

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
«Защита компьютерной информации в клиентских приложениях»

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

Екатеринбург
2008

Федеральное агентство по образованию

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

В.В.Бакланов

ЗАЩИТА КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ В КЛИЕНТСКИХ ПРИЛОЖЕНИЯХ

Учебное пособие

Научный редактор проф., д-р техн. наук Н.А. Гайдамакин

Екатеринбург

УДК 691.3.07
ББК 32.973.26

Рецензенты:

кафедра систем и технологий защиты информации Уральского государственного университета путей сообщения (зав. кафедрой проф., д-р физ.-мат. наук Ю.И.Ялышев)

доц., канд. физ.-мат. наук О.Н. Соболев (в/часть 69617, г. Екатеринбург)

Автор: В.В.Бакланов

Защита компьютерной информации в клиентских приложениях:
учебное пособие / В.В.Бакланов. Екатеринбург: ГОУ ВПО УрГУ, 2005. 84 с.

ISBN

В пособии рассматриваются вопросы, связанные с защитой компьютерной информации, обрабатываемой с помощью распространенных клиентских приложений Microsoft Word 8.0-11.0 и Microsoft Internet Explorer 5*. Излагаются алгоритмы и механизмы обработки информации, влияющие на защиту компьютерной информации от угроз конфиденциальности, целостности и доступности. Анализируются известные атаки на клиентские приложения. Особое внимание уделяется защите от возможности внедрения, запуска и распространения вредоносных программ. Теоретический материал закрепляется выполнением лабораторных работ, которые содержат элементы исследовательской деятельности.

Учебное пособие предназначено для студентов, обучающихся по специальностям 075200 - Компьютерная безопасность, 075500 - Комплексное обеспечение информационной безопасности автоматизированных систем, 075600 - Информационная безопасность телекоммуникационных систем.

Библиогр.: 21 назв. Рис. 16

УДК 691.3.07
ББК 32.973.26

ISBN

© ГОУ ВПО «Уральский государственный университет», 2005

© В.В.Бакланов, 2005

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
1. ИССЛЕДОВАНИЕ УЯЗВИМЫХ МЕСТ И ЗАЩИТНЫХ МЕХАНИЗМОВ ТЕКСТОВОГО ПРОЦЕССОРА MICROSOFT WORD	6
1.1. Общие сведения о документах Word	6
1.2. Механизмы образования информационного «технологического мусора»	8
1.3. Угрозы, способствующие внедрению, распространению, функционированию и сокрытию вредоносных программ	13
1.4. Способы обнаружения и нейтрализации вредоносного программного кода	21
2. ЛАБОРАТОРНАЯ РАБОТА № 1	34
2.1. Исследование механизмов образования информационного «технологического мусора»	34
2.2. Определение приоритетов в выполнении автоматически исполняемых макросов	36
2.3. Исследование приоритетов в запуске событийных процедур, связанных с открытием документов	41
2.4. Исследование механизма вирусного заражения документов и шаблонов Word	42
2.5. Исследование механизма защиты от вредоносных макросов	44
2.6. Блокирование просмотра программного кода, реализованного в некоторых макровирусах	46
2.7. Разработка процедуры для блокирования и нейтрализации вредоносного макрокда	46
2.8. Контрольные вопросы	47
3. ИССЛЕДОВАНИЕ ЗАЩИТНЫХ МЕХАНИЗМОВ БРАУЗЕРА MICROSOFT INTERNET EXPLORER	49
3.1. Алгоритмы функционирования браузера Internet Explorer	49
3.2. Анализ опубликованных уязвимостей браузера Internet Explorer ..	64
4. ЛАБОРАТОРНАЯ РАБОТА № 2	69
4.1. Подготовка к проведению исследований	69
4.2. Имитация атак на отказ в обслуживании	72
4.3. Исследование возможности копирования файлов в кэш браузера ..	72
4.4. Исследование возможности запуска файлов из кэша браузера	76
4.5. Исследование прочих возможностей внедрения и запуска файлов ..	78
4.6. Выявление элементов управления ActiveX, зарегистрированных в качестве безопасных	78
4.7. Контрольные вопросы	80
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	82

ВВЕДЕНИЕ

Защита компьютерной информации с позиций администратора безопасности привычно ассоциируется с операционной системой. Действительно, большинство защитных механизмов, от разграничения доступа пользователей к файлам до шифрования данных, размещено именно в операционной системе. Вместе с тем определенная часть задач по защите компьютерной информации ложится на программные приложения, с которыми приходится работать пользователю. Это средства создания и редактирования текстовых документов, электронные таблицы, системы управления базами данных, почтовые клиентские программы и Интернет-браузеры. Так, для операционной системы Windows* документ Word – это просто файл, который она распознает только по характерному расширению .doc. Опасные функции, содержащиеся в этом документе, в том числе потенциально опасный программный код, видны только его «родной» программе – текстовому процессору Word (для которого, кстати сказать, расширение имени файла .doc не является значимой информацией). На уровне приложения определяется и механизм шифрования конфиденциального документа, и стратегия резервирования и восстановления содержимого файлов на случай сбоя, и защита от вредоносных программ в макросах.

От браузера, с помощью которого осуществляется получение информации в Интернет, зависит не только производительность поиска и отображения этой информации, но и защита всей компьютерной системы от сетевых атак и проникновения вредоносных программных компонентов.

Большинство программных приложений в отношении обрабатываемой информации выполняют роль самостоятельных операционных систем. Как справедливо отметил один из руководителей подразделений фирмы Microsoft после разработки офисного пакета образца 1997 года: «Теперь Microsoft Office – это настоящая операционная система, а Visual Basic for Application – ее API»

В этом учебном пособии рассматривается защита компьютерной информации в клиентских приложениях, и в качестве примеров рассматриваются две весьма примечательные программы (точнее не программы, а целые программ-

ные пакеты), разработанные в фирме Microsoft, – текстовый процессор Word версий 8.0-11.0 и браузер Internet Explorer версий 5*. В настоящее время опубликовано много книг по защите серверов, предоставляющих услуги сети Интернет. Защита серверов безусловно важна, но следует признать, что пользователей в компьютерном мире гораздо больше, и от их безопасности в первую очередь зависит компьютерная безопасность общества в целом.

Материал пособия состоит из двух тем. В теоретической части каждой темы излагаются способы и средства защиты информации, реализованные в рассматриваемых приложениях, а также типичные компьютерные атаки, которые когда-либо были успешно проведены информационными злоумышленниками по причине уязвимостей, допущенных программистами фирмы Microsoft. Теоретический материал дополняется лабораторным практикумом, позволяющим обучаемым самостоятельно исследовать некоторые уязвимости и защитные механизмы прикладных программ.

Современные уязвимости программных приложений Microsoft Word и Microsoft Internet Explorer, обнаруженные рядом исследователей, включая и автора, в данном пособии не рассматриваются, поскольку оно должно служить целям безопасности и не превращаться в наставление по реализации преступных замыслов.

1. ИССЛЕДОВАНИЕ УЯЗВИМЫХ МЕСТ И ЗАЩИТНЫХ МЕХАНИЗМОВ MICROSOFT WORD

Текстовый процессор Word в течение ряда лет фактически является стандартной средой подготовки и редактирования электронных документов. Этому в немалой степени способствуют несомненные удобства использования, предусмотренные его разработчиками из фирмы Microsoft. Одновременное редактирование нескольких документов, использование редакторов текста, таблиц, рисунков, диаграмм, формул, встроенная система проверки правописания, наследование настроек форматирования – это только немного из тех возможностей, которые обеспечивает Word. С помощью установки многочисленных параметров, видоизменения меню и командных панелей, создания собственных макросов или использования программ других авторов опытный пользователь может до неузнаваемости изменить свою рабочую среду.

Microsoft Word победил многих своих конкурентов именно за счет своих широчайших возможностей. Но за удобства надо платить, а избыточная функциональность почти всегда небезопасна. И если разработчики текстового процессора в достаточной мере предусмотрели защиту компьютерных данных от угроз целостности, включая возможность восстановления поврежденного документа, то для защиты конфиденциальных данных стандартных мер защиты явно недостает.

1.1. Общие сведения о документах Word

В многооконном режиме работы текстовый процессор может работать со многими документами, но только один из них (ActiveDocument) в конкретный момент времени является активным и имеет фокус ввода.

Документ Word – это файл с весьма сложным форматом. Кроме непосредственно текста, в документ входят внедренные объекты (графика, видео, звук, элементы управления), а также сегменты для размещения интерпретируемого программного кода. Это обусловлено тем, что Microsoft Word поддерживает

технологии OLE 2.0 и информация записывается в файл (и считывается из него) в виде нескольких потоков данных с помощью специального программного интерфейса (API). Сложная структура файла позволяет скрыть в нем реализованные защитные механизмы. Но, по справедливому мнению известного специалиста по компьютерной безопасности Брюса Шнайера, сложность – это проклятие для безопасности, и сложная структура файла является причиной уязвимости содержащейся в нем информации.

Для документов и шаблонов Word характерна блочная структура. Размер файла всегда кратен 512 байтам (размер стандартного сектора на магнитном диске). Созданный документ (а созданным он может считаться только после первого сохранения файла) уже содержит в себе многочисленные элементы форматирования и настройки. Даже если в документ не вставлено ни одного символа, он уже имеет объем около 20 Кб (в Word 9.0 – 11.0 объем «пустого» файла документа составляет 19968 байт или 39 секторов). Блочная структура файла с частично заполненными блоками является причиной использования документов Word в качестве стеганографических контейнеров.

Документам Word по умолчанию присваивается расширение файла .doc. Но программа winword.exe различает свои документы не по расширению, а по «магическому» числу (сигнатуре) в заголовке файла. Этим числом являются 4 байта E011CFD0h по нулевому смещению от начала файла (в шестнадцатеричном дампе файла эти байты следует читать справа налево). На самом деле это сигнатура, характерная не только для документов Word, но и для многих сложных файлов, организованных по форматам структурированного хранилища (structured storage). Для документов, созданных в программой winword.exe версий 8.0 – 11.0, в файле есть еще одна сигнатура A5ECh, размещенная по смещению 200h или 300h от начала файла (тоже читается справа налево).

В Microsoft Word и его файлах организовано множество оригинальных и любопытных механизмов, но в рамках данного пособия мы остановимся только на некоторых аспектах, имеющих отношение к обеспечению безопасности компьютерной информации.

1.2. Механизмы образования информационного «технологического мусора»

Структура файла Word во многом определяется тем, в каком режиме происходит его «ручное» и автоматическое сохранение. Сохранять или не сохранять вновь созданный файл – личное дело пользователя. Периодичность дальнейшего принудительного сохранения документа с помощью меню «Файл»-«Сохранить» или кнопки «Сохранить» на основной командной панели определяется многими факторами: важностью документа, внимательностью пользователя, степенью его доверия к компьютеру и его программному обеспечению и др. Но Microsoft Word часть ответственности за сохранение документа берет на себя. Автоматически, с интервалом от 5 до 10 минут (скорее всего, с использованием датчика случайных чисел), редактируемый файл автоматически сохраняется в двух каталогах: каталоге для временных файлов операционной системы (обычно это каталог C:\Windows\Temp или %TEMP%) и в рабочем каталоге, в котором хранится файл редактируемого документа. В меню «Сервис» – «Параметры» – «Сохранение» автосохранение документа можно отменить, сняв «галочки» с пунктов «Автосохранение каждые ... минут», «Разрешить фоновое сохранение», «Всегда сохранять резервную копию». Но мнение пользователя для офисного приложения решающим не является. Во всяком случае в рабочем каталоге документа резервные файлы создаются независимо от настроек. При продолжительном сеансе работы над документом или его многократном редактировании у файла появляются десятки или даже сотни копий. Если документ велик и редактируется долго (а программа не определяет, работает ли в данное время пользователь с документом, или ушел на несколько часов, оставив компьютер включенным), то под резервные файлы на диске тратится довольно много места.

При закрытии документа его копии из рабочего каталога логически удаляются, а, как известно, логическое удаление файлов не затрагивает блоков данных, и они могут храниться на диске еще долгое время. За удаление времен-

ных файлов из каталога TEMP ни операционная система, ни текстовый процессор вообще не отвечают.

С режимом автоматического сохранения связан механизм восстановления данных в случае внезапного (не зависящего от пользователя) завершения работы с документом. Это может произойти из-за сбоя в работе текстового процессора, «глюка» операционной системы или просто из-за выключения питания. В этом случае самый последний из временных файлов используется программой Word для восстановления документа. В Word версии 8.0 этот последний временный файл располагался в каталоге %TEMP% и имел характерное расширение .asd. При очередной загрузке winword.exe проверял, есть ли в каталоге %TEMP% какой-нибудь файл с расширением .asd, и если «магическое» число этого файла соответствовало документу Word, документ автоматически открывался в текстовом процессоре.

Характерно, что механизм восстановления данных в Microsoft Office 97 был очень уязвим. Файл автосохранения, имеющий расширение .asd, открывался в Word без какой-либо проверки на наличие макросов. Злоумышленник мог предварительно создать произвольный документ в формате Word, внедрить в него вредоносный макрос, изменить расширение файла с .doc на .asd и внедрить полученный файл в каталог для временных файлов (тем более, что во всех версиях Windows*, кроме XP, этот каталог является общим и доступным для записи любому зарегистрированному пользователю). Запланированное вредоносное действие выполнялось автоматически при очередном запуске Word. К счастью, разработчики своевременно исправили этот дефект, и все последующие версии Word проверяют файлы автосохранения на предмет наличия макросов точно так же, как и любой другой файл. А начиная с версии Word 10.0, неизвестно куда подевались и сами файлы asd-типа. Но Word все равно восстанавливает содержание последней редакции документа в случае аварийного завершения работы, и делает это по временной отметке временного файла, сохраненного последним. Кстати сказать, все временные файлы имеют псевдослучайные имена, состоящие из 8 символов, и это связано с одной из старых атак [18]. Временные фай-

лы Word имеют 3 фиксированных символа ~WR, один символ, постоянный для конкретного каталога и конкретного документа, и 4 случайные цифры. Все временные файлы имеют расширение .tmp.

Режим «быстрого сохранения» ведет к созданию так называемого комплексного или быстро сохраняемого (complex или fast-saved) файла. В этом режиме при каждом сохранении файл перезаписывается лишь частично, и в него записываются только «приращения» к ранее сохраненной информации. По сути, при каждом сохранении создается новый файл, содержащий вновь введенный текст и внедренные объекты, а все, что было сохранено раньше, остается без изменения. Комплексный файл похож на айсберг, и пользователь из текстового процессора Word видит только верхушку этого айсберга. Предыдущие редакции документа в этом режиме не просматриваются, и увидеть документ целиком можно только с помощью какого-либо шестнадцатеричного редактора (для этого хорошо подходит view'ер файлового менеджера Far, позволяющий просматривать файлы в различной кодировке, включая UNICODE).

Сложная блочная структура файла не всегда позволяет освободиться от информации, ранее присутствовавшей в документе, но удаленной при последующем редактировании. Предположим, что пользователь при разработке документа включил в него конфиденциальные данные, которые были сохранены в файле. В процессе дальнейшей работы пользователь исключил из документа некоторые фрагменты с конфиденциальными сведениями и вновь сохранил файл. Естественно, он считает, что электронный документ уже не содержит конфиденциальных данных и поэтому не принимает мер к защите файла от копирования и распространения. Однако «мусор» прошлых редакций во многих случаях в файле сохраняется. Особенно актуально это для режима «быстрого сохранения». Грошовая экономия времени на сохранении файла в результате приводит к двум неприятностям: чрезмерно большому размеру файлов и заведомому наличию в них «технологического мусора». Следует помнить, что режим «Быстрого сохранения» установлен в Word по умолчанию и пользователю

необходимо убрать «галочку» против параметра «Разрешить быстрое сохранение» в меню «Сервис» – «Параметры» – «Сохранение».

Многое в накоплении «технологического мусора» зависит и от операционной системы. Так, операционные системы Windows 9* выделяют для сохранения файлов неочищенные блоки (страницы) оперативной и дисковой памяти. С учетом того, что внутри файла Word может быть довольно много свободного места, внутрь документа могут попасть фрагменты иных документов. Особенно это опасно, когда в несекретные документы попадают фрагменты ранее удаленных файлов с конфиденциальным содержанием. Аналогичная опасность имеет место при одновременной работе с секретными и несекретными документами. В этом случае конфиденциальная информация может попасть в другой документ с неочищенными страницами оперативной памяти. Семейство операционных систем Windows NT 5.0, к счастью, лишено такого недостатка.

Существуют и иные механизмы утечки конфиденциальной информации, основанные на косвенных признаках конфиденциальности.

1. При создании нового документа тестовый процессор фиксирует первую введенную пользователем строку (цепочка символов, завершенная нажатием <ENTER>). При сохранении документа пользователю предлагается эта строка в качестве имени файла. Обычно пользователь выбирает иное имя, но первая строка тем не менее сохраняется в «Свойствах» документа в качестве его названия. Название конфиденциального файла рекомендуют выбирать так, чтобы оно не раскрывало суть документа. Но первая строка может быть и заголовком документа, и грифом его секретности. В дальнейшем пользователь может зашифровать документ и придумать его файлу безопасное название, но в свойствах документа в открытом виде по-прежнему сохраняется первая введенная строка. Возможность шифрования свойств документа предусматривается только с 10-й версии Word, и пользователь должен указать это явно.
2. В «Свойствах» документа присутствует статистика, из которой можно почерпнуть довольно интересные сведения. Например, распечатывался ли до-

кумент и когда это происходило в последний раз, сколько раз он редактировался и сколько времени пользователь потратил на работу с документом. Статистика указывает количество страниц, абзацев, предложений, слов и символов. Не зная содержания зашифрованного документа, сопоставляя размер его файла с этими данными, можно составить представление о его содержании (текст, рисунки, таблицы и т.д.).

3. Автоматическая проверка правописания безусловно полезна как начинающим, так и опытным пользователям, ведь ошибки возникают не только из-за незнания грамматики или орфографии, но и вследствие обычных «очепяток». В некоторых случаях Word предлагает довольно забавные варианты синтаксиса русских слов. Для проверки правописания Word использует базовые словари, например CUSTOM.DIC, расположенный в каталоге OFFICE. Но пользователь, профессионально работающий на компьютере, обычно не ограничивается базовым словарем. Если введенного слова нет в словаре, процедура проверки орфографии выделяет это слово красной волнистой линией. Если пользователь уверен в правильном написании слова, он может добавить его в словарь, но этот словарь представляет собой обычный текстовый файл, в который добавленные слова записываются в виде отдельных строк. Словарь не шифруется, и при довольно продолжительной работе в нем накапливается достаточно слов, чтобы создать представление о роде «литературной» деятельности пользователя (мы по-прежнему предполагаем, что пользователь работает с конфиденциальными документами и их файлы хранятся в зашифрованном виде).

По вышеуказанным причинам программная продукция Microsoft, несмотря на свою популярность, лишь ограниченно пригодна для обработки конфиденциальной информации. Для использования текстового процессора Word для разработки и редактирования конфиденциальных документов необходимо разработать и установить дополнительное программное обеспечение (например, в форме макросов).

1.3. Угрозы, способствующие внедрению, распространению, функционированию и сокрытию вредоносных программ

Одна из особенностей процессора Word заключается в использовании шаблонов для создания новых документов. Шаблон – это заготовка, образец для вновь создаваемого документа. В шаблоне содержатся стили, автотекст, командные панели и макросы, которые копируются в документ, созданный на основе данного шаблона. Сосредоточение в шаблоне настроек форматирования, характерных для документов определенных типов, позволяет пользователям существенно экономить свое время и автоматически переносить прежние настройки на каждый новый документ. Файлы вновь созданных шаблонов лишь немного больше по объему файлов «пустых» документов. По структуре они практически идентичны документам. Документы и шаблоны взаимно преобразуются: документ можно сохранить, как шаблон, а шаблон – открыть в Word, как документ. Именно «благодаря» шаблонам получила распространение такая категория вредоносных программ, как макровирусы. Шаблоны являются промежуточной средой для инфицирования: зараженный документ инфицирует шаблон, а через зараженный шаблон вредоносный код копируется в каждый вновь создаваемый или открываемый для просмотра или редактирования документ. Кстати, разнообразие механизмов копирования программного кода, реализованных в Word, делают возможным инфицирование документов и без посредства шаблонов.

В Word имеются три типа шаблонов, которые отличаются «сферой влияния», продолжительностью действия, а также приоритетом при выполнении событийных процедур.

1. Обычный или глобальный шаблон реализован в виде отдельного файла с именем Normal.dot. Например, в Windows 9* этот шаблон обычно располагается в одной из трех папок:

- **C:\Program Files\Microsoft Office\Шаблоны**
- **C:\WINDOWS\Application Data\Microsoft\Шаблоны**
- **C:\WINDOWS\All Users\Application Data\Microsoft\Шаблоны**

Этот шаблон присутствует в среде Word всегда и «работает» при открытии любого документа. Новые документы, создаваемые из меню «Файл» – «Создать», заимствуют формат и настройки этого шаблона. Normal.dot хранит в себе основные пользовательские настройки. С помощью файлового монитора, настроенного на контроль файлов, открываемых winword.exe, можно увидеть, что Normal.dot – это файл, с которым наиболее интенсивно «общается» Word во время сеанса работы. Имя глобального шаблона фиксировано и не подлежит изменению. При загрузке Word ищет файл с таким именем в заданном каталоге, и если не находит, генерирует временный файл, который при первом сохранении или завершении работы Word преобразуется в новый глобальный шаблон. Вновь созданный файл Normal.dot в Word версий 9.0 и выше имеет объем 29 Кб (29696 байт или 58 блоков).

Normal.dot наиболее интенсивно собирает «технологический мусор», накапливая в себе все ранее установленные настройки. Всего за несколько недель работы размер Normal.dot может увеличиться от нескольких десятков до нескольких сотен килобайт. Удаление файла-шаблона считается наиболее надежным способом избавления и от вирусов в макросах, и от надоевших настроек.

2. Шаблон, присоединенный к документу (Document.AttachedTemplate, attach – прикрепленный). «Прикрепить» шаблон к документу можно «вручную», через диалоговое окно «Сервис» – «Шаблоны и надстройки», с помощью верхней командной кнопки «Присоединить», либо на программном уровне с помощью инструкции: **Document(N).AttachedTemplate = "имя_шаблона"**. Настройки и командный код присоединенного шаблона добавляются к настройкам и коду глобального шаблона, но действуют только на один документ. Событийные процедуры, хранящиеся в присоединенном шаблоне, исполняются только тогда, когда соответствующий документ (к которому шаблон прикреплен) является активным. Имя присоединенного шаблона фигурирует в свойствах документа. Присоединенный шаблон распознается по сигнатуре в заголовке файла и может иметь любое имя и расширение.

3. Дополнительные общие (AddIns) шаблоны. Они названы общими, поскольку сфера их действия распространяется на все открываемые документы. Но время их действия ограничено, поскольку эти шаблоны присоединяются на один сеанс работы. Подключить общие шаблоны тоже можно как «вручную», так и с помощью программного кода. Подключение производится через меню «Сервис» – «Шаблоны и надстройки» командной кнопкой «Добавить». Однажды присоединенный общий шаблон будет постоянно отображаться в нижнем окне редактирования диалогового окна «Шаблоны и надстройки», его подключение на сеанс производится установкой «галочки» против соответствующего имени. Таким же образом «галочку» можно убрать, а выделенный затем шаблон – удалить (командная кнопка «Удалить»). Программное присоединение, подключение, отключение и удаление общих шаблонов производится с помощью свойств и методов объекта AddIns. В системном реестре и системных файлах такие шаблоны не регистрируются. «Упоминание» об именах общих шаблонов можно найти в «теле» файла Normal.dot.

Однако общие шаблоны можно подключать на более длительный срок, чем один сеанс. Для этого имя шаблона с обязательным расширением .dot должно быть помещено в папку автозагрузки приложений Microsoft Office, которая по умолчанию имеет адрес C:\Program Files\Microsoft Office\Office\STARTUP. Путем изменения настроек реестра эту папку можно переименовать и перенести в любое другое место.

Масса потенциальных удобств для опытного пользователя и еще большее число потенциальных угроз связаны с возможностью включения в документы и шаблоны интерпретируемого программного кода, который упрощенно называют макросами. На самом деле макрос – это событийный код открытых процедур из числа открытых (Public) модулей проекта. Изложение большинства вопросов, связанных со средой программирования VBA, можно найти в ряде других источников [2,6,14]. В данном пособии рассматриваются только некоторые сведения, имеющие непосредственное отношение к антивирусной защите.

В среде текстового процессора Word интерпретируемый код может постоянно размещаться в:

- глобальном шаблоне Normal.dot,
- документе формата Word,
- шаблоне, присоединенном к документу,
- дополнительном общем шаблоне,
- скомпилированном файле надстройки (динамической библиотеке).

Проект, или программный проект – это часть шаблона или документа, содержащая программный код. В свою очередь проект включает в себя программные модули, пользовательские формы (диалоговые окна) и ссылки. Наибольший интерес для антивирусной защиты представляют программный модуль NewMacros, в который пользователи записывают свои макросы, и встроенный (постоянно присутствующий в шаблоне или документе) модуль ThisDocument. Оба модуля интенсивно используются вредоносными программами для размещения в них программного кода и данных. В каждом модуле создаются более мелкие фрагменты кода, именуемые процедурами и функциями. Различие между ними в том, что функция способна возвращать какие-либо значения (например, результаты вычислений), а процедура – нет.

Вредоносные программы обычно состоят из нескольких процедур (реже – функций), размещенных в одном из двух перечисленных программных модулей. Процедура – это самостоятельный блок кода, заключенный между операторами Sub и End Sub. Процедуры могут быть закрытыми (в этом случае перед оператором Sub ставится ключевое слово Private – конфиденциальный) и открытыми (с ключевым словом Public или без него). Различие между ними в том, что открытую процедуру можно вызывать из любой другой процедуры в данном проекте, а закрытую – только из процедур данного модуля. Большинство процедур, способных содержать вредоносный код, являются открытыми.

Следующая особенность программной среды VBA связана с событиями. 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. Обработка событий позволяет вредоносной программе совершать инфицирование, производить деструктивные действия и маскироваться.

Реагировать на события могут только процедуры, причем в своем большинстве – только открытые процедуры. Для того чтобы процедура могла перехватывать факт наступления события, она должна быть названа определенным именем. Различают три группы событийных процедур:

1. События, обрабатываемые автоматически запускаемыми макросами:

AutoExec – обрабатывается при запуске Word или загрузке глобального шаблона,

AutoNew - при создании нового документа,

AutoOpen - при открытии существующего документа,

AutoClose - при закрытии документа,

AutoExit - при закрытии Word или выгрузке глобального шаблона.

Считается, что в программном противоборстве выигрывает программа, которая запускается первой. Самым первым из запускаемых макросов является AutoExec, поскольку он сопровождает запуск самого текстового процессора. На основе автомакросов было написано большое число вредоносных программ. Но пользователь может «вручную» заблокировать выполнение автомакросов, удерживая в нажатом состоянии при открытии Word и при других перечисленных событиях клавишу Shift. Автомакросы можно отключить и программно, запуская в начале сеанса процедуру, содержащую строку: Word-

Basic.DisableAutoMacros. В этом случае автоматические макросы становятся пассивными на весь сеанс работы (до выключения Word).

2. События, связанные с документами. Процедуры, обрабатывающие эти события, размещаются в постоянном модуле ThisDocument и являются закрытыми (Private). Их имена содержат слово Document и через символ подчеркивания – действие, производимое над этим документом:

```
Private Sub Document_Open()  
  'Обработка события, связанного с открытием документа  
End Sub
```

```
Private Sub Document_New()  
  'Обработка события, связанного с созданием документа  
End Sub
```

```
Private Sub Document_Close()  
  'Обработка события, связанного с закрытием документа  
End Sub
```

Нетрудно заметить, что три автомакроса и процедуры, связанные с документами, обрабатывают одни и те же события. Отличия между ними заключаются только в очередности их исполнения при наступлении событий.

```
AutoOpen() - Document_Open()  
AutoNew() - Document_New()  
AutoClose() - Document_Close()
```

События, связанные с документами, наиболее часто используются вредоносными программами с целью инфицирования. Например, события Document_Open() и Document_New() являются сигналом для вредоносного макрокда, уже «захватившего» глобальный шаблон, о том, что открывается или создается новый документ, и неплохо бы его «заразить».

3. События, связанные с активизацией элементов управления Word, например.

```
Sub FileSave()  
  Сохранение активного документа или шаблона  
End Sub
```

Sub ToolsMacro()

Открытие диалогового окна "Макрос"

End Sub**Sub ViewVBCode()**

Вход в интегрированную среду разработки VBE

End Sub

Имена этих процедур образованы из типовых команд Word, а полный список таких команд можно посмотреть в диалоговом окне «Макрос». Для этого следует вызвать диалоговое окно (это делается одним из трех способов):

- «Сервис» – «Макрос» – «Макросы»,
- «Вид» – «Панели инструментов» – «Visual Basic», а затем нажать командную кнопку с треугольничком в появившейся панели,
- Alt + F9.

Затем следует выбрать «Макросы из» - «Команд Word», и эти команды в алфавитном порядке будут отображены в диалоговом окне. Смысл некоторых команд можно понять из их названия, например, команда FilePrint должна соответствовать пункту меню «Файл» – «Печать». В случае затруднений следует обратить внимание на описание команд Word в нижней части окна «Макрос», которое выводит подсказку по каждой выбранной команде.

Для того чтобы сделать процедуру событийной, надо просто назвать ее подходящим именем. Например, назвав процедуру EditCopy(), можно будет перехватывать 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.

Далеко не все команды Word из предложенного списка могут трансформироваться в событийные процедуры. Скорее всего программисты фирмы Microsoft еще не успели запрограммировать обработку событий для каждой из заре-

зеврированных команд. В любом случае надо попробовать, и самой простой пробой является вывод сообщения, например:

```
Sub EditCopy()  
MsgBox «Копирование в буфер»  
End Sub
```

после чего следует проверить, реагирует ли созданная процедура на нужное событие (в данном случае на выделение фрагмента документа и нажатие командной кнопки «Копировать»).

Событийные процедуры третьего вида позволяют сделать компьютерную программу по настоящему интеллектуальной. Практически с каждым пунктом меню, с каждой командной кнопкой может быть связано событие. Благодаря этому ни одно действие пользователя не остается незамеченным. Вредоносная программа, содержащая событийные процедуры и перехватывающая действия пользователя, может гибко реагировать на ситуацию и при угрожающих действиях пользователя вовремя скрыться.

Возможны ситуации, когда событийные процедуры, имеющие одно и то же имя, могут присутствовать и в документах, и в шаблонах. Если предположить, что первичное инфицирование происходит из постороннего документа, то очень важно определить, какая из программ – вредоносная или антивирусная – победит в противоборстве. Для этого нужно определить приоритеты, с которыми исполняются событийные процедуры одного типа из документов и шаблонов. Затем следует определить приоритет в выполнении родственных процедур различного типа. Установить эту очередность читателям предлагается самостоятельно по методике, приведенной к заданиям на лабораторную работу. На основе проведенных исследований можно будет установить, какую именно процедуру нужно разместить в текстовом процессоре, чтобы успешно противодействовать вредоносным программам.

1.4. Способы обнаружения и нейтрализации вредоносного программного кода

Реализованная в Microsoft Office защита от вредоносных макросов далеко не отвечает своему назначению, поскольку по сути является средством обнаружения макросов вообще. Ее реализация в различных версиях Word различается.

В Word 8.0 для установки этой защиты следует установить флажок для параметра «Защита от вирусов в макросах» в меню «Сервис» – «Параметры» – «Общие». После этого winword.exe начинает проверять каждый открываемый документ на предмет наличия в нем группы блоков, выделенных для макросов. Если эти признаки обнаружены, программа выводит диалоговое окно, изображенное на рис. 1, предлагая пользователю на выбор три решения: отключить макросы, не отключать макросы, не открывать документ. Что на самом деле со-

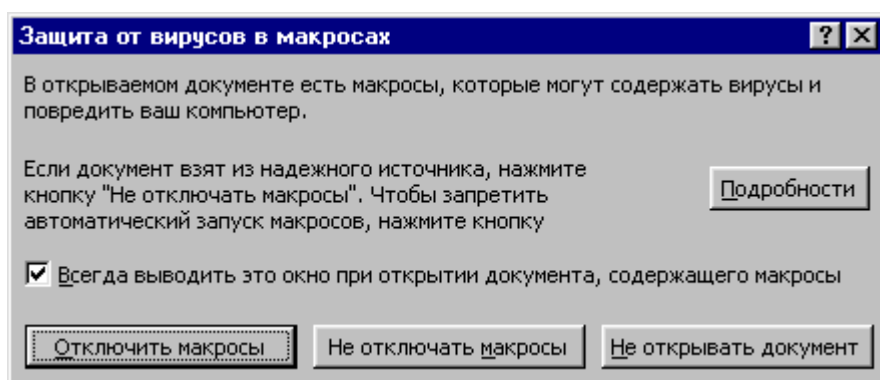


Рис. 1. Окно предупреждения о наличии макросов в открываемом документе Word 8.0

держится в открываемом документе, пользователю неизвестно, и он вынужден принимать решение «вслепую», основываясь на своей подозрительности и жизненном опыте. Можно предполо-

жить, что текстовый процессор анализирует программный код найденных макросов на наличие в них известных опасных сигнатур. Но это не так, что легко доказать. Создадим в Word 8.0 новый документ. Затем откроем редактор Visual Basic for Application, для чего выберем в меню «Сервис» – «Макрос» – «Редактор Visual Basic». В редакторе откроем окно проектов, найдем проект документа (рис. 2) и дважды «щелкнем» по встроенному модулю ThisDocument. В окне редактора откроется еще одно дочернее окно с текстом программы. В только что созданном документе еще никаких программ не содержится, и мы запишем в

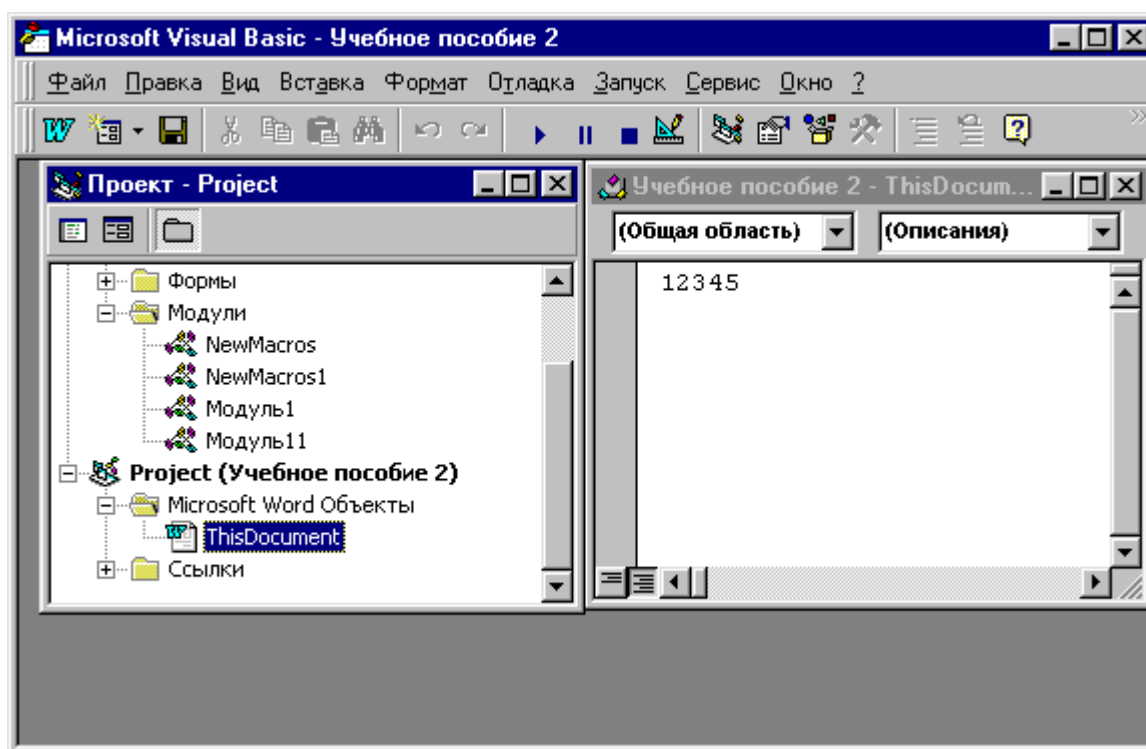


Рис.2. Создание макроса в документе

это окно произвольную комбинацию символов, например, цифры «12345», которые не только не «вредоносны», но в принципе никакой программой не являются. После этого документу присваивается какое-нибудь имя, например, «Документ с макросом», он сохраняется и закрывается. После установки «Защиты от вирусов в макросах» этот документ вновь открывается, и бдительный Word тут же предупреждает нас об опасности. Кстати, нетрудно убедиться, что при сохранении документа с любым кодом размер его файла автоматически увеличивается на несколько килобайт – это программа резервирует около десятка 512-байтных блоков для размещения в документе программного кода. По-видимому, само присутствие в файле этих дополнительных блоков и вызывает срабатывание системы «защиты».

У этого эксперимента есть довольно забавное продолжение. Для этого следует открыть «инфицированный» документ, войти в редактор VBE и удалить из окна кода модуля ThisDocument ранее записанную строку цифр. После этого документ вновь сохраняется и закрывается. Можно предположить, что теперь система защиты ничего подозрительного не обнаружит. Как бы не так! Бди-

тельная система по-прежнему предупреждает нас об опасности при каждой попытке открыть этот документ. Оказывается, при удалении кода дополнительные блоки, выделенные файлу, отключаются лишь частично, и документу приходится сохранять «запах» макросов до конца своей жизни.

В последующих версиях Word программисты ликвидировали свою оплошность. Теперь защита от вирусов в макросах не срабатывает на произвольную строку символов. Можно убедиться в том, что произвольные символы, записанные в окно кода любого программного модуля, при сохранении файла из него удаляются. Теперь удаление программного кода сопровождается полным отключением от файла дополнительных «программных» блоков. Но система защиты не становится от этого эффективнее. Она по-прежнему срабатывает не на вредоносную сигнатуру, а только на несколько ключевых слов, обозначающих начало процедуры или функции. Можно отметить еще несколько особенностей этой защиты. Установить или снять «Защиту от вирусов в макросах» в Word 8.0 можно не только руками пользователя (точнее – щелчком мыши). Это

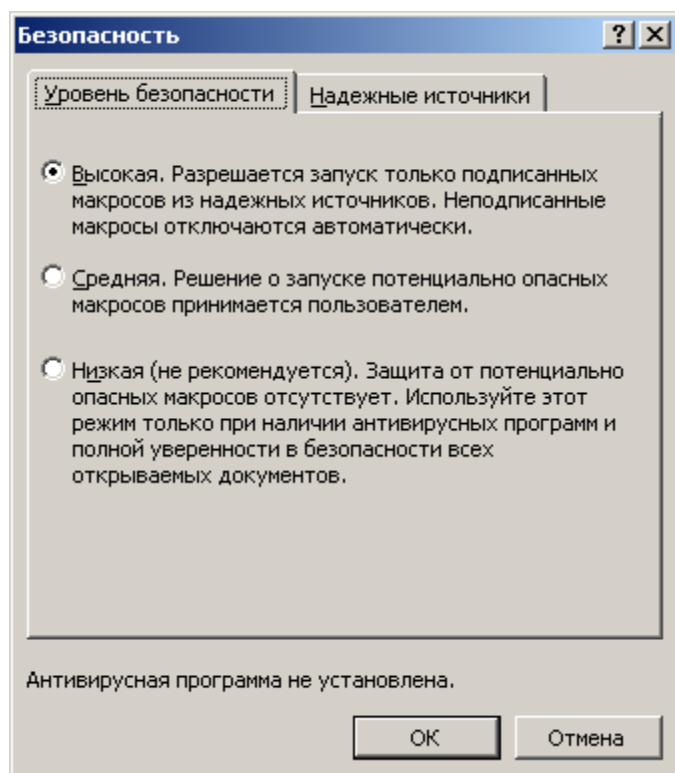


Рис. 3. Установка уровней защиты от макросов в Word 9.0 – 11.0

же действие можно выполнить в программе с помощью команды `Options.VirusProtection = True` (или `False`). Подобная команда часто встречалась в тексте макровирусов, которые могли по своему усмотрению программно включать и отключать эту защиту.

Начиная с Word 9.0 программисты предусмотрели не два, а три уровня защиты от макровирусов, причем их установка производится в отдельном окне «Безопасность» (рис. 3). Высокий уровень защиты отключает все макросы, низкий –

разрешает их выполнение, а средний приводит к выводу стандартного предупреждения пользователю. При этом изменить уровень защиты из программы с помощью объектов и методов Word уже невозможно – это можно делать только “Вручную”. Но защита от этого прочнее не стала. Изменить уровень защиты в программе по-прежнему можно, ведь все настройки хранятся в системном реестре, а возможности языка VBA легко позволяют манипулировать реестром – были бы только права доступа к соответствующему разделу. Две строки, приведенные ниже, показывают, как сбрасывается защита от вирусов в макросах в системном реестре. Указанные строки можно нередко встретить в тексте вредоносных программ:

```
System.PrivateProfileString("", HKCU\Software\Microsoft\Office\8.0\Word\Options", "EnableMacroVirusProtection")= 0
```

‘ так отключается защита от вирусов в макросах в Office 97

```
System.PrivateProfileString("", "HRCU\Software\Microsoft\Office\9.0\Word\Security", "Level")=1 ‘так устанавливается низкий
```

уровень защиты от макросов в Office 2000, XP, 2003.

Защита от макросов действует не только в отношении документов, но и при открытии шаблонов. Все рассматриваемые версии Word обнаруживают макросы в общих шаблонах и шаблонах, присоединенных к документам. В Word 8.0 программа «доверяет» глобальному шаблону и не подвергает его проверке. С одной стороны, это несомненная уязвимость, поскольку инфицирован-

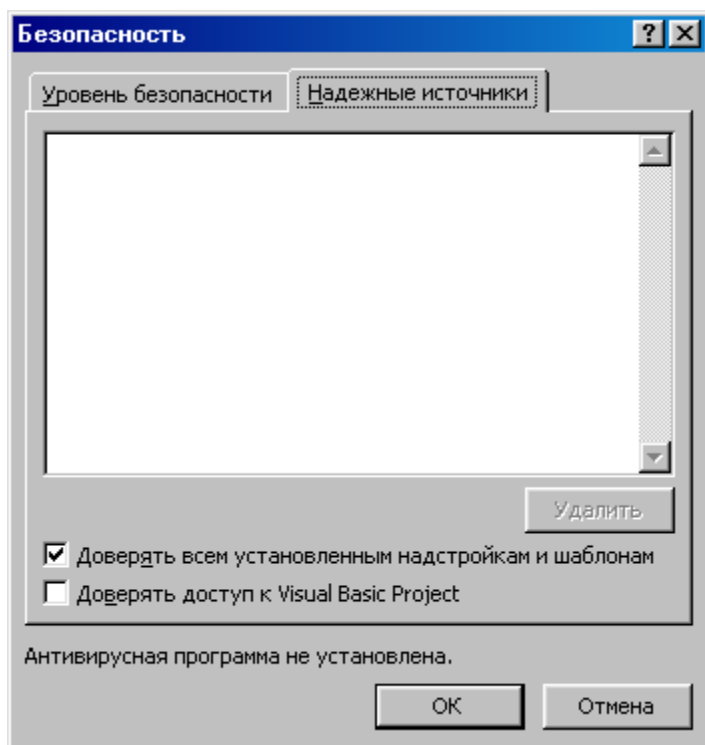


Рис.4. Информация о надежных источниках программного кода

ный шаблон может беспрепятственно реализовывать вредоносные функции и заражать документы. Но, с другой стороны, бесконечные предупреждения пользователя о присутствии макросов в Normal.dot приведут либо к полному отказу от использования макросов (стоило их для этого придумывать?), либо к снятию защиты от вирусов, что еще опаснее. Компромиссный вариант реализован в следующих версиях Word – текстовый процессор доверяет своему глобальному шаблону, если тот находится в определенном каталоге. Если этот каталог изменяется, защита от макросов будет фиксировать их присутствие в Normal.dot при открытии Word. Но доверительные отношения к шаблонам и надстройкам тоже определяются из установок диалогового окна «Безопасность» (рис. 4). Следует обратить внимание на то, что доверительные отношения могут быть установлены только ко всем установленным шаблонам и надстройкам: выборочная установка доверия не допускается.

Для защиты глобального шаблона предусмотрена еще одна мера. В меню «Сервис» – «Параметры» – «Сохранение» можно установить «Запрос на сохранение шаблона». Если в течение сеанса шаблон изменился, при завершении работы Word выдаст пользователю предупреждение с запросом, следует ли сохранить эти изменения. По замыслу разработчиков, если пользователь внес эти изменения сам, он согласится их сохранить, если нет – предпримет меры по выяснению причин изменений. Но если глобальный шаблон инфицируется, проникший в него вредоносный макрос в числе первых мер отключает такой запрос. Строка **Options.SaveNormalPrompt = False** является обычной для большинства макровирусов.

Еще одной, и довольно результативной, мерой защиты глобального шаблона от инфицирования является установка пароля на программный проект Normal.dot (вкладка Protection «Свойств» проекта). Фиксация установленного пароля на просмотр программного кода в глобальном шаблоне будет произведена только после завершения сеанса работы в Word.

По мнению разработчиков основной целью такой защиты являлся режим ограничения просмотра и редактирования программного кода от пользователя, который может скопировать код и бесплатно использовать его в своих проектах, т.е. охрана авторских прав создателя программы. Однако проект при этом не шифруется, и программное содержимое документа или шаблона может быть просмотрено в ASCII-кодировке с использованием любого view'ера, например, в файловом менеджере Far. Так что своей цели программисты Microsoft не добились, но кое-что полезное сделали. Если в открываемом документе содержится

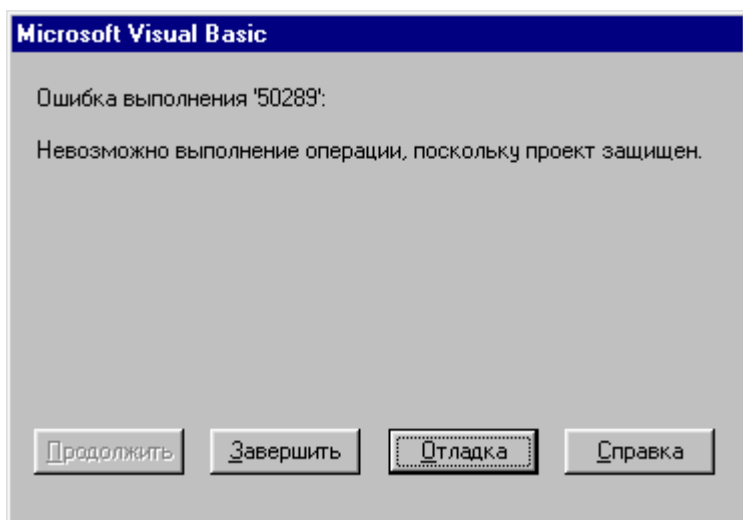


Рис. 5. Окно, выводимое при попытке инфицировать глобальный шаблон, закрытый паролем

код, пытающийся инфицировать глобальный шаблон, редактор VBE выводит окно предупреждений, изображенное на рис.5. При нажатии на кнопку «Отладка» пользователь откроет редактор VBE и ему будет показан фрагмент опасного кода, вызвавшего предупреждение.

Пусть не все пользователи разбираются в языках программирования, такой подход все же

более предпочтителен, чем расчет на интуицию пользователя.

Для пользователя установка пароля на проект тоже в определенной мере неудобна. Например, попытки записать собственный макрос, просмотреть уже имеющиеся макросы или открыть проект будут сопровождаться требованием

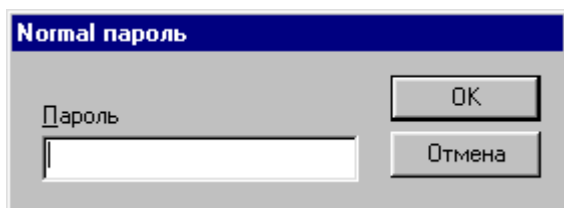


Рис. 6. Запрос пароля при доступе к защищенному проекту

ввести пароль (рис.6). Необходимо отметить, что установить пароль на программный проект можно только вручную (т.е. с помощью мыши). Инструкции, позволяющей сделать то же самое в тексте программы, в языке VBA не предусмотрено.

Начиная с версии 9.0 в Microsoft Word предусмотрен еще один заслон против вредоносных макросов, не препятствующий использованию полезных программ. Речь идет о цифровой подписи программного кода программистами, которым пользователь может доверять. Наличие цифровой подписи подтверждает, что документ с макросом принадлежит владельцу подписи и с момента его составления не изменялся. При модификации программного кода посторонним цифровая подпись пропадает. Поэтому макросы следует подписывать только после их тестирования и подготовки к распространению, поскольку при любом изменении исходного текста подписанного проекта макроса его цифровая подпись удаляется. Цифровая подпись отключает защиту от вирусов в макросах для документов, обладающих такой подписью.

Автор макроса, желающий предоставить свою программу для широкого использования, должен запустить исполняемый файл, генерирующий цифровую подпись. Этот файл носит имя selfcert.exe и постоянно размещается в том же каталоге, куда был инсталлирован winword.exe (например, c:\program files\microsoft office\office 10). Если этого файла нет, следует повторно запустить программу инсталляции Microsoft Office в режиме добавления компонентов проекта и выбрать для установки «Цифровую подпись для проектов VBA». При

запуске программы selfcert.exe она выводит окно (рис. 7). Следует обратить внимание на подсказку, выведенную в окне: «Этот тип сертификата не выполняет проверку подлинности». Поскольку самостоятельно созданный цифровой сертификат не был выдан официальным центром

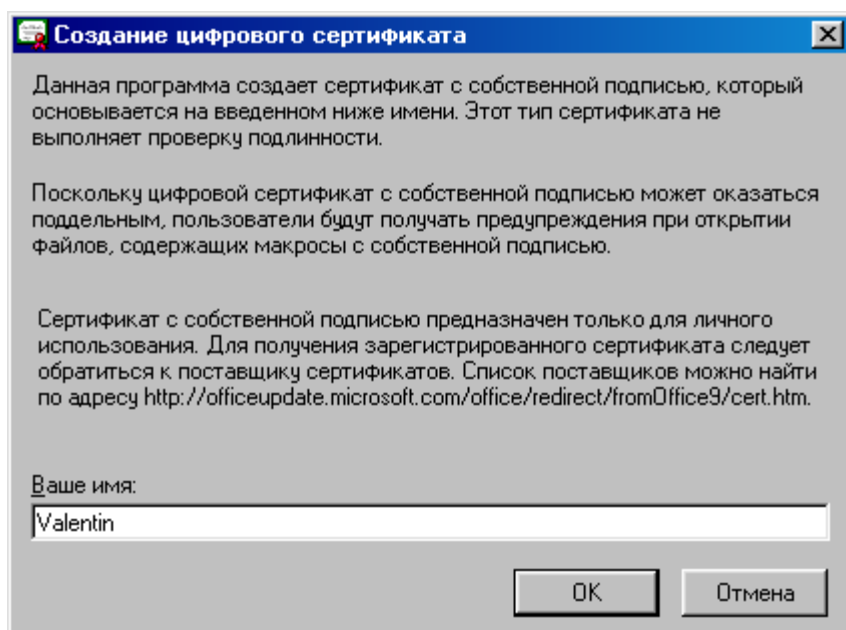


Рис. 7. Генерация цифровой подписи

сертификации, подписанные с его использованием проекты макросов называют проектами с автоподписью. Сертификаты, созданные пользователем самостоятельно, рассматриваются как неподтвержденные и при высоком или среднем уровне безопасности приводят к выводу предупреждения системы безопасности. Действующие в организации правила использования средств цифровой подписи Microsoft Office могут запрещать использование таких сертификатов, а также запуск макросов с автоподписью другими пользователями.

Если пользователь часто использует макросы от известного ему автора или фирмы, он может внести их в список надежных источников (см. окно «Безопасность»). Word, открывая документ с такими макросами, проверяет наличие цифровой подписи, их источники, и при совпадении разрешает открывать документ и запускать макросы без вывода предупреждений об опасности. Следует обратить внимание на то, что эта защита не определяет присутствия в документе событийных макросов. Цифровая подпись устанавливается на программный проект целиком, и нет возможности отделить доверенные модули, процедуры и функции друг от друга.

Защита, построенная на человеческой доверчивости, всегда уязвима. Но, с другой стороны, жить и работать не доверяя никому, тоже нельзя. В этом случае придется научиться все делать самостоятельно (в этом случае пользователь по крайней мере должен доверять своему профессионализму).

Антивирусной защите в большинстве случаев следует доверять, особенно если пользователь не очень хорошо разбирается в сложной структуре Word. Опытному пользователю можно посоветовать быть более подозрительным, особенно если часто приходится работать с документами, созданными другими авторами, либо ранее использовавшимися на других компьютерах. Вредоносные программы, кроме инфицирования, могут проявлять себя самыми разнообразными действиями, какие только могут прийти в голову вирусмейкерам. Например, при сохранении документов файлы могут сохраняться «пустыми» или шифроваться со случайными паролями, которые впоследствии невозможно подобрать. Наиболее часто встречающихся в тексте слова вредоносная программа может

заменить словами нецензурного содержания. Пункты меню и командные панели могут измениться до неузнаваемости, и текстовый процессор станет неуправляемым. Наконец, внедренная программа может не выполнять ничего откровенно вредоносного, но она будет выбирать из редактируемого текста фразы с интересным содержанием и сохранять их для последующей передачи злоумышленнику.

Косвенными признаками исполнения вредоносного кода из документов или шаблонов следует считать:

- Замедление или неестественное выполнение операций при работе с файлами, текстом, таблицами, рисунками.
- Выдачу сообщений об ошибочных действиях, к которым пользователь не имеет никакого отношения.
- Попытки в ходе редактирования документа установить сетевое соединение и прочее.

У вредоносных программ имеются и статические демаскирующие признаки, позволяющие обнаружить опасный код на этапе его хранения. В этом немалая роль принадлежит пользователю. Интерпретируемый код макросов представляет собой осмысленный текст на английском языке и обнаружить его можно визуально, если воспользоваться программами, позволяющими полностью просматривать документ (в файлах формата Word нельзя заранее сказать, где именно должен размещаться модуль программы). 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 изстораживающих симптомов инфицирования. Увидеть эти имена можно в диалоговых окнах Word или редактора VBE (см рис.8).

Диалоговые окна «Макрос», «Шаблоны и надстройки», «Организатор», дочерние окна редактора VBE позволяют пользователю обнаружить присутствие присоединенных шаблонов, модулей и процедур.

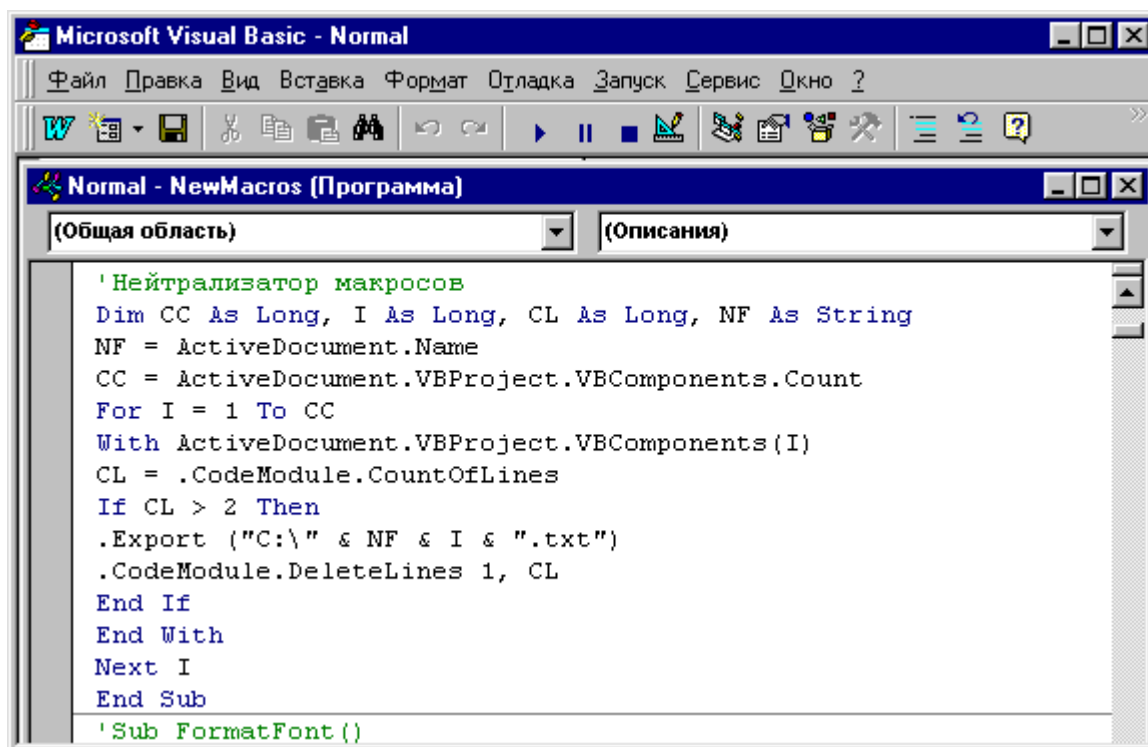


Рис.8. Окно кода редактора Visual Basic

- Окно «Макрос» (рис.9). В окне видимы все открытые процедуры всех открытых модулей документа, Normal.dot и присоединенных шаблонов.
- Окно «Расположение» меню «Сервис – Параметры». В окне просматриваются полные маршруты (Path) к глобальному шаблону Normal.dot и папке автозагрузки.

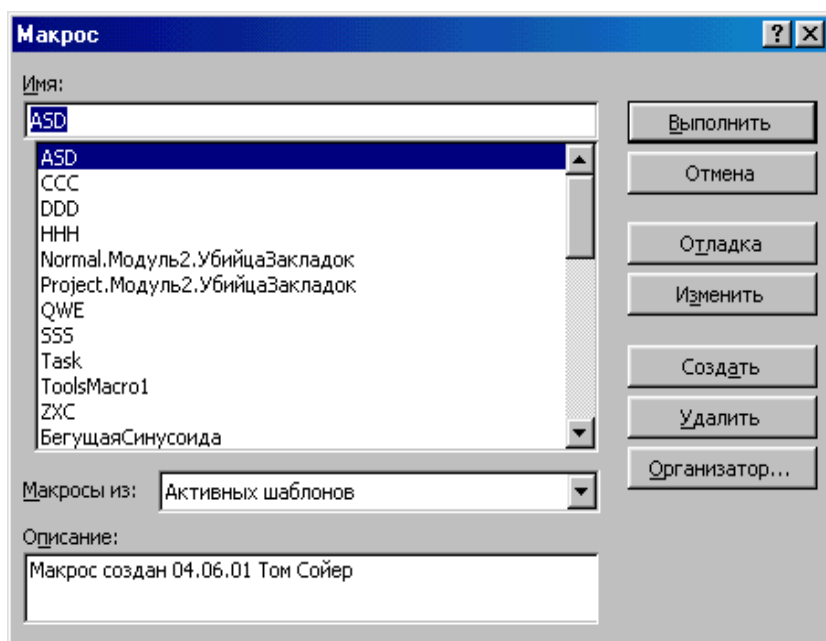


Рис.9. Диалоговое окно «Макрос»

- Окно «Организатор». Видны названия всех модулей и форм, размещенных в глобальном шаблоне Normal.dot и активном документе.
- Окно «Шаблоны и надстройки». Видны наименования шаблонов и надстроек, под-

ключенных к глобальному шаблону.

- Окна редактора VBE. Видны названия проектов, модулей и процедур, а также интерпретируемые коды макросов.

Чтобы не допустить обнаружения вредоносного макроса со стороны пользователя, вирмейкеры используют разнообразные способы, препятствующие просмотру программных проектов, модулей, макросов. Одним из наиболее распространенных способов противодействия любопытному пользователю является блокирование опасных элементов управления, позволяющих получить доступ к диалоговым окнам. Первые поколения макровирусов с этой целью отключали командные кнопки или опасные пункты меню. При этом элемент управления можно полностью убрать с командной панели или меню, либо сделать «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 (BuildKey-  
Code(wdKeyF8, wdKeyAlt)).Disable
```

Еще один из весьма распространенных способов маскировки вредоносного кода является классическим методом отвлекающих действий. Он заключа-

ется в генерации ошибки в ответ на действия пользователя. Например, пользователь пытается открыть диалоговое окно «Макрос» или редактор Visual Basic. Но нажатие на командную кнопку или выбор соответствующего пункта меню приводит к такому

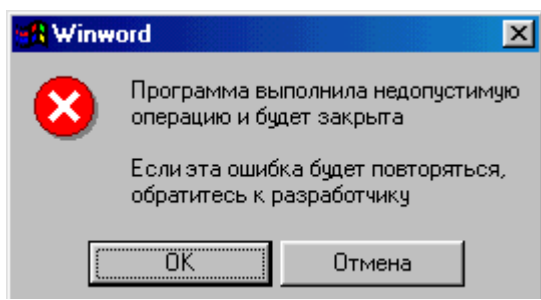


Рис. 10. Сообщение об ошибке

сообщению (рис.10). А реализуется это в тексте вредоносной программы 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
```

Более сложным вариантом является использование «подставных» диалоговых окон, построенных на основе пользовательских форм. При вызове диалогового окна отображается подставное окно, в котором никаких признаков наличия «чужих» макросов обнаружить нельзя. Но, с другой стороны, окно с полными функциями запрограммировать достаточно сложно – в окне «Макрос» имеется три окна редактирования, комбинированный список, полоса прокрутки и семь командных кнопок, причем каждый элемент окна связан с соответствующей процедурой. Поэтому чаще всего во вредоносных программах предлагается не полнофункциональное диалоговое окно, а муляж, причем в этом окне пользователь не сможет увидеть и тех макросов, которые он сам записал. Щелчок курсором по любой командной кнопке этой формы, как правило, сопровождается выводом сообщения о системной ошибке, так что этот способ скрывтия немногим отличается от описанного выше.

Для опытных пользователей Word, искушенных в программировании на VBA, можно порекомендовать написать собственный макрос, защищающий от вредоносного кода в открываемых для просмотра и редактирования документах. Прежде всего надо создать событийную процедуру, которая будет называться FileOpen(). Эта процедура заменит собой программу, которая обычно выполняется текстовым процессором при открытии документа через меню «Файл»-«Открыть». Можно предусмотреть на командной панели дополнительную кнопку и связать ее с выполнением защитной процедуры. Тогда для своих, заведомо безопасных документов пользователь будет использовать обычную процедуру открытия документа, а для подозрительных – отдельную кнопку.

Созданная процедура в первую очередь должна содержать команду, отключающую на сеанс все автоматически исполняемые макросы. Следующим шагом является сканирование всех программных модулей открываемого документа (если они есть) и копирование их содержимого в буфер памяти. Наконец, в записанном буфере производится поиск заведомо опасных сигнатур, присутствующих в инструкциях инфицирования или обращающихся на запись к файловой системе или системному реестру. Вот далеко не полный перечень этих сигнатур (при необходимости количество обязательных символов в сигнатурах можно еще сократить): «CodeModule», «OrganizerCopy», «OrganizerDelety», «Kill», «RegWrite», «CreateObject».

2. ЛАБОРАТОРНАЯ РАБОТА № 1

Цель работы:

1. Исследовать эффективность защитных механизмов текстового процессора Microsoft Word.
2. Детально разобраться с алгоритмами инфицирования, используемыми вредоносными макросами.
3. Исследовать причины образования технологического информационного «мусора», возникающего при обработке конфиденциальных документов в программной среде Word.
4. Разработать собственный вариант антивирусной программы, позволяющей обнаруживать и нейтрализовать вредоносные макросы.

Для выполнения лабораторной работы можно использовать любую версию Microsoft Word – от 8.0 до 11.0. При использовании 8-й версии (Microsoft Office 97) можно наблюдать некоторые из уязвимостей, которые в последующих версиях были устранены. Версия операционной системы Windows тоже не особенно критична, но в системах семейства Windows 9* явно можно отслеживать накопление «технологического» мусора за счет передачи файлам неочищенных блоков виртуальной памяти.

При инсталляции офисных приложений необходимо специально указать на необходимость установки справочной системы для среды программирования Visual Basic for Application – по умолчанию она не устанавливается.

2.1. Исследование механизмов образования информационного «технологического мусора»

- Установите в меню Word «Сервис» – «Параметры» – «Сохранение» временной интервал автосохранения документа, равный 1 минуте. В этом же окне установите по выбору один из режимов «Всегда создавать резервную копию» или «Разрешить быстрое сохранение».
- Создайте новый документ и сохраните его с произвольным именем.

- Поработайте в режиме свободного набора текста не менее 5 минут (например, откройте один из справочных файлов Windows и перепишите (либо построчно скопируйте) в свой документ несколько абзацев). Несколько раз сохраните документ «вручную».
- Проверьте объем созданного документа и сопоставьте его с количеством введенных символов (количество символов в тексте можно узнать в «Свойствах» документа: «Файл» – «Свойства» – «Статистика»). На каждый символ в кодировке UNICODE требуется 2 байта. Чем вызвана несоразмерность объема файла с его содержанием?
- Найдите все временные файлы, созданные приложением во время работы над документом. Можете ли вы идентифицировать временные файлы Word по их именам? Можно ли узнать, каким именно документам соответствуют те или иные временные файлы? Просмотрите один из временных файлов с помощью файлового менеджера Far в кодировке UNICODE (F3 – просмотр файла, Shift+F8 – изменение кодировки символов).
- Закройте созданный документ и приложение Word. Какие из временных файлов при этом были логически удалены? Можете ли вы при необходимости восстановить их? Каким образом?
- Вновь откройте созданный файл в Word. Продолжайте работать с ним, добавляя новые фрагменты (желательно большого размера), изменяя и удаляя прежние. После каждого добавления сохраняйте файл. После 7-8 минут работы оцените размер файла и сопоставьте его с реальным объемом текста.
- Сделайте выводы в отношении режимов «Быстрого сохранения» и создания резервных копий.
- Попробуйте определить, какой из резервных (временных) файлов используется для восстановления документа в случае сбоя в работе. Для этого принудительно завершите сеанс работы Word из системы. Это можно сделать, нажав комбинацию клавиш Ctrl+Alt+Del, выделив в выведенном окне программу Microsoft Word и выбрав кнопку «Завершить задачу». После очередного запуска Word автоматически загрузит «поврежденный» документ.

2.2. Определение приоритетов в выполнении автоматически исполняемых макросов

Для выполнения задания необходимо создать автоматически запускаемую событийную процедуру, и расположить ее в одном из документов и в различных шаблонах. Пользуясь модальными свойствами окна сообщения, можно отслеживать очередность активизации событийных процедур в файлах формата Microsoft Word.

- Установите низкий уровень защиты от вирусов в макросах. Для этого откройте окно «Безопасность» («Сервис» – «Макрос» – «Безопасность» – «Уровень безопасности») и установите требуемый уровень защиты, при котором вы сможете свободно создавать и использовать макросы без предупреждающих сообщений о возможной опасности. В таком режиме можно работать только с теми файлами, которые вы сами создаете и редактируете.
- Откройте диалоговое окно «Макрос» (меню «Сервис» – «Макрос» – «Макросы»). Для этого же можно нажать командную кнопку «Выполнить макрос» на панели инструментов Visual Basic. В открывшемся окне ввести имя процедуры AutoOpen и нажать кнопку «Создать». Запускается редактор VBA и в нем открывается окно текста программ (Code Window). Тем самым вы уже создали событийную процедуру, которая должна автоматически выполняться при открытии в Word любого документа. Между строками Sub ... – End Sub вставьте строку вызова окна сообщений, как это показано ниже.

```
Sub AutoOpen()  
MsgBox "Работает процедура глобального шаблона"  
End Sub
```

- Создайте новый документ. Повторно откройте окно «Макрос», найдите на нем командную кнопку «Организатор» и нажмите ее. Word выведет новое диалоговое окно, с помощью которого можно копировать макросы из шаблона в документ и обратно. Обратите внимание на то, что копируется не одна процедура, а весь модуль NewMacros. Для этого выделите NewMacros и на-

жмите кнопку «Копировать». Аналогичный модуль кода будет создан в новом документе.

- Измените сообщение, которое должно выводиться при запуске автомакроса из документа. Для этого откройте окно редактора VBA («Сервис» – «Макрос» – «Редактор Visual Basic»), найдите или откройте в нем дочернее окно менеджера проектов («Проект» – «Project»). Далее выберите название проекта, соответствующее имени документа, щелкните мышью по значку «+» слева от названия и откройте программный модуль NewMacros. В окне текста программ измените текст сообщения, как это указано ниже:

```
Sub AutoOpen()  
MsgBox "Работает процедура документа"  
End Sub
```

Можно предложить альтернативный вариант копирования программного кода. Для этого, находясь в редакторе VBA, следует открыть два окна текста программ: шаблона и документа, выделить текст программы в одном окне, скопировать его в буфер, а затем вставить в другое окно.

- Сохраните документ под именем «Документ с сообщением» в папке «Мои документы» и закройте его.
- Создайте второй документ. По аналогии с предыдущими пунктами скопируйте в него автомакрос с сообщением, а затем измените выводимое сообщение на приведенное ниже:

```
Sub AutoOpen()  
MsgBox "Работает процедура общего шаблона"  
End Sub
```

- Сохраните этот документ как общий шаблон («Файл» – «Сохранить как»: Тип файла – шаблон документа, Имя файла – «Общий шаблон с сообщением»). Для того чтобы сделать общий шаблон загружаемым автоматически, необходимо сохранить его в папке StartUp. Путь к этой папке можно узнать, открыв диалоговое окно «Сервис» – «Параметры» – «Расположение», строка «автозагружаемые». После сохранения закройте созданный общий шаблон.

- Откройте диалоговое окно «Сервис» – «Шаблоны и надстройки». С помощью кнопки «Добавить» найдите местонахождение сохраненного общего шаблона и выберите его. Убедитесь, что имя шаблона появилось в среднем окне «Общие шаблоны и надстройки» и отмечено галочкой. Щелкните кнопку «ОК» и окно закроется.
- Создайте третий документ. По аналогии с вышеприведенной методикой скопируйте в него автомакрос и измените текст сообщения следующим образом:

```
Sub AutoOpen()  
MsgBox "Работает процедура присоединенного шаблона"  
End Sub
```

- Сохраните третий документ в папке «Мои документы» с именем «Присоединенный шаблон» и типом файла «Шаблон документа», после чего закройте его.
- Откройте ранее сохраненный «Документ с сообщением». Какое из сообщений будет выведено на экран?
- С помощью меню «Сервис» – «Шаблоны и надстройки» откройте диалоговое окно «Шаблоны и надстройки» и с помощью кнопки «Присоединить» найдите ранее сохраненный «Присоединенный шаблон». Сохраните изменения в документе и закройте его вместе с приложением. Теперь у вас имеются три типа шаблонов и один документ, и в каждом из них размещена одна и та же автоматически запускаемая процедура. Поскольку окно сообщений MsgBox обладает модальностью, одновременно может быть выведено только одно сообщение. По очередности вывода сообщений можно узнать, какой из вышеупомянутых файлов имеет приоритет в исполнении кода.
- Найдите в папке «Мои документы» файл с именем «Документ с сообщением» и откройте его. Какое сообщение будет выведено на экран? Что можно сказать о макросах, хранимых в других файлах?
- Откройте еще какой-либо документ. Если нет документов формата Microsoft Word, можно открыть документ другого формата. Какое сообщение вы получаете в этом случае?

- Закройте файл «Документ с сообщением». Не закрывая приложения, вновь откройте этот же документ, удерживая нажатой левую кнопку Shift клавиатуры. Какое сообщение выведено на экран? Почему?
- Войдите в редактор VBE и удалите программный код макроса AutoOpen() из файла «Документ с сообщением». Сохраните изменения в документе и закройте его.
- Вновь откройте «Документ с сообщением». Какое сообщение вы получили на этот раз? Укажите в отчете приоритет в выполнении автоматически запускаемых макросов из файлов Microsoft Word. Усматриваете ли вы в заданном приоритете угрозу инфицирования глобального шаблона через уже зараженный документ? Закройте «Документ с сообщением».
- В редакторе VBE откройте окно кода NewMacros глобального шаблона Normal.dot и создайте новую процедуру AutoExec() следующего вида:

```
Sub AutoExec()
```

```
WordBasic.DisableAutoMacros ‘эта команда действует в течение всего сеанса
```

```
End Sub
```

- Закройте и вновь откройте Microsoft Word. Сопровождается ли открытие текстового процессора какими-либо сообщениями?
- Откройте файл «Документ с сообщением». Удалось ли вам нейтрализовать автомакрос в документе или присоединенном к нему шаблоне? Закройте файл «Документ с сообщением».
- Удалите процедуру AutoExec() в глобальном шаблоне. Создайте в модуле NewMacros глобального шаблона событийную процедуру, которая будет обрабатывать события, связанные с открытием документов.

```
Sub FileOpen()
```

```
WordBasic.DisableAutoMacros
```

```
MsgBox «Процедура отключения автоматических макросов сработала»
```

```
Dialogs(wdDialogFileOpen) .Show
```

```
End Sub
```


- Закройте и вновь откройте Microsoft Word. В очередной раз откройте файл «Документ с сообщением» (обратите внимание на то, что событийная процедура FileOpen() не работает при открытии недавно открываемых файлов, помеченных в нижних строчках меню «Файл». Какое сообщение было выведено на этот раз? Удалось ли вам нейтрализовать автоматические макросы в документе или присоединенном к нему шаблоне? Закройте «Документ с сообщением»? Удалите или прокомментируйте код процедуры FileOpen() в глобальном шаблоне. Еще несколько раз откройте и закройте «Документ с сообщением». Срабатывают ли автоматические макросы в документе? Почему?
- Если на компьютере установлен файловый менеджер Far, просмотрите файл «Документа с сообщением» в различных кодировках (просмотр F3, переключение кодировок в режиме просмотра – Shift+F8) до и после закрытия паролем. Найдите в теле файла программный код.
- Установите парольную защиту на программный код «Документа с сообщением». Для этого щелкните правой кнопкой мыши по заголовку проекта документа «Project(Документ с сообщением)». В контекстном меню выберите «Свойства проекта». В появившемся диалоговом окне выберите закладку «Защита» (Protection), установите защиту на просмотр кода и дважды введите какой-либо простой пароль. После этого закройте редактор VBE, сохраните и закройте документ.
- Откройте «Документ с сообщением» с установленной парольной защитой на просмотр кода. Срабатывает ли код событийных процедур документа? Попробуйте открыть окна кода модулей ThisDocument и NewMacros. Опишите результат. Происходит ли шифрование кода процедур при установке пароля на проект?
- Открыв диалоговое окно «Шаблоны и настройки», отсоедините от документа «Присоединенный шаблон» и отключите «Общий шаблон». Удалите эти шаблоны из каталогов, в которых они располагаются. Закройте файл «Документ с сообщением».

- Сделайте выводы относительно исследования приоритетов в запуске автомакросов и возможностей нейтрализации опасного программного кода в открываемых документах Word.
- Удалите процедуры Sub FileOpen() и Sub AutoExec() из глобального шаблона.

2.3. Исследование приоритетов в запуске событийных процедур, связанных с открытием документов

- Измените текст сообщения в процедуре AutoOpen() в программном модуле NewMacros() глобального шаблона:

```
Sub AutoOpen()  
MsgBox "Работает процедура NewMacros шаблона"  
End Sub
```

- Откройте модуль ThisDocument глобального шаблона и введите в окно кода следующую процедуру:

```
Private Sub Document_Open()  
MsgBox "Работает процедура ThisDocument шаблона"  
End Sub
```

- Закройте и вновь откройте Microsoft Word. Сопровождается ли открытие текстового процессора какими-либо сообщениями?
- Откройте любой из документов (кроме файла «Документ с сообщением»). Какими сообщениями Word сопровождает открытие документа? В какой последовательности? Закройте документ.
- Откройте «Документ с сообщением». Какая из процедур выполняется?
- Войдите в редактор VBE, найдите модуль ThisDocument документа и вставьте в него следующую процедуру:

```
Private Sub Document_Open()  
MsgBox "Работает процедура ThisDocument документа"  
End Sub
```

- Сохраните, закройте и вновь откройте «Документ с сообщением». Отрадите в отчете, в какой последовательности выводятся сообщения.
- Сделайте выводы, отразите их в отчете.

2.4. Исследование механизма вирусного заражения документов и шаблонов Word

Существуют по меньшей мере три способа копирования или переноса программного кода из документа в документ, из шаблона в шаблон, или из шаблона в документ и обратно. Исследуйте их.

- Установите низкий уровень защиты от вирусов в макросах.
- Найдите каталог, в котором постоянно хранится глобальный шаблон. Для этого откройте меню «Сервис» – «Параметры» – «Расположение» и прочитайте строку «Шаблоны пользователя». Удалите или переименуйте используемый глобальный шаблон Normal.dot (это можно сделать только при закрытом приложении Word), после чего запустите и вновь закройте Word (без закрытия приложения глобальный шаблон будут недоступен). Убедитесь в появлении нового файла с названием Normal.dot.
- Перенесите нижеприведенный фрагмент кода в программный модуль ThisDocument глобального шаблона. Используя встроенную справочную систему редактора VBA, проведите анализ приведенного ниже фрагмента кода и разберитесь, каким образом происходит вирусное заражение. Что является источником инфицирования – документ или шаблон? Где должен располагаться данный фрагмент? Модифицируйте макроскод таким образом, чтобы инфицирование происходило в другую сторону. Напишите процедуру, производящую вирусное заражение и документов, и глобального шаблона. Проверьте ее в действии. *После окончания проверки удалите инфицированный документ и глобальный шаблон.*

```
Private Sub Document_Open()  
Dim AD, NT As Object  
Set AD = ActiveDocument.VBProject.VBComponents(1). Code-  
Module  
Set NT = NormalTemplate.VBProject.VBComponents(1). Code-  
Module  
AD.InsertLines 1, NT.Lines(1, NT.CountOfLines)  
End Sub
```

- Проведите анализ второго варианта вирусного заражения. Для этого запишите макрос, сопровождающий процедуру копирования программных модулей типа NewMacros из документов в шаблоны и обратно. Выберите в меню «Сервис» – «Макрос» – «Начать запись», назначьте макросу произвольное имя, а затем уже в режиме записи откройте диалоговое окно «Макрос», выберите кнопку «Организатор» и скопируйте модуль NewMacros из шаблона в документ. Закройте окно «Организатор» и остановите запись. Войдите в редактор VBA и посмотрите записанный код. Измените название макроса на Sub FileSave(). Данная событийная процедура будет инфицировать документ при его сохранении. По аналогии создайте еще одну событийную процедуру (назовите ее FileOpen()), которая будет заражать шаблон при открытии уже инфицированного документа. Опробуйте обе процедуры в действии.
- Произведите копирование путем экспорта-импорта программного кода с использованием текстовых файлов (им можно присваивать любые имена и расширения). Первая строка кода, приведенного ниже, экспортирует программное содержимое встроенного модуля ThisDocument (он всегда обозначается первым номером) из глобального шаблона в текстовый файл, расположенный на логическом разделе C:\. Вторая строка вставляет содержимое этого же текстового файла в программный модуль NewMacros активного документа. Заключите эти строки в событийную процедуру, связанную с открытием или сохранением документа, и проверьте, как происходит инфицирование. Обратите внимание на содержимое созданного текстового файла – в нем кроме кода должен содержаться заголовок программного модуля, который считается необязательным.

```
NormalTemplate.VBProject.VBComponents(1).Export ("C:\ABCD.txt")
```

```
ActiveDocument.VBProject.VBComponents(2).CodeModule. AddFromFile "C:\ABCD.txt"
```

- Сравните между собой три приведенных способа вирусного инфицирования (все в равной степени встречаются в тексте вредоносных макросов). Каковы достоинства и недостатки каждого из способов? Выделите в приведенных фрагментах характерную сигнатуру, позволяющую обнаруживать опасный код в исследуемых макросах. Отрадите результаты исследований в отчете.

2.5. Исследование механизма защиты от вредоносных макросов

- Установите средний уровень безопасности защиты от вредоносных макросов (меню «Сервис» – «Макрос» – «Безопасность»). Теперь при обнаружении макрокда в документе или шаблоне система будет выводить пользователю запрос.
- Создайте новый документ и сохраните его в папке «Мои документы» под именем «Документ с макросом». Закройте документ и оцените его объем в байтах. Сколько в данном пустом файле 512-байтных блоков?
- Откройте сохраненный документ. Сопровождается ли его открытие предупреждением о макросах?
- Откройте редактор VBE (Alt-F11), затем его дочернее окно «Проект», «щелкните» дважды левой кнопкой мыши по строке Project(Документ с макросом), а затем по выпавшей строке ThisDocument. Откроется окно кода встроенного программного модуля документа, в которое следует ввести с клавиатуры подряд несколько символов (например, «12345»). Закройте окно редактора VBA, сохраните и закройте документ. Изменился ли размер сохраненного файла? Почему?
- Вновь откройте «Документ с макросом». Сопровождается ли его открытие сообщением об опасности? Откройте редактор VBA и посмотрите, что случилось с ранее записанной в модуль кода строкой.
- Запишите в окно кода модуля ThisDocument этого документа начальную и заключительную строчки процедуры, например:

Private Sub AAA()

End Sub

- Закройте окно редактора, сохраните и закройте документ. Проверьте, изменился ли его размер. Сколько 512-байтных блоков добавилось к документу?
- Вновь откройте «Документ с макросом». Сработала ли защита от вредоносных макросов?
- Откройте редактор VBE и полностью удалите каркас процедуры из модуля ThisDocument. Закройте редактор, сохраните и закройте документ. На сколько блоков уменьшился его размер?
- Вновь откройте документ. Сопровождается ли его открытие предупреждением об опасности?
- Попробуйте установить минимальную строку в модуле кода, на которую «срабатывает» механизм безопасности. Запишите в модуль кода документа слово “Sub”, после чего сохраните, закройте и вновь откройте документ. Если защита не сработала и размер файла не увеличился, добавьте имя процедуры, например «Sub A». Зафиксируйте момент, когда защита отнесется к внесенной записи как к полноценному и опасному макросу.
- Аналогичную проверку проведите в отношении ключевого слова Function (функция)
- Найдите раздел системного реестра, в котором располагается параметр защиты от вредоносных макросов. Методом проб установите соответствие между уровнем защиты, установленным в окне «Безопасность», и значением параметра.
- Доверяет ли Word собственному глобальному шаблону и общим шаблонам на предмет возможности присутствия в них вредоносного кода? Соотносится ли такое доверие с именем каталога, в котором шаблоны по умолчанию размещаются? Попробуйте переместить Normal.dot в другой каталог (предварительно изменив его расположение («Сервис» – «Параметры» – «Расположение»))

- Можно ли считать надежной реализованную защиту от вирусов в макросах? Могут ли злоумышленники использовать что-либо для ее обхода?

2.6. Блокирование просмотра программного кода, реализованного в некоторых макровирусах

- Создайте новый документ. Откройте окно программного модуля ThisDocument этого документа и создайте там две событийные процедуры:

Sub ToolsMacro()

‘ Открытие диалогового окна “Макрос”

End Sub

Sub ViewVBCode()

‘ Вход в интегрированную среду разработки VBE

End Sub

- Попробуйте открыть диалоговое окно «Макрос», либо войти в редактор VBA. Почему ваши попытки не удаются? Подобные способы защиты часто используются в текстах вредоносных макросов для того, чтобы пользователь не мог визуально проверить наличие постороннего программного кода. Попробуйте обойти эту блокировку, не закрывая документ. Результаты отразите в отчете.
- Закройте созданный документ без сохранения.

2.7. Разработка процедуры для блокирования и нейтрализации вредоносного макрокда

На основании проведенных исследований напишите текст макроса, нейтрализующего вредоносный программный код в открываемых для редактирования документах. Необходимо удовлетворить следующим требованиям:

- Макрос должен быть пригоден для использования экспертом-вирусологом при исследовании новых вредоносных программ.

- Макрос должен обеспечить полную безопасность при открытии любого инфицированного документа при установленном низком уровне защиты от вредоносного кода.
- Код, содержащийся в макросе, должен располагаться в глобальном шаблоне. В его функции входит обнаружение макросов в документе, нейтрализация автоматически выполняемых процедур, выдача предупреждения пользователю с выводом текста опасного кода. По решению пользователя вредоносный программный код должен вырезаться из открываемого документа и сохраняться в текстовом файле.

Завершите работу и убедитесь, что все созданные вами файлы и шаблоны удалены!

2.8. Контрольные вопросы

1. Особенности формата документов и шаблонов Word.
2. Основные причины образования в среде Word информационного «технологического мусора».
3. Организационные и технологические способы защиты конфиденциальной информации, обрабатываемой в среде Word, от случайного распространения.
4. Размещение интерпретируемого программного кода в документах и шаблонах Word.
5. Событийные процедуры и их использование во вредоносных макросах.
6. Приоритеты исполнения событийных процедур, связанных с документами Word.
7. Основные механизмы вирусного инфицирования документов и шаблонов Word.
8. Реализация доверительных отношений к макросам, написанным другими авторами.
9. Защита от вирусов в макросах в различных версиях Word.
10. Защита программных проектов в документах и шаблонах Word.

11. Сравнительная эффективность средств программной защиты от внедрения и запуска вредоносных макросов.
12. Возможности визуального обнаружения вредоносного программного кода в программной среде и документах Word.

3. ИССЛЕДОВАНИЕ ЗАЩИТНЫХ МЕХАНИЗМОВ БРАУЗЕРА MICROSOFT INTERNET EXPLORER

3.1. Алгоритмы функционирования браузера Internet Explorer

Ввиду широкой распространенности и известности Web-технологий возможность внедрения и запуска кода из гипертекстовых документов является объектом изучения хакерами всего мира. Обнаруженные уязвимости очень быстро становятся известными администраторам и пользователям и достаточно своевременно устраняются. В настоящее время издано немало книг и статей, описывающих атаки на Web-клиентов [1,3,11,13,16,17].

Рассмотрим, как происходит процесс запроса, получения и обработки документов в .html-формате со стороны браузера Microsoft Internet Explorer 5.0 в операционных системах Windows 98, Me и NT 5.0. Основное внимание при этом уделим механизмам гиперссылок, так как благодаря им становится возможным копирование файла на локальный компьютер и его открытие с целью чтения, интерпретации или запуска.

Клиентский запрос обычно инициируется активным действием пользователя (щелчком мыши или вводом адреса нужного ресурса в адресной строке браузера). При этом браузер посылает на Web-сервер запрос с указанием URL (Uniform Resource Locator - унифицированный адрес ресурса). URL обычно состоит из трех частей: протокола для взаимодействия с сервером, имени сервера и имени ресурса на этом сервере. Синтаксис URL выглядит следующим образом:

prefix://host.domain[:port]/path/filename

Номер порта обычно пропускается. Префикс указывает на протокол или расположение ресурса. Он может принимать одно из значений: telnet, news, wais, gopher, ftp, http или file. Протоколы и службы telnet, news, wais, gopher используются редко и рассматриваться не будут. Большей частью нас будут интересовать префиксы http и file. Http указывает на расположение ресурса на Web-

сервере, file – на расположение ресурса на локальном компьютере (на том же, где расположен браузер) или на ftp-сервере с анонимным доступом.

Если имя сервера содержит его IP-адрес, например `http://10.187.56.3/abcd.html`, то соединение клиента и сервера происходит непосредственно путем установления TCP-сеанса. При использовании доменного имени вначале происходит его преобразование в IP-адрес путем обращения к DNS-серверу. Логические анализаторы адресной строки Web-серверов и браузеров позволяют использовать разнообразный синтаксис. Так, при обращении к файловой системе своего компьютера IP-адрес может не указываться, например: `file://c:/windows/calc.exe`. В полном имени файла можно указывать либо прямые, либо обратные слешы, причем после префикса file можно указывать двойной или тройной прямой слеш. В некоторых случаях префикс file может вообще не указываться.

Web-страница состоит из контейнерного ресурса (html-файла), который может включать в себя встроенные ресурсы типа изображения (статического или динамического) или звука. Загрузка Web-страницы предусматривает возможность отдельной передачи контейнера и каждого из встроенных ресурсов, причем они вместе или по отдельности могут размещаться на отдельных серверах. Язык HTML предусматривает много тэгов и параметров, в которых можно указать URL.

Первый, и наиболее очевидный способ состоит в создании ссылки на имя ресурса. Если адрес будет указан не полностью, браузер использует в качестве первой части URL адрес последней страницы. Например:

```
<A HREF = "http://abc.def.ru/filename.html"> Ссылка </A>  
<A HREF = "File://C:/Windows/Calc.exe"> Ссылка </A>
```

‘кавычки в адресе не обязательны

Использование тэга `` предусматривает возможность перехода по гиперссылке (загрузки очередного файла) лишь тогда, когда пользователь предпримет соответствующее активное действие («щелчок» кнопкой мыши, нажатие клавиши Enter).

Web-сервер – это программа, которая обрабатывает HTTP-запросы (при обращении к файловой системе локального компьютера запросы обрабатывает сам браузер). Сервер читает и синтаксически анализирует HTTP-запрос, преобразует URL в имя файла, проверяет его наличие и разрешение на доступ пользователя к нему, формирует и передает ответ. HTTP-протокол предусматривает передачу в заголовке сообщения-ответа поля Content-Type, указывающего на тип передаваемого файла. Например, image/gif, text/html, application/x-javascript, application/binary и т.д.

Интерпретация гипертекстового документа в соответствии с языком разметки HTML возможна только после копирования файла на клиентский компьютер. Браузер может сохранять файлы в автоматическом режиме в строго определенном месте (кэше) либо в произвольном месте файловой системы по желанию пользователя. Браузеру известны некоторые типы файлов, и он знает, как с ними обращаться. К таким файлам относятся известные типы изображений, звуков и видео различных форматов. Если браузер не может открыть и интерпретировать (отобразить, проиграть) файл, он обращается к разделу HKCR системного реестра, находит там ассоциативную связь с нужным приложением

и запускает его, передавая полное имя документа.

Если загружаемый ресурс является известным исполняемым или интерпретируемым файлом, браузер запрашивает пользователя, как следует поступить с этим объектом. Не существует различия в том, откуда

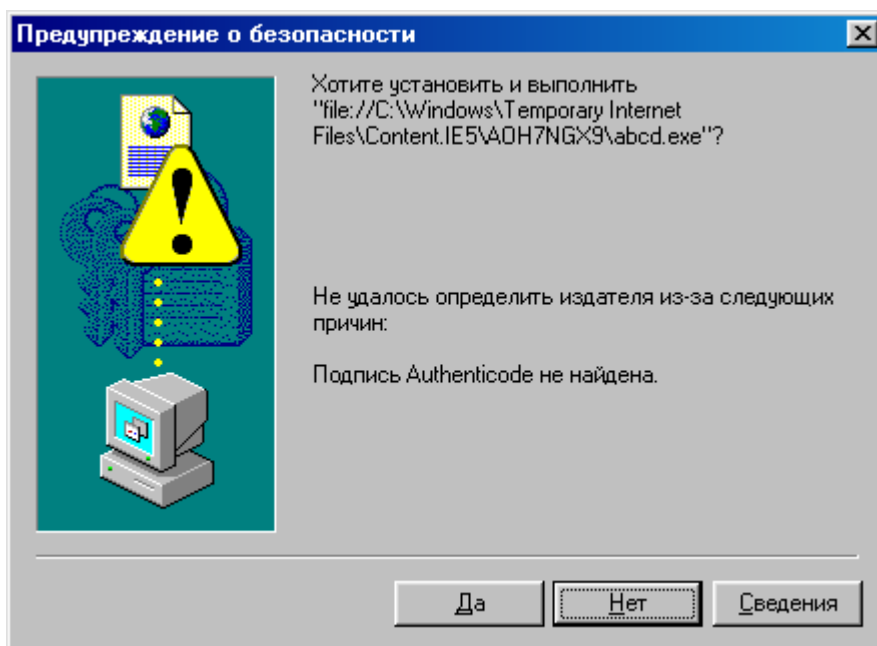


Рис. 11. Предупреждение об опасности запуска файла, выводимое пользователю браузером IE

загружается файл: с удаленного сервера или из собственной файловой системы. При подобной форме запуска пользователь увидит диалоговое окно обозревателя «Загрузка файла», в котором будет указано имя загружаемого файла с запросом, что с ним делать: запустить или сохранить на диске (рис. 11).

Вариант с сохранением файла, заданного с префиксом `file://` (или `file:///`) в другом каталоге, эквивалентен обычному копированию. В случае, если пользователь решит запустить файл, будет выведено сообщение о том, что не удалось найти электронную подпись, идентифицирующую автора программы. Запуск произойдет только после повторного согласия пользователя. Если пользователь обратится с запросом к справочной системе Windows, ему будет рекомендовано сохранить файл на диске, чтобы потом перед его открытием или запуском принять меры предосторожности, в качестве которых предлагаются:

- проверка файла на наличие в нем вирусов,
- сохранение открытых файлов и завершение всех приложений,
- отключение от Интернета и других сетевых соединений.

Второй вариант загрузки файла по гиперссылке заключается во включении в состав html-документа графического файла. Для этого в тексте документа используется тэг:

** ,**

где `image_URL` – унифицированный адрес ресурса для графического файла. Если данный файл размещается на той же странице, где расположен основной документ, допустимо использовать относительный адрес (например, если html-файл скопирован по адресу **`http://abcd.ru/page1`**, то относительный адрес **`/picture3`** будет означать для браузера полный адрес **`http://abcd.ru/page1/picture3`**). Браузер запоминает адреса скопированных ресурсов, и это можно использовать в дальнейшем.

Третий вариант – использование тэга **<EMBED>**, который используется для встраивания звуковых файлов, например:

<EMBED SRC="sound.mid">

Четвертый способ загрузки (открытия) файлов связан с использованием фреймовых структур (frames – рамка, отдельная область окна просмотра):

<FRAME SRC="URL">

Загрузка подобных «мультимедийных» файлов с помощью тэгов <EMBED>, , <FRAME> осуществляется автоматически, без какого-либо участия со стороны пользователя.

Если загружаемые ресурсы не представляют какую-либо опасность, браузер помещает их в свой кэш, расположенный в дисковом пространстве. Кэш является важным промежуточным пунктом между Web-сервером и браузером. Его функция не ограничивается накоплением ранее скопированных документов на случай необходимости их повторного просмотра. Он еще должен изолировать файловую систему клиентского компьютера от внешних угроз. Дисковый кэш представляет собой каталог, который в ОС Windows NT 5.0 расположен в %Userprofile%\ LocalSetting\ Temporary Internet Files\ Content.IE5\, а в Windows 9* соответственно в каталоге %Win-dir%\Temporary Internet Files\Content.IE5\.. В дальнейшем

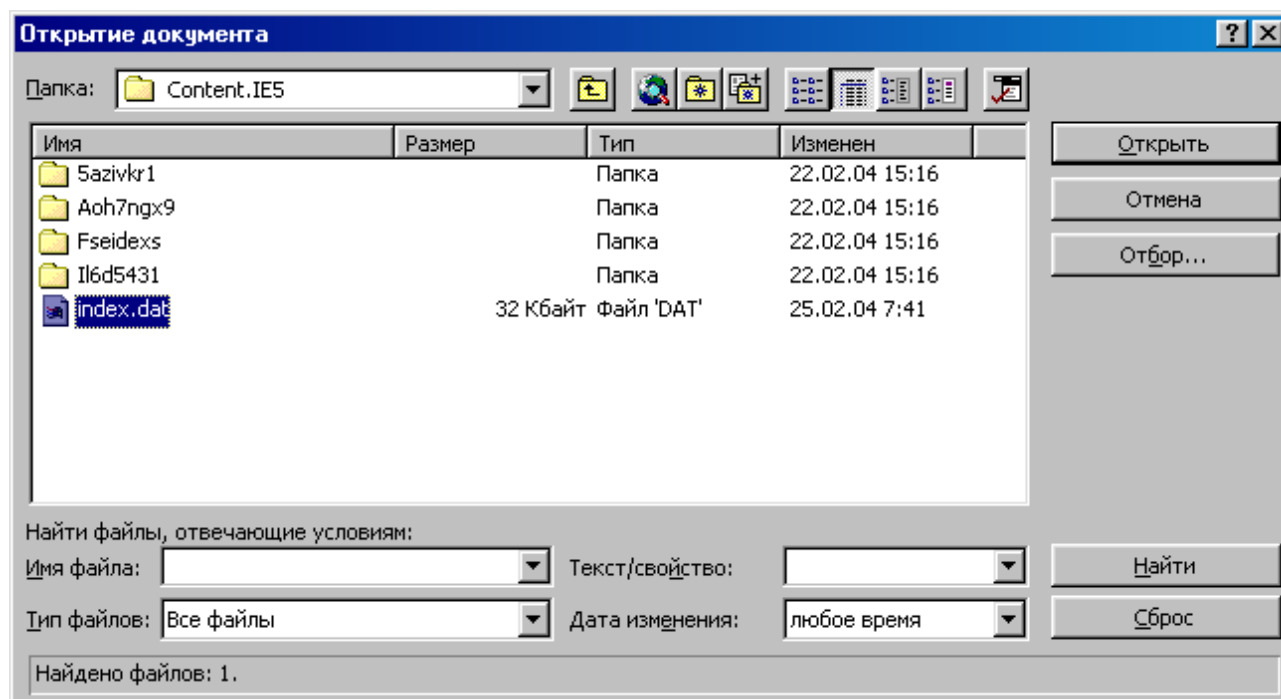


Рис. 12. Содержимое каталога **Temporary Internet Files** при просмотре в Word 9.0

данный каталог будем сокращенно именовать TIF.

Внутри каталога TIF создается определенное число подкаталогов, имена которых представляют случайную комбинацию из восьми цифр и заглавных латинских символов. В данные подкаталоги помещаются файлы, переданные по HTTP-протоколу с Web-узла. Случайный выбор имени подкаталога должен сильно затруднить потенциальному злоумышленнику возможность обращения извне к помещенному внутри него файлу (рис. 12).

Наиболее естественной формой внедрения и запуска вредоносного программного кода является его копирование из Web-сервера в дисковый кэш браузера с последующей попыткой запуска на исполнение ранее внедренного файла из следующего html-документа.

Отображение информации в окнах браузера производится уже после того, как отображаемый файл будет отправлен Web-сервером и поступит в кэш, по мере считывания тэгов и интерпретации разметки документа-контейнера. Копируется в кэш не каждый файл. Пользователь может копировать файлы не в кэш, а в один из других каталогов локальной файловой системы.

Информация в кэше защищается не только от внешнего, но и от внутреннего доступа. При использовании для просмотра этого каталога штатных средств операционной системы (а таким средством по умолчанию является Проводник – Explorer), можно увидеть содержимое каталога как бы в кривом зеркале. Открыв каталог TIF обычным образом: («Мой компьютер» – логический диск – Windows – Temporary Internet Files), мы увидим, что файлы отображаются в нем не как в обычном каталоге, а в два столбца – в первом указывается имя файла, во втором – его адрес в Интернет.

Оболочка Explorer.exe устанавливает особую форму обращения к этому каталогу (к правам доступа это никакого отношения не имеет). Нетрудно убедиться в том, что:

- можно беспрепятственно удалить из этого каталога любой отображаемый файл;

- скопировать какой-нибудь файл в этот каталог из другого каталога локальной файловой системы невозможно ни путем перетаскивания (drag and drop), ни через меню «Копировать» – «Вставить» соответствующих папок;
- обратное копирование файлов из TIF в другие папки с помощью Explorer происходит успешно;
- на попытку открыть или запустить какой-либо из находящихся в кэше файлов браузер либо никак не реагирует (в случае html-файлов), либо отвечает на это выводом окна сообщений. Установлено, что подобный режим отображения информации и ограничения в работе с файлами вызывается наличием в каталоге TIF и каждом из его подкаталогов файла инициализации desktop.ini. Оболочка Explorer при открытии каталога «видит» .ini-файл, читает его, находит в реестре по указанному CLSID нужный сервер автоматизации и передает ему управление. Данный сервер (.dll, .osx или .exe-файл) интерпретирует обращения к другим файлам внутри этого каталога нужным образом.

Если посмотреть содержимое каталога TIF с помощью файлового менеджера Far или текстового процессора Word, мы обнаружим там совершенно иную структуру (рис. 12). В нем присутствует уже упомянутый выше файл desktop.ini и каталог Content.IE5, в котором содержатся подкаталоги с именами из восьми псевдослучайных символов. Внутри этих подкаталогов содержатся имена файлов, между именем и расширением которых вставлены символы [1]. Запуск либо открытие этих файлов из Far проходит успешно.

Использование сценариев внутри документа позволяет расширить возможности загрузки файлов. В частности, использование встроенной объектной модели Internet Explorer позволяет автоматически выполнить загрузку объекта по указанному URL:

```
<SCRIPT LANGUAGE="VBScript">  
Navigate URL  
</SCRIPT>
```


Еще один вариант загрузки ресурсов связан с использованием метода `Open`. Данный метод позволяет создать новое окно браузера. Формат вызова метода имеет вид:

`Window.Open (URL, Имя_нового_окна, Параметры)`

URL содержит ссылку на документ, который должен быть открыт в окне браузера.

Браузеры содержат встроенные механизмы для отображения данных определенных типов, например текста и графических образов в формате GIF и JPEG. Для отображения данных, которые они не умеют отображать сами, браузеры могут вызывать внешние приложения. Например, для выполнения Java-апплета браузер должен передать имя этого апплета и параметры его вызова виртуальной Java-машине и отобразить результаты ее работы в своем окне. Инициализацию элементов управления в гипертекстовых документах рекомендуется производить с помощью тэга `<OBJECT>`. Этот способ более универсален и позволяет обращаться к нужному серверу автоматизации по зарегистрированному идентификатору класса CLSID и произвольному имени. Если объект в системе не зарегистрирован, возможна его подкачка (копирование) из Интернет. Предусматривается обновление версии сервера автоматизации и прочие возможности.

Тэг `<OBJECT>` позволяет разработчикам HTML-страниц указывать браузеру тип объекта, включаемого в документ, с тем, чтобы браузер принял решение: будет он его отображать сам или передаст исполнение внешней программе. В самом общем случае разработчик должен передать браузеру следующую информацию об объекте:

- Тип и местоположение включаемого объекта.
- Тип и местоположение данных для объекта (например, формат и расположение файла, содержащего графический образ).
- Дополнительные параметры.

Полный синтаксис тэга-контейнера `<OBJECT>` имеет следующий вид:

<OBJECT

ACCESSKEY=клавиша
ALIGN=ABSBOTTOM | ABSMI | DDLE | BASELINE | BOTTOM
LEFT | MIDDLE | RIGHT | TEXTTOP | TOP
CLASS=имя_класса
CLASSID=идентификатор_объекта
CODE=имя_файла
CODEBASE=url_адрес[#version=a,b,c,d]
CODETYPE=media_тип
DATA=url_адрес
DATAFLD=имя_столбца
DATASRC=#идентификатор-источника
HEIGHT=целое_число
ID=идентификатор
LANG=язык
LANGUAGE=JAVASCRIPT | JSCRIPT | VBSCRIPT | VBS
NAME=имя
STYLE=правила_CSS
TABINDEX=целое_число
TITLE=текст
TYPE=MIME_тип
WIDTH=целое_число >

Опишем назначение основных атрибутов:

- Атрибут CLASSID. Компоненты ActiveX используют для идентификации специальный атрибут – идентификатор класса CLASSID (сокращенно CLSID). Значение параметра CLASSID представляет собой строку следующей формы:

CLASSID = "clsid: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

Первые восемь цифр CLSID указывают анализатору кода браузера на то, что это элемент управления ActiveX. Остальная часть идентификатора представляет собой уникальный номер этого элемента. Идентификатор класса предназначается для определения, установлен ли данный компонент в системе или еще нет. Если такой идентификатор найден, то используется локальная копия компонента. В принципе, если разработчику гипертекстового документа заранее известно, что используется безопасный и безусловно присутствующий в операционной системе компонент ActiveX, на него не требуется ссылаться с

помощью тэга <OBJECT>. Альтернативный вариант запуска мобильного кода для этого случая будет описан ниже.

- Атрибут CODEBASE задает сетевой адрес компонента, по которому браузер может загрузить его на компьютер пользователя в случае отсутствия. Одновременно этот атрибут обозначает базовый URL для ресурса в атрибуте DATA, если его местоположение задано относительным URL. Этот же атрибут используется для задания местоположения динамических библиотек, используемых управляющими элементами ActiveX. Форма задания атрибута:

```
CODEBASE      =      "http://www.windowatch.com/security.ocx  
#Version = 1,0,0,0"
```

Строка может содержать несколько адресов – на тот случай, если один из серверов окажется недоступным. После адреса элемента управления задается его версия в виде: #Version = a,b,c,d, где a и b – соответственно старшее и младшее слово максимально доступной на сервере версии элемента управления, а c и d – старшее и младшее слово минимально доступной на сервере версии. Значения a,b,c,d могут быть установлены равными «-1». В этом случае элемент управления с сервера загружается, если дата его выпуска позже даты его установки на объектовом компьютере.

- Атрибут DATA задает URL файла данных, необходимых для использования объектом. Какими должны быть тип и формат файлов данных – разработчиками не документируется.
- ID - внутреннее наименование элемента управления ActiveX с целью ссылки на него в сценарии.
- Атрибуты WIDTH и HEIGHT задают размеры окна для отображения объекта по горизонтали и вертикали соответственно. Большинство браузеров требуют, чтобы эти атрибуты задавались всегда, т.к. они не могут вычислить необходимый размер окна автоматически.
- Атрибут NAME задает название объекта и используется только в тех случаях, когда данный объект является частью формы.

- Атрибут CODETYPE указывает тип объекта, заданного атрибутом CLASSID. Этот атрибут является необязательным, но рекомендуемым. Если атрибут задан, его значением по умолчанию считается значение атрибута TYPE.
- Атрибут TYPE указывает тип данных, заданных атрибутом DATA. Этот атрибут является необязательным, но рекомендуемым.

Задание всех параметров не является обязательным, и большинство из них для исследования механизмов защиты браузера интереса не представляют. По меньшей мере, один из аргументов в тэге <OBJECT> должен присутствовать. Анализ атрибутов тэга <OBJECT> производится одной из оболочек (Shell) операционной системы Windows* (iexplore.exe в режиме on-line или explorer.exe в режиме автономного чтения off-line). Для анализа используются библиотечные функции из файлов mshtml.dll и тому подобных. Разбор производится в следующей очередности: Обнаружив в HTML-документе тэг <OBJECT>, браузер вызывает функцию CoGetClassObjectFromURL, которая представляет собой механизм Internet

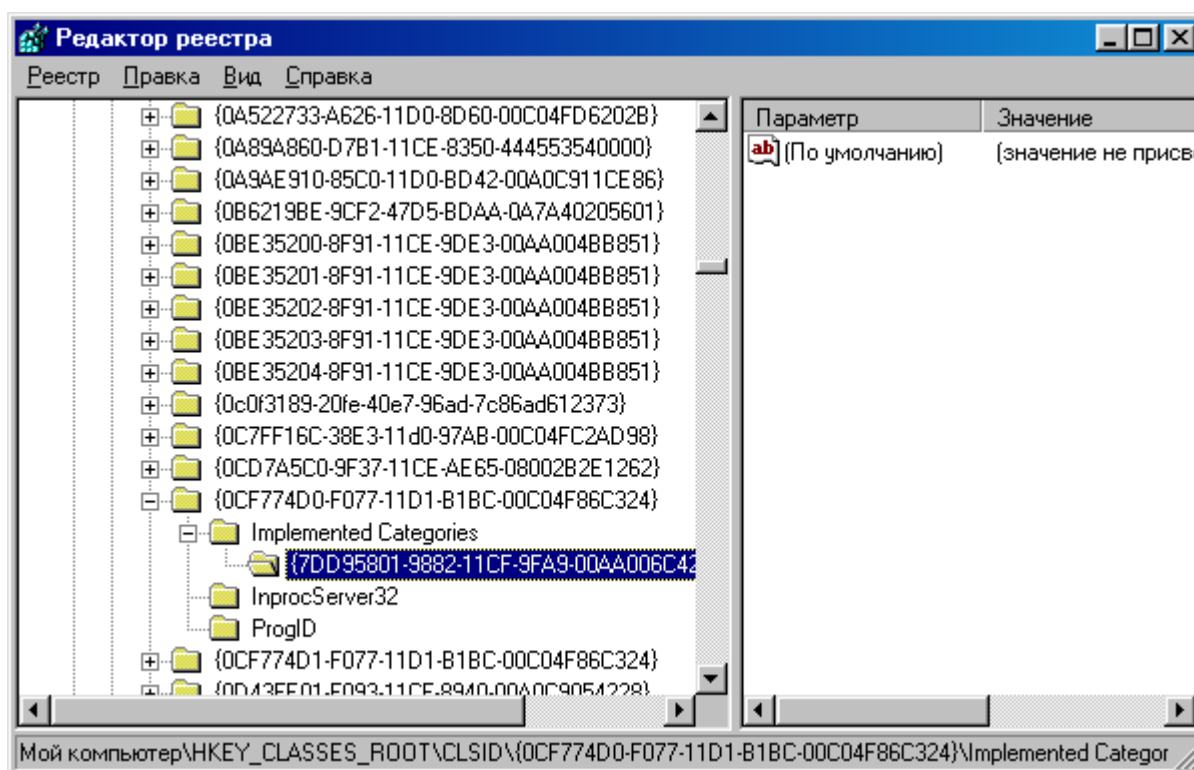


Рис. 13. Поиск CLSID в системном реестре

Component Download Service. Функции передается заданный в тэге идентификатор CLSID, и она ищет его в разделе HKEY_CLASSES_ROOT\CLSID системного реестра. Если такой идентификатор найден, для него проверяется наличие подключа Implemented Categories (реализованные категории) со свойством "safe for scripting" (безопасный для исполнения в сценариях). Это свойство выражается цифровой последовательностью 7DD95801-9882-11CF-9FA9-00AA006C42C4 (Рис. 13). При регистрации идентификатора объекта в операционной системе вызывается следующая функция CoGetClassObject, которая в случае успешного выполнения возвращает “фабрику классов” для данного компонента. Если безопасность присутствующего в HTML-файле элемента управления установлена, разрешается исполнение в сценарии его методов, невзирая на установленную защиту браузера и желание пользователя. С таким элементом управления ActiveX можно работать при установленном уровне защиты High браузера - до проверки уровня защиты дело просто не доходит.

В зависимости от того, где располагается найденный элемент управления, исполнение его кода реализуется различным образом. Если он расположен во внутреннем сервере, то файл является динамической библиотекой (файл с расширением .dll) и исполняется, будучи загруженным в адресное пространство клиентского процесса (браузера). Если элемент управления расположен в локальном сервере, он представляет собой исполняемый (.exe или .osx) файл и запускается в виде самостоятельного процесса. Методы, указанные для данного интерфейса, интерпретируются элементом управления и преобразовываются в вызовы функций Win32API.

Если установлено, что элемент управления не является безопасным, проверяются установленные уровни защиты браузера для соответствующей зоны Интернет, из которой скопирован данный гипертекстовый документ. Исполнение кода разрешается, если установлен низкий уровень защиты. При высоком уровне элемент управления не выполняется (его присутствие в документе игнорируется), при среднем уровне предупреждается пользователь, и запуск кода происходит только при его согласии.

Если нужный идентификатор класса CLASSID не обнаружен, проверяется наличие в тэге <OBJECT> атрибута CODEBASE. Управление передается функции CoGetClassObjectFromURL, которая пытается загрузить сервер автоматизации с сетевого адреса, заданного атрибутом CODEBASE.

При наличии атрибута CODEBASE браузер обращается по URL, заданному атрибутом. С этого адреса на клиентский компьютер копируется файл, поименованный в строке (возможные расширения файлов: .exe, .dll, .osx, .inf или .cab). Загруженный файл проверяется на наличие цифровой подписи. При отсутствии таковой для пользователя выводится предупреждение о том, что загруженный элемент управления не подписан. В противном случае пользователю выводится информация о загруженном элементе управления с электронной подписью. Если пользователь доверяет этой подписи, элемент управления будет помещен в соответствующий каталог (.dll, .exe и .osx файлы помещаются в системный каталог Windows) и зарегистрирован в разделе HKCR реестра. При установке на компьютер пользователя элемента управления ActiveX, помеченного разработчиком безопасным при инициализации и использовании в сценариях, в разделе системного реестра этого элемента управления создается ключ Implemented Categories (Реализованные категории), содержащий два подключа, отмечающие соответствующие категории безопасности элемента управления. Подключ {7DD95801-9882-11CF-9FA9-00AA006C42C4} соответствует безопасности при инициализации, а подключ {7DD95802-9882-11CF-9FA9-00AA006C42C4} – безопасности при использовании в сценарии.

Если идентификатор класса обнаружен и атрибут CODEBASE присутствует, проверяется соответствие версии элемента управления. Причем, если в атрибуте CODEBASE указана более новая версия или более поздний срок ее создания, браузер загрузит обновленную версию и заменит ею старый файл. Происходит ли это с уведомления пользователя, или автоматически, неясно.

При наличии в тэге <OBJECT> атрибута DATA с указанным URL происходит копирование по данному адресу файла данных (поскольку предполагает

ся, что этот файл нужен элементу управления). Как связывается копирование данных с другими атрибутами – не документировано.

Существует еще один способ запуска элементов управления ActiveX. Он заключается в использовании тэга <Script>, внутри которого содержится интерпретируемый код (сценарий) на одном из языков, из которых основными являются VBS и JS. Присутствие в HTML-документе сценария позволяет обеспечить интерактивный режим работы с пользователем, автоматическое изменение содержимого документа, динамическое отображение его частей и т.д. Сами по себе языки VBS и JS могут считаться полностью безопасными, поскольку в них не предусмотрены функции, позволяющие обращаться к файловой системе, работать с системным реестром и запускать процессы. Все это становится возможным, если в сценарии объявлен сервер автоматизации (точнее – входящий в его состав компонент или элемент управления). Загрузка элемента управления происходит при объявлении объектной переменной (пример приведен для сценария на языке VBS).

Set Obj = CreateObject(«ProgID») где ProgID = Application.Class

- Программный идентификатор ProgID позволяет загружать объект с помощью записей реестра. В первой строке создается переменная типа Object с именем Obj. Затем интерпретатор, используя вызов функции CLSIDFromProgID, находит в разделе системного реестра HKCR ключ Application.Class. Если запись обнаружена, из нее извлекается идентификатор класса CLSID. Если такой записи нет, процедура прерывается.
- Вызывается функция CoGetObject, которой передается найденный CLSID. Если операция выполнена успешно, загружается и запускается файл или модуль (с расширениями .exe, .dll, .ocx), хранящий объект.

Пример сценария, использующего сервер сценариев WScript.Shell (Windows Script Host Shell Object, обеспечивается исполняемым файлом C:\WINDOWS\SYSTEM\WSHOM.OCX):

```

<SCRIPT language=VBScript>
On Error Resume Next
Set WshShell = CreateObject("WScript.Shell")
WshShell.Run"Calc.exe" 'Пример запуска «Калькулятора»
</SCRIPT>

```

Браузер, встречая тэг <SCRIPT>, загружает в адресное пространство своего процесса динамическую библиотеку VBScript.dll (находится в системном каталоге Windows). Код библиотеки интерпретирует инструкции языка VBS.

Встречая инструкцию CreateObject, интерпретатор обращается к системному реестру (ключ HKCR), находит ProgID, соответствующий ему идентификатор класса CLSID, а затем - полное имя (адрес) сервера автоматизации (библиотеки). Экземпляр объекта загружается в память процесса, и его интерфейс (пространство объектов, свойств и методов) становится доступным интерпретатору. При создании объекта методом CreateObject он загружается в память и остается соединенным с интерпретатором VBS до завершения сценария. Строки:

```

<object
id="WshShell"
classid="clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B"
></object>

```

и

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

для браузера (и вызываемых им функций Win32API) являются эквивалентными.

Вызов подобного объекта при установленном пороге защиты от среднего и выше, безусловно, покажется браузеру опасным, и он выведет пользователю соответствующее предупреждение. Проверка допустимости присоединенных элементов управления ActiveX производится через системный реестр тем же порядком, что и в случае использования тэга <OBJECT>.

Библиотека типов реализована в виде исполняемого файла с расширениями .osx, .dll или .exe. Каждый класс объектов, размещенный в этом файле, зарегистрирован в системном реестре под своим уникальным номером (CLSID-идентификатор класса). Сервер автоматизации в целом может содержать опас-

ные свойства и методы, но класс объектов, содержащийся в данном исполняемом файле, при этом может быть безопасным для исполнения.

3.2. Анализ опубликованных уязвимостей браузера Internet Explorer

Как уже отмечалось выше, возможности удаленных атак узлов с сети Интернет занимают мысли многих исследователей. В доступной литературе и на многочисленных сайтах в Интернет обсуждается и анализируется достаточно представительный список уязвимостей Web-клиентов, позволяющих внедрить и запустить на компьютере пользователя вредоносный программный код. Наибольший перечень относится к распространенному браузеру Internet Explorer и поддерживающим его механизмам операционной системы. Перечислим наиболее существенные уязвимости, отраженные в многочисленных изданиях типа «Секреты хакеров».

В первую очередь подлежат оценке угрозы, которые могут быть реализованы с помощью скриптлетов без включения в них элементов ActiveX. Языки программирования Jscript, VBScript в целом безопасны, поскольку в них не предусмотрено инструкций, позволяющих работать с файлами и реестром, а также запускать другие программы. Однако с их помощью можно реализовать атаки на отказ в обслуживании. Современные браузеры и операционные системы достаточно хорошо защищены от подобных атак. Но защита эта касается только ресурсов компьютера, а не самого пользователя. Даже обычный цикл с выводом модальных окон сообщений в VBS (`S = MsgBox(«Юзер – козел», 4144, «С хакерским приветом»)`) не позволит пользователю закрыть обозреватель до тех пор, пока не закроет все окна сообщений с полученным «приветствием».

Интернет-браузер является для компьютерной системы своего рода окном в мир, и через это, иной раз чрезмерно распахнутое окно, в клиентский компьютер могут проникнуть вредоносные программы.

Первая опасность такого рода связана с возможностью удаленного запуска программы с опасными функциями, заведомо присутствующей на атакуемом компьютере. В первую очередь такими программами являются исполняемые

файлы, способные совершать деструктивные действия. К ним относятся: `format.com`, `fdisk.exe`, `deltree.exe`, `regedit.exe`, `debug.exe`, `run32dll.exe` и другие. По большому счету опасность представляет любая программа, позволяющая обращаться в режиме чтения и/или записи к объектам файловой системы, манипулировать системным реестром, либо запускающая другие программы на клиентской машине (особенно – с параметрами в командной строке). Одно дело запустить команду `format`, которая запросит пользователя, что ей надо отформатировать, совершенно другое – запустить команду безусловного форматирования конкретного логического диска без согласия пользователя **`format c: /u/autotest`**)

Вторая опасность возникает при внедрении в файловую систему атакуемого компьютера посторонней программы с вредоносными функциями и ее немедленным запуском.

Третья опасность возникает в случае копирования внешней программы с отложенным ее запуском на исполнение. Например, она может быть скопирована на диск в одном сеансе, а быть запущенной – только при очередном включении компьютера. Либо внедрение происходит через один гипертекстовый документ, а запуск – через другой.

Проанализируем «наставления и рекомендации» по удаленным атакам на клиентские компьютеры с использованием уязвимостей современных браузеров.

1. В ряде источников по хакерским атакам указывается, что некоторые компоненты ActiveX, такие как Excel, PowerPoint, Scriptlet.typelib и др., зарегистрированы в реестрах ряда версий операционных систем как безопасные для исполнения (“safe for scripting”). Таким образом, представляется возможность присоединения к сценарию, вставленному в гипертекстовый контейнер, и использование объектов, свойств и методов этих элементов управления, которые обладают практически неограниченными возможностями в отношении файловой системы и реестра ОС клиентского компьютера.
2. Отмечается возможность автоматического обновления версии зарегистрированного компонента ActiveX с любого из серверов, указанных тэга

<OBJECT>. Утверждается, что браузер, обнаружив на сервере новую версию исполняемого файла или динамической библиотеки, ранее зарегистрированных в системе, автоматически «скачает» ее, не запрашивая при этом разрешения пользователя и, таким образом, обновит функциональные возможности, предоставляемые этим элементом управления. Авторы идеи не указывают, при каких обстоятельствах браузер обращается за новой версией к указанному параметром CODEBASE серверу. Сомнения заключаются в том, что при наличии в реестре CLSID ранее зарегистрированного объекта для браузера обращаться к ресурсам Интернет вообще не требуется.

3. Еще один вариант внедрения файла связан с использованием параметра DATA тэга <OBJECT>. Предлагается вообще не указывать CLSID компонента:

```
<OBJECT DATA= "http://abcd.com/file_name.exe" TYPE=
"text/html" WIDTH=1 HEIGHT=1>
```

Файл данных должен подкачиваться, если задан компонент (сервер автоматизации), которому эти данные стали нужны. Подразумевается, что файл данных не может относиться к числу исполняемых, командных или скриптов. По-видимому, объявлением фальшивого типа файла программу-браузер предлагается ввести в заблуждение. Однако браузер запрограммирован на лексический анализ URL и имя (и тип) файла выделит без проблем. В другом источнике предлагается загружать файл данных непременно с другого сетевого адреса (по отношению к адресу, с которого был скопирован первоначальный html-файл, включающий вышеуказанную строку).

4. Указывается на возможность копирования в Интернет базы данных Access (файл с расширением .mdb, который копируется браузером при указании его местонахождения параметром DATA тэга <OBJECT>). После этого, якобы, запускается приложение Access, в нем открывается скопированная база данных и управление получает интерпретируемый код макроса, который можно предварительно разместить в этой базе данных. Только после этого браузер оповещает пользователя о возможной опасности.

5. Отмечается опасность, связанная с возможностью запуска любого исполняемого файла из файловой системы локального компьютера путем использования любого ненулевого параметра CLSID. Фрагмент кода, встроенного в html-файл, выглядит при этом следующим образом:

```
<OBJECT CLASSID="CLSID:10000000-0000-0000-0000-000000000000" CODEBASE="C:\Windows\Calc.exe"></OBJECT>
```

6. Перед изложением очередной уязвимости необходимо пояснить следующее. Когда в адресной строке браузера указан URL сетевого ресурса, браузер устанавливает с сервером TCP-сеанс и по протоколу верхнего уровня (http, ftp и др.) запрашивает нужный ресурс. Если указанный файл не является опасным для загрузки, он копируется во внутренний кэш браузера, каким является каталог Temporary Internet Files. Одновременно с этим файл открывается пользователю в окне браузера. Таким образом, файл оказывается в контексте локальной зоны безопасности, для которой браузер оказывает полное доверие. Если извне удастся запустить этот файл, он может выполнить любые опасные действия по отношению к клиентскому компьютеру. Для того чтобы местонахождение файла осталось неизвестным, он записывается в подкаталог со случайным именем: **%windir%\Temporary Internet Files\Content.IE5\XXXXXXXX\filename[1].ext**, где XXXXXXXX – псевдослучайная алфавитно-цифровая строка длиной в 8 символов. Случайно генерируемое имя каталога делает почти невозможным прямой запуск файла из кэша по указанному пути.

Из «хакерской» литературы известно, что можно записать и запустить на объектовом компьютере файлы, имеющие расширение .chm. Данные файлы являются компилированными html-документами и широко используются для поддержки справочных систем Windows. Обращение к этому файлу активизирует приложение hh.exe. Выводится окно справочной системы, в которой поддерживаются гипертекстовые ссылки. Гиперссылки в .chm-файле в принципе могут быть использованы для запуска приложений.

Авторы идеи предлагают на первом этапе поместить в произвольный html-файл, к которому пользователь с большой вероятностью будет иметь доступ, до десятка одинаковых chm-файлов с различными именами. Расчет делается на то, что браузер не воспринимает файлы справочной системы в качестве опасных. Их загрузка в кэш происходит под видом графических файлов с использованием тэга:

```
<IMG SRC="chm1.chm" WIDTH=1 HEIGHT=1>
```

Html-контейнер и псевдографические файлы будут скопированы в кэш браузера (каталог Temporary Internet Files). Там они будут помещены в один из подкаталогов, имеющих случайно сгенерированное имя. Это делает почти невозможным прямой запуск файла из кэша по его полному имени.

Следующим этапом является вычисление полного пути к этим файлам, для чего требуется угадать случайную строку из 8 символов. Для определения пути предлагается довольно замысловатый способ. С этой целью в первом гипертекстовом контейнере, кроме ссылок на подгружаемые «графические» файлы, создается ссылка на второй гипертекстовый документ, который непременно должен загружаться с другого Web-сервера (почему – неясно). Можно имитировать загрузку с другого сервера, задав браузеру тот же URL в модифицированном виде (например, <http://www.abcd.ru> вместо <http://abcd.ru>). Благодаря тому, что текущий документ с использованием свойств и методов компонента Internet Explorer может получить адрес предыдущего документа, представляется возможность извлечь нужную строку адреса и отделить от нее путь к нужному файлу. После этого происходит непосредственно открытие chm-файла с помощью метода Internet Explorer

```
Window.showHelp("c:\dir\hostfile.chm") .
```

При этом делаются попытки открыть каждый из внедренных в кэш .chm-файлов. Открытый файл автоматически запускает исполняемый файл из локальной файловой системы.

4. ЛАБОРАТОРНАЯ РАБОТА № 2

Цель работы:

- Изучить механизм взаимодействия браузера Microsoft Internet Explorer с Web-браузером
- Исследовать защитные механизмы браузера, препятствующие возможности внедрения и запуска вредоносного программного кода на клиентском компьютере.
- Выявить и оценить потенциальные уязвимости Web-протоколов и их программной поддержки со стороны операционных систем Windows*

4.1. Подготовка к проведению исследований

Для имитации Интернет в лабораторной работе используется локально размещенный на клиентском компьютере Web-сервер «Eserv» версии 2.95, объединяющий в себе функции ftp-сервера, почтового сервера, сервера новостей. DNS-сервер не используется, и обращение к Web-серверу производится по его IP-адресу. Поскольку канальный, сетевой и транспортный уровни передачи данных влияния на HTTP-протокол не оказывают, создание подобной виртуальной среды оказывается достаточно удобным для проведения необходимых исследований (рис.14).

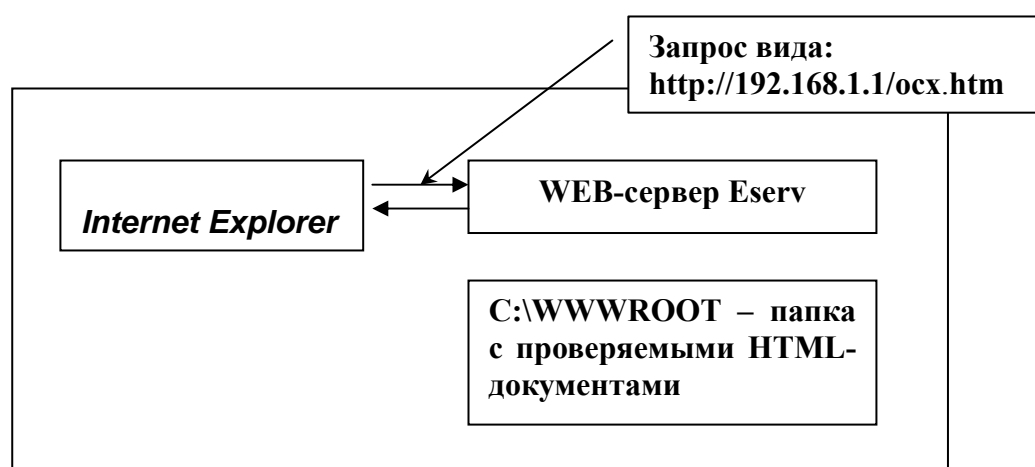


Рис. 14. Браузер IE и Web-сервер под управлением ОС Windows 2000

Инсталляция Web-сервера производится в следующей очередности:

1. Установите оптический диск с программой в привод CD ROM. Если механизм autorun.inf в реестре не отключен, автозагрузка произойдет автоматически. В противном случае следует открыть корневой каталог диска, найти там командный файл, соответствующий используемой версии операционной системы, и запустить его.
2. По умолчанию программа будет установлена в каталог C:\Program Files\Eserv2\. Следует убедиться в ее правильной инсталляции и создать на «Рабочем столе» ярлык на исполняемый файл Web-сервера.

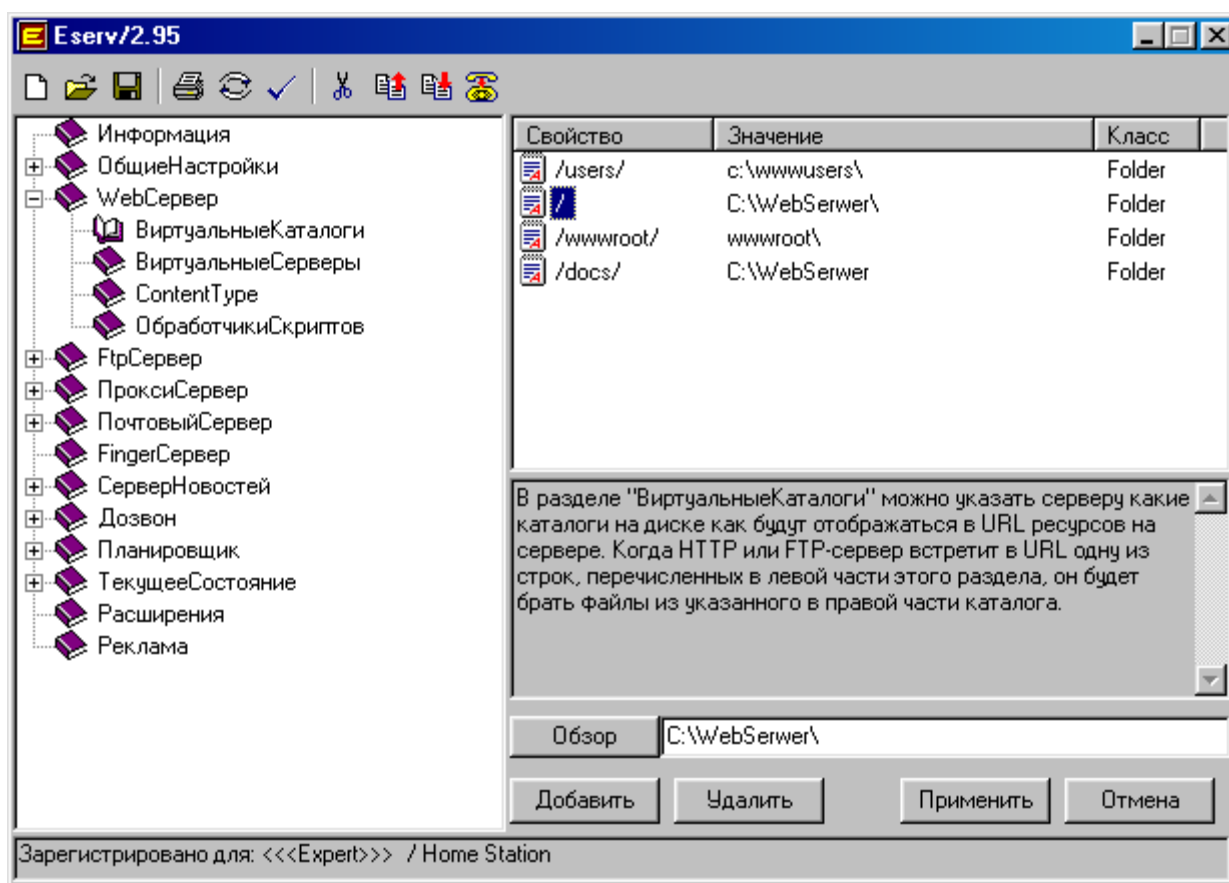
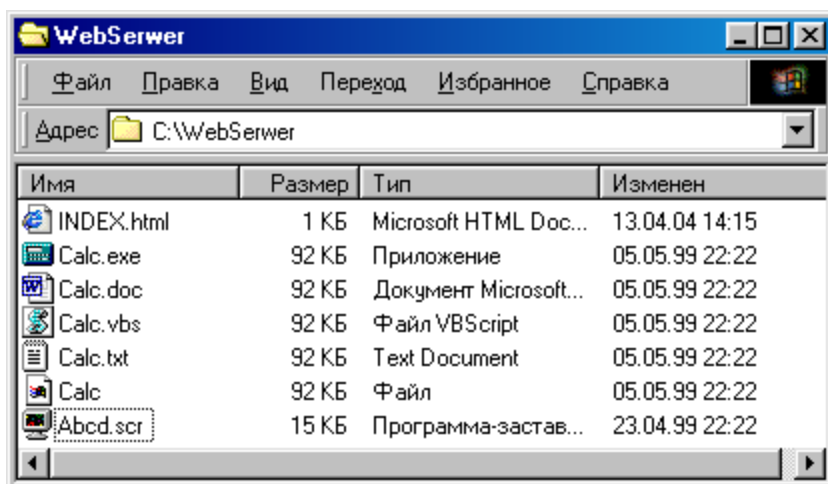


Рис.15. Окно Web-сервера Eserv

3. В удобном месте файловой системы создайте папку с произвольным именем, в которой будут размещаться Интернет-ресурсы Web-сервера. Предположим, она будет называться VebServer.

4. Запустите «Сеанс MS DOS» или «Командную строку» и введите команду ip-config. Прочитайте и запишите на память IP-адрес сетевого интерфейса компьютера. Если IP-адрес не установлен, следует через меню кнопки «Пуск» произвести настройку сетевого оборудования и программного обеспечения.
5. Запустите Web-сервер и убедитесь в выводе диалогового окна приложения (рис. 15). Выберите в списке левой части окна «WebСервер» – «ВиртуальныеКаталоги» и дважды «щелкните» мышью по значку корневого каталога сервера в правом окне.
6. С помощью кнопки «Обзор» под правым окном задайте имя ранее созданного каталога для размещения Web-ресурсов, после чего нажмите кнопку «Применить».
7. Запустите браузер Internet Explorer и в его адресной строке задайте IP-адрес Web-сервера (например, <http://192.168.0.1>). В окне браузера должен раскрыться каталог сервера.
8. Сверните рабочее окно Web-браузера до размера пиктограммы на «Панели задач».
9. Настройте браузер Internet Explorer «Сервис» – «Свойства обозревателя» – «Безопасность» на безопасный, но функциональный обзор:
 - разрешить выполнение активных сценариев,
 - разрешить выполнение сценариев элементов управления ActiveX, помеченных как безопасные,



- Загрузку и исполнение неподписанных элементов управления ActiveX производить с разрешения пользователя.
- Остальные настройки безопасности – по

Рис.16. Подготовка файлов для копирования на клиентский компьютер

умолчанию либо на свое усмотрение.

- С помощью текстового редактора «Блокнот» создайте в рабочем каталоге Web-сервера каркас html-документа по следующему образцу:

```
<html>
<head>
<title>Исследование уязвимостей IE</title>
</head>
<body>
` Сюда следует вставлять нижеприведенные фрагменты
</body>
</html>
```

- Сохраните созданный файл под именем index.htm. Этот документ будет автоматически открываться при обращении к Web-серверу. Путем пробного запуска Internet Explorer убедитесь, что документ открывается в рабочем окне браузера.

4.2. Имитация атак на отказ в обслуживании

1. Вставьте в каркас html-файла index.html следующий фрагмент со скриптом:

```
<SCRIPT language=VBScript>
On Error Resume Next
For I=1 To 10
Message = MsgBox("Юзер - козел", 4144, "С хакерским приветом")
Next
</SCRIPT>
```

Запустите браузер. Можете ли вы закрыть приложение? Как вы классифицируете эту атаку? Создает ли она опасность для операционной системы или обрабатываемой информации?

4.3. Исследование возможности копирования файлов в кэш браузера

1. Скопируйте в рабочий каталог Web-сервера исполняемый файл Calc.exe («Калькулятор»). Этот файл будет использоваться в качестве муляжа «вредо-

носной» программы. Создайте в этом же рабочем каталоге еще четыре копии этого файла и каждому из файлов присвойте характерное расширение: .exe, .doc, .vbs, .txt. Один из файлов оставьте без расширения.

2. Найдите в системных каталогах и скопируйте в рабочий каталог Web-сервера один из небольших файлов экранной заставки (расширение .scr). Переименуйте его в Abcd.scr. (рис. 16). Путем пробного запуска убедитесь, что экранная заставка запускается без сбоев.
3. Вставьте в каркас созданного .html-файла ссылку на запуск файла из локальной файловой системы клиентского компьютера, например:

** Ссылка ** (в Windows NT 5.0 этот файл располагается в системном каталоге). Сохраните измененный файл и вновь запустите браузер. Будет выведен документ с гиперссылкой, по которой необходимо «щелкнуть» мышью. Зафиксируйте в отчете реакцию системы на подобный запуск. Удалите из html-файла внедренный фрагмент и сохраните изменения.

- Вставьте в гипертекстовый контейнер index.html следующие ссылки на ранее созданные файлы «вредоносных» программ. В образцах указан конкретный IP-адрес, вместо которого следует указать аналогичный адрес своего хоста:

- Запустите Internet Explorer и зафиксируйте в отчете, что будет выведено в его окне. Произошел ли запуск приложений или их открытие в других приложениях?
- С помощью оболочки «Проводник» найдите, где располагается кэш браузера (каталог Temporary Internet Files). Осмотрите содержимое этого каталога после копирования Web-ресурсов. Чем отличается отображение находящихся там файлов? Какие из файлов не попали в кэш? Почему?

- Попробуйте с помощью «Проводника» выполнить некоторые манипуляции с содержимым каталога Temporary Internet Files:
- Скопируйте какой-нибудь файл в этот каталог из другого каталога локальной файловой системы.
- Откройте или запустите какой-либо из находящихся в кэше файлов. Прокомментируйте полученные результаты.
- Убедитесь в том, что обратное копирование файлов из TIF в другие папки, либо удаление файлов из этого каталога с помощью Explorer происходит успешно.
- Запустите файловый менеджер Far и с его помощью вновь войдите в каталог Temporary Internet Files. Как можно объяснить различия в представлении отображаемой информации? Какие имена присваиваются подкаталогам и как распределяются в них скопированные файлы?
- Проведите еще несколько экспериментов с кэшем браузера:
- Скопируйте внутрь каталога TIF несколько произвольных файлов с помощью Far. Убедитесь, что копирование прошло успешно и Far отображает присутствие скопированных файлов в папке TIF. Теперь вновь проверьте содержимое папки TIF с помощью оболочки Explorer. Удалось ли обнаружить там скопированные файлы? Почему?
- С помощью Far скопируйте в папку TIF файл «Калькулятор» и запустите его оттуда из командной строки «Пуск» – «Выполнить». Почему в этом случае запуск программы «Калькулятор» выполняется успешно?
- Удалите все подкаталоги кэша. Вновь создайте в гипертекстовом контейнере ссылки на «рисунки», но теперь с указанием размеров окон:

```
<IMG SRC="http://192.168.0.1/calc.exe" WIDTH=1 HEIGHT=1>
<IMG SRC="http://192.168.0.1/calc" WIDTH=1 HEIGHT=1
<IMG SRC="http://192.168.0.1/calc.doc" WIDTH=1 HEIGHT=1>
<IMG SRC="http://192.168.0.1/calc.vbs" WIDTH=1 HEIGHT=1>
<IMG SRC="http://192.168.0.1/calc.txt" WIDTH=1 HEIGHT=1>
```

- Вновь запустите браузер. Чем отличается информация, выведенная при открытии гипертекстового документа? С помощью Far исследуйте кэш браузе-

ра на предмет изменений, которые там произошли, отразите свои наблюдения в отчете. Вновь удалите подкаталоги кэша.

- Измените вид ссылок на загружаемые Web-ресурсы. На этот раз постарайтесь внедрить файлы под видом звуковых фрагментов.

```
<EMBED SRC="http://192.168.0.1/calculator.exe" WIDTH=1 HEIGHT=1>  
<EMBED SRC="http://192.168.0.1/calculator" WIDTH=1 HEIGHT=1  
<EMBED SRC="http://192.168.0.1/calculator.doc" WIDTH=1 HEIGHT=1>  
<EMBED SRC="http://192.168.0.1/calculator.vbs" WIDTH=1 HEIGHT=1>  
<EMBED SRC="http://192.168.0.1/calculator.txt" WIDTH=1 HEIGHT=1>
```

- Запустите браузер. С помощью Far исследуйте кэш браузера на предмет произошедших там изменений, отразите их в отчете. Какие из копируемых файлов на этот раз попали в каталог TIF, в каких подкаталогах они оказались?
- Попробуйте скопировать исполняемый файл calculator.exe в кэш браузера с помощью атрибута CODEBASE тэга <OBJECT>

```
<OBJECT CODEBASE ="http://192.168.0.1/calculator.exe"></OBJECT>
```

Удачно ли завершилось копирование? Попробуйте таким же путем скопировать файлы с другими расширениями.

- Скопируйте исполняемый файл calculator.exe в кэш браузера с помощью атрибута DATA тэга <OBJECT>

```
<OBJECT DATA ="http://192.168.0.1/calculator.exe"></OBJECT>
```

Удачно ли завершилось копирование? Аналогично скопируйте файлы с другими расширениями.

- Сделайте выводы в отношении возможности копирования в кэш браузера файлов различного типа с использованием разнообразных механизмов.
- Поместите в гипертекстовый документ элемент управления ActiveX следующего содержания:

```
<SCRIPT language=VBScript>
```

```
On Error Resume Next
```

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

```
WshShell.Run"Abcd.scr" 'Запуск экранной заставки
```

```
</SCRIPT>
```

Запустите браузер. Произошла ли загрузка и запуск документа с элементов управления ActiveX?

4.4. Исследование возможности запуска файлов из кэша браузера

Существуют несколько способов создания гиперссылки на ранее внедренный в кэш исполняемый файл. Один из них основывается на принципиальном предназначении дискового кэша. Если в индексном файле браузера уже имеется сетевой адрес, на который указывает гиперссылка, и с момента записи файла прошло немного времени, браузер обратится к кэш-памяти и извлечет нужный файл из нее. Следовательно, для запуска находящегося в кэше исполняемого файла нужно повторно сослаться на его сетевой ресурс из гипертекстового документа. При этом следует иметь в виду, что внедрение и запуск файлов должны производиться с использованием различных тэгов или их атрибутов. Так, для внедрения файлов должны использоваться тэги , <EMBED>, атрибут DATA тэга <OBJECT>, а для запуска – тэги <HREF>, <FRAME>, метод Window.Open, атрибут CODEBASE тэга <OBJECT>. Для исследования такой возможности:

1. Найдите в файловой системе какой-либо файл справки с расширением .chm (большинство таких файлов располагаются в папке %windir%\Help). Убедитесь в том, что этот файл запускается. Присвойте ему имя abcd.chm и с помощью Far скопируйте в один из подкаталогов кэша браузера. Зафиксируйте полный путь к этому файлу. Создайте в файле index.htm в рабочем каталоге браузера сценарий, позволяющий удаленно запустить внедренный файл (в примере указано условное имя подкаталога):

```
<SCRIPT>
window.showHelp(C:\Windows\Temporary Internet
Files\GJFTHBHD\
abcd.chm)
</SCRIPT>
```

Запустите браузер и понаблюдайте за результатом. Удалось ли запустить файл справочной системы? Какие угрозы можно реализовать путем такого запуска? Что необходимо знать злоумышленнику для успешного удаленного запуска?

- Для удаленного запуска ранее внедренных файлов злоумышленнику необходимо выяснить или подобрать случайное имя подкаталога, в который он был записан. Авторами «Секретов хакеров» предлагается два варианта:
- Вывести в окно браузера содержимое индексного файла **index.dat** и передать это изображение на удаленный узел, с которого организуется внедрение и запуск специальной программы. Для этого Г.Гунински предлагает использовать в гипертекстовом документе вставку следующего вида:

```
<OBJECT DATA="file://C:/Windows/Temporary Internet  
Files/Content.IE5/ index.dat" TYPE="text/html" WIDTH=200  
HEIGHT=200></OBJECT>
```

- Учитывая, что методы встроенного в браузер компонентной модели объектов Internet Explorer позволяют вернуть адрес предыдущего документа, Г.Гунински предлагает довольно замысловатый способ. Создается гипертекстовый документ, который содержит в себе ссылку на несколько (до десятка и более) файлов-псевдорисунков. В нем же создается ссылка на второй гипертекстовый документ, который должен загружаться с другого Web-сервера. Файлы-ресурсы перекачиваются в кэш браузера, где произвольным порядком размещаются в подкаталогах со случайными номерами. Далее можно определить имя того подкаталога, в который попал первый html-файл. Благодаря тому, что текущий документ с использованием свойств и методов компонента Internet Explorer может получить адрес предыдущего документа, представляется возможность извлечь нужную нам строку адреса и отделить от нее путь к нужному файлу. При этом мы исходим из предположения, что в подкаталог с уже известным именем попал и один из файлов-псевдорисунков, который теперь предстоит запустить. Это делается путем перебора всех имен ранее внедренных файлов в известном нам каталоге (не следует забывать, что брау-

зер вставляет три дополнительных символа [1] между именем и расширением каждого файла.

Вам предлагается исследовать возможность реализации предложенных методов.

4.5. Исследование прочих возможностей внедрения и запуска файлов

1. Исследуйте возможность запуска исполняемого файла из гипертекстового документа, предварительно скопированного пользователем в файловую систему клиентского компьютера (например, на «Рабочий стол»).
2. Поместите копию гипертекстового документа на «Рабочий стол». Исследуйте возможности запуска одного из исполняемых файлов, заведомо имеющегося в файловой системе («Блокнот», «Калькулятор», экранная заставка и др.) из html-файла с использованием атрибутов CODEBASE и DATA тэга <OBJECT>. Результаты отразите в отчете.
3. Последовательно измените расширение гипертекстового документа на .htt, .hta, .chm и попробуйте открыть их на «Рабочем столе». Вставляя в документы тэги запуска исполняемых файлов, присутствующих в системе, или сценариев ActiveX, оцените степень опасности.
4. Исследуйте возможность локального запуска элемента управления ActiveX из гипертекстовых документов с расширениями .html, .hta, .htt, .chm.

4.6. Выявление элементов управления ActiveX, зарегистрированных в качестве безопасных

1. Исследуйте элементы управления ActiveX, помеченные как безопасные для исполнения с целью выявления тех из них, которые имеют возможность обращаться к файловой системе локального компьютера, системному реестру, либо запускать исполняемые файлы на клиентском компьютере.
2. Визуально проанализируйте содержимое ключа HKCR системного реестра на предмет выявления элементов управления, зарегистрированных как безопасные для исполнения («safe for scripting»), но имеющих возможность рабо-

тать с файлами, реестром и запускать другие приложения. Для поиска используйте редактор реестра («Правка» – «Найти») с заданием строки {7DD95802-9882-11CF-9FA9-00AA006C42C4}, свидетельствующей о безопасности.

3. Установите идентификаторы класса CLSID, имеющие ключ Implemented Categories с разделом {7DD95802-9882-11CF-9FA9-00AA006C42C4}, соответствующим безопасности при использовании в сценарии. По CLSID определите имена и местоположение серверов автоматизации, реализующие безопасные функции.
4. Для просмотра свойств и методов доверенных элементов управления используйте браузер объектов редактора VBE офисного пакета Microsoft Office. Для этого найденные доверенные объекты необходимо идентифицировать по именам файлов серверов автоматизации с расширениями .osx, .dll, создать на них ссылку в браузере объектов и затем просмотреть на предмет наличия объектов, свойств и методов, позволяющих работать с файловой системой, реестром, и запускать другие программы. Поиск необходимых объектов, свойств и методов производить по общепринятым обозначениям: CopyFile, CreateTextFile, RegDelete и др.
5. Ответьте на вопрос: можно ли доверять серверу сценариев на основании декларированных свойств и методов? Почему? Можно ли автоматизировать поиск «безопасных» элементов управления?
6. Исследуйте вариант внедрения вредоносного кода с обновлением версий элементов управления ActiveX путем задания новой версии и ее расположения в Интернет с помощью атрибута CODEBASE тэга <OBJECT>. Проверку производить в следующей очередности:
 - выбрать в системном реестре (раздел HKCR) элемент управления с установленными метками безопасности для инициализации и исполнения (см. общие сведения). Этот элемент должен гарантированно присутствовать во всех версиях операционных систем Windows* и почти наверняка быть бесполезным для использования,

- по параметрам, указанным в системном реестре, либо свойствам файла сервера сценариев ActiveX установить его версию,
- создать копию файла элемента управления в рабочем каталоге Web-браузера и с помощью любого шестнадцатеричного редактора модифицировать ее (при отсутствии редактора можно использовать отладчик Debug с параметром «е»). В теле файла следует найти строку FileVersion (в Unicode), убедиться в том, что заданное после этой строки числовое значение совпадает с версией, выводимой в «Свойствах» файла, а затем увеличить цифру;
- создать гипертекстовый документ, который содержит тэг <OBJECT> с параметром CODEBASE, ссылающийся на удаленный ресурс с «обновленной» версией;
- запустить браузер и проследить, что произошло с «обновленной» версией сервера сценариев (не была скопирована, оказалась скопированной в кэш браузера, пользователю было выведено предупреждение об обновлении версий, файл в системном каталоге обновился автоматически?)

Сделайте вывод в отношении реальной защищенности клиентских компьютеров при работе в Интернет.

4.7. Контрольные вопросы

1. Как происходит запрос и получение Интернет-ресурсов с использованием универсальных браузеров?
2. Виды ссылок на ресурсы в файлах html-формата.
3. Способы подключения компонентов ActiveX к сценариям в html-файле.
4. Основные уязвимости Web-протоколов, позволяющие внедрять и запускать программный код.
5. Механизмы доверительных отношений к компонентам ActiveX, зарегистрированным в операционной системе в качестве безопасных.
6. Свойства каталога для временных файлов Интернет.
7. Интернет-атаки на пользователей и на отказ программной среды клиентского компьютера.

8. Способы запуска исполняемых программ, постоянно установленных в операционной системе.
9. Опубликованные рекомендации по внедрению постороннего программного кода.
10. Организационные и технологические меры защиты браузеров от удаленных атак из Интернет.

Библиографический список

1. Режим доступа: <http://www.guninsky.com>
2. 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.
3. 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.
4. Ахметов К.С. Microsoft Internet Explorer 4.0 для всех/ К.С.Ахметов, А.Г.Федоров. М.: КомпьютерПресс, 1997. 336 с.
5. Кришнамурти Б. Web-протоколы. Теория и практика/ Б.Кришнамурти, Дж.Рексворд. М.: ЗАО «Издательство БИНОМ», 2002. 592 с.
6. Биллиг В.А. VBA в Office 2000. Офисное программирование/В.А.Биллиг. М.: Издательско-торговый дом «Русская Редакция», 1999. 480 с.
7. 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.
8. Дарахвелидзе П.Г. Разработка Web-служб средствами Delphi / П.Г.Дарахвелидзе, Е.П.Марков. СПб.: БХВ-Петербург, 2003. 672 с.
9. Дунаев С.Б. Технологии Интернет-программирования / С.Б.Дунаев. СПб.: БХВ-Петербург, 2001. 480 с.
10. 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.
11. Касперски К. Укрощение Интернета / К.Касперски. М.: СОЛОН-Р, 2002. 288 с.
12. Лукацкий А.В. Вопросы информационной безопасности, возникающие при использовании технологий Java и ActiveX. Тематический выпуск № 2 / А.В.Лукацкий. М.: 1998.

13. Матросов А.В. HTML 4.0./ А.В.Матросов, А.О.Сергеев, М.П.Чаунин. СПб.: БХВ-Петербург, 2001. 672 с.
14. Орлов А.А. VBA: для тех, кто любит думать / А.А.Орлов. М.: СОЛОН-Р, 2002, 240 с.
15. Попов А.В. Командные файлы и сценарии Windows Script Host / А.В.Попов. СПб.: БХВ-Петербург, 2002. 320 с.
16. Скэмбрей Д. Секреты хакеров. Безопасность Windows 2000 – готовые решения : пер. с англ./Джоел Скэмбрей, Мак-Клар Стюарт. М.: Издательский дом «Вильямс», 2002. 464 с.
17. 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.
18. Таненбаум Э. Современные операционные системы. 2-е изд. / Э.Таненбаум. СПб.: Питер, 2002. 1040 с.
19. Торрес Дж. Скрипты для администратора Windows. Специальный справочник / Дж. Торрес. СПб.: Питер, 2002. 336 с.
20. Хоникатт Д. Реестр Windows 2000: уч. пос.; пер. с англ. / Дж. Хоникатт. М.: Издательский дом «Вильямс», 2000. 320 с.
21. Рофэйл Э. СОМ и СОМ+. Полное руководство: пер. с англ./ Эш Рофэйл, Ясер Шохауд. Киев : ВЕК+, Киев : НТИ; М.: Энтроп, 2000. 560 с.

Учебное пособие

Валентин Викторович Бакланов

ЗАЩИТА КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ В КЛИЕНТСКИХ ПРИЛОЖЕНИЯХ

Редактор – *Н.П.Кубыщенко*

Компьютерная верстка – авторская

Подписано к печати 16.06.2005		Формат 60x84 1/16	
Бумага типографская	Офсетная печать	Усл. печ. л. 4,88	
Уч. - изд. л. 6,0	Тираж	Заказ	Цена “С”

Редакционно-издательский отдел ГОУ ВПО УГТУ-УПИ

620002, Екатеринбург, ул. Мира, 19

Ризография НИЧ ГОУ ВПО УГТУ-УПИ

620002, Екатеринбург, ул. Мира, 19