Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Севастопольский государственный университет»

Исследование безопасности программного обеспечения информационных систем в среде отладчика OllyDbg

Методические указания

к выполнению лабораторной работы

для студентов, обучающихся по направлению

09.03.02 “Информационные системы и технологии”

дневной и заочной формы обучения

Севастополь 2019

Исследование безопасности программного обеспечения информационных систем в среде отладчика OllyDbg. Методические указания к лабораторным занятиям по дисциплине «Технические средства информационных систем» / Сост. А.В. Волкова, В.С. Чернега – Севастополь: Изд-во СевГУ, 2019 – 10 с.

Методические указания предназначены для проведения лабораторных работ по дисциплине «Технические средства информационных систем». Целью методических указаний является помощь студентом в изучении и исследовании методов защиты программного обеспечение с помощью отладчика OllyDbg. Излагаются теоретические и практические сведения необходимые для выполнения лабораторной работы, программа исследований, требования к содержанию отчета.

Методические указания рассмотрены и утверждены на методическом семинаре и заседании кафедры информационных систем (протокол № 1 от 30 августа 2019 г.)

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

Рецензент: Кротов К.В., канд. техн. наук, доцент кафедры ИС

Лабораторная работа

Исследование безопасности программного обеспечения информационных в среде отладчика OllyDbg

1 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.

2 Основные теоретические положения

Для обеспечения безопасности программного обеспечения, необходимо знать инструменты и методы обхода средств защиты, которыми пользуются злоумышленники. Чтобы научиться создавать защищенные программы, сначала нужно изучить способы обхода средств защиты (протекторов).

В качестве инструмента исследования безопасности программного обеспечения будет использоваться отладчик OllyDbg. В процессе отладки проводится анализ поведения программы путем исполнения ее в различных режимах (пошаговое выполнение, пошаговое выполнение с заходом в процедуру и т.п.).

2.1 Анализатор исполняемых файлов

При исследовании безопасности программного обеспечения в первую очередь нужно узнать какой упаковщик или протектор был использован. Для такой проверки используются анализаторы исполняемых файлов. Кроме информации об упаковщике или протекторе они также предоставляют много других полезных данных: каким компилятором был создан файл, размеры и названия секций, коэффициент сжатия, точку входа и дизассемблированный фрагмента кода на ней, и др.

Самый популярный анализатор исполняемых файлов – PEiD (PE iDenliller v0.95 by snaker, Qweilon & Jibz), существующий в нескольких версиях. Анализ производится по внутренней и внешней базе сигнатур, есть несколько уровней сканирования от быстрого до глубокого, имеется возможность обрабатывать целые каталоги. Возможности анализатора легко расширяется внешними плагинами, а сигнатуры хранятся в отдельном текстовом файле. Для работы с базой сигнатур написана специальная программа PEiDSO. Разработчикам плагинов в комплекте прилагается SDK с примерами на разных языках программирования и описанием API.

Одним из наиболее распространенным способом защиты программного продукта является процедура запроса пароля.

Алгоритм работы программы следующий:

1) программа запрашивает пароль у пользователя;
2) программа что-то делает с паролем (шифрует или получает хешфункцию);
3) сравнивается полученная функция с правильным паролем.

Пример такой программы показан на рисунке 2.1:

```
// Функция ввода пароля
Me_Pass=Password_is_In();
// Сравнение
if(Me_Pass == "Pas1234")
          MessageBox(0,"PASSWORD OK","Password",MB_OK);
else
          MessageBox(0,"password FALSE","Password",MB_OK);
```

Рисунок 2.1 – Пример кода программы, сравнивающей пароли

Для обхода такой защиты злоумышленнику достаточно найти место сравнения паролей и либо переписать пароль, либо изменить программу так, чтобы любой вводимый пароль считался правильным.

Рассмотрим пример вскрытия программы слабой защищенности. Запустим программу CRACKME1.EXE, расположенную в папке лабораторных работ и проследим за ее выполнением (см. рисунок 2.2).

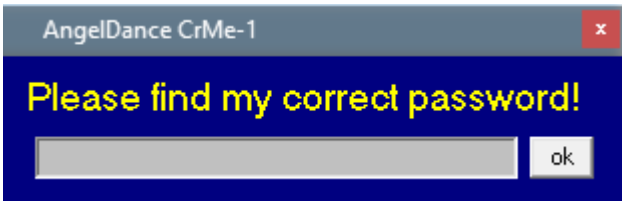

Рисунок 2 – Главное окно программы

Вводим произвольный пароль, например «123», нажимаем OK, получаем сообщение о неверном пароле (рисунок 2.3).

Рисунок 2.3 – Окно программы с сообщением о неверном пароле

Для вскрытия программы вначале следует определить:

1) зашифрована программа или нет;

2) на каком языке написана программа (эта информация понадобится для определения функций, на которые ставятся точки останова, т.к. необходимо знать, на какие собственно функции их ставить – у каждого языка программирования они свои).

Для этого воспользуемся бесплатной утилитой PEiD. Запустим утилиту PEiD, в результате чего появляется окно, показанное на рисунке 2.4.

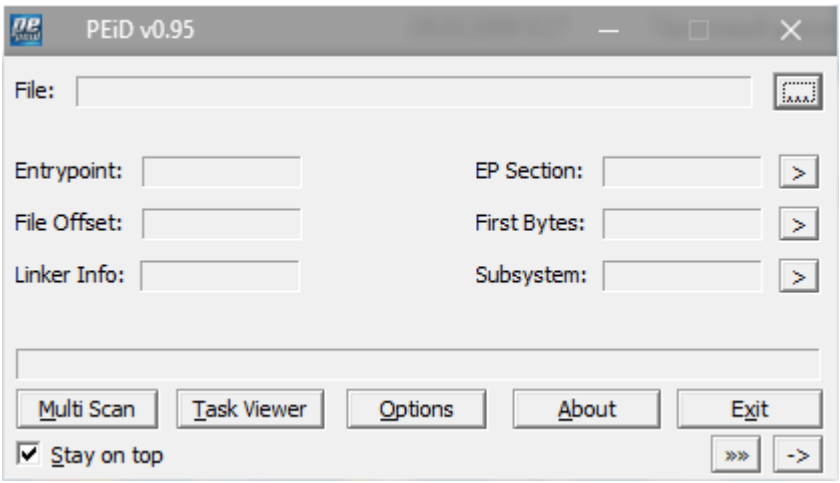

Рисунок 2.4 – Окно программы PEiD

Нажав кнопку, расположенную справа от поля File, необходимо найти и открыть файл CRACKME1.EXE (можно просто перетащить файл CRACKME1.EXE в окно PEiD с помощью мыши.) Программа PEiD проанализирует файл и выведет информацию о нем в своем окне. Как видно из рисунка 2.5, в нижней строке написано «Borland C++ 1999». Это значит, что исследуемая программа написана на языке программирования C++ фирмы Borland и ничем не зашифрована и не запакована (имеются в виду шифровальщики, или протекторы, и упаковщики сторонних разработчиков, такие как UPS, ASP Pack и т.д.).

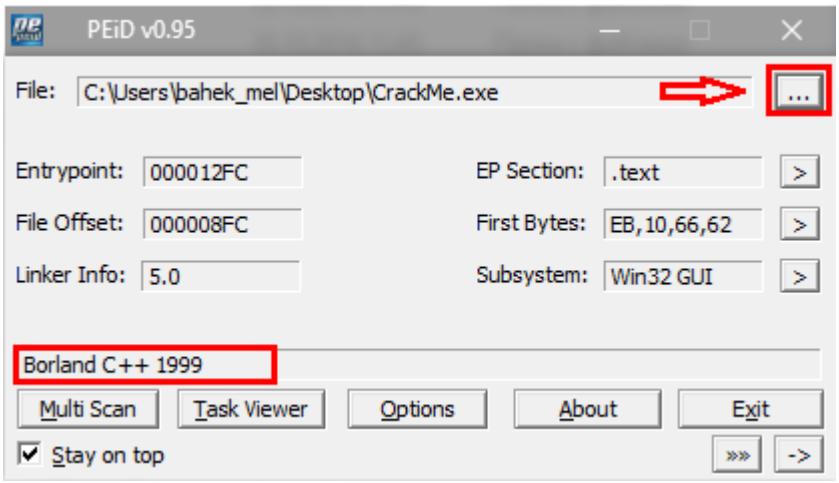

Рисунок 2.5 – Результат анализа исследуемого файла

Следует заметить, что для повышения защищенности программ существует множество способов обмана той или иной программы-идентификатора. Зная это, злоумышленник не полагается только на одну программу-идентификатор протекторов/упаковщиков. Обычно для анализа программы он использует несколько утилит.

Затем, выявив, что исследуемая программа ничем не запакована, следует запустить отладчик OllyDbg и загрузить в него программу (см. рисунок 2.6). OllyDbg остановился на самом начале файла, т.е., на точке входа в программу. К сожалению (к счастью для хакера), большинство программистов не шифруют в программе текстовые строки, выводимые на экран монитора.

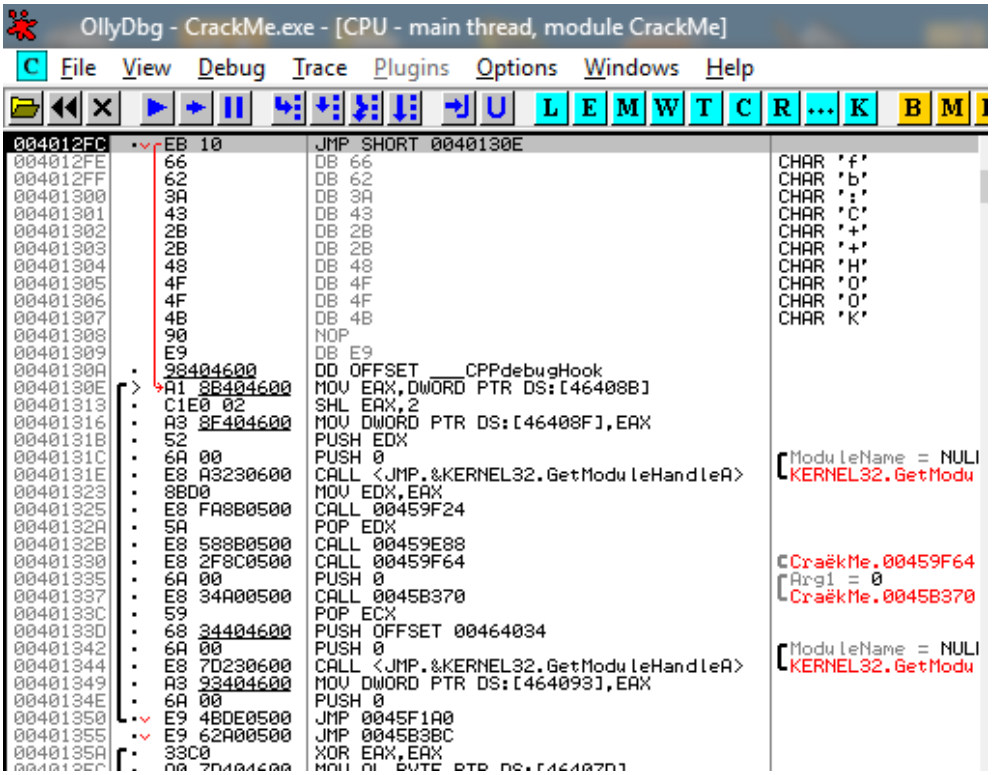

Рисунок 2.6 – Главное окно отладчика OllyDbg
после загрузки программы CRACKME1.EXE

Поэтому нужно попытаться отыскать в теле программы строки, выводимые па экран (например, «password FALSE»). Для этого, в окне кода, нажав правую кнопку мыши и выбрав в контекстном меню команд последовательность Search for → All referenced text strings необходимо выполнить поиск всех текстовых строк в файле. Отладчик выдаст все найденные текстовые строки (рисунок 2.7).

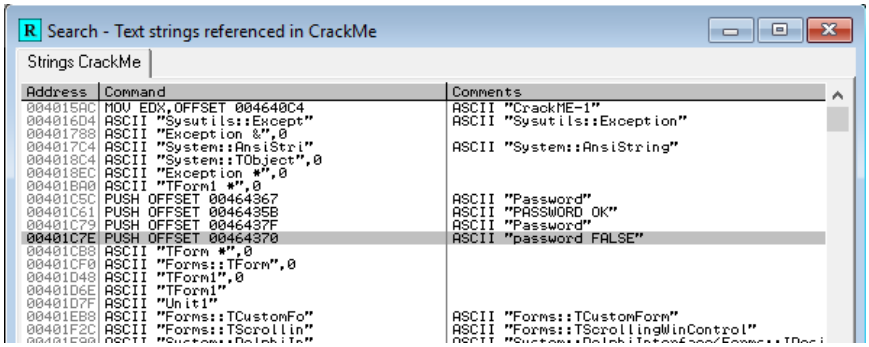

Рисунок 2.7 – Строки, которые выводятся в окне программы

В результате визуального анализа обнаруживаем, что сообщение в исходном тексте программы о неправильном пароле не зашифровано. Поэтому можно перейти на команду, которая выполняется в случае неправильного ввода пароля, путем двойного щелчка мышкой на строчке «password FALSE». На рисунке 2.8 виден вызов функции MessageBoxA. Не важно, что делается в программе после того, как выводится сообщение о неверном пароле, необходим сам пароль. Поэтому необходимо пролистать содержимое главного окна отладчика немного вверх до тех пор, пока не появится сообщение о том, что введен правильный пароль.

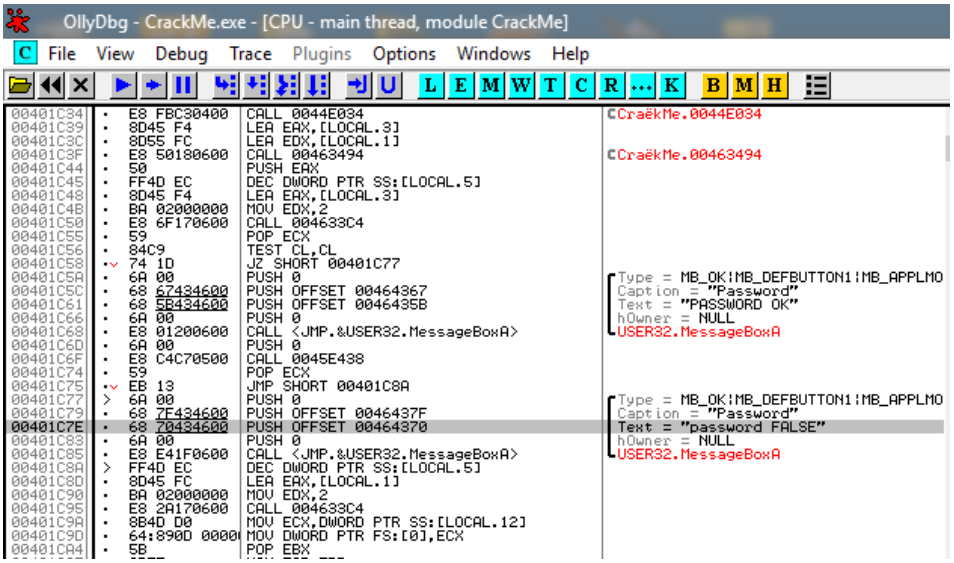

Рисунок 2.8 – Код вывода сообщения о неправильном пароле

Стрелочки, которые находятся левее кода инструкции показывают направление перехода (см. рисунок 2.9). То есть откуда или куда происходит переход после срабатывания логики в программе.

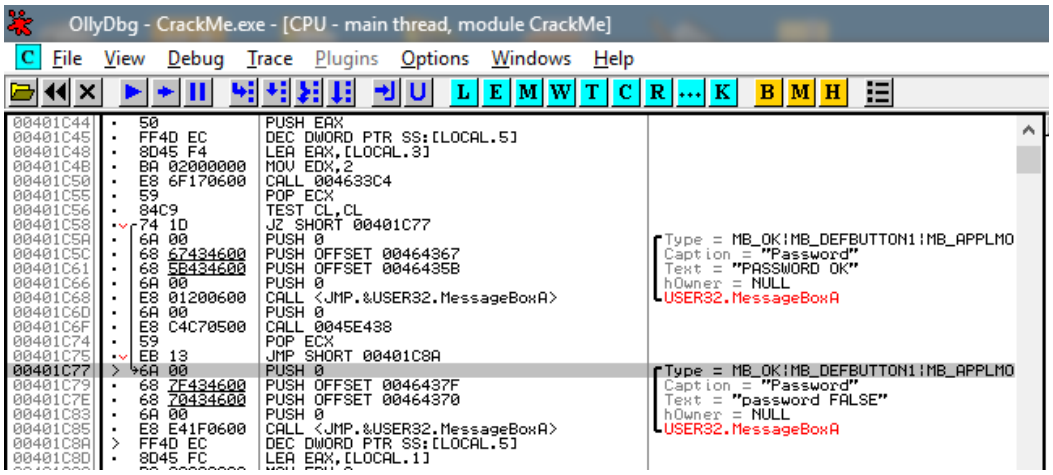

Рисунок 2.9 – Дизассемблерный код вызова функции MessageBoxA

Если нажать мышкой на эту стрелочку (или просто выделить строку рядом со стрелкой), то можно наглядно увидеть этот переход. Выделив мышкой данную строку, можно увидеть, что переход на сообщение о неверном пароле осуществляется с адреса **00401C58**. В ассемблере функция JZ осуществляет условный переход. Исходя из этого, сделаем вывод, что некая функция проверяет пароль на правильность и, если он верен, переходит на строку по адресу **00401C77,** а если не верен, то на строку по адресу **00401C5A**.

Таким образом, функция проверки правильности пароля находится выше строки:

```
00401C58   74 1D    JZ SHORT 00401C77
```

Пролистав код немного вверх, необходимо обратить внимание на функцию ассемблера CMP (рисунок 2.10), которая выполняет сравнение, а после – осуществляет переход.

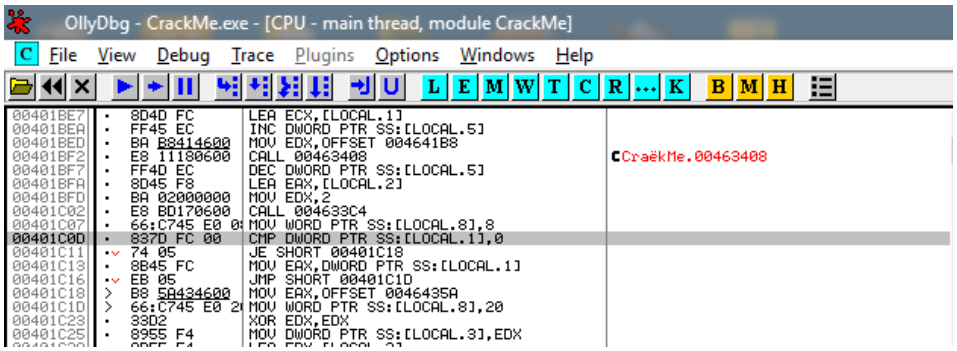

Рисунок 2.10 – Дизассемблерный код функции сравнения CMP

Команда ассемблера CMP выполняет сравнение двух операндов. А после сравнения сразу стоит переход JE (см. рисунок 2.11).

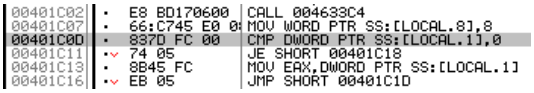

Рисунок 2.11 – Условный переход JE

Поставив точку останова (клавиша F2) на строке, содержащей CMP, необходимо запустить программу в отладчике (клавиша F9). Программа запустилась, но в отладчик не попала. Это значит, что до сих пор код, где была поставлена точка останова, не выполнялся (см. рисунок 2.12).

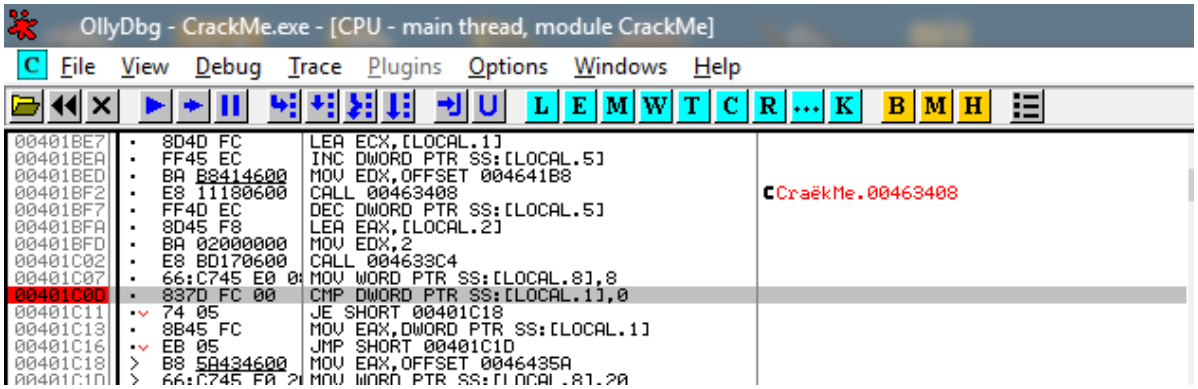

Рисунок 2.12 – Установка точки останова

В появившемся окне программы CRACKME1.EXE необходимо ввести любой пароль и нажать кнопку OK. Теперь программа попала в отладчик. Перейдя в отладчик можно увидеть символьную строку, похожую на пароль (рисунок 2.13).

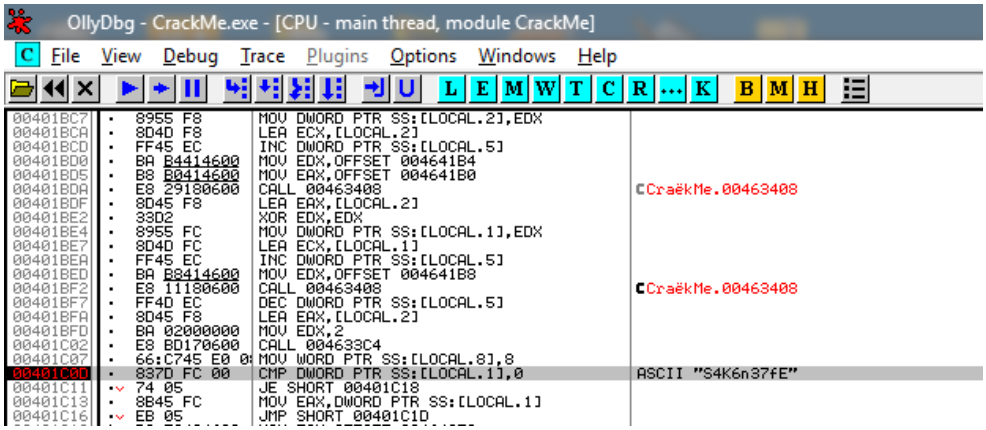

Рисунок 2.13 – Окно дизассемблерного кода, содержащего строку пароля

Дальше необходимо трассировать программу с помощью клавиши F8, до момента, изображенного на рисунке 2.14, запомнить значение в регистре EAX и потом нажать кнопку F9.

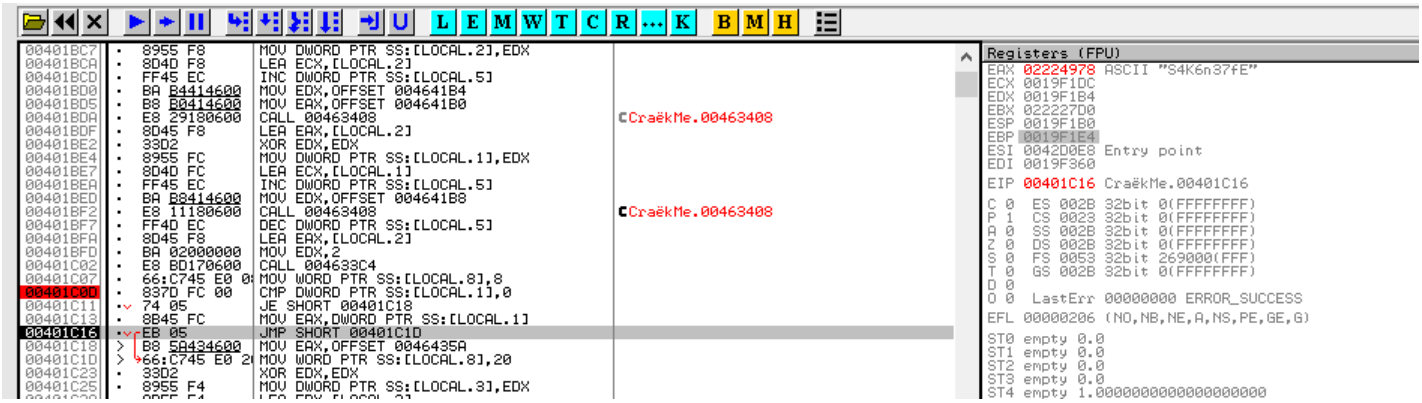

Рисунок 2.14 – Окно регистров, содержащего строку пароля в регистре EAX

В окне регистров, а точнее, в регистре EAX – искомый пароль **S4K6n37fE.** Можно попробовать ввести его в окне исследуемой программы (рисунок 2.15).

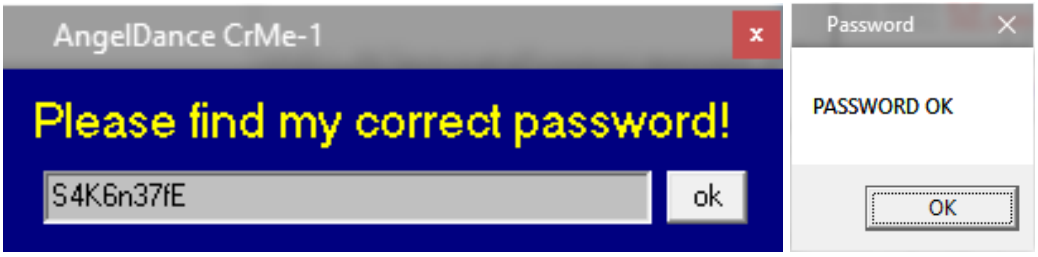

Рисунок 2.15 – Окно с сообщением о вводе правильного пароля

Таким образом, парольная защита пароля вскрыта.

Для уменьшения вероятности взлома программы ее разработчику целесообразно учитывать следующие рекомендации.

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.

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.

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.

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.

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.

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.

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.

Ни в коем случае не предпринимать никаких действий сразу после проверки. Наилучший шаг - выждать день-два (или хотя бы минут 15). Причем все действия по проверке следует как можно дальше отнести от выдачи сообщений и прочих действий, предпринимаемых при обнаружении неправильного кода.

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.

3 Программа выполнения работы

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

3.2 Исследовать способы парольной защиты в программе CRACKME1.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.

3.3 С помощью отладчика OllyDbg исследовать способы парольной защиты программ CRACKME2.EXE, CRACKME3.EXE и CRACKME4.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.

3.4 С помощью отладчика OllyDbg исследовать способ защиты программы CRACKME5.EXE. Определите на каком языке написана программа. В данной программе ключ генерируется по введенному в первом поле имени.

3.5 Разработать рекомендации по усилению защиты вскрытия пароля.

4 Содержание отчета

4.1 Цель работы.

4.2 Скриншоты исследования работы программы в отладчике OllyDbg с описанием последовательности действий и их результатов.

4.3 Рекомендации по усилению защиты вскрытия пароля.

4.4 Выводы.

5 Контрольные вопросы

5.1 Расскажите о регистрах общего назначения процессоров типа Pentium и приведите примеры их использования.

5.2 Расскажите особенностях использования индексных регистров процессора?

5.3 Для каких целей применяются регистры ST0-ST7 и XMM0-XMM7?

5.4 Что означает понятие «трассировка программы»?

5.5 Каким образом тестируется содержимое регистров общего назначения?

5.6 Какие изменения происходят в регистрах при выполнении команд PUSH и POP?

5.7 Расскажите о способах защиты от вскрытия паролей?

5.8 Какие функции используются для вывода запроса пароля?

5.9 По каким косвенным признакам можно локализовать процедуру сравнения пароля?

5.10 Продемонстрируйте на лабораторной установке процедуры установки и снятия точек останова.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Казарин О.В. Безопасность программного обеспечения компьютерных систем / О.В. Казарин. – М.: Изд-во МГУЛ, 2003. – 212 с.
2. Крейгон Х. Архитектура компьютеров и ее реализация / Х. Крейгон. – М.: Мир, 2004. – 416 с.
3. Таненбаум Э. Архитектура компьютера. 6-е изд. / Э. Таненбаум, Т. Остин . – СПб.: Питер, 2013. – 816 с.
4. Федоров Д.Ю. Основы исследование безопасности программного обеспечения [Электронный ресурс] // Санкт-Петербургский государственный инженерно-экономический университет, 2012. URL://http://pycode.ru/files/lab1.pdf (дата обращения 26.09.2016).