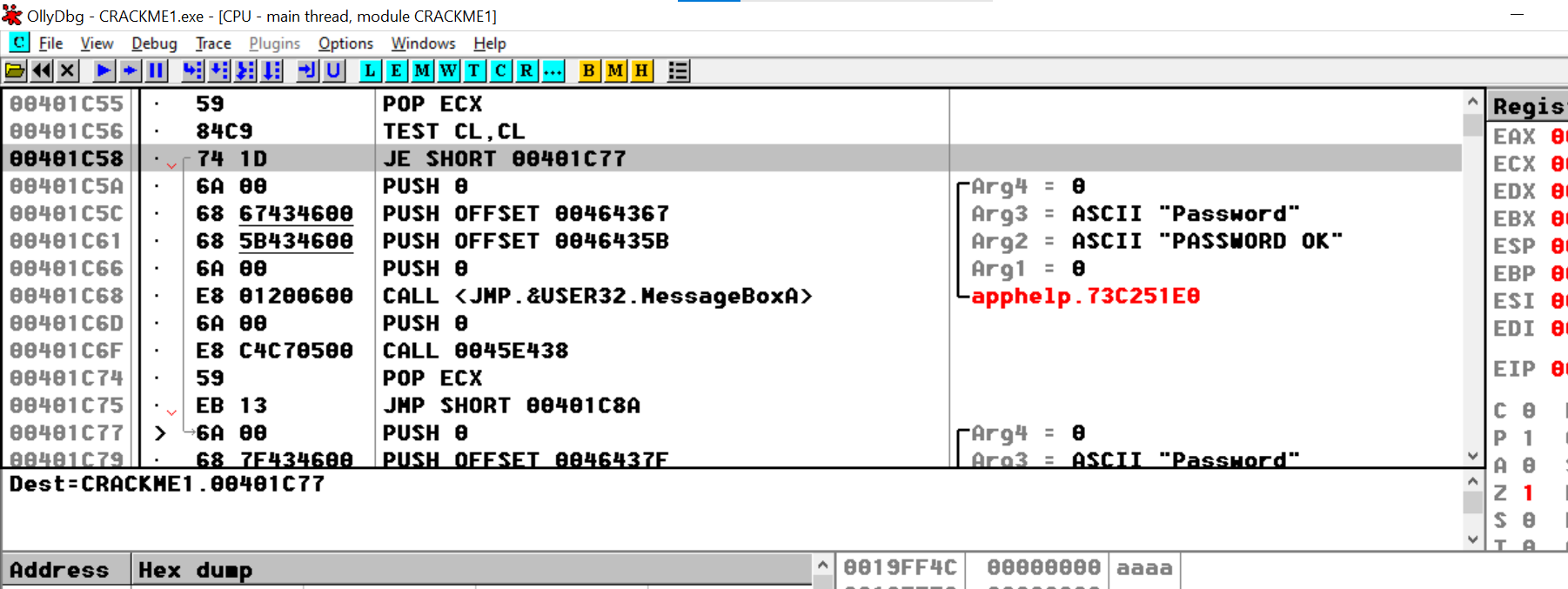


Рисунок 5 – Место перехода переход на сообщение о неверном пароле осуществляется с адреса 00401C58

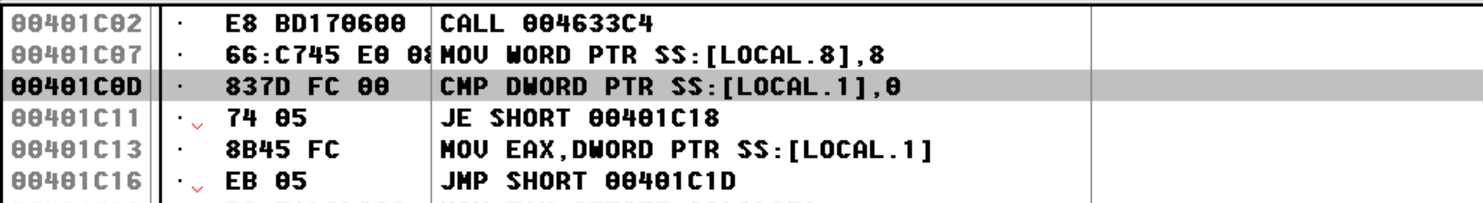


Рисунок 6 – Команда ассемблера CMP выполняет сравнение двух операндов, а после сравнения сразу стоит переход JE

Далее была проделана следующая последовательность действий:

1) Точка останова (F2) поставлена на строке с CMP.

2) Программа запущена с помощью F9, введен любой пароль, программа попала в отладчик.

3) С помощью F8 выполнена трассировка программы до тех пор, пока в регистре EAX не обнаружена строка с предполагаемым паролем.

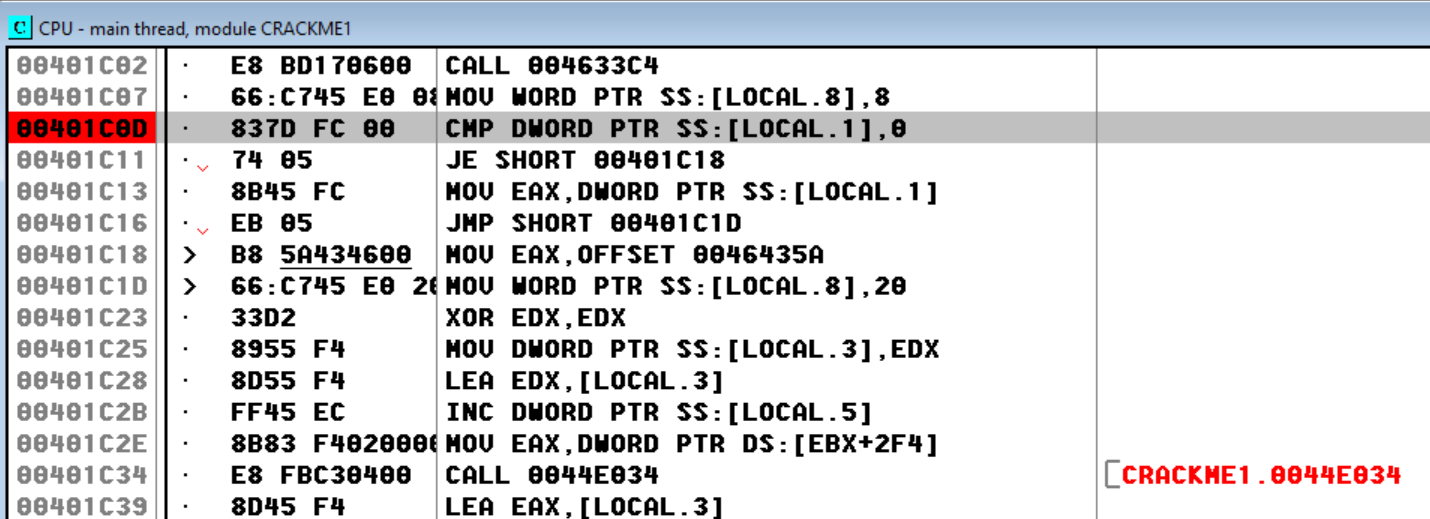


Рисунок 7 – Была добавлена точка останова

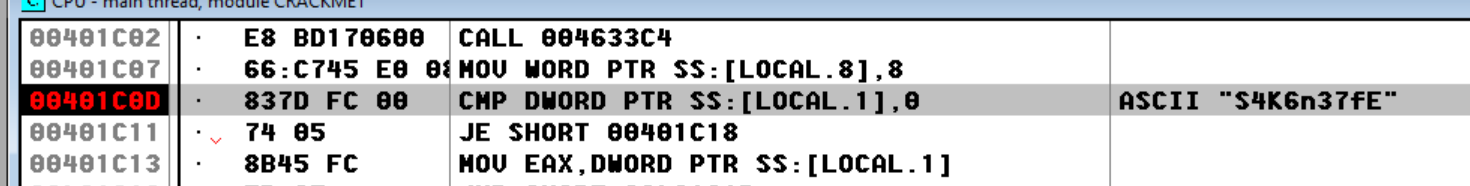


Рисунок 8 – Искомый пароль

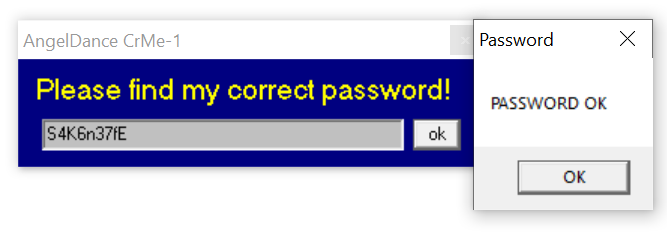


Рисунок 9 – Защита программы взломана

Далее требовалось изменить программу таким образом, чтобы она принимала любой пароль вне зависимости от его правильности.

Для этого была найдена строка с адресом 00401C58, в которой переход с помощью условного оператора JE осуществлялся на строку 00401C77, которая запускает функцию вывода сообщения о неправильном пароле. Исходя из этого, можно сделать вывод, что необходимо изменить только адрес строки, на которую оператор JE совершает переход, а именно – на адрес строки с функцией о выводе сообщения о верно введенном пароле. Адрес строки о верно введенном пароле – 00401C5A.

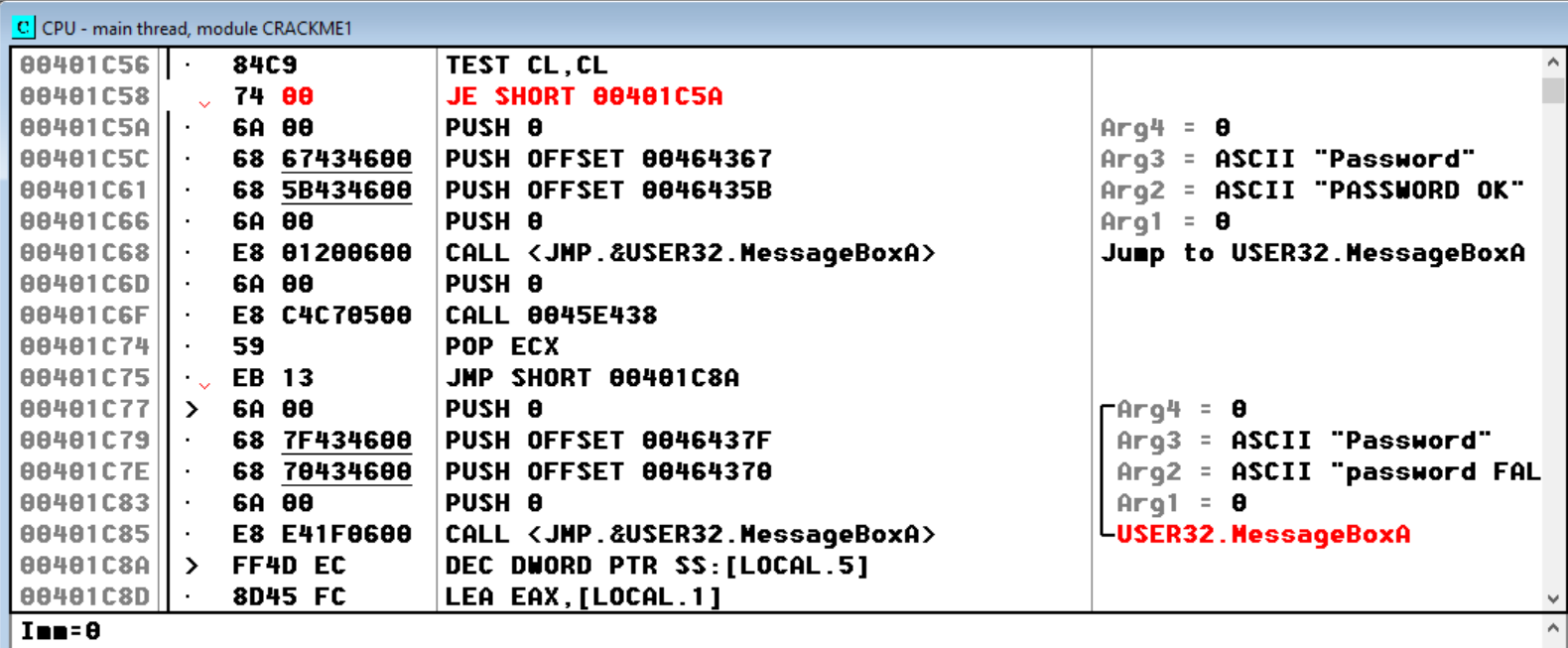


Рисунок 10 – Изменения строки перехода

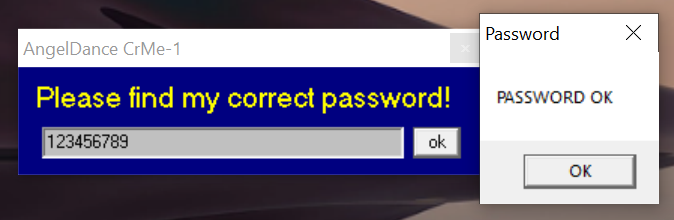


Рисунок 11 – Программа принимает любой пароль за верный

CRACKME2.EXE

Как видно, исследуемая программа написана на языке программирования C++ фирмы Borland и ничем не зашифрована и не запакована.

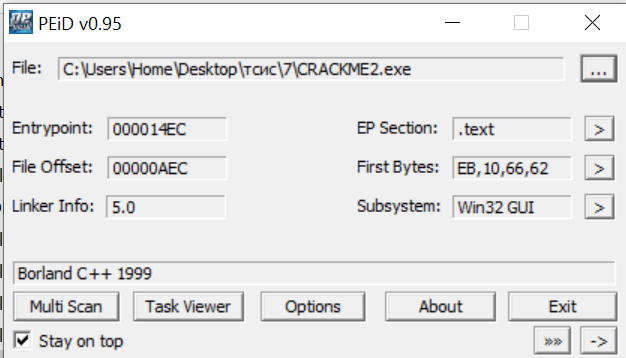


Рисунок 12 – Результат анализа приложения CRACKME2.EXE

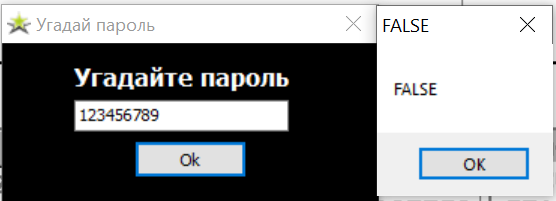


Рисунок 13 – Реакция программы на ввод пароля

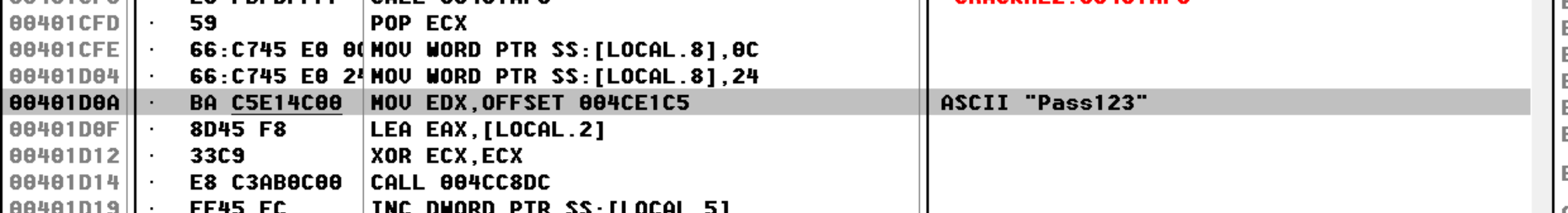


Рисунок 14 – Найдена строка предполагаемого пароля с помощью

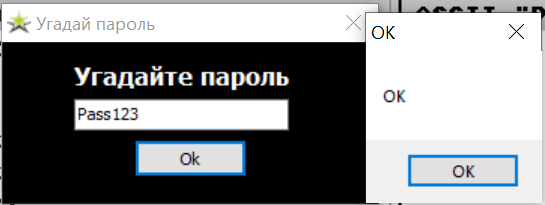


Рисунок 15 – Результат ввода обнаруженного пароля

Далее требовалось изменить программу таким образом, чтобы она принимала любой пароль вне зависимости от его правильности.

Была найдена строка, с которой начинается функция вывода сообщения о верно введенном пароле. Затем была найдена строка, в которой осуществляется переход на эту строку – это строка 00401D3F. Оператор JE обуславливает переход на сообщение о верном пароле только в случае совпадения того, что пользователь ввел, с настоящим паролем. В остальных случаях программа пропускает этот переход и идет дальше до функции вывода сообщения о неверном пароле.

Исходя из этого, можно заключить, что достаточно заменить оператор условного перехода JE на обычный безусловный JMP, чтобы программа в любом случае выдавала сообщение о верно введенном пароле.

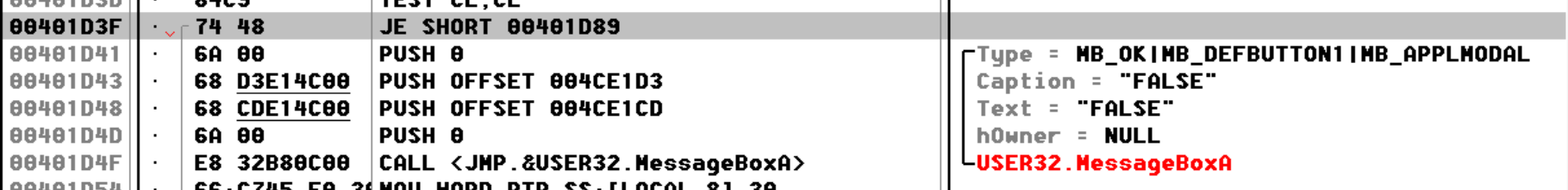


Рисунок 16 – Исходный код перехода

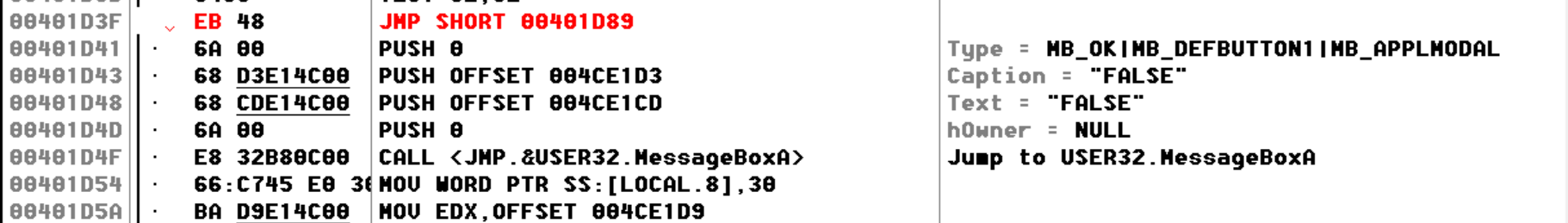


Рисунок 17 – Измененный на безусловный переход

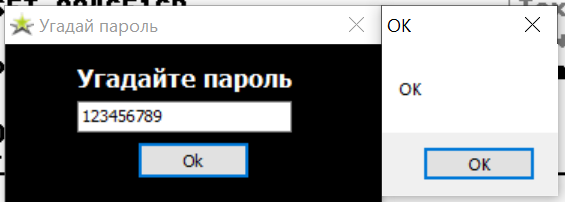


Рисунок 18 – Программа принимает любой введенный пароль за верный

CRACKME3.EXE

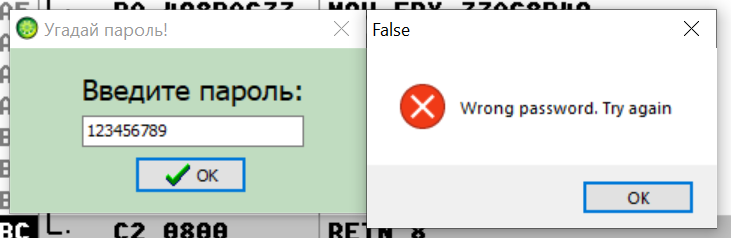


Рисунок 19 – Реакция программы на ввод случайного пароля

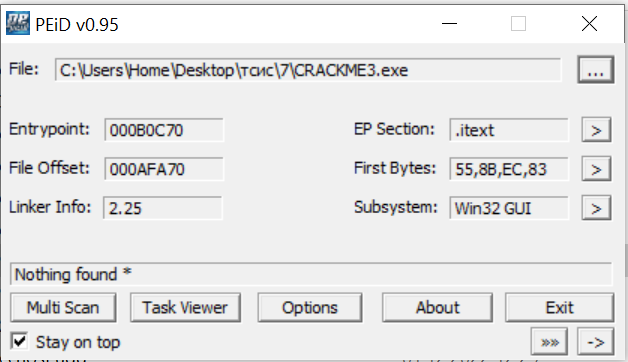


Рисунок 20 – Результат анализа приложения CRACKME3.EXE

Утилита не смогла определить язык разработки, однако далее в поиске строк была обнаружена информация о том, что программа написана на языке Delphi.

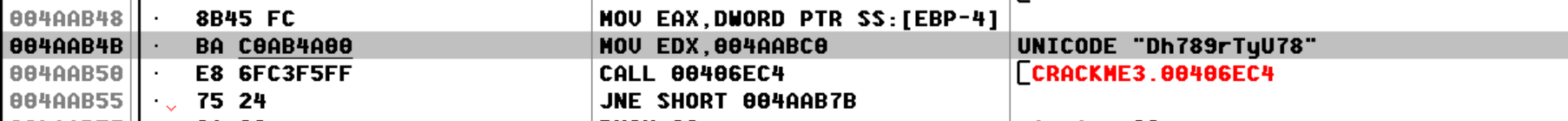


Рисунок 21 – Участок кода с сообщениями о правильности пароля

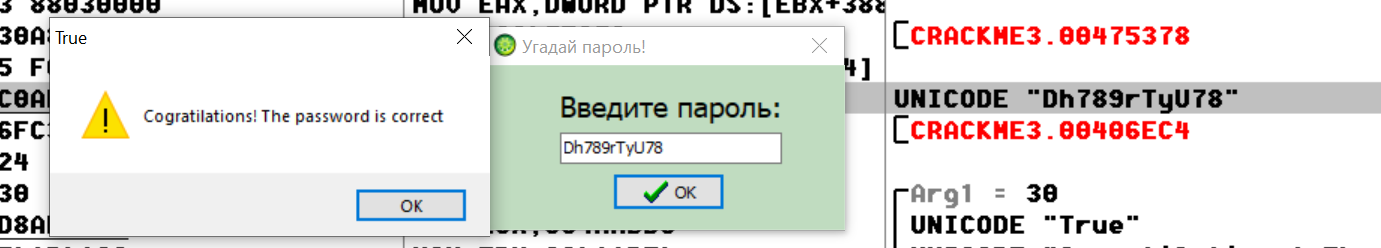


Рисунок 22 – Результат ввода предполагаемого пароля

Чтобы программа могла принять любой пароль, необходимо понять, что происходит на этапе ввода пароля и вывода соответствующего сообщения. Перед сообщением о верно введенном пароле был обнаружен условный переход JNE, который осуществляет переход программы к части кода, выводящей сообщение о неверном пароле в случае введения такового. Если пароль введен верный, программа выводит соответствующее сообщение, затем «обходит» с помощью безусловного JMP (строка 004AAB79) сообщение о неверном пароле и завершается.

Из этого следует, что достаточно изменить значение строки в операторе JNZ на такое, чтобы программа продолжала свою работу с сообщения о верном пароле.

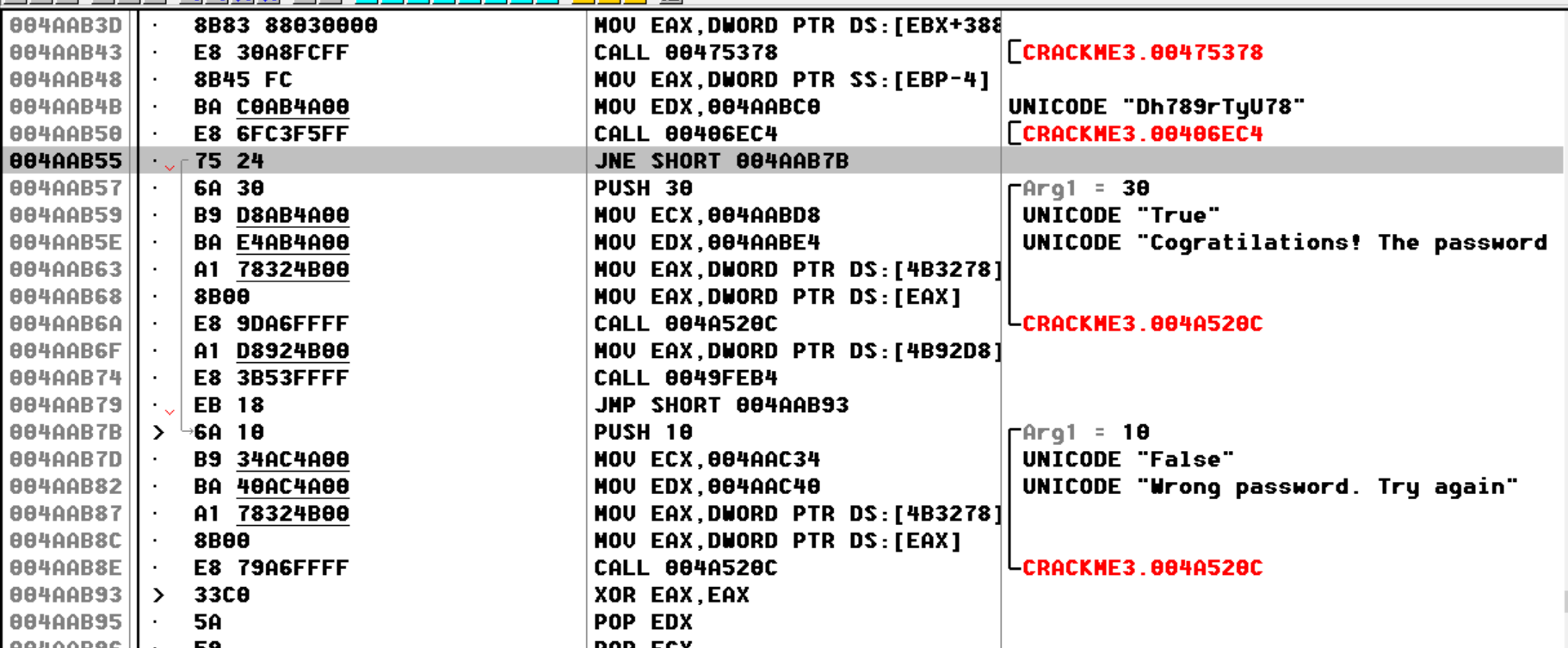


Рисунок 23 – Исходный код перехода

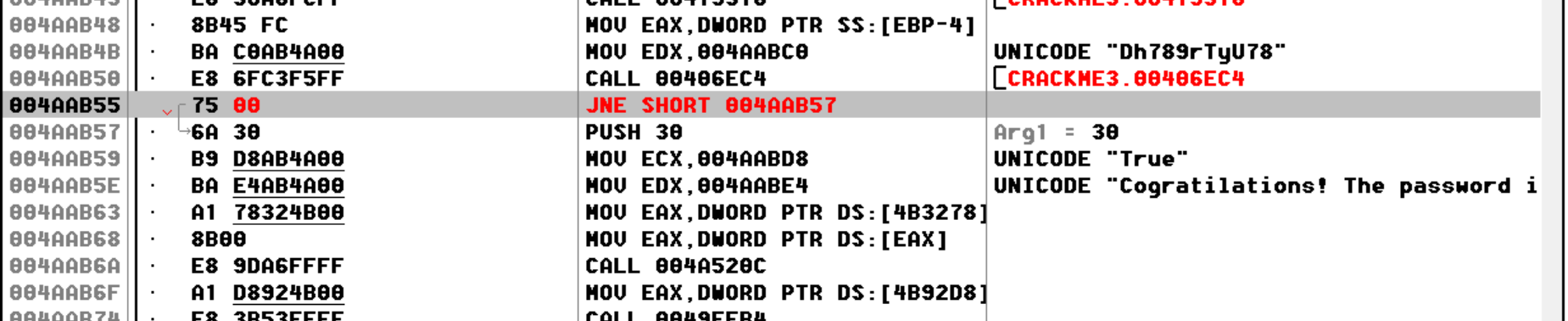


Рисунок 24 – Измененный код перехода

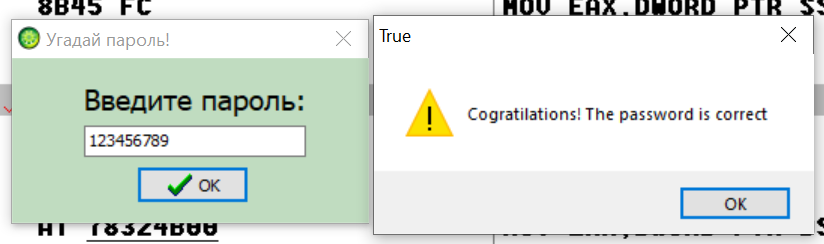


Рисунок 24 – Результат работы измененной программы

CRACKME4.EXE

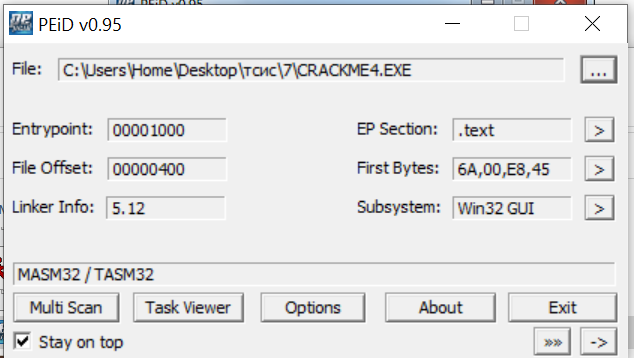


Рисунок 25 – Приложение CRACKME4.EXE в PEiD

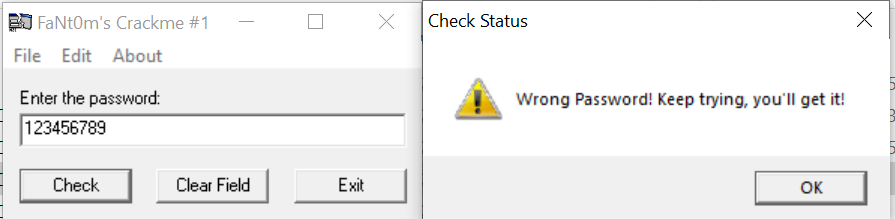


Рисунок 26 – Реакция программы на ввод случайного пароля

Перед функциями выведения сообщений была обнаружена строчка предполагаемого пароля.

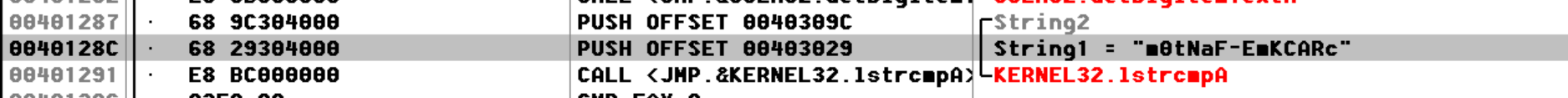


Рисунок 27 – Предполагаемый пароль

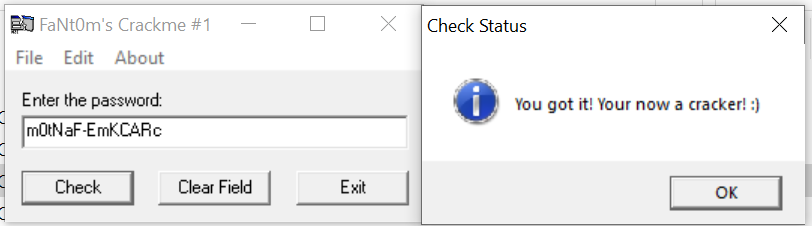


Рисунок 28 – Результат на ввод верного пароля

Чтобы программа пропускала любое введенное значение, нужно исследовать ближайшие условные и безусловные переходы. Условный переход JE (в том случае, если мы ввели правильный пароль) заставляет программу «перескочить» участок кода, в котором выводится сообщение о неправильном пароле. Поэтому достаточно заменить его на безусловный оператор JMP, чтобы программа каждый раз «перескакивала» данный участок кода, а не только тогда, когда пользователь ввел верный код.

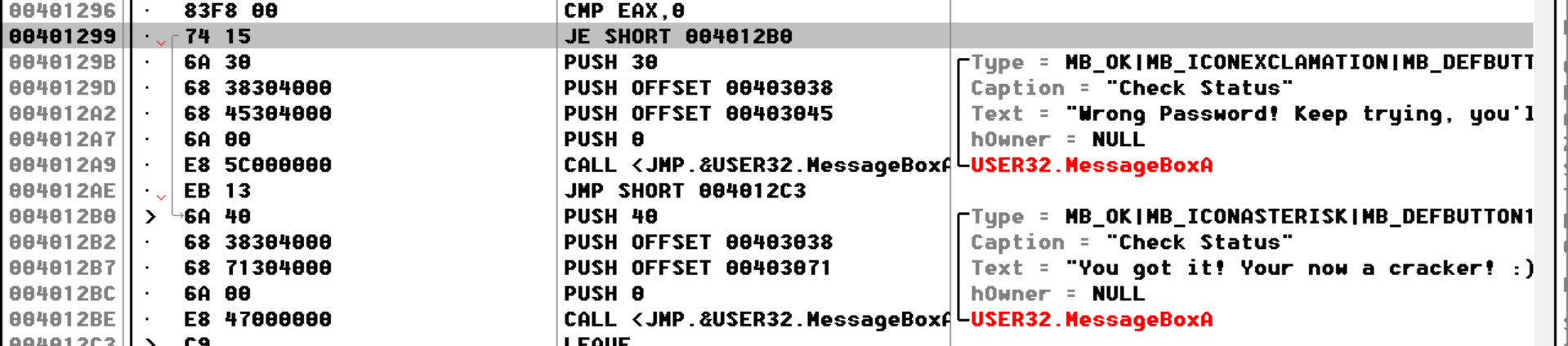


Рисунок 29 – Исходный код перехода

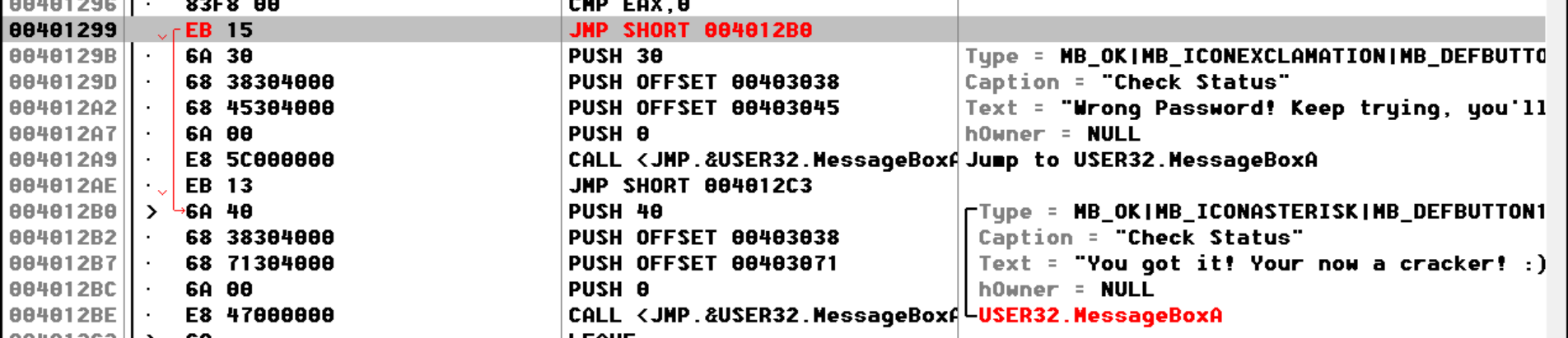


Рисунок 30 – Измененный код перехода

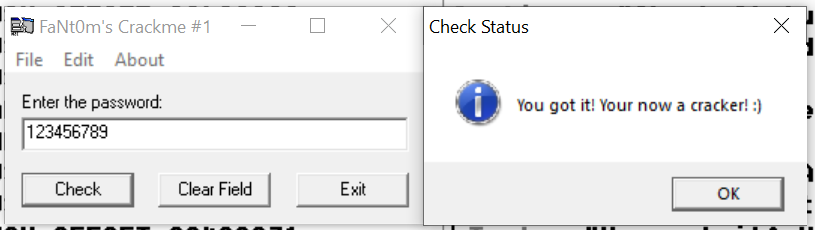


Рисунок 31 – Результат работы измененной программы

CRACKME5.EXE

Как и предыдущая программа, CRACKME5 написана на ассемблере.

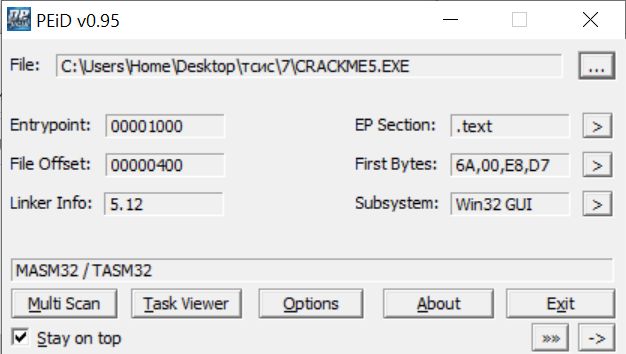


Рисунок 32 – Результат анализа программы через утилиту

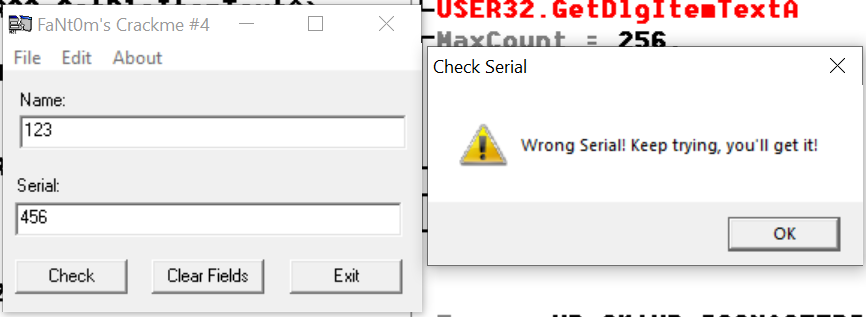


Рисунок 33 – Реакция программы на случайные name и serial

С помощью поиска строк были найдены сообщения о правильности пароля, а также предполагаемые пароли – AD и 45.

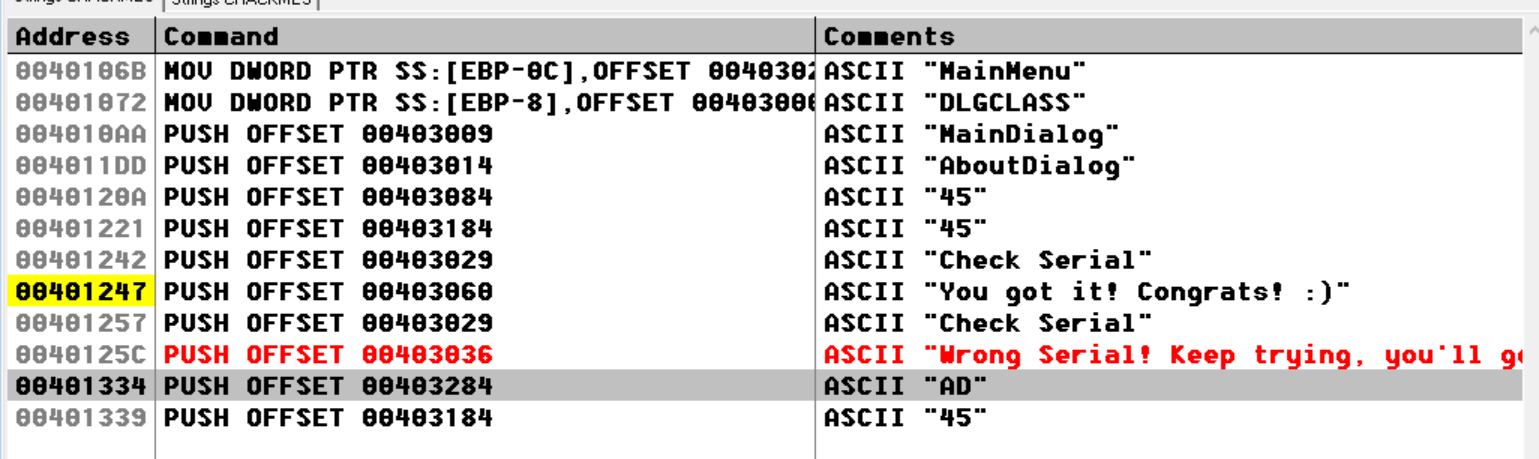


Рисунок 34 – Предполагаемые name и serial

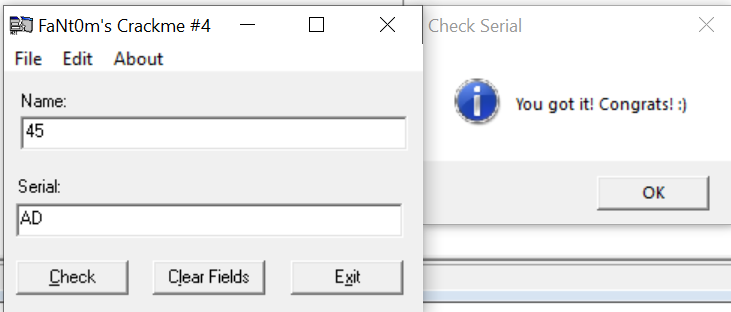


Рисунок 35 – Взлом программного пароля

Был найден оператор условного перехода JE, который в случае неправильного пароля «перекидывает» выполнение программы на функцию вывода сообщения о неверном пароле. Достаточно было просто заменить адрес строки, на которую совершает переход оператор, на адрес строки с функцией вывода сообщения о верном пароле.

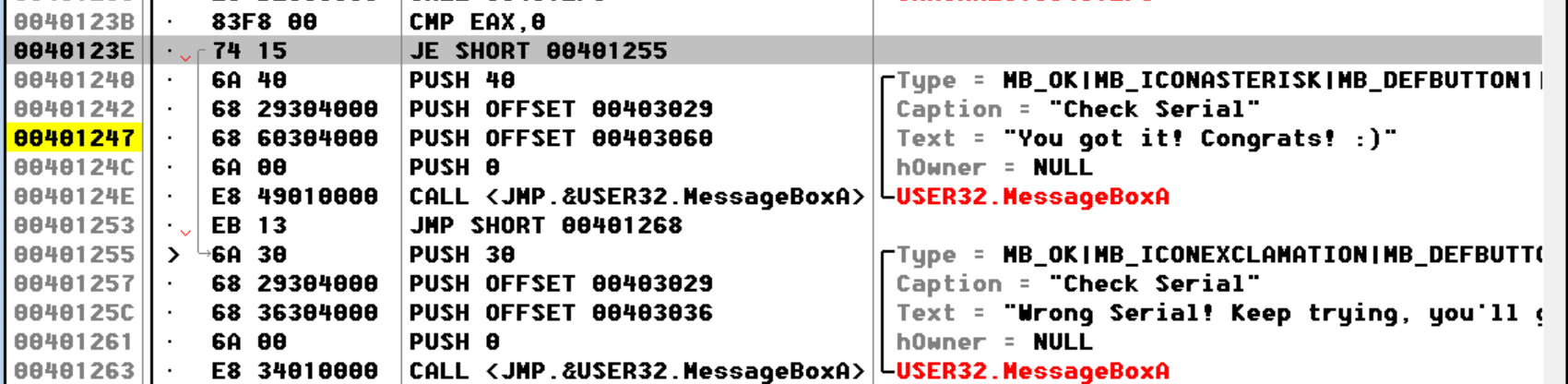


Рисунок 36 – Исходный код перехода

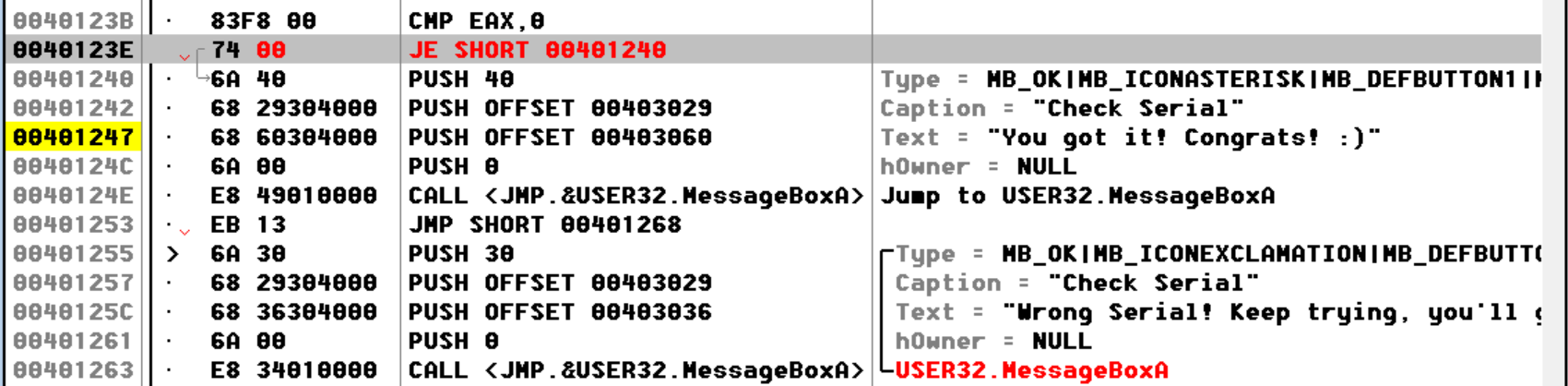


Рисунок 37 – Измененный код перехода

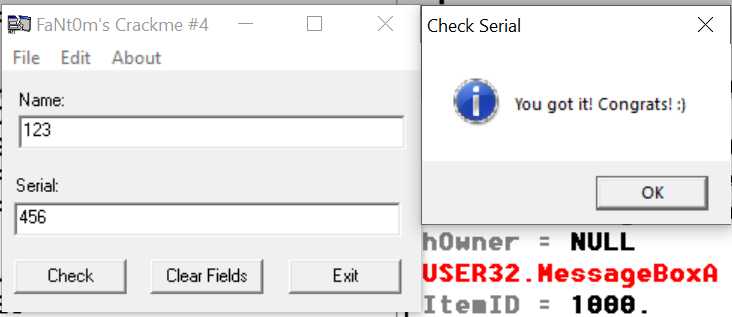


Рисунок 38 – Реакция программы на случайные name и serial

Рекомендации по усилению парольной защиты

1) Следует как можно меньше применять стандартные функции (особенно API) и библиотеки визуальных компонентов VCL (Visual Component Library), а вместо них использовать команды языка ассемблера. Это связано с тем, что современные дизассемблеры умеют распознавать стандартные процедуры высокоуровневых языков, библиотек API и VCL. Злоумышленник может ставить точки останова на вызов функций по их имени и тем самым проще обнаружить место запроса пароля.

2) Применять нестандартный способ ввода пароля. Наипростейший путь - написать свой визуальный компонент для ввода регистрационного кода. Он конечно должен будет обрабатывать события от клавиатуры, но момент считывания кода нельзя поймать избитыми методами.

3) Не хранить введенный код в одном месте. Если введенный код или регистрационный номер хранить в одном месте, то достаточно легко установить точку останова на зону памяти, в которой размещен введенный код.

4) Не хранить введенный код открытым текстом. Целесообразно завести в программе 5-10 переменных типа STRING и после ввода кода переписать введенное значение в них. Делать это лучше всего не в одном месте, а распределить по программе. Таким образом, поиск даст злоумышленнику множество адресов, по которым будет находиться введенный код.

5) Не следует анализировать введенный код идентификации (пароль) сразу после его ввода. Чем дальше ввод кода от его анализа, тем лучше. Самое разумное - после ввода кода поблагодарить пользователя за сотрудничество и сообщить, что со временем будет выполнена регистрация программы. А анализ кода произвести, например, через 1-2 минуты в совершенно другом месте программы.

6) Не проверять пароль одним алгоритмом. Рекомендуется разработать 2-3 алгоритма проверки. Проверки следует выполнять в различных местах программы с достаточно большим временным разносом. Взломав первый алгоритм, хакер может не догадываться о существовании еще нескольких, которые проявятся со временем.

7) Не проверять код только в одном месте и не создавать для проверки функцию. Злоумышленнику достаточно найти и отключить эту проверку, и защита взломана. Если проверок несколько, они разные и распределены по программе, то взлом затрудняется.

8) Ни в коем случае не предпринимать никаких действий сразу после проверки.

9) Целесообразно применять различные отвлекающие маневры. Кроме реальных функций проверки кода очень неплохо сделать пару бутафорских, которые будут вызываться после ввода кода. Желательно проводить активные манипуляции с введенным значением, выдавать сообщения о некорректности введенного кода и т.д., отвлекая внимание злоумышленника от реальной проверки.

ВЫВОДЫ:

В ходе выполнения работы были исследованы методы защиты программного обеспечения информационных систем и ее нейтрализации, приобретены практические навыки исследования и отладки программ с помощью пакета OllyDbg. Были взломаны 5 программ путем отыскания их паролей с помощью отладчика, а также изменены коды программ таким образом, что каждый введенный пароль считался верным.