МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
О.Г. Локтионова
_____ 2017 г.

Изучение и отладка приложений win32 и способы изменения хода их выполнения с помощью отладчика уровня пользователя

Методические указания по выполнению лабораторной работы по дисциплине «Безопасность операционных систем и баз данных» для студентов укрупненной группы специальностей 10.00.00

Курск 2017

УДК 621.(076.1)
Составители: М.О. Таныгин.

Рецензент
Кандидат технических наук, доцент кафедры
информационной безопасности *И.В. Калуцкий*

Изучение и отладка приложений win32 и способы изменения хода их выполнения с помощью отладчика уровня пользователя: методические указания по выполнению лабораторной работы по дисциплине «Программно-аппаратные средства обеспечения информационной безопасности» / Юго-Зап. гос. ун-т; сост.: М.О. Таныгин. Курск, 2017. 21 с.: ил., ,Библиогр.: с. 21.

Излагаются методические указания по выполнению лабораторной работы на персональной ЭВМ. Изучается методы отладки и изучения приложений с помощью отладчика уровня пользователя.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальностям и направлениям подготовки «Информационная безопасность автоматизированных систем», «Информационная безопасность».

Предназначены для студентов укрупненной группы специальностей 10.00.00 дневной формы обучения.

Текст печатается в авторской редакции
Подписано в печать . Формат 60x84 1/16.
Усл. печ. л. Уч. –изд.л. Тираж 30 экз. Заказ . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

ОГЛАВЛЕНИЕ.

1. ЦЕЛЬ РАБОТЫ.

Ознакомление с простым отладчиком уровня пользователя Olly Debugger, изучение простейших методов изучения и отладки программ, а также методов изменения хода выполнения программ и реверсинга на примере «взлома» простых приложений Win32.

2. ОБЩИЕ СВЕДЕНИЯ ОБ ИССЛЕДОВАНИИ И ОТЛАДКЕ ПРОГРАММ.

2.1. Цели исследования программ.

Задача анализа программного обеспечения возникает в самых различных ситуациях. Предположим, например, что компьютерную сеть нашей организации поразил неизвестный компьютерный вирус. Для того, чтобы создать способный с ним бороться антивирус, необходимо вначале понять принципы функционирования этого вируса. А для этого необходимо провести анализ машинного кода вируса.

Другой пример. Систему защиты информации нельзя считать надежной до тех пор, пока не проведена экспертиза, которая показала, что известные методы преодоления систем защиты данного класса не работают для тестируемой системы. Другими словами, эксперт или группа экспертов становятся в позицию злоумышленника и пытаются преодолеть защиту, реализуемую тестируемой системой. Если преодолеть защиту не удается, значит, систему защиты можно рекомендовать для практического использования, в противном случае эта система нуждается в доработке. Такая экспертиза должна включать в себя опробование всех известных методов преодоления защиты, в том числе и методов, связанных с анализом программного обеспечения системы защиты. Таким образом, при проведении экспертизы системы защиты информации неизбежно возникает задача анализа программного обеспечения этой системы.

2.2. Средства исследования программ.

Основным инструментальным средством исследования программ являются отладчики, сочетающие в себе функции дизассемблера – программы, переводящей двоичные коды

исполняемого файла в ассемблерные команды, и трассировщика – программы, позволяющей организовывать пошаговое исполнение программ, создавать точки останова, просматривать содержимое регистров процессора.

Отладчиками являются либо модулями сред разработки, либо отдельными приложениями, предназначенными для поиска ошибок в приложениях. Как правило, модули сред разработки рассчитаны на отладку приложений в стадии разработки. Поиск ошибок в уже скомпилированных приложениях при отсутствии исходного кода осуществляется отладчиками уровня пользователя.

В силу того, что многозадачные операционные системы работают в защищенном режиме процессора, они, как правило, реализуют несколько колец защиты на процессоре. Windows реализует два кольца защиты – ring 0 для ядра системы, драйверов, загрузчиков и низкоуровневых системных библиотек и ring 3 для сервисов, библиотек API и пользовательских процессов. В следствие этого отладчики так же делятся на отладчики уровня ядра и уровня пользователя. Мы будем знакомиться с последними на примере бесплатного отладчика OllyDbg, сочетающего в себе мощные возможности с доступным пользовательским интерфейсом.

3. ИССЛЕДОВАНИЕ ПРОГРАММЫ С ПОМОЩЬЮ ОТЛАДЧИКА УРОВНЯ ПОЛЬЗОВАТЕЛЯ **OLLYDBG.**

3.1. Интерфейс отладчика

OllyDbg –программа с MDI-интерфейсом, которая позволяет отображать в различных окнах ту или иную информацию об отлаживаемом приложении. После загрузки приложения в отладчик появляется окно CPU (рис. 1).

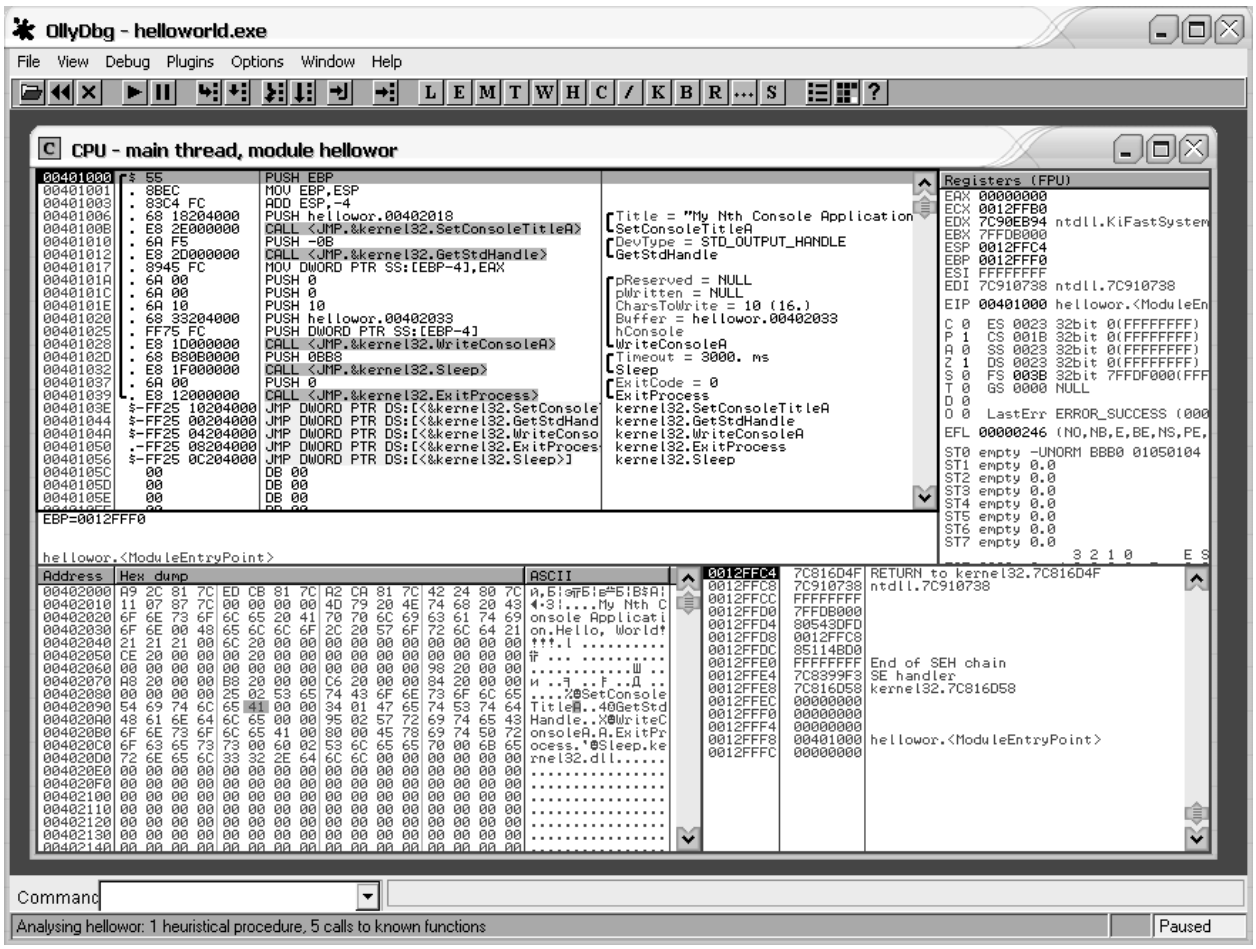

Рис. 1 – Загруженное в OllyDbg приложение helloworld.exe

Как видно из рисунка, все PE-файлы (в том числе 32-битные с расширением .dll) должны начинаться с DOS-заголовка. Обычно он служит для того, чтобы в случае запуска программы из операционной системы, не знающей о PE-формате, такой, как DOS, запускалась DOS-stub. Большинство компиляторов используют Int 21h, сервис 9, чтобы напечатать строку “This program cannot run in DOS mode”. Но поскольку DOS-заглушка – это полноценное DOS-приложение, не исключено другое её применение.

Окно CPU условно разделено на 4 части: дизассемблированную секцию кода (при загрузке регистр следующей исполняемой инструкции EIP содержит адрес точки входа приложения, этот адрес отмечен чёрным в окне дизассемблированного кода), окно регистров, окно дампа памяти и окно стека (по аналогии с окном кода, адрес, содержащийся в ESP, указывающий на вершину стека, выделен чёрным).

Ярко-голубые кнопки на панели инструментов предназначены для отображения различных окон (аналогично пунктам меню View):

- Log window
- Executable modules
- Memory
- Threads
- Windows
- Handles
- CPU
- Patches
- Call stack
- Break points
- References
- Run trace
- Source

3.2. Анализ кода исполняемого файла.

При загрузке исполняемого файла OllyDbg пытается проанализировать модуль с целью замены адресов вызовов именами функций, установления внутримодульных и межмодульных связей, выявления логических блоков и т.д. Результаты анализа появляются справа от инструкций в виде комментариев и/или логических скобок. Так же внутри инструкций вызовов адреса процедур по возможности замещаются их мнемоническими именами. Иногда результаты анализа бывают неверными (например, если при компиляции исполняемого файла был применен запаковщик). Неверные данные анализа мешают восприятию кода, поэтому в этом случае можно очистить эти данные, щёлкнув правой кнопкой мыши на любой инструкции и выбрав пункт Analysis – Remove analysis from module. Так же перед инструкциями могут встречаться символы $ или >, означающие, что из какого-либо места в программе произошёл вызов ($) или (без)условный переход (>) по текущему адресу. Перейти на место вызова/перехода можно с помощью пункта контекстного меню Go to – CALL from (address) / JUMP from (address).

3.3. Пошаговое выполнение, трассировка и точки останова.

Чаще всего поиск ошибок в программах осуществляется с помощью ручной трассировки с отслеживанием тех или иных изменений в регистрах, в стеке (в частности, в локальных переменных) или в памяти области данных (в частности, в

глобальных переменных). OllyDbg позволяет выполнять программу пошагово, до точки останова, до возврата из функции, либо осуществлять автоматическую трассировку с указанными параметрами.

Рассмотрим эти возможности на примере трассировки программы helloworld.exe.

- Откройте OllyDbg.exe.
- Нажмите F3 и найдите исполняемый файл helloworld.exe.
- Как можно увидеть, работа программы начинается с выполнения инструкции PUSH EBP по адресу 0x00401000 (адрес виртуальный).
- Нажмите F7 (Step Into, аналогично соответствующему пункту из меню Debug).
- При нажатии Step Into отладчик выполняет ровно одну инструкцию процессора, заходя во все встречающиеся циклы и вызовы. Обратите внимание на то, что EIP стал равен 0x00401001, а также на изменившееся окно отображения содержимого стека – его верхний элемент теперь содержит значение EBP, а указатель вершины уменьшился на 4.
- Нажмите F7 еще два раза.
- Регистр EBP в результате выполнения двух инструкций стал равен ESP-4. Этим программа выделяет так называемый буфер для локальных переменных в стеке. В данной программе он равен 4. Обращение к локальным переменным в дальнейшем осуществляется относительно EBP. OllyDbg позволяет показывать как абсолютные адреса в окне стека, так и относительные (от EBP и от ESP). Для этого из контекстного меню стека следует выбрать Address – Relative to EBP/ESP/Selection. Выбрав отображение адресов относительно EBP, можно увидеть, что регистры различаются на 4 байта (на двойное слово).
- Нажмите F7 еще раз.
- Следующая инструкция PUSH hellowor.0402018 помещает соответствующий адрес в стек (как показывает анализ, этот адрес служит аргументом для последующего вызова процедуры). Для того, чтобы посмотреть, что находится по этому адресу, нажмите правой кнопкой мыши на этой инструкции и выберите пункт Follow in dump – immediate constant из появившегося контекстного меню. В окне дампа по этому адресу начинается строчка “My Nth Console Application”. Следующая инструкция вызовет API-функцию

SetConsoleTitleA. Аргументы функциям по соглашению *stdcall*, которого придерживаются многие компиляторы, передаются через стек от последнего к первому. По этому же соглашению, возвращаемое значение содержится в регистре EAX.

- Нажмите F7 еще два раза.
- Мы попадаем в модуль kernel32.dll, где выполняется функция SetConsoleTitleA. Как видно, используя только пошаговое исполнение, трудно добиться нужного результата, потому что не всегда нужно знать, что происходит при исполнении кода системных функций.
- Чтобы вернуть управление к модулю helloworld.exe, нажмите Ctrl+F9 (Execute till return) и F7 еще раз.
- Следующие две инструкции поместят очередной аргумент в стек и вызовут функцию API GetStdHandle.
- Нажмите F8 (Step over) два раза.

Step over так же, как и Step into, пошагово выполняет инструкцию за инструкцией, но без вхождений в тела циклов или вызовов. После второго нажатия F8 значение регистра EAX становится равно 7. Следующая инструкция MOV DWORD PTR SS:[EBP-4], EAX пересылает содержимое этого регистра в отведенное под локальные переменные место в стеке.

Теперь поставим точку останова по адресу 0x0040102D (push 0bb8).

- Для этого выделите мышью или стрелками на клавиатуре инструкцию по адресу 0x0040102D и нажмите F2 (Toggle breakpoint). Повторное нажатие F2 уберет точку останова.
- Адрес напротив выбранной инструкции стал красным.
- Теперь нажмите F9 (Run).

Программа будет исполняться, пока значение адреса в EIP не будет равно 0x0040102D. После этого выполнение программы останавливается и управление вновь переходит к отладчику. Результат выполненного кода можно увидеть, переключившись на окно программы с помощью сочетания клавиш Alt-Tab (или любым другим способом):

Рис. 2 – Результат пошагового выполнения программы.

- Нажмите F9 еще раз, и через несколько секунд (из-за выполнения API Sleep) выполнение программы будет завершено.
- Нажмите Ctrl+F2 для перезапуска программы и уберите точку останова, выделив инструкцию и нажав F2.

3.4. Изменение хода выполнения программы.

Отладчики позволяют в процессе выполнения программы изменять значения регистров процессора, а также памяти области данных, стека или же самого кода программы. Этим чаще всего пользуются взломщики для обхода защиты программ от несанкционированного использования. В частности, ими применяются реверсинг (восстановление исходного кода программы) для написания генераторов ключей (т.н. кейгенов) или ассемблирование на ходу (изменение части кода программы во время её выполнения с помощью отладчика) для написания программ-взломщиков (т.н. крэков, в т.ч. офсетных крэков – текстовых документов, описывающих последовательность действий для самостоятельного взлома программы заменой, как правило, нескольких байтов файла с исполняемым модулем, используя простой шестнадцатеричный редактор).

В данной лабораторной работе рассмотрен простейший случай обхода защиты программы с помощью ключа и серийного номера. Следует отметить, что в настоящих коммерческих приложениях редко используется какой-либо один метод защиты, что уменьшает вероятность взлома программ. В данной работе эти методы защиты подробно рассматриваться не будут.

Важно понимать, что использование отладчика по отношению к программам не всегда связано с незаконной деятельностью. В частности, многие фирмы легально используют отладочные средства для локализации программных продуктов.

Итак, исследуемая программа называется CrackMeEx.exe, она написана на C++ с использованием Win32 API (рис. 3).

Рис. 3 – Главное окно исследуемой программы.

Главное окно программы содержит два текстовых поля и кнопку. В одном из полей записан т.н. ID Key, идентификационный ключ, генерируемый программой при запуске. Во второе поле ввода предлагается ввести серийный номер продукта, который, очевидно, связан с идентификационным ключом. При нажатии кнопки Register введенный серийный номер проверяется программой и в случае несовпадения с правильным выводится сообщение об ошибке (рис. 4).

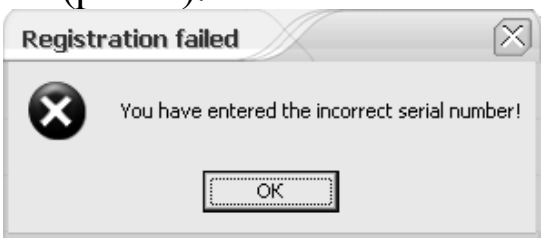

Рис. 4 – Сообщение о неверно введенном серийном номере.

Нашей задачей является получение правильного серийного номера, а затем – изменение программы таким образом, чтобы для регистрации подходил любой введенный серийный номер.

3.4.1. Получение «правильного» серийного номера.

- Откройте OllyDbg и загрузите исполняемый модуль CrackMeEx.exe в отладчик.

Трассировка программы Windows вручную займёт слишком много времени, потому как от точки входа до необходимого нам цикла в оконной процедуре процессор может исполнить не одну тысячу инструкций. Поэтому необходимо найти место, где поставить точку останова. Целесообразнее всего начать поиск с сообщения о вводе неверного серийного номера.

• Нажмите правой кнопкой мыши в окне дизассемблера и выберите пункт Search for – All referenced text strings.

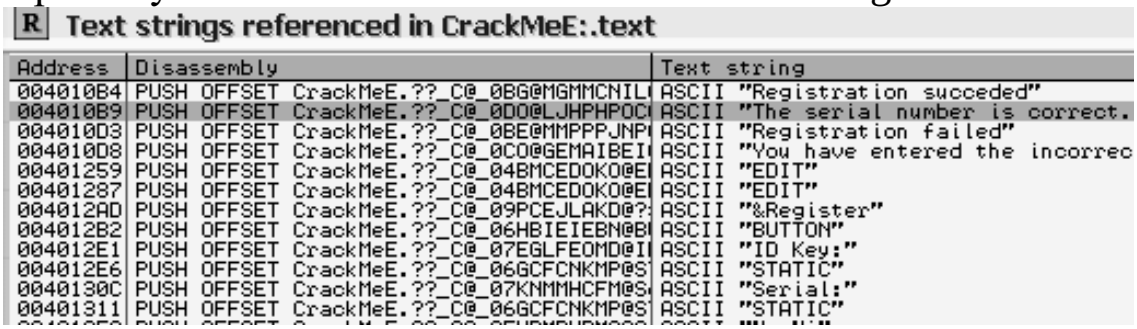

Рис. 5 – Ссылки на текстовые строки в модуле.

Вверху списка мы видим уже знакомое нам сообщение «You have entered the incorrect serial number». Мы также видим, что программа обращалась к адресу, в котором содержится эта строка, с помощью инструкции PUSH по адресу 0x004010B9, что позволяет предположить, что в этом месте эта строка была помещена в стек как аргумент для вызова функции API MessageBox, отображающей окно с сообщением.

• Выберите это сообщение и нажмите Enter (Follow in disassembler) (рис. 6).

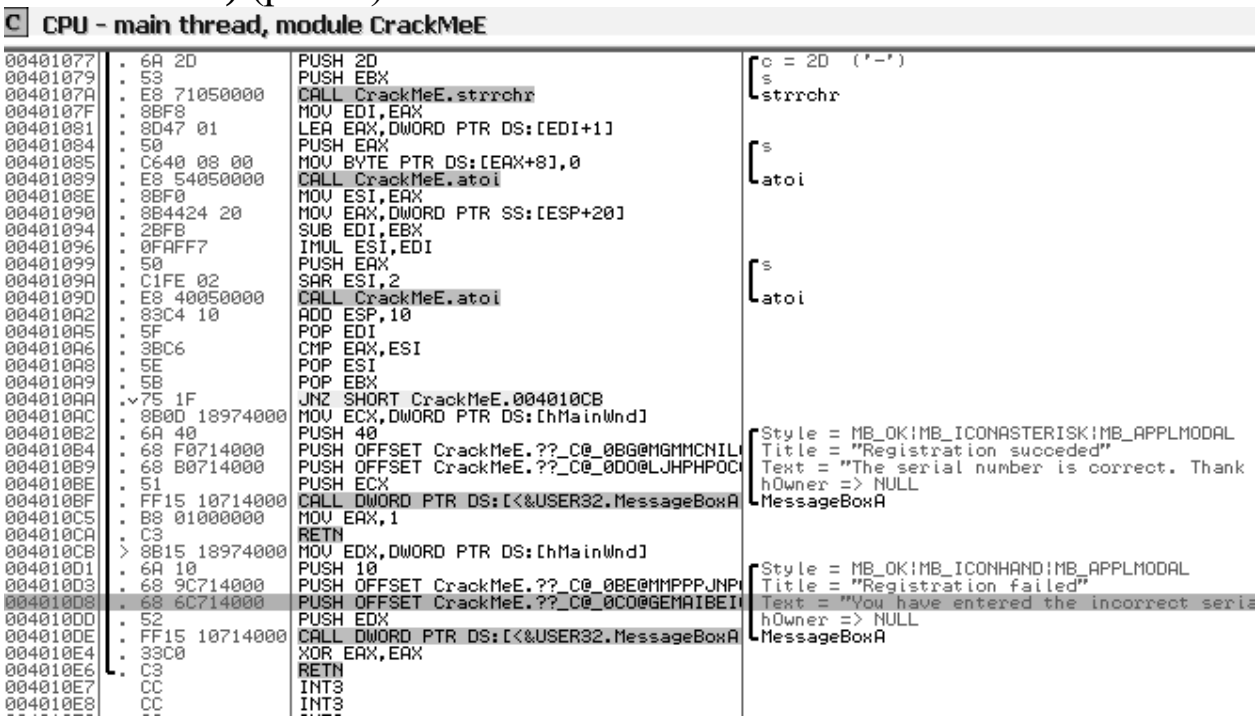

Рис. 6 – Место помещения в стек сообщения о неверном серийном номере.

Окно дизассемблера теперь показывает на место в программе, где выводится сообщение о неверном серийном номере.

Внимательно изучите этот фрагмент кода. Немного выше происходит вывод сообщения о правильном серийном номере («The serial number is correct. Thank you for the registration!»). Видно, что после вызова функции MessageBox в первом случае регистр EAX становится равным единице, а во втором случае – 0 (с помощью инструкции XOR EAX, EAX). Это позволяет предположить, что проверка серийного номера происходит в отдельной функции (подпрограмме), которая возвращает 1, если серийный номер введен верно, и 0 в противном случае (в результате анализа OllyDbg выделяет эту процедуру большой черной скобкой слева от инструкций в окне дизассемблера).

- Прокрутите текст программы вверх до начала подпрограммы, проверяющей серийный номер. Выделите первую инструкцию функции (адрес 0x00401070) и нажмите F2 для помещения точки останова по указанному адресу (рис. 7).

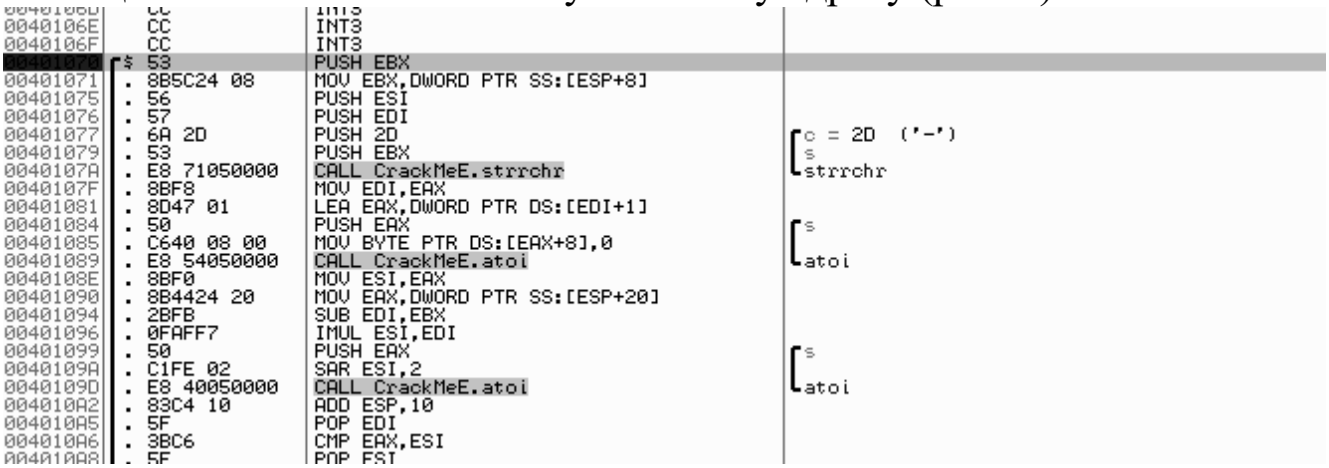

Рис. 7 – Точка останова в начале процедуры проверки серийного номера.

- Нажмите F9 (Run), введите число 12345 в поле Serial и нажмите кнопку Register.

Выполнение программы было прервано на указанном адресе. Обратите внимание на область стека.

Рис. 8 – Стек в момент исполнения подпрограммы проверки серийного номера.

Мы видим, как введенный нами серийный номер и сгенерированный программой идентификационный ключ были последовательно помещены в стек.

Ниже по тексту программы мы видим вызовы стандартных функций C strrchr (производит поиск указанного символа в строке, начиная с правого её конца, в данном случае – символа “-“) и atoi (конвертирует строку в число). Очевидно, что для генерации серийного номера используются числа, находящиеся после символа “-“.

- Нажимайте F8 (Step over) до первого вызова atoi.

Мы видим, что в число переводятся первые 8 цифр, следующие за символом “-“ идентификационного ключа (эти цифры хранятся по адресу, который содержит регистр EAX, нажмите правой кнопкой на регистре EAX и выберите пункт Follow in dump контекстного меню, чтобы увидеть в окне дампа эти цифры).

- Нажмите F8 еще один раз.

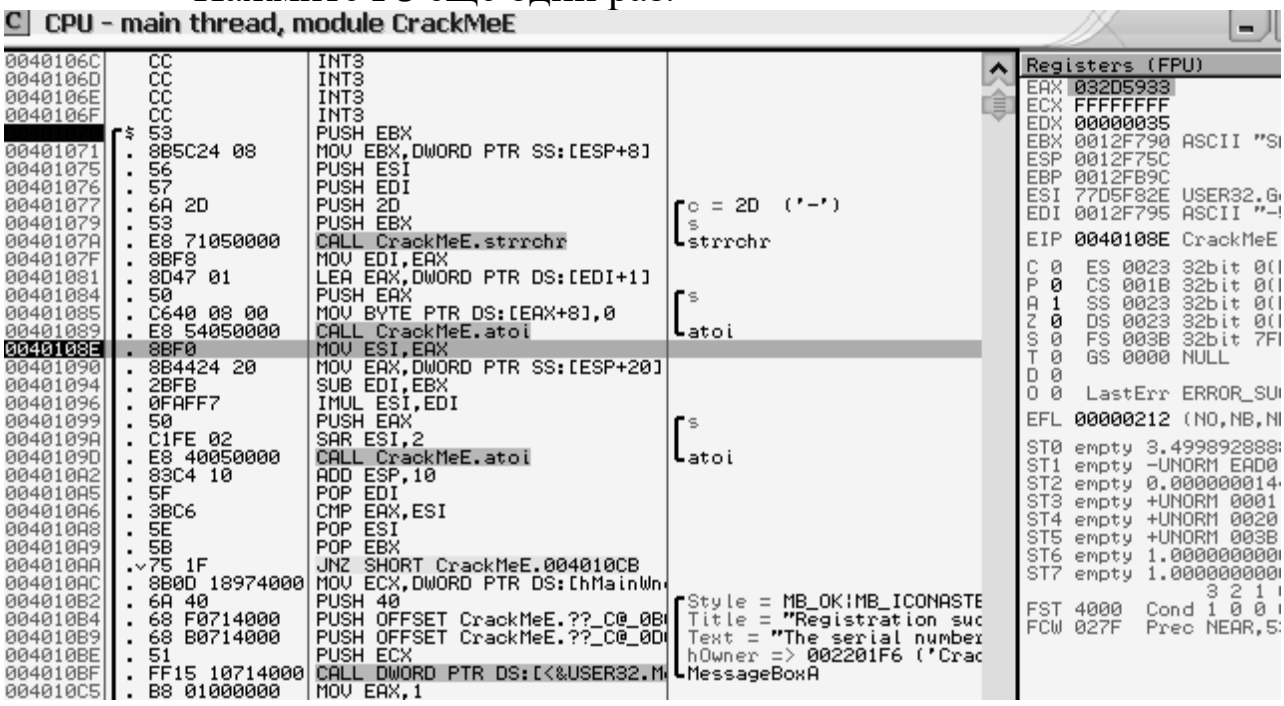

Рис. 9 – Результат выполнения atoi в первый раз.

Мы видим, что значение регистра EAX стало равно 0x032D5933 (рис. 9).

- Выделите этот регистр и нажмите Enter (Modify) (рис. 10).

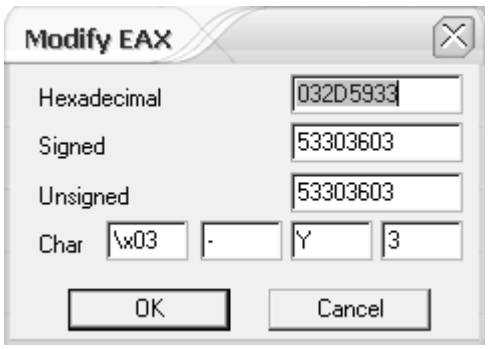

Рис. 10 – Окно изменения значения регистра.

Мы видим, что в десятичной форме регистр EAX содержит число, составленное из тех самых первых восьми цифр идентификационного ключа. Следующей инструкцией это число помещается в регистр ESI. Далее с ним производятся некоторые математические операции, в результате которых получается искомый серийный номер. Подробно эти операции мы рассматривать не будем.

- Нажимайте F8 до тех пор, пока atoi не будет выполнена во второй раз. Убедитесь, что в регистре EAX содержится введенный нами неправильный серийный номер (12345).

Следующие две инструкции предположительно выравнивают стек после математических операций. После этого следует инструкция CMP EAX, ESI, сравнивающая содержимое этих двух регистров. Как мы уже убедились выше, в ESI содержится сгенерированный программой серийный номер, а в EAX – введенный нами неправильный серийный номер. Следовательно, этой инструкцией программа сравнивает введенный нами серийный номер с полученным ей «эталоном».

- Поскольку нашей целью был этот самый «эталон», выделите регистр ESI, нажмите Enter и скопируйте десятичное значение этого регистра (66629503).

Теперь проверим полученный нами серийный номер.

- Нажмите Alt+F2 (Close) и выйдите из отладчика. Запустите программу.
- Введите в поле серийного номера полученный ранее номер и нажмите Register.

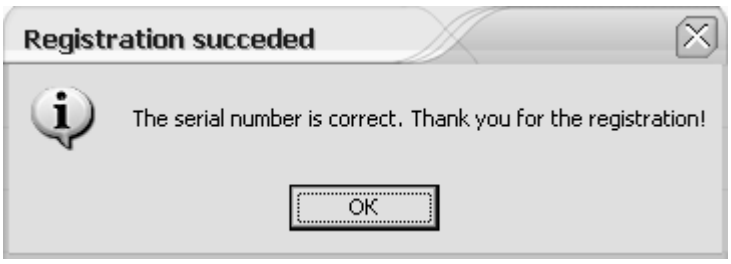

Рис. 11 – Сообщение об успешном завершении регистрации.

Мы видим, как после вывода сообщения об успешной регистрации (рис. 11) поля ввода и кнопка стали неактивными (это условный знак того, что программа теперь зарегистрирована). Однако такой метод взлома годится для разового использования (поскольку каждый раз программой генерируются разные идентификационные ключи, которые, кроме того, зависят от имени компьютера). Поэтому обычно для таких программ пишут генераторы ключей (используя код рассмотренной выше процедуры проверки серийного номера) или вносят изменения в код самой программы.

3.4.2. Изменение кода программы.

Наша цель – добиться того, чтобы программа осуществляла регистрацию при любом введенном номере. Обладая уже полученными о ней сведениями, нам не составит труда это осуществить.

- Запустите OllyDbg, откройте модуль CrackMeEx и перейдите к подпрограмме проверки серийного номера. Там найдите инструкцию сравнения регистров EAX и ESI.

Существует множество способов заставить программу делать то, что нам нужно, и никакого универсального подхода к этому нет. К примеру, нам вполне достаточно заменить эту инструкцию сравнения регистров EAX и ESI на сравнение двух одинаковых регистров. Или же убрать из кода программы условный переход, чтобы независимо от результатов сравнения всегда выполнялась часть кода, отражающая поведение программы на правильно введенный номер.

- Установите точку останова на начало подпрограммы проверки серийного номера (если она не установлена), запустите программу, введите любое число в поле серийного номера, например, 12345, и нажмите Register.

- Нажимайте F8, пока не дойдёте до адреса 0x004010A5 (непосредственно перед инструкцией сравнения).
- Выберите инструкцию CMP EAX, ESI и нажмите Space (Assemble).

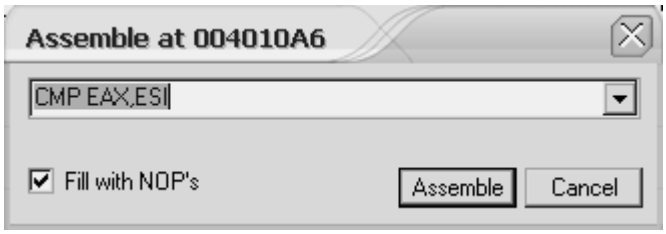

Рис. 12 – Окно ассемблирования по указанному адресу.

- Введите в строку новую инструкцию CMP EAX,EAX и нажмите Assemble.
- Нажмите F9 и увидите сообщение об успешной регистрации.

3.4.3. Создание патча для программы.

Патч (от англ. patch – заплатка) – это своеобразная заплатка, предназначенная для какого-либо изменения в оригинальной программе. Официальные издатели порой выпускают патчи в виде программ для исправления каких-либо недочётов в своих продуктах, либо для их обновления для новой версии. Взломщики, как правило, делают патчи для того, чтобы обходить систему защиты программы (они называются крэками). Причём патчи далеко не всегда являются программами. Чаще всего они представляют собой инструкцию, как «своими руками» исправить несколько байт программы, используя простой шестнадцатеричный редактор. Мы создадим такой патч для CrackMeEx.

- Запустите OllyDbg, откройте модуль CrackMeEx и перейдите к подпрограмме проверки серийного номера. Там найдите инструкцию условного перехода JNZ.
- Нажмите Assemble и введите новую инструкцию NOP.
- Таким образом, независимо от результатов, полученных после сравнения регистров, программа выполнит процедуру успешной регистрации.
- Нажмите / на панели инструментов, чтобы открыть окно патчей.
- Здесь мы видим все изменения, которые когда-либо были применены отладчиком к программе (рис. 13).

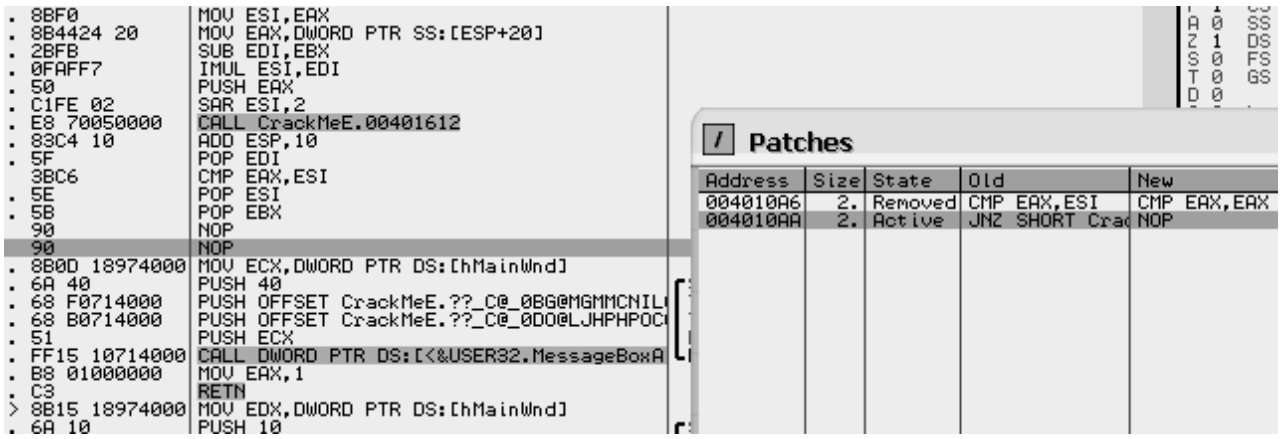

Рис. 13 – Измененный код программы и окно патчей.

- Выберите только что измененный код в окне патчей и нажмите Space (Restore original code).
- Статус патча сменился на Removed, а код программы был заменен исходным. Таким образом, OllyDbg позволяет удобно управлять всеми изменениями, вносимыми в исполняемые модули.
- Теперь необходимо выписать адрес инструкции, которая должна быть изменена, а так же опкоды старой и новой инструкций:

```
.004010AA        75       90
.004010AB        1F       90
```

Первое число в строке показывает смещение, по которому нужно заменить байт (второе число в строке) новым байтом (третье число в строке). Все числа в шестнадцатеричной форме.

Для получения полноценного офсетного патча следует преобразовать виртуальный адрес во внутрифайловое смещение. Можно сделать это самостоятельно, в данном случае отняв адрес загрузки (0x00400000) от найденного адреса, а можно воспользоваться HIEW.

```
000010AA     75       90
000010AB     1F       90
```

Теперь проверим патч в действии.

- Закройте OllyDbg и запустите любой шестнадцатеричный редактор, например, HIEW.
- Найдите смещение 0x10AA. Байт по заданному смещению должен быть равен 75, а следующий за ним – 1F (рис. 14).

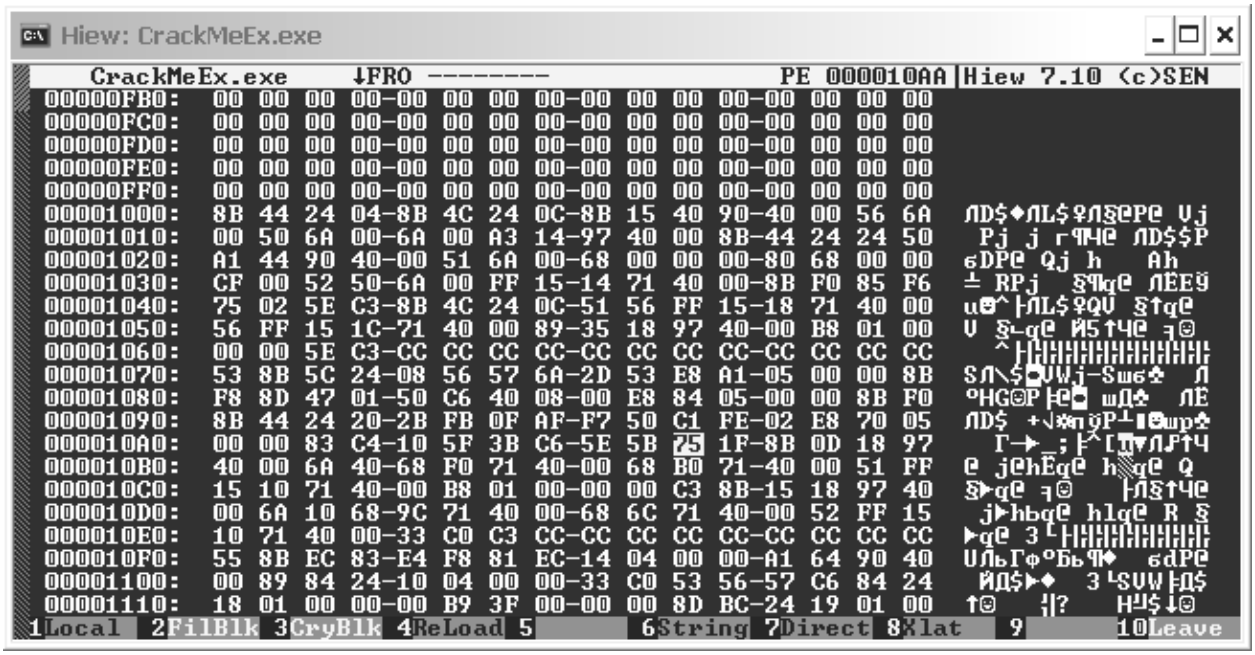

Рис. 14 – Байты, которые необходимо заменить.

- Замените найденные опкоды новыми (90 и 90), сохраните изменения и выйдите из редактора (рис. 15).

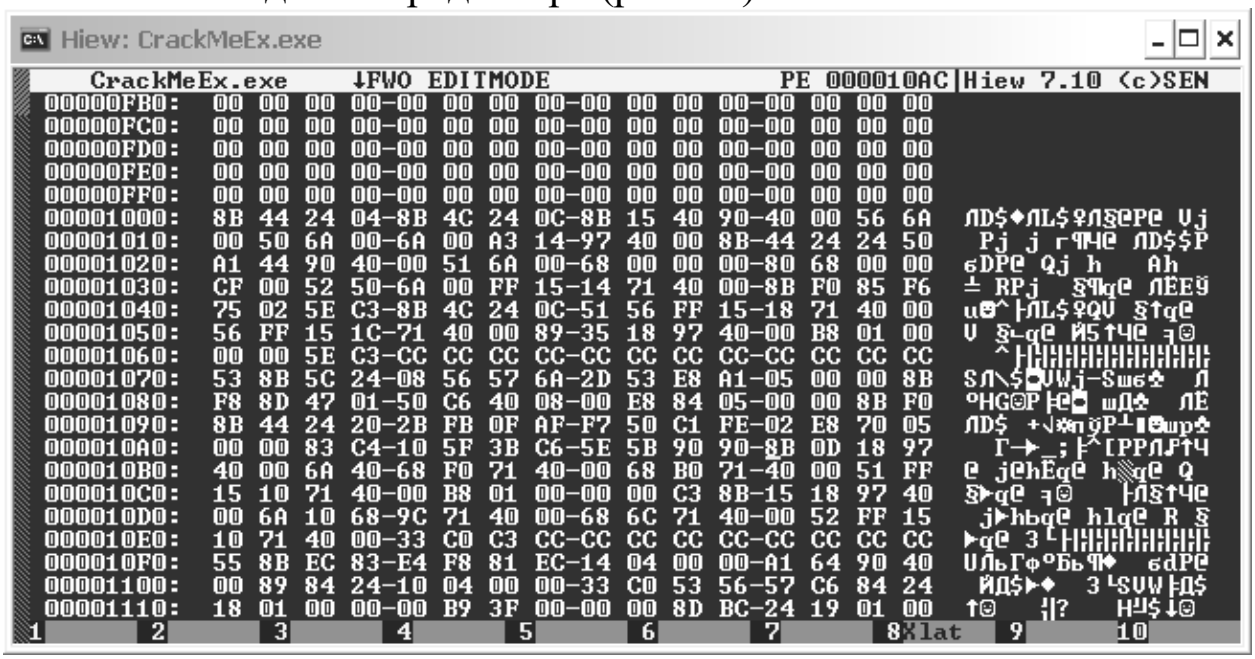

Рис. 15 – Внесенные изменения.

- Запустите программу, введите любое число и нажмите Register (рис. 16).

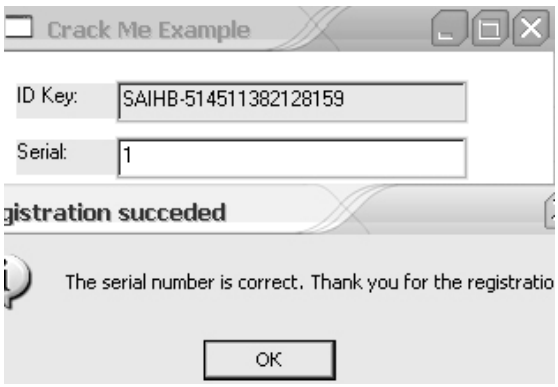

Рис. 16 – Успешная регистрация при любом введенном номере.

3.4.4. Исследование алгоритма формирования ключа.

Исследование алгоритмов работы программ является наиболее трудоёмким и долговременным процессом. Для этого трассируется участок кода процедуры проверки ключа и номера (с адреса 00401071 по адрес 004010A5 (рис. 17) с заходом в процедуры преобразования строк в число (по клавише F7) и отслеживается изменение содержимого регистров процессора.

```
00401071  . 8B5C24 08      MOV EBX,DWORD PTR SS:[ESP+8]
00401075  . 56             PUSH ESI
00401076  . 57             PUSH EDI
00401077  . 6A 2D          PUSH 2D                      [c = 2D  ('-')
00401079  . 53             PUSH EBX                     [s
0040107A  . E8 71050000    CALL CrackMeE.strrchr        [strrchr
0040107F  . 8BF8           MOV EDI,EAX
00401081  . 8D47 01        LEA EAX,DWORD PTR DS:[EDI+1]
00401084  . 50             PUSH EAX                     [s
00401085  . C640 08 00     MOV BYTE PTR DS:[EAX+8],0    [atoi
00401089  . E8 54050000    CALL CrackMeE.atoi
0040108E  . 8BF0           MOV ESI,EAX
00401090  . 8B4424 20      MOV EAX,DWORD PTR SS:[ESP+20]
00401094  . 2BFB           SUB EDI,EBX
00401096  . 0FAFF7         IMUL ESI,EDI
00401099  . 50             PUSH EAX                     [s
0040109A  . C1FE 02        SAR ESI,2                    [atoi
0040109D  . E8 40050000    CALL CrackMeE.atoi
004010A2  . 83C4 10        ADD ESP,10
004010A5  . 5F             POP EDI
004010A6  . 3BC6           CMP EAX,ESI
```

Рис. 17 – Участок кода, проверяющий правильность ключа

4. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ.

1) Для номера своего варианта выбрать исполняемый файл.
2) Проанализировать алгоритм формирования серийного номера, написать на языке высокого уровня программу генерации серийного номера по случайному ключу.
3) Создать офсетный патч для своей программы, приводящий к успешной регистрации для любой вводимой пары значений.

5. СОДЕРЖАНИЕ ОТЧЁТА.

1) титульный лист;

2) скриншот программы своего варианта;

3) скриншоты основных этапов выполнения работы;

4) описание алгоритма генерации серийного номера и листинг программы, его реализующей;

5) скриншот двоичного редактора при создании патча.

6. ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ.

1) Для чего необходима отладка программных средств?
2) Какие существуют методы и средства отладки программ?
3) Чем отличаются отладчики уровня пользователя от отладчиков уровня ядра?
4) Назовите выполняемые программными системами защиты информации машинные команды, которые являются наиболее критичными с точки зрения обеспечения информационной безопасности.
5) Назовите основные приёмы изменения хода исполнения программ.
6) Что такое патч и каким образом происходит его создание?

7. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) Гордеев, А.В., Молчанов, А.Ю. Системное программное обеспечение – Спб.: Питер, 2001. – 736с. ил.

2) Рихтер, Дж. Windows для профессионалов: создание эффективных WIN32 приложений с учетом специфики 64-х разрядной версии Windows. – СПб.: Издательский дом «Питер», 2001.