Jet Info

ИНФОРМАЦИОННЫЙ БЮЛЛЕТЕНЬ № 1 (152)/2006

Скрытые каналы – еще не известные угрозы информационным ресурсам

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Скрытые каналы – еще не известные угрозы информационным ресурсам

СОДЕРЖАНИЕ

Проблемы информационной безопасности вызывают сегодня повсеместную заинтересованность: разрабатываются системы защиты информации, обеспечиваются их организационно-нормативная поддержка и сопровождение. При этом всегда существует вопрос, насколько полон список угроз, для ликвидации которых используются и организационные, и технические меры, а кроме того, насколько актуальны все угрозы, входящие в этот список.

Динамика развития информационных технологий заставляет постоянно заглядывать вперед, определяя не только какие технологии окажутся востребованными в ближайшее время, но и какого рода угрозы и риски появятся в связи с развитием информационных технологий.

Одну из таких потенциальных угроз — скрытые каналы передачи данных — рассматривают авторы статей, представленных в настоящем номере журнала. Сегодня это, несомненно, не первостепенная угроза, однако ее особенностью является то, что средства ее предотвращения практически отсутствуют, а значит, реализация данной угрозы может привести к значимому ущербу для информационных систем.

Мы предлагаем две точки зрения на эту проблему. Аргументы в поддержку обеих представляются значимыми, однако, не выделяя ни одну из них, предоставляем возможность читателям Jet Info сделать свои выводы.

Мы предупредили, что такая угроза существует.

РАСПРЕДЕЛЕННЫЕ АТАКИ НА РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

Доктор физико-математических наук А.А. Грушо,
кандидат физико-математических наук Е.Е. Тимонина

Многоагентные системы создавались для интеграции неоднородных приложений. Но и противник также нуждается в интеграции и координации своих действий для того, чтобы атаковать распределенную систему. Противник может также строить свою многоагентную систему для проведения распределенной атаки на распределенную систему. Чтобы оценить опасность таких атак, мы рассмотрим требования, которым должна удовлетворять такая враждебная многоагентная система. Мы также анализируем возможность выполнения этих требований[1].

ВВЕДЕНИЕ

Рассмотрим сеть, состоящую из узлов и каналов между ними. Распределенную информационную систему можно представить как приложения в узлах сети, связанные некоторой интегрирующей системой, которая позволяет координировать использование различных приложений в решении задач. Интеграция может быть достигнута с помощью многоагентной системы (МАС).

Некоторые МАС описаны в научной литературе, например, стандарт FIPA для интеллектуальных агентов [30]. System Research Institute опубликовал свои результаты в Open Agent Architecture [23]. Много работ посвящено COUGAAR [14]. МАС присутствует в GRID [21]. Мы говорим о высоко критичных распределенных системах [24], если потери велики при серьезных сбоях в таких системах. Мы предполагаем наличие злоумышленника и то, что распределенная система может подвергнуться его атаке для нанесения ущерба владельцу информации и системы.

Для предотвращения ущерба мы должны выяснить, каким образом противник может атаковать распределенную систему. Это поможет понять, как определить безопасность распределенной системы и каким образом она может быть достигнута.

Любая атака может быть разбита на два этапа. Первый включает сбор информации об атакуемой системе и организацию атаки, второй — проведение самой атаки. На первом этапе разрабатывается план для всех участников атаки с информацией о том, что следует делать в определенных обстоятельствах во время проведения атаки. На втором этапе атаки противник стремится получить доступ к уязвимым точкам, через которые наносится ущерб, а затем скрывает следы атаки, чтобы предотвратить обвинения против атакующих.

Мы предполагаем, что любая атака на отдельный хост распределенной системы не приводит к фатальному ущербу. Атака должна быть распределенной, т. е. представлять собой распределенную атаку как скоординированные действия, произведенные в различных подсистемах распределенной информационной системы с целью нанесения ущерба.

Рассмотрим, что необходимо использовать противнику для выполнения распределенной атаки. Конечно, существуют механизмы безопасности в распределенной системе. К ним относятся аудит или системы обнаружения вторжений, контроль целостности, контроль доступов, межсетевые экраны и др. Как минимум, первый этап атаки должен быть «невидимым» для механизмов защиты.

При организации атаки необходим механизм интеграции деятельности всех противников. Рассмотрим ситуацию, когда они представлены программно-аппаратными агентами в компьютерной среде. Тогда интеграция действий противника в атаке может осуществляться с помощью враждебной многоагентной системы (ВМАС). Чтобы решать задачи по интеграции деятельности злоумышленника, ВМАС должна представлять собой сеть. Для этого необходимы коммуникации между агентами, и каналы должны быть скрытыми от средств защиты.

В настоящей статье авторы пытаются найти ответ на вопрос, могут ли данные свойства быть выполнены.

В основу ВМАС заложены три хорошо известные модели. Первая — это *модель невлияния*

[1] Работа поддержана грантом РФФИ, проект № 04-01-00089

[1,2,17,18,25,27]. Если выполняются условия невлияния, то можно доказать, что агенты «невидимы» для механизмов защиты. Примеры реализации «невидимых» агентов будут приведены далее.

Вторая модель — *модель скрытых каналов*. Она определяет возможность существования коммуникаций, «невидимых» для механизмов защиты. Частично такие коммуникации внутри компьютера описаны в работе [13]. Скрытые каналы в открытых сетях часто называют стеганографией. В рассматриваемой задаче скрытые от средств защиты каналы должны существовать в условиях, когда используются механизмы защиты, которые, «не видя» скрытый канал, могут препятствовать его существованию. В таких случаях мы говорим о скрытых каналах, преодолевающих защиту.

Третья модель — *Open Agent Architecture* (OAA) [23]. Она показывает, как BMAC может быть сделана интеллектуальной.

Мы не строили большую BMAC, но проведенные эксперименты подтверждают все положения настоящей статьи.

«НЕВИДИМЫЕ» АГЕНТЫ

В стандарте FIPA [30] агент определяется как сущность, способная вести себя автономно, выполнять план и распознавать другие сущности в протоколах, в которых они участвуют. Есть по крайней мере два протокола, в которых каждый агент BMAC является участником. Первый — протокол взаимодействия агентов противника. Второй — это легальный протокол, содержащий агента в виде встроенного в некоторую область кода и процесса, использующего ресурсы компьютера.

Пусть компьютерная система разделена на два уровня — High и Low. Если субъект на уровне High может выполнять все свои действия и способен «видеть» действия субъекта на уровне Low, но любой субъект на уровне Low не может «видеть» никаких действий или их результатов на уровне High, то тогда система удовлетворяет условию невлияния. Если враждебный агент находится на уровне High, а механизмы защиты находятся на уровне Low, и выполняются условия невлияния, то агент не может быть «увиден» средствами защиты.

Мы должны ответить на вопрос, каким образом агент, будучи участником легального протокола, остается «невидимым» для механизмов защиты. Рассмотрим следующую примитивную схему, моделирующую механизмы «невидимости».

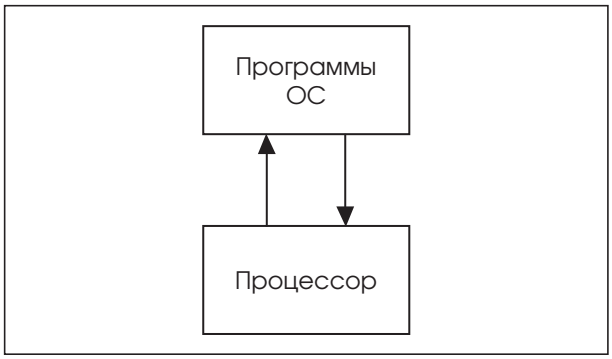

Рис. 1 Схема работы компьютера

На рис. 1 приведена нормальная схема работы компьютера. Отметим, что процессор только выполняет команды, и ему все равно, кто их передает.

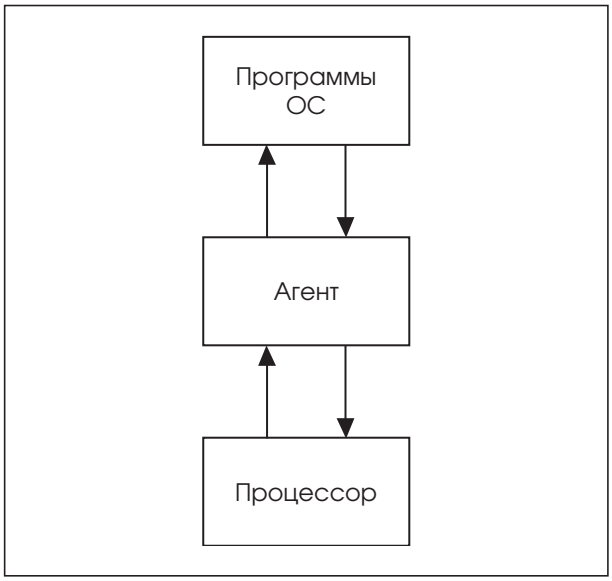

Рис. 2 Враждебный агент в схеме компьютера

На рис. 2 агент «видит», что «хотят» сделать программы, куда обратиться, что и как обработать. Тогда программный агент может использовать процессор до работы легальных программ. То есть агент может «подправить» все так, как нужно ему, затем обработать реальный запрос и передать легальным программам ту информацию, которую ему нужно.

Эта модель описывает теоретическую возможность работы «невидимого» агента. На практике можно разместить агента в ядре операционной системы таким образом, что он будет контролировать и модифицировать всю деятельность системы безопасности. Данная конструкция реализует метод «men-in-the-middle» [3], она была выполнена в некоторых практических разработках. Программы такого типа можно найти в Интернете. Они называются руткиты.

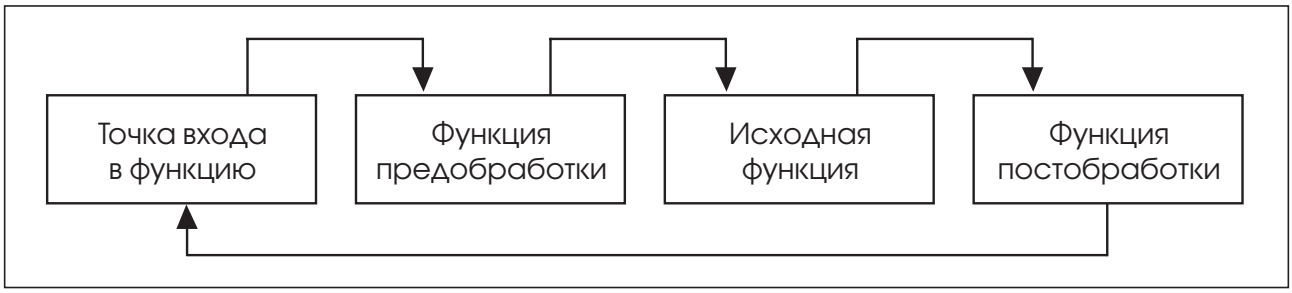

Рис. 3 Процесс перехвата вызовов функции API

Большинство реализаций современных руткитов могут прятать от пользователя файлы, папки и ключи реестра, скрывать запущенные программы, системные службы, драйверы и сетевые соединения. То есть злоумышленник имеет возможность создавать файлы и ключи реестра, запускать программы, работать с сетью, и эта активность не будет обнаружена пользователями, в том числе и администратором системы.

В настоящее время известно около двух десятков подобных проектов, в рамках которых разрабатываются способы маскировки файлов, работающих процессов, сеансов межсетевого взаимодействия и т.п. Приведем несколько примеров.

Hacker Defender является программным агентом, функционирующим на платформах Windows NT 4.0/2000/XP, а также на более поздних версиях систем на базе Windows NT. Основная идея данной программы состоит в перезаписи некоторых сегментов памяти во всех запущенных процессах. При этом описанные действия не должны оказывать влияния на стабильность работы системы или работающих процессов. Программа может быть абсолютно «невидима» для всех остальных процессов, в том числе и для средств защиты [28,29]. Тем самым противник имеет возможность скрывать файлы, процессы, системные службы и драйверы, ключи реестра и их значения, открытые порты. Программа также осуществляет маскировку своих действий в памяти и прячет идентификаторы скрываемых процессов. Кроме того, этот агент инсталлирует скрытый «blackdoor», регистрируется как скрытый системный сервис и инсталлирует скрытый системный драйвер. Технология «blackdoor» позволяет внедрять редиректор (сетевое программное обеспечение, эмулирующее доступ прикладных программ к удаленной файловой системе как к локальной).

После запуска программы Hacker Defender перечисляет все доступные для нее процессы в системе, а затем пытается перехватить определенные вызовы API [12]. Перехват заключается в замене первых байт кода функции на безусловный переход к новому коду функции, предварительно сохраненному в адресном пространстве программы. Для этого выясняется адрес необходимой функции, затем в памяти программы отводится место под код нового варианта функции и ее первоначального кода, который сохраняется для дальнейшего использования. Таким образом, когда пользовательская программа вызывает функцию API, например, NtQuerySystemInformation, вызов передается функции предобработки данных. Затем может быть вызвана исходная функция, которая, в свою очередь, вернет результаты функции постобработки. Функция постобработки модифицирует данные, которые вернула исходная функция, например, удаляя некоторые записи.

Процесс перехвата вызовов API изображен на рис. 3.

Получается, что программы, которые будут использовать перехваченные вызовы API, получат информацию не о реальном положении дел в системе, а уже обработанные агентом данные. Hacker Defender перехватывает функции запуска новых процессов, что позволяет агенту передавать нужный код через новые программы, запускаемые пользователем.

FU также является известным программным агентом, исходный код которого опубликован 03.09.2004. FU состоит из двух компонент: драйвера Windows 2000/XP (msdirectx.sys) и собственно исполняемого файла (fu.exe). Если драйвер успешно установлен, то противнику необходимо просто запустить файл fu.exe на исполнение. Заметим, что при этом противнику вовсе не нужно обладать какими-либо особыми правами, в том числе и правом запуска приложений. Данная программа непосредственно манипулирует объектами ядра операционной системы. FU модифицирует список PsActiveProcessList, содержащий список активных процессов, информацию из которого получает ZwQuerySystemInformation. При этом процесс остается существовать в качестве «свободного» потока и будет нормально функционировать, поскольку распределение процессорного времени Windows основано на потоках, а не на процессах. FU позволяет нарушителю скрывать файлы, каталоги и про-

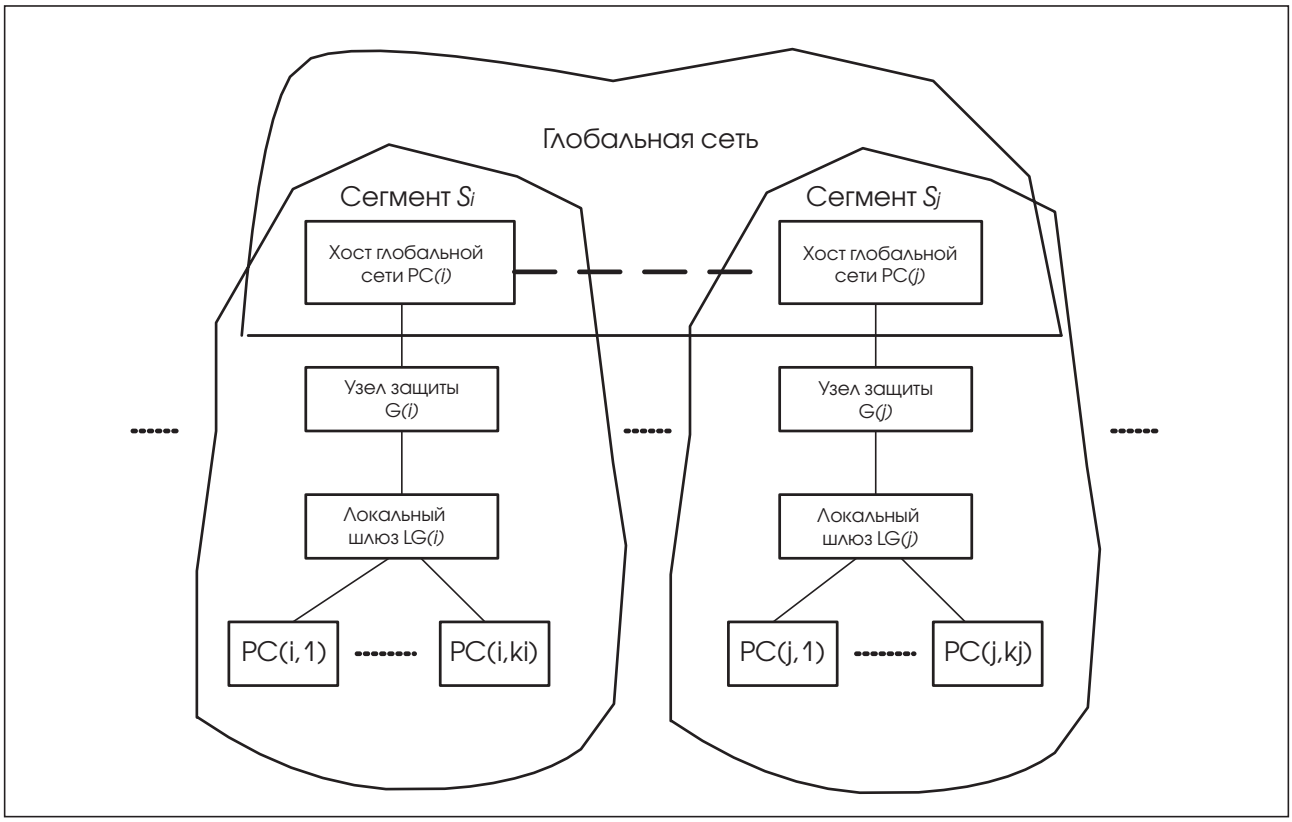

Рис. 4 Схема глобальной сети

цессы, а также изменять привилегии процесса, поднимая его до уровня администратора.

AFX Rootkit представляет собой программного агента, который функционирует на платформах Windows NT/2000/XP/2003. Текущая версия данной программы позволяет нарушителю скрывать процессы и их идентификаторы, файлы и директории, значения системного реестра, сервисы, сетевые соединения TCP/UDP, иконки в панели задач. При установке этого агента создается директория с уникальным именем, в которой противник сможет размещать необходимые ему данные, в том числе исполняемый файл (root.exe). Заметим, что данная директория абсолютно «невидима» другими процессами, файлами, библиотеками и т.д., однако все программы внутри директории не являются скрытыми друг от друга.

«Невидимый» агент может сделать другие файлы или процессы «невидимыми». Более того, «невидимый» процесс может решить проблему построения «невидимых» коммуникаций между агентами противника в рамках одного компьютера.

В работе [1] мы доказали, что если есть сеть автоматов, каждый из которых удовлетворяет условию невлияния, и коммуникации на уровне High «невидимы» для механизмов безопасности, тогда сеть автоматов также удовлетворяет условию невлияния.

СКРЫТЫЕ КАНАЛЫ

Итак, мы описали, каким образом можно построить «невидимый» канал между агентами на уровне High компьютерной системы с одной операционной системой. Но распределенная система основана на сети, в которой уровень Low, как правило, находится под контролем системы безопасности. Например, можно использовать снифферы. Здесь нет канала на уровне High. Тогда «невидимость» канала между агентами противника, использующими уровень Low, означает, что агенты уровня High могут прятать информацию в легальных каналах уровня Low. Здесь возникают три проблемы. Первая — спрятать спуск информации с уровня High на уровень Low от механизмов защиты. Вторая — это преодоление механизмов защиты, предназначенных для уничтожения скрытых каналов. И третья проблема — способность агентов быть интеллектуальными и использовать стойкие методы стеганографии.

Первая проблема может быть решена, так как организация встраивания информации уровня High в предложенных далее скрытых каналах возможна с помощью программно-аппаратных агентов в компьютерной среде. В самом деле, предположим, что функция организации скрытого канала для связи с внешней средой реализуется програм-

мно-аппаратным агентом. Тогда данный агент находится в конкретном компьютере под охраной другого агента, который делает его «невидимым» для средств защиты.

Показать, что функции построения скрытого канала доступны для программно-аппаратного агента, помогут примеры решения второй проблемы — преодоление механизмов защиты, предназначенных для уничтожения скрытых каналов.

Рассмотрим схему сети (рис. 4).

Пусть имеется m+1 сегментов локальных вычислительных сетей, S_0, S_1,...,S_m, в каждом из которых есть рабочие станции со своими локальными адресами и шлюзы, соединяющие локальные сети с глобальной сетью (например, Интернет). Пусть s_0, s_1, ..., s_m — адреса шлюзов сегментов локальных вычислительных сетей S_0, S_1, ..., S_m, которые представляют эти сегменты в глобальной сети. Мы считаем, что для общения между собой рабочие станции различных сегментов используют виртуальную частную сеть (VPN), которая работает на основе протокола IPSec следующим образом. Пакет от рабочей станции с адресом a в сегменте S_i должен быть передан на рабочую станцию с адресом b в сегменте S_j. Пакет передается так:

- от машины с адресом a к локальному шлюзу LG(i) в сегменте S_i;
- далее пакет от LG(i) попадает на узел защиты G(i), в котором пакет шифруется и инкапсулируется в пакет (пакеты) с адресом отправителя s_i и адресом получателя s_j, пакеты имеют одинаковую длину и другие одинаковые параметры;
- инкапсулированный пакет из G(i) попадает на хост глобальной сети PC(i), который направляет его через глобальную сеть на аналогичный хост PC(j);
- далее пакет направляется на узел защиты G(j), на выходе которого восстанавливается исходный пакет, посланный от a к b, этот пакет поступает на локальный шлюз LG (j);
- LG (j) отправляет пакет к абоненту b в сегменте S_j.

В глобальной сети и в каждом сегменте S_0, S_1 ,..., S_m имеются программно-аппаратные агенты противника, которые для выполнения враждебных функций должны получать инструкции от программно-аппаратного агента противника [7] из глобальной сети (ПГС). Будем считать, что противник из глобальной сети через своих агентов полностью контролирует хосты PC(k), $k = 0, 1, ..., m$. Программно-аппаратные агенты внутри сегментов локальной сети S_0, S_1, ..., S_m контролируют соответственно компьютеры LG(j), $j = 0, 1, ..., m$. Мы считаем, что узлы защиты G(i) сделаны правильно так, что никто из злоумышленников не может их контролировать. Таким образом, управление программно-аппаратным агентом в любом из сегментов со стороны противника из глобальной сети связано с построением канала связи от PC(j) к LG(j). Утечка информации связана с построением канала от LG(j) к PC(j).

Ни один из хостов глобальной сети (в частности, каждая машина PC(j), $j = 0, 1,..., m$) не знает внутренние адреса сегментов. В силу того, что шифрование и формирование пакетов для отправки через глобальную сеть происходит на узлах защиты, противник не может построить канал взаимодействия с программно-аппаратным агентом сегмента, используя шифртекст или служебные атрибуты пакетов. Таким образом, мы предполагаем, что единственными зависимыми параметрами, известными на PC(j) и LG(j) для входного потока пакетов, являются адреса отправителя пакетов. При передаче пакетов от LG(j) к PC(j) единственными зависимыми параметрами являются адреса их получателей. Эта зависимость связывает внутренние адреса каждого сегмента и внешний адрес s соответствующего шлюза.

Скрытый канал, не требующий обучения, строится следующим образом [8]. Возьмем часто встречающиеся адреса s_1 и s_2 из множества $S = \{s_1, ..., s_m\}$. Модулируем поток пакетов из PC(0) в G(0) следующим образом. Опишем, например, код для 1. При передаче 1 любой очередной пакет с исходящим адресом s_1 передается после передачи нечетного числа пакетов с другими адресами. Поэтому все расстояния между пакетами с исходящими адресами s_1 являются четными. Рассмотрим последовательность расстояний между исходящими адресами, полученными в LG(0). Все расстояния между адресами пакетов из сегмента S_1 — четные (кроме ошибок вида потери или вставки пакета, которые мы считаем достаточно редкими). Агент в LG(0), обладая достаточными вычислительными ресурсами и памятью, считает четность или нечетность расстояний между повторяющимися адресами и выявляет отклонение четных расстояний от случайного распределения длин расстояний между повторяющимися адресами из S_1. Аналогично, пакет с исходящим адресом s_2 передается через нечетное число пакетов с другими адресами. Таким образом, преобладание четных расстояний между адресами из S_1 и четных расстояний между адресами из S_2 определяет 1 в коде.

Аналогично строится код для передачи 0 и x, означающий конец передачи кода 1 или 0.

Пропускная способность данного канала однозначно связана с другим параметром, связанным с данным способом скрытой передачи информации. Это число N переданных пакетов для восстановления одного бита переданной информации.

Полученные асимптотические оценки показывают, что вероятность правильно принять сигнал 1 или 0 или *x* агентом в LG(0) стремится к 1 при *N* стремящемся к ∝.

Пусть адреса упорядочены s*1* < s*2* < ... < s*m*. Рассмотрим другой метод общения агентов. Этот метод требует обучения. Процедура обучения необходима для того, чтобы максимально полно восстановить на LG(0) порядок.

Процедура обучения состоит в следующем. PC(0) получив очередные k пакетов с адресами s*i1*, s*i2*, ..., s*ik*, упорядочивает их в соответствии с возрастанием и посылает данные пакеты на узел защиты для последующей передачи LG(0). Мы доказали [5], что LG(0) в реальном масштабе времени может проводить сопоставление полученных данных и восстанавливать порядок. Тогда можно передавать информацию упорядоченными k-группами адресов, переставляя их на PC(0) в нужном порядке..

Пример скрытого канала по времени, который также преодолевает IPSec, исследован в [16]. В данном случае для скрытой передачи информации используется задержка между передачей последовательных пакетов в потоке. В указанной работе рассмотрены четыре различных модели потока (с дискретным временем и дискретным (пакетным) трафиком, с непрерывным временем и дискретным трафиком, с непрерывным трафиком и временем, а также непрерывные трафик и время с ограничением скорости передачи). В рамках каждой из моделей для трех алгоритмов подавления (задерживающих пакеты с ограничением числа задерживаемых пакетов, величины задержки, значения средней скорости передачи) оценена зависимость пропускной способности скрытого канала от параметров алгоритма подавления. Следует отметить, что ни один из рассмотренных алгоритмов не способен полностью перекрыть скрытый канал, а может лишь снизить его пропускную способность.

При использовании задержек между пакетами в качестве «контейнера» для скрытой передачи может применяться не весь поток TCP-пакетов, а лишь серия пакетов, отправляемых подряд без подтверждения.

Возможны и другие каналы преодоления IPSec, но они более уязвимы к уничтожению. Например, можно модулировать передачу длинами пакетов, если IPSec не производит выравнивания длин пакетов.

Рассмотрим скрытые каналы, преодолевающие PROXY-серверы [15].

Прежде, чем подробно объяснить механизмы преодоления защиты, опишем общую идею. PROXY-сервер, в отличие от IPSec, устанавливает TCP-соединения с непосредственно связанными с ним абонентами. При этом данные, передаваемые пакетами, собираются на PROXY-сервере, а затем передаются при образовании TCP-соединения с таким же сервером на приемном конце. Сложность преодоления надежно защищенного PROXY-сервера состоит в том, что враждебный агент может манипулировать или наблюдать манипуляции с пакетами, а не с данными. Однако в основе преодоления PROXY-сервера лежит простая идея управления порядком данных с помощью задержек пакетов, передающих эти данные. Более подробно рассмотрим следующую модель.

Пусть, как и раньше, имеется $m+1$ сегментов локальных вычислительных сетей $S_0, S_1, ..., S_m$, в каждом из которых имеются рабочие станции со своими локальными адресами и шлюзы, используемые для соединения локальных сетей с глобальной сетью (например, Интернет). Пусть $s_0 < s_1 < ... < s_m$ упорядоченные адреса шлюзов сегментов локальных вычислительных сетей $S_0, S_1, ..., S_m$, представляющих эти сегменты в глобальной сети. Мы считаем, что для общения между собой рабочие станции различных сегментов используют виртуальную частную сеть (VPN), которая работает следующим образом. Данные от рабочей станции с адресом *a* в сегменте S_i должны быть переданы на рабочую станцию с адресом *b* в сегменте S_j. Схема связи изображена на рис. 4, а связь организуется следующим образом.

1. От машины с адресом a пакеты с передаваемыми данными, посылаются к внутреннему шлюзу – машине LG(*i*) в сегменте S_i.
2. Далее пакеты попадают на шлюз – узел защиты G(*i*). Он работает как PROXY-сервер, который устанавливает соединение с машиной с адресом *a* и восстанавливает данные, передаваемые от рабочей станции с адресом *a* на рабочую станцию с адресом *b*. Восстановленные данные, поступающие на PROXY-сервер от абонентов сегмента S_i, выстраиваются в очередь в соответствии с порядком их восстановления на PROXY-сервере. Далее PROXY-сервер шифрует данные в очереди на ключе получателя – PROXY-сервера G(*j*), в сегменте которого находится получатель данных. После этого G(*i*) устанавливает соединение с G(*j*) и передает туда пакеты, содержащие зашифрованные данные.
3. Пакет из G(*i*) попадает на хост глобальной сети PC(*i*), который направляет его через глобальную сеть на аналогичный хост PC(*j*).
4. Далее пакет направляется на узел защиты G(*j*), где восстанавливаются и выстраиваются в очередь данные, передаваемые на S_j. При подходе соответствующих данных в этой очереди

PROXY-сервер УЗ(j) расшифровывает их и устанавливает соединение в сегменте S_j с рабочей станцией с адресом b. Данные передаются пакетами через внутренний шлюз LG(j).

5. LG(j) отправляет пакеты к абоненту b в сегменте S_j.

Как и раньше, в глобальной сети и в каждом сегменте $S_0, S_1, ..., S_m$ имеются программно-аппаратные агенты противника, которые для выполнения своих враждебных функций должны получать инструкции от программно-аппаратного агента противника из глобальной сети (ПГС). Будем считать, что ПГС полностью контролирует компьютеры PC(j), $j = 0, 1, ..., m$. Программно-аппаратные агенты противника внутри сегментов локальной сети $S_0, S_1, ..., S_m$ контролируют соответственно компьютеры соответственно LG(j), $j = 0, 1, ..., m$. Мы считаем, что узлы защиты G(i) сделаны правильно так, что никто из противников не может их контролировать. Таким образом, управление программно-аппаратным агентом в любом из сегментов со стороны противника из глобальной сети связано с построением канала связи от PC(j) к LG(j). Утечка информации связана с построением канала от LG(j) к PC(j).

Как и раньше мы предполагаем, что единственными зависимыми параметрами, известными на PC(j) и LG(j) для входного потока пакетов, являются адреса отправителя пакетов. При передаче пакетов от LG(j) к PC(j) единственными зависимыми параметрами являются адреса получателей пакетов. Эта зависимость выражается в связи множества внутренних адресов каждого сегмента и внешнего адреса s соответствующего шлюза и считается в данной задаче известной.

LG(i) может влиять на порядок очереди отправляемых данных следующим образом. Протокол TCP гарантирует восстановление данных. Если какой-то пакет не получен, то протокол TCP передает запрос на получение пропущенных данных. Пусть на PROXY-сервере установлено два соединения с абонентами сегмента a_1 и a_2, которые передают данные A_1 и A_2 соответственно на PROXY-сервер G(i). Естественно, что те данные, которые восстановлены первыми, будут первыми поставлены в очередь на передачу в глобальную сеть. Пусть агент в LG(i) хочет, чтобы последовательность данных в очереди на отправку была: $A_1 A_2$. Если пакеты, передающие A_1, заканчиваются быстрее, чем пакеты, передающие A_2 (агент наблюдает поток пакетов и может осуществлять их задержку, чтобы убедиться в том, что передача данных закончена), то агент не предпринимает никаких действий. Если пакеты, передающие A_2, заканчиваются быстрее, чем пакеты, передающие A_1, то агент на LG(i) задерживает или уничтожает один из пакетов (например, последний) из серии пакетов, передаваемых от a_2. После этого он выжидает, когда закончится поток пакетов, передающих файл A_1, и посылает последний пакет с данными A_2. Тогда, если верно его предположение о том, что серия пакетов от a_1 одному абоненту в другом сегменте содержала данные одного соединения A_1, а аналогичная серия от a_2 одному абоненту в другом сегменте содержала данные другого соединения A_2, то указанная процедура поменяет естественный порядок данных в очереди на G(i) на требуемый порядок данных. Разумеется, такая же процедура перестановки на узле G(j) возможна и на потоке входящих данных с помощью агента на PC(j). Отметим, что указанная процедура носит вероятностный характер. Ясно, что вероятность правильной перестановки увеличивается для больших серий пакетов, передаваемых при отправке больших файлов.

Несмотря на возможность ошибки, можно построить язык скрытой передачи данных, использующий данный алгоритм перестановки данных в очереди. Пусть агент в LG(0) передает информацию агенту в PC(0) в глобальной сети. Тогда агент в LG(0) знает, какие данные идут на адрес получателя s_j, $j = 1, ..., m$, серверов G(j). С помощью алгоритма перестановки данных в очереди агент в LG(0) упорядочивает очередные пары пришедших данных.
Блоки полученной последовательности данных разбиваются на пакеты. Серии этих пакетов передаются по соответствующим адресам. В серии пакетов могут вкрапливаться отдельные пакеты, связанные, например, с установлением других соединений. Поэтому агент в PC(0) не должен учитывать эти вкрапления. Полученные таким образом серии однозначно соответствуют переданной единице. Передача нуля осуществляется последовательностью непересекающихся монотонно убывающих пар адресов. Последовательность с непересекающимися чередующимися возрастающими и убывающими парами адресов однозначно определяет символ x, соответствующий разделительному символу. При этом мы предполагаем, что PROXY-сервер последовательно устанавливает соединения с другими PROXY-серверами в соответствии с получившейся в буфере очередью данных. При этом данные, передаваемые по одному адресу, передаются в одном соединении.

В связи с описанными скрытыми каналами решена [15] задача оценки r для устойчивого выделения 1, 0 и x, а также устойчивости передачи к сбоям.

В модели потока, создаваемого протоколом TCP, для создания скрытого канала могут быть использованы задержки подтверждения. Отправив один или несколько пакетов с данными, узел ожида-

ет подтверждения их получения, и лишь затем продолжает передачу. Число пакетов, которые могут быть переданы подряд без ожидания подтверждения, определяется размером «окна» принимающей стороны (этот параметр используется для управления скоростью передачи и задает объем данных, которые получатель способен принять, не прерывая передачу [26]). Таким образом, в Интернете задержки между пакетами, отправленными подряд, определяются параметрами канала передачи пакетов.

Задержки же между пакетами, перед отправкой которых узел ожидал подтверждения поставки предыдущих пакетов, в большей степени определяются скоростью обработки пакетов узлами, а также особенностями протокола уровня приложений (задержка отправки пакетов может быть вызвана ожиданием какого-либо события приложением). Поэтому возможна манипуляция ПГС задержками подтверждений, которая может быть использована в статистическом [13] скрытом канале низкой пропускной способности.

Для построения скрытого канала возможно использование модуляции скорости передачи информации через Интернет. Как и в предыдущем случае, такой скрытый канал преодолевает IPSec.

В связи с тем, что пропускная способность каналов Интернет не безгранична, а число пользователей сети неуклонно растет, поставщиками услуг Интернета повсеместно применяются ограничения пропускной способности предоставляемых пользователям каналов связи. Поскольку при этом обычно абонент соединен с оператором скоростной линией связи (например, локальной сетью), ограничение скорости передачи производится искусственно, программными средствами шлюза. Как правило, алгоритм ограничения скорости основан на следующей схеме: поступающие по «быстрому» каналу на шлюз пакеты помещаются в буфер, а затем постепенно отправляются по «медленному» каналу. При этом фактически пропускная способность «медленного» канала может быть выше заданного ограничения. Задержки при отправке пакетов, помещенных в буфер, выбираются таким образом, чтобы средняя скорость передачи соответствовала установленным ограничениям.

Чаще всего наибольшая нагрузка на канал связи возникает при последовательной передаче больших объемов данных. При этом для ускорения передачи протокол TCP формирует пакеты определенной длины (как правило, 1500 байт) и полностью использует приемное «окно» получателя. В потоке возникают серии пакетов, отправленных с минимальными (для физического канала) интервалами. После обработки алгоритмом ограничения скорости эти интервалы увеличиваются для обеспечения заданной средней скорости. Изменение скорости передачи [11] происходит за счет манипуляции интервалами между пакетами. Тогда шлюз, через который проходят пакеты, имеет возможность встраивать в поток пакетов скрытую информацию с помощью манипуляции скоростями. Для того чтобы интервалы между пакетами, определяющие скорость передачи, не изменились на маршруте от данного шлюза до получателя, серия пакетов не должна полностью помещаться в буфер ни на одном из шлюзов на этом маршруте. В противном случае скрытая информация, содержащаяся в потоке, окажется искаженной.

Во избежание искажения интервалов для прохождения медленного канала при модуляции необходимо выбирать интервалы таким образом, чтобы они были не короче интервалов, соответствующих самому медленному каналу на маршруте. Оценка скорости самого медленного канала может быть произведена путем измерения средней скорости передачи.

Опишем способ скрытой передачи информации агенту, контролирующему единую точку входа [9].

Рассмотрим схему аутентификации пользователя, обращающегося по протоколу LDAP на сервер директорий. Предположим, что там есть программно-аппаратный агент нарушителя безопасности, который знает, как работает справочная служба, но не имеет предметно-ориентированной информации и знаний по использованию своих возможностей. Можем считать, что потенциал такого агента сравним с возможностями администратора сервера в том случае, если он обладает предметно-ориентированной информацией и имеет конкретное задание по администрированию информации на сервере.

Пусть сервер аутентификации обслуживает большую распределенную сеть с большим числом пользователей и разграниченными правами доступа к ресурсам системы. Для того чтобы войти в сеть и получить права доступа к определенным ресурсам, пользователь посылает на сервер аутентификации пароль, передается по транспортной сети в шифрованном виде, причем система шифрования не зависит от идентификатора пользователя. Пусть нарушитель безопасности является легальным пользователем систем. Он может подключиться к сети на своем компьютере в случае успешной аутентификации на сервере и хочет получить доступ к запрещенным для него ресурсам системы. Права доступа к ресурсам для любого пользователя записаны в дереве директорий сервера аутентификации. Если злоумышленник пытается получить

доступ к запрещенным для него ресурсам, это отражается в данных аудита.

Для решения задач нарушителя безопасности самым простым способом является передача специального пароля, известного его программно-аппаратному агенту на сервере аутентификации, с запросом необходимых для него запрещенных прав к ресурсам системы. В этом случае программно-аппаратный агент дает права пользователю на требуемые ресурсы и может очистить данные аудита на сервере идентификации/аутентификации. Однако, имея распределенную систему аудита, доступ к запрещенной для этого пользователя информации будет отражен в других системах аудита, например, в данных аудита СУБД. При правильно работающей системе анализа данных аудита служба безопасности получит информацию о том, что данный пользователь с данной рабочей станции получил ценную информацию. Поэтому, войдя на рабочую станцию от своего имени, нарушитель безопасности должен обращаться со специальным паролем к агенту на сервере аутентификации, называя вымышленный идентификатор, то есть от чужого имени.

Таким образом, приходим к задаче создания в системе с помощью программно-аппаратного агента на сервере аутентификации ложного пользователя с большими правами по доступу к ресурсам системы. При наличии специального пароля, известного программно-аппаратному агенту, эта задача легко решается. Чтобы предотвратить такую угрозу, между системой приема сообщений от пользователя и сервером аутентификации установим узел защиты, который шифрует истинные значения идентификаторов и паролей. Пусть в дереве директорий в качестве имен используются шифртексты идентификаторов. Тогда программно-аппаратный агент не сможет получить специальный пароль от нарушителя безопасности и приписать ему особые права.

Однако эта защита может быть преодолена следующим образом. Нарушитель безопасности посылает любой настоящий идентификатор и случайный пароль к нему. После шифрования на узле защиты в дереве пользователей идентифицируется некоторый пользователь по присланному идентификатору. Естественно, присланный пароль этого пользователя отличается от истинного пароля. Выявляя этот факт, программно-аппаратный агент принимает решение о создании нового пользователя, который является листом в дереве директории для истинного пользователя, а его идентификатор соответствует присланному паролю, зашифрованному узлом защиты. После этого он посылает на компьютер нарушителя безопасности сигнал о неправильной аутентификации полученного идентификатора пользователя. Получив этот сигнал, нарушитель безопасности посылает на сервер аутентификации идентификатор, в котором присутствует выбранное до того имя легального пользователя, дополненное переданным ранее паролем. Для данного пользователя он выдумывает специальный пароль и передает его для аутентификации ложного пользователя. Агент нарушителя находит имя ложного пользователя, созданного им на предыдущем шаге, и в качестве аутентификатора записывает шифртекст переданного пароля. Затем он посылает сигнал о правильной аутентификации, приписывает данному пользователю максимальные права, стирает данные аудита о создании нового пользователя и о присвоении ему прав, временно заканчивает свое функционирование и маскирует свое присутствие.

Потом в системе появляется ложный пользователь с большими правами. Доступ происходит по легальной процедуре идентификации и аутентификации с любой машины сети. Ложный пользователь может существовать столько времени, сколько потребуется службе безопасности на выяснение по другим данным аудита или определение по другим косвенным данным наличия в сети дополнительного получателя ценной информации. В случае контроля директории ложный пользователь может быть обнаружен в дереве директорий и уничтожен. Тогда процедура создания нового ложного пользователя должна повториться.

В случае ошибки, когда ложный пользователь создается случайно (входящий пользователь неправильно называет свой пароль), доступ к этому пользователю никто не будет иметь, и он останется незаметным. Если процедура образования ложного пользователя не завершена, агент через некоторое время его уничтожит.

Интеллектуальный программно-аппаратный агент нарушителя на сервере аутентификации может быть настроен на получение инструкций через скрытый канал. Допустим, нарушителем безопасности является группа k легальных пользователей системы, которые знают фиксированное число m, известное также программно-аппаратному агенту. Нарушители безопасности инициируют вход в систему и выход из нее так, чтобы иметь возможность первыми из пользователей за ограниченное время набрать по m вхождений в систему.

Пусть агент обладает достаточными вычислительными возможностями для оценки промежутка времени, необходимого для m вхождений в систему любых других пользователей. Если это время не превосходит заданного порога, то агент выделяет данную группу пользователей как своего

союзника. После m вхождений и выходов из системы группа нарушителей продолжает работать в обычном режиме и при этом начинается выработка языка общения агента с группой нарушителей. Если в заданном промежутке группа нарушителей вошла в систему ровно один раз, то их перестановке (порядки появления на сервере) соответствует двоичный вектор длины k. Этот вектор передается агентом к членам группы нарушителей в форме ответов о правильной или неправильной аутентификации. Достаточно сделать первую посылку вектора из всех 0 в ответ на заведомо ложные пароли, так как все двоичные вектора можно считать лексикографически упорядоченными. Тогда 2^k перестановок порядка вхождения в систему данной группы воссоздадут у агента и членов группы нарушителей код, который можно использовать при дальнейшем общении агента и группы нарушителей. Любая другая комбинация вхождений в систему уже не будет считаться у агента значимой.

Рассмотрим скрытый канал через Интернет с помощью HTML [10]. В данном случае речь пойдет о задаче построения скрытого канала между далеко расположенными корреспондентами.

Пусть пользователь A хочет передать короткое сообщение пользователю B через Интернет. Предположим, что пользователь A имеет доступ к браузеру, а пользователь B имеет доступ к аудиту обращений на некоторую страницу Интернет. Все обращения пользователя A к информационным ресурсам Сети отслеживаются контролером U. Задача состоит в том, чтобы построить скрытый от U канал передачи информации от пользователя A к пользователю B.

В литературе описано множество скрытых каналов для решения этой задачи. Однако доказать стойкость сокрытия факта передачи от контролера U эти методы не позволяют. Предлагаемый здесь метод позволяет доказывать такую стойкость для небольших сообщений. Однако ограниченные рамки статьи не позволяют привести это доказательство.

Изложим содержание метода. С помощью последовательности гипертекстовых ссылок пользователь A попадает на сайт, к которому имеет доступ абонент B, и выбирает оговоренный секретным ключом документ. Пусть $R_1, ..., R_N$ — последовательность возможных гипертекстовых ссылок в этом документе. Тогда скрытое сообщение от пользователя A к пользователю B можно закодировать перестановкой вызовов этих гиперссылок $R_{j1}, ..., R_{jN}$. Перестановка определяется с помощью вызовов и меток времени, находящихся в журнале аудита.

Если $B_1, ..., B_N$ сайты, в которых абонент B может получить доступ к данным аудита, то перестановками $B_{i1}, ..., B_{iN}$ можно кодировать допустимые сообщения.

Пусть вместо пользователя B на сервере находится программно-аппаратный агент B_1, связанный с пользователем B, который отслеживает порядок вызова гиперссылок в определенном тексте. Тогда построенная этим агентом перестановка может быть передана пользователю B, который может находиться где угодно и может общаться со своим агентом с помощью произвольного скрытого канала. Такое взаимодействие возможно с помощью протокола HTTP. Маршрут передачи информации может быть еще более сложным для создания неотслеживаемости взаимодействия пользователей A и B.

Возникает вопрос о способе кодирования сообщения с помощью перестановок. Один из вариантов кодирования можно построить следующим образом. Рассматривая некоторую нумерацию перестановок, относим номеру данной перестановки двоичное представление этого номера. Тогда сообщение можно писать на любом доступном языке, используя для его кодирования двоичные векторы. Использование данного метода ограничено вычислительными возможностями при нумерации перестановок. Приведем некоторые значения числа перестановок, показывающие возможности кодирования сообщений для различных N: $5! = 120$, а $10! = 3628800$.

РАСПРЕДЕЛЕННЫЕ АТАКИ С ПОМОЩЬЮ BMAC

Чтобы осуществить распределенную атаку, должна быть построена «невидимая» сеть агентов. В предыдущих разделах мы обсуждали, каким образом это можно сделать. На всех шагах распределенной атаки BMAC должна быть интеллектуальной.

Для моделирования интеллектуальных свойств BMAC мы использовали результаты исследований по Open Agent Architecture [23]. Пусть существует три класса агентов в BMAC. Первый класс состоит из агентов-посредников. Второй класс включает в себя мета-агентов. Третий класс составляют агенты интерфейса с внешними сетями.

Агент-посредник представляет собой специализированного служебного агента, который отвечает за координацию взаимодействия агентов, их связь и кооперацию, а также за «невидимость» мета-агентов, связанных с ним. «Невидимость» агентов-посредников может быть реализована методами, описанными в соответствующем разделе

выше. В некоторых случаях посредник обеспечивает хранение данных для других типов агентов, то есть предоставляет им функции «blackboard» для их взаимодействия. Посредники должны быть связаны между собой в некоторый кластер. В частном случае это может быть иерархическая структура.

Предположим, что посредники могут взаимодействовать друг с другом в определенной последовательности. Если мета-агент *A* выполняет план *P* и ему необходим сервис *S*, то он обращается к агенту-посреднику *F* за сервисом. Агент-посредник *F* посылает сообщение другим агентам-посредникам о необходимости сервиса *S*. Каждый агент-посредник *F'* делает запрос своим мета-агентам на сервис *S*. Если *F'* получает отказ, то он передает запрос *F* на сервис *S* следующему агенту-посреднику *F''* и т.д. Если *F'* получает положительный ответ от одного из своих мета-агентов, то он передает этот положительный ответ *F*. Пропускная способность такого канала может быть низкой, но первый шаг атаки не нуждается в скорости. Все мета-агенты связаны между собой через агентов-посредников. Каждый мета-агент имеет своего агента-посредника для взаимодействия в BMAC. Мета-агенты исполняют роль ассистентов посредников при координации деятельности всех агентов. Мета-агенты могут быть сделаны «невидимыми» с помощью своих агентов-посредников. Агент-посредник и мета-агент могут играть роль «men-in-the-middle» в легальном протоколе. Это путь контроля и модификации любого легального протокола внутри одного компьютера.

Агенты-посредники и агенты интерфейса играют роль «men-in-the-middle» в легальных протоколах связи. Основная проблема здесь — сделать их быстро работающими. Агенты интерфейса могут связываться друг с другом с помощью скрытых каналов, если они находятся в различных сегментах сети под охраной своих агентов-посредников.

Мета-агенты могут создаваться и могут уничтожаться. Коды агентов могут передаваться через агентов-посредников или агентов интерфейса.

Рассмотрим, каким образом BMAC работает в распределенной атаке.

Мета-агенты используют методы распознавания для сканирования окружающей среды. Они находят точки для модификации информации и для выстраивания «men-in-the-middle». Мета-агенты могут иметь информацию о выполнении атаки или могут ее не иметь. В последнем случае они должны найти сервис в форме, содержащей план, что делать дальше. План атаки может передаваться по сети.

Когда каждый агент, являющийся участником атаки, имеет свой план, тогда первый шаг атаки закончен. Низкая пропускная способность скрытых каналов определяет слабую синхронизацию распределенной атаки. Поэтому агент противника проходит активную фазу атаки почти автономно.

Если распределенная система разрушается, то это приводит к ущербу, но в таком случае и BMAC также разрушается. После восстановления распределенной системы BMAC должна создаваться с самого начала. Поэтому противник должен предпочесть нанесение ущерба с помощью ошибочных вычислений или Византийского поведения [21].

Рассмотрим пример. На первом шаге BMAC находит точки доставки данных приложениям. Тогда мета-агенты готовят планы модификации входных или выходных данных для выбранных приложений. Планы могут также содержать генератор случайных чисел для выбора момента модификации данных. Реализация атаки состоит в случайной модификации входных или выходных данных с помощью «невидимого» агента противника. Распределенная система делает ошибки, но все тесты приложений и связи не находят сбоев. Медленное изменение интенсивности ошибок может привести к разрушению отдельных приложений.

ЗАКЛЮЧЕНИЕ

Безопасность определяется знанием возможных атак. Наиболее опасные атаки на распределенные системы — это распределенные атаки. Такие атаки могут осуществляться враждебными многоагентными системами. BMAC может быть «невидимой» и интеллектуальной. Координация агентов в BMAC также может быть «невидимой» и осуществляться с помощью скрытых каналов.

Эти условия могут удовлетворяться с помощью реализации модели невлияния, скрытых каналов и соответствующей организацией агентов противника.

Литература

1. Грушо А.А., Тимонина Е.Е., Модель невлияния для сети. Обозрение прикладной и промышленной математики, т. 7 (1), Москва: ТВП, 2000.
2. Грушо А.А., Тимонина Е.Е., Двойственность многоуровневой политики безопасности. Тез. конф. «Методы и технические средства информационной безопасности», С.-Петербург, 2000.

3. Грушо А.А., Тимонина Е.Е., Применко Э.А. Анализ и синтез криптоалгоритмов. Курс лекций. — Йошкар-Ола: изд-во Марийского филиала Московского открытого социального университета, 2000.
4. Грушо А.А., Тимонина Е.Е., Языки в скрытых каналах. Труды XXX международной конференции «Информационные технологии в науке, образовании, телекоммуникации, бизнесе», Украина, 2003.
5. Грушо А.А., Тимонина Е.Е., Оценка времени обучения агента для организации скрытого канала. Дискретная математика, Т. 15 (2), 2003.
6. Грушо А.А., Тимонина Е.Е., Преодоление защиты от скрытого канала. Обозрение прикладной и промышленной математики, Т. 10 (3), Москва: ТВП, 2003.
7. Грушо А.А., Тимонина Е.Е. Проблемы компьютерной безопасности. Сб. научных докладов «Информационные технологии в производстве, медицине, психологии и этике» Академии информационных управленческих технологий. — М.: Центр Управления Полетами, 2003.
8. Грушо А.А., Тимонина Е.Е. Стохастические скрытые каналы. Материалы международн. семинара «Информатика и общество» I&S'04 (10-24 января). — Низкие Татры, 2004.
9. Грушо А.А., Тимонина Е.Е. Нарушение безопасности систем с единой точкой входа. Материалы 8-ой международной конференции «Связь-2004», озеро Иссык-Куль, 22-29 августа, 2004, Новосибирск: ЗАО РИЦ Прайс Курьер, 2004.
10. Грушо А.А., Тимонина Е.Е. Скрытые каналы с использованием HTML. Труды XXXII международной конференции и III международной конференции молодых ученых «Информационные технологии в науке, образовании, телекоммуникации и бизнесе. IT+S&E'2005», Украина, 2005.
11. Гусев А.В. Скрытый канал с модуляцией скорости передачи. Труды XXXII международной конференции и III международной конференции молодых ученых «Информационные технологии в науке, образовании, телекоммуникации и бизнесе. IT+S&E'2005», Украина, 2005.
12. Неббет Г. Справочник по базовым функциям API Windows NT/2000. Пер. с англ. — М.: Издательский дом «Вильямс», 2002.
13. Тимонина Е.Е. Скрытые каналы (обзор). Jet Info: изд-во компании «Джет Инфо Паблишен», 2002. — 14(114).
14. COUGAAR Architecture Document. A BBN Technologies Document, Version for Cougaar 11.4, 2004.
15. Galatenko A., Grusho A., Kniazev A., Timonina E. Covert Channels through PROXY Server. The Third International Workshop «Information Assurance in Computer Networks. Methods, Models, and Architectures for Network Security», St. Petersburg: Springer, LNCS 3685, 2005 (to be printed).
16. B. Giles. J., Hajek B. An Information-theoretic and Game-theoretic Study of Timing Cannels. IEEE Transactions on Information Theory, 2002.
17. Goguen J.A., Meseguer J. Security Policies and Security Models. Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, 1982.
18. Goguen J.A., Meseguer J. Inference Control and Unwinding. Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, 1984.
19. Grusho A.A., Timonina E.E. Construction of the Covert Channels. International Workshop «Information Assurance in Computer Networks. Methods, Models, and Architectures for Network Security», St. Petersburg: Springer, LNCS 2776, 2003.
20. A Guide to Understanding Covert Channel Analysis of Trusted Systems. National Computer Security Center, NCSC-TG-030, Ver. 1, 1993.
21. Gumnopoulos L., Dritsas S., Gritzalis S., and Lambrinoudakis C. GRID Security Review. International Workshop «Information Assurance in Computer Networks. Methods, Models, and Architectures for Network Security», St. Petersburg: Springer, LNCS 2776, 2003.
22. Herard J. etc. Validation of Communication in Safety--Critical Control Systems. Nordtest Report TR 543, 2003.
23. Martin D.L., Cheyer A.J., Moran D.B. The Open Agent Architecture: A Framework for Building Distributed Software Systems. Artifical Intelligence Center SRI International, 1998.
24. McLean J., Heitmeyer C. High Assurance Computer Systems: A Research Agenda. Center for High Assurance Computer Systems Naval Research Laboratory, Washington, DC 20375, 1995.
25. Moskowitz I.S., Costich O. L. A classical Automata Approach to Noninterference Type Problems. Procced. Of the Computer Security Foundations Workshop 5, Franconi, NH: IEEE Press., 1992.
26. RFC 793. Transmission Control Protocol, 1981.
27. Rushby J. Noninterference, Transitivity, and Channel-Control Security Policies. Technical Report CSL-92-02, 1992.
28. Father H. Hooking Windows API. Techniques of hooking API functions on Windows, 2002.
29. Father H. Invisibility on NT boxes. How to become unseen on Windows NT, 2003.
30. Foundation for Intelligent Physical Agents, Standard: FIPA TC Communications, 2002.

О КАНАЛАХ СКРЫТЫХ, ПОТАЙНЫХ, ПОБОЧНЫХ. И НЕ ТОЛЬКО

Доктор физико-математических наук В.А. Галатенко

О скрытых каналах

Jet Info не первый раз обращается к теме скрытых каналов. В 2002 году ей был посвящен отдельный номер (см. [1], [2]), так что в данной работе предполагается, что читатель знаком с основами этой области знания; в противном случается рекомендуется перечесть статью [2]. Тем не менее, автор с самого начала хотел бы заметить, что тематика скрытых каналов в ее традиционной трактовке представляется ему несколько надуманной, формальной. Пик исследований в области скрытых каналов приходится на середину 1980-х годов, когда была опубликована «Оранжевая книга» Министерства обороны США, в которой, начиная с класса безопасности B2, было введено требование анализа скрытых каналов. В результате бороться со скрытыми каналами стали, в основном, не ради реальной безопасности, а ради успешной сертификации. Кроме того, скрытые каналы из-за, в общем-то, случайной ассоциации с классами B2 и выше исследовались почти исключительно в контексте многоуровневой политики безопасности, с обязательным упоминанием субъектов HIGH и LOW, моделями невлияния и прочими премудростями. Все это бесконечно далеко от реальных проблем типичных современных информационных систем, да и публикуемые результаты по большей части носят очевидный характер и не представляют ни теоретического, ни, тем более, практического интереса. В статье [2] объяснены концептуальные причины подобного положения дел.

В частности, едва ли целесообразно задаваться вопросом возможности организации скрытых каналов для управления враждебной многоагентной системой (BMAC). Если BMAC удалось построить, взломав множество удаленных систем и внедрив в них вредоносное программное обеспечение (ВПО), то, очевидно, для этого нашлись коммуникационные ресурсы подходящей скрытности, более чем достаточные и для последующего управления.

В середине 1980-х годов была предложена систематическая методология выявления скрытых каналов по памяти (см. [3]), ключевым элементом которой является матрица разделяемых ресурсов. В сетевой среде, в Интернет, легальных разделяемых ресурсов сколько угодно — например, выделяемое пользователям пространство на общедоступных сайтах. Можно воспользоваться и полями заголовков IP-пакетов (скажем, контрольная сумма — превосходный кандидат на эту роль), и начальными порядковыми номерами при установке TCP-взаимодействия (см. [4]). Можно организовать и практичные скрытые каналы по времени, например, кодируя единицу отправкой пакета в определенный интервал времени, составляющий миллисекунды (см. [5]).

С появлением мощных многопроцессорных систем с общей памятью полоса пропускания скрытых каналов подскочила до мегабит в секунду и продолжает увеличиваться с ростом быстродействия аппаратуры (см. [6]). Это, конечно, серьезная проблема, однако для ее решения достаточно отказаться от разделения подобных систем между субъектами с различным уровнем допуска.

Проблема скрытых каналов — это проявление более общей проблемы сложности современных информационных систем. В сложных системах скрытые каналы были, есть и будут, так что бороться нужно с причиной, а не со следствием.

В самом общем виде метод борьбы со сложностью систем можно сформулировать как «проведение объектного подхода с физическими границами между объектами». Процессоры не должны разделяться не только между субъектами, но и между потоками управления. Пользовательская сеть должна быть физически отделена от административной. Вообще говоря, компоненты системы не должны доверять друг другу: процессор может не доверять памяти, сетевая карта — процессору и т.п. При выявлении подозрительной активности компоненты должны поднимать тревогу и применять другие защитные меры (например, дисковый контроллер может зашифровать файлы, сетевой контроллер — блокировать коммуникации и т.п.). В общем, на войне как на войне. Если организовать физические границы невозможно, следует воспользоваться виртуальными, формируемыми в первую очередь криптографическими средствами. Более подробное изложение этих вопросов можно найти в работе [7].

Скрытые каналы можно не только выявлять, но и ликвидировать или зашумлять «не глядя». Как пояснено в [2], для этого служат разного рода нормализаторы, сглаживающие нагрузку на процессор, энергопотребление, время вычисления определенных функций, сетевой трафик и т.п. Например, ядро операционной системы Asbestos [8] в ответ на запрос о создании порта возвращает новый порт с непредсказуемым именем, поскольку возможность создания портов с заданными именами может служить скрытым каналом.

Накладные расходы на нормализацию могут быть велики, отчего функционирование легальных субъектов может существенно замедляться, так что следует искать и находить разумный компромисс между информационной безопасностью и функциональной полезностью систем.

С точки зрения борьбы со сложностью скрытые каналы имеют следующее неприятное свойство. Разделяемые ресурсы, присутствующие на любом уровне информационной системы, начиная с самого нижнего, аппаратного, могут быть использованы на всех вышележащих уровнях, вплоть до прикладного, для организации утечки информации. Централизованный арбитр доступа к памяти в многопроцессорной системе, разделяемый несколькими процессорами кэш второго уровня, устройство управления памятью — все это может служить каналом утечки. Таким образом, при анализе скрытых каналов необходимо рассматривать систему в целом. Попытка проведения так называемой составной сертификации, когда система оценивается на основе ранее проведенных испытаний отдельных модулей или уровней, ведет к пропуску скрытых каналов. Проблема усугубляется тем, что в описании отдельных модулей или уровней необходимые детали могут быть опущены как несущественные. Казалось бы, какая разница, как устроена очередь инструкций, выбранных микропроцессором для исполнения? Однако и это может быть важно для безопасной работы приложения (см. [6]). Операционная система, успешно прошедшая сертификацию при испытаниях на «голой» аппаратуре, содержит скрытые каналы заметной пропускной способности, если выполняется под управлением монитора виртуальных машин. В общем, разделяемый ресурс — это та самая горошина, которую настоящая принцесса почувствует через любое количество перин. И об этом необходимо помнить.

Подход на основе скрытых каналов активно используется для оценки степени несовершенства реализации таких защитных сервисов, как анонимизаторы и их сети, а также пополнение трафика. Это представляется естественным, так как анонимизация и пополнение трафика — разновидности нормализации, предназначенной для ликвидации скрытых каналов. Если нормализация оказалась несовершенной, значит, скрытые каналы остались. Насколько несовершенной? Настолько, насколько велика утечка информации.

Несовершенство анонимизаторов можно оценивать как пропускную способность скрытых каналов утечки информации об отправителе и/или получателе (см. [9]). Для отдельных анонимизаторов удается получить точное значение, для сетей анонимизаторов — оценку сверху.

Согласно текущим тенденциям, все большая часть Интернет-трафика шифруется (см. [10]). Шифрование защищает содержимое и заголовки пакетов, дополнение пакетов препятствует получению информации путем анализа их размеров. Однако криптография сама по себе не защищает от анализа поведения пакетов, то есть их распределения во времени, в результате чего может пострадать приватность пользователей. Кроме того, временной анализ SSH-трафика существенно упрощает несанкционированный доступ к пользовательским паролям. Пополнение трафика на канальном уровне — эффективная защитная мера против подобного анализа. Поток данных в канале приобретает заранее заданный характер. Некоторые пакеты задерживаются, а в канал, когда нужно, отправляются фиктивные данные. Это в принципе. На практике же весьма непросто реализовать пополнение так, чтобы наблюдаемый трафик в точности следовал заранее заданному распределению, так что у злоумышленника остается возможность скоррелировать пополненный полезный трафик. Несовершенство реализации пополнения можно оценить

как пропускную способность скрытого канала, основанного на варьировании межпакетных интервалов. Оказывается, что в идеальных условиях этот скрытый канал допускает практическое использование. К счастью, в реальной загруженной сети с множеством потоков данных высокий уровень шума в канале затрудняет действия злоумышленника.

Применение аппарата скрытых каналов для оценки степени несовершенства архитектуры и/или реализации сервисов безопасности представляется весьма перспективным направлением исследований.

Красивое применение методов передачи данных, характерных для скрытых каналов по времени, в беспроводных сенсорных сетях удалось найти авторам работы [11]. Одна из главных проблем сенсорных сетей — снижение энергопотребления. Если двоичные значения передаются по беспроводной сети обычным образом, то можно считать, что на это тратится энергия, пропорциональная их логарифму. Однако значения можно передавать и молчанием: послать стартовый бит, заставляющий получателя включить счетчик, выждать время, соответствующее значению, и послать стоп-бит. В результате экономится энергия, но тратится время (пропорциональное значению), однако передачу можно оптимизировать — молчание отлично мультиплексируется, каскадируется и быстро переадресуется.

Конечно, описанный метод передачи данных относится к разряду забавных диковинок. В целом, в настоящее время скрытые каналы являются почти исключительно академически-сертификационной областью. В этом контексте интересна работа [12], в которой исследуется проблема полноты анализа скрытых каналов. Вводится понятие полного набора скрытых каналов, элементы которого в совокупности порождают максимально возможную скрытую утечку информации (аналогом полного набора может служить базис в векторном пространстве). По мере выявления скрытых каналов их совокупность можно проверять на полноту (с помощью сформулированных в [12] критериев) и получать в результате оценку потенциально возможной утечки информации. Еще один очень важный аспект работы [12] — описание архитектурного подхода к построению систем, облегчающего анализ скрытых каналов. Выявлять по одному скрытые каналы в произвольной информационной системе — задача бесперспективная; целесообразно строить системы неким регулярным образом и затем подвергать их систематическому анализу с учетом их специфики.

На практике ни злоумышленники, ни производители средств информационной безопасности не уделяют скрытым каналам сколько-нибудь заметного внимания. Причина в том, что в современных информационных системах более чем достаточно «грубых» уязвимостей, допускающих несложное использование, поэтому и атакующие, и защищающиеся предпочитают пути наименьшего сопротивления, что вполне естественно. Первые эксплуатируют очевидные «дыры», вторые пытаются их прикрыть.

Потребителям тоже не до скрытых каналов — им бы от червей и вирусов врукопашную отбиться, да найти деньги на прошлогодний снег в упаковке с надписью «системы предотвращения вторжений с известными сигнатурами». И еще терпеливо выслушать поучения производителей дырявого ПО за отсутствие дисциплины управления многочисленными корректирующими заплатами для этого самого ПО.

По поводу уязвимостей есть две новости, и обе хорошие. Первая — проблем с безопасностью базового ПО становится меньше, поэтому злоумышленники более активно эксплуатируют уязвимости приложений. Новость вторая — приложений много. А ведь есть еще фишинг и другие методы морально-психологического воздействия... Поэтому время скрытых каналов, если и придет, то не очень скоро.

Чтобы осознать, сколь скромное место занимают скрытые каналы среди других проблем информационной безопасности, даже если ограничиться только дефектами программного обеспечения, целесообразно рассмотреть классификацию подобных дефектов, предложенную в статье [13] в контексте разработки средств статического анализа исходных текстов с целью выявления ошибок, чреватых возникновением уязвимостей.

Дефекты в ПО могут быть внесены намеренно или по небрежности. Первые подразделяются на злоумышленные и незлоумышленные. Злоумышленные дефекты — это лазейки, логические и временные бомбы; незлоумышленные — скрытые каналы (по памяти или по времени) и несогласованные пути доступа.

Дефекты, внесенные непреднамеренно, делятся на:

- ошибки проверки правильности данных (ошибки адресации, в том числе переполнение буферов, некачественные проверки значений параметров, неверное размещение проверок, неадекватная идентификация/аутентификация);
- ошибки абстракции (повторное использование объектов, раскрытие внутреннего представления);
- асинхронные дефекты (проблемы параллельного выполнения, включая ситуации опере-

жения, активные и пассивные тупики, разрывы между временами проверки и использования, а также наличие нескольких ссылок на один объект);
- ненадлежащее использование подкомпонентов (утечка ресурсов, непонимание распределения ответственности);
- ошибки функциональности (дефекты обработки исключительных ситуаций, прочие дефекты безопасности).

Чтобы понять, как дефекты безопасности могут быть внесены в программное обеспечение намеренно, но не злоумышленно, рассмотрим скрытый канал, образующийся в дисковом контроллере при оптимизации обслуживания запросов по алгоритму лифта (обращения к диску обрабатываются не в порядке поступления, а по мере того, как штанга с головками достигает запрошенных блоков, см. статью [14], в которой представлен систематический подход к выявлению скрытых каналов по времени). Злоумышленный отправитель информации может влиять на порядок и, следовательно, время обработки обращений, контролируя направление перемещения штанги с головками путем выдачи собственных запросов к диску в определенном порядке. Здесь в роли разделяемого ресурса, допускающего (злоумышленное) целенаправленное воздействие, выступает общая очередь запросов к дисковым блокам, а также текущая позиция и направление движения штанги. Данный дефект естественно считать внесенным преднамеренно, но не злоумышленно, поскольку скрытый канал образовался не из-за ошибки реализации, а вследствие принятого проектного решения, направленного на оптимизацию функционирования системы.

Самую большую и практически важную группу дефектов, внесенных по небрежности, составляют ошибки проверки правильности данных, точнее, недостаточный контроль входных данных перед их использованием. Разработка методов недопущения или выявления подобных ошибок — задача первостепенной практической важности. А скрытые каналы могут подождать...

О потайных каналах

Как отмечено в работе [15], в настоящее время происходит становление так называемой многоаспектной информационной безопасности, когда делаются попытки учесть весь спектр интересов (порой конфликтующих между собой) всех субъектов информационных отношений, а также все виды конфигураций информационных систем, в том числе децентрализованные, не имеющие единого центра управления.

Безопасность зависит от субъекта. У пользователя своя безопасность, у поставщика информационного наполнения — своя (и пользователь здесь может считаться врагом). Появляются новые аспекты безопасности, такие как управление цифровыми правами. Эта тенденция особенно наглядно проявляется в применении потайных каналов.

Напомним (см. [2]), что скрытыми считаются нестандартные каналы передачи информации. Нестандартные способы передачи информации по легальным каналам (именуемыми в данном контексте обертывающими) получили название потайных (subliminal channels) или стеганографических (stego channels) каналов. Общие сведения о них приведены в статье [2]. Потайные каналы используют тогда, когда имеется легальный коммуникационный канал, но что-либо (например, политика безопасности) запрещает передавать по нему определенную информацию.

Отметим, что между скрытыми и потайными каналами имеется два важных отличия. Во-первых, вопреки названию, никто не пытается скрыть существование скрытых каналов, просто для передачи информации используют сущности, для этого изначально не предназначенные, созданные для других целей. Напротив, потайной канал существует только до тех пор, пока о нем не узнал противник. Во-вторых, считается, что время передачи информации по скрытому каналу не ограничено. В противоположность этому, время передачи по потайному каналу определяется характеристиками обертывающего канала. Например, если для тайной передачи информации применяется графический образ, то передать можно только то, что удается поместить в этот образ, не нарушая скрытности.

В целом, потайные каналы гораздо практичнее скрытых, поскольку у них есть легальная основа — обертывающий канал. Потайные (а не скрытые) каналы — наиболее подходящее средство для управления враждебной многоагентной системой. Но в них нуждаются не только злоумышленники. Потайные каналы могут эффективно применяться поставщиками информационного наполнения, встраивающими в него скрытые «цифровые водяные знаки» и желающими контролировать его распространение, соблюдение потребителями цифровых прав. Еще один пример, ставший клас-

сическим, — применение потайного канала премьер-министром Великобритании Маргарет Тэтчер, которая, чтобы выяснить кто из ее министров виновен в утечках информации, раздала им варианты одного документа с разными межсловными промежутками.

Разумеется, при весьма общих предположениях потайные каналы нельзя не только устранить, но даже обнаружить (например, в сжатом JPEG-образе всегда найдется место для скрытой информации). По отношению и к скрытым, и к потайным каналам справедливо приведенное в статье [16] положение «Вы всегда можете послать бит».

Содержательным является вопрос о емкости и устойчивости подобных каналов, которые определяются не только полосой пропускания обертывающего канала и характеристиками шума в нем, но и максимальным размером полезной (скрываемой) нагрузки, а также функцией-детектором допустимости передаваемой информации (см., например, статью [17] и цитированные в ней источники, среди которых мы выделим работу [18]).

Проблематика потайных каналов давно и плодотворно исследуется с позиций теории информации, получено много интересных теоретически и важных практически результатов.

Обратим внимание на возможность и эффективность совместного использования скрытых и потайных каналов в сетевой среде. Так, в работе [19] описана реализация сети анонимизаторов (см. [15]) с помощью HTTP-серверов и клиентов. Web-серфинг служит обертывающим каналом. В роли узлов сети анонимизаторов выступают HTTP-серверы, а взаимодействие между ними осуществляется по скрытым каналам в HTTP/HTML при посредничестве ничего не подозревающих клиентов (в первую очередь — с помощью средств перенаправления запросов и активного содержимого, встроенных, например, в рекламные баннеры, присутствующие на посещаемой Web-странице). В результате можно достичь не только невозможности ассоциации между отправителем и получателем сообщений, но и реализовать более сильное свойство — скрытность (даже в присутствии глобального наблюдателя). Оказывающиеся невольными посредниками Web-серферы пополняют подлежащее анализу множество анонимности, затрудняя тем самым получение наблюдателем полезной информации.

(Разумеется, и злоумышленники, и разработчики защитных средств осознают возможности и проблемы, связанные с использованием HTTP в качестве обертывающего канала. Например, в статье [20] описана обучаемая система Web Tap, выявляющая аномалии в исходящих HTTP-транзакциях.)

Отметим также очевидную связь между интеллектом встроенных агентов (или элементов многоагентной системы) и требуемой пропускной способностью потайных или скрытых каналов для взаимодействия с ними. В заметке [21] приведен пример высокоинтеллектуальной троянской программы, являющейся экспертной системой, встроенной в доверенную (с многоуровневой политикой безопасности) стратегическую систему управления военными поставками и перемещением войск и способной определять по поставкам и перемещениям, возможно ли на следующей неделе начало наступательных военных действий. Если подобная программа будет каждый день передавать всего один бит информации (возможно/невозможно), это окажется весьма ценным для стратегического планирования. В то же время, согласно формальным требованиям «Оранжевой книги», скрытые каналы с полосой пропускания менее одного бита в десять секунд при аудите доверенных систем могут вообще не рассматриваться. (Редкий случай, когда «Оранжевая книга» делает послабление и, как оказывается, напрасно.)

Мораль состоит в том, что при анализе потайных и скрытых каналов вообще и их пропускной способности в частности нужно учитывать специфику информационных систем, ценность информации и семантику взаимодействия. В противном случае результаты анализа рискуют оказаться бессодержательными.

О побочных каналах

Побочные каналы можно считать частным случаем скрытых. В роли (невольных) передатчиков в подобных каналах выступают штатные компоненты информационных систем, а в роли приемников — внешние наблюдатели, применяющие соответствующее оборудование. Чаще всего с помощью побочных каналов измеряется время видимых операций (временные атаки на RSA стали общим местом), их энергопотребление и/или побочные электромагнитные излучения и наводки (ПЭМИН), но для атак могут применяться и акустические каналы, идет ли речь о цифровом замке сейфа или процессоре персонального компьютера, обрабатывающего секретный ключ (см. [22]).

Побочные каналы представляют собой, вероятно, наиболее наглядное проявление многоаспектности современной информационной безопасности. В роли атакующих на информационные системы (информационное наполнение, банков-

ские карты, SIM-карты сотовых телефонов и т.п.), как правило, выступают их владельцы, располагающие значительным временем и соответствующим инструментарием. В сочетании с принципиальной невозможностью управления физическим доступом, перечисленные факторы делают атаки с использованием побочных каналов особенно опасными.

Объектами атак с использованием побочных каналов чаще всего становятся криптографические компоненты информационных систем, точнее, их секретные ключи. Например, в статье [23] описана атака разбиением на SIM-карты сотовых телефонов (точнее, на алгоритм COMP128, применяемый для аутентификации пользователей и выработки сеансовых ключей), проводимая путем измерения энергопотребления с целью клонирования этих карт. Атаку удалось отточить до такой степени, что для определения секретного 128-битного ключа оказывается достаточно всего восьми измерений с адаптивно выбираемыми входными данными! То есть злоумышленнику достаточно получить SIM-карту буквально на минуту.

Весьма наглядно опасность атак, основанных на дифференциальном анализе энергопотребления, проиллюстрирована в статье [22]. В 1998 году Брюс Шнейер писал, что в галактике не хватит кремния, а у Солнца — времени жизни для реализации атаки методом грубой силы на секретный ключ (112 бит) алгоритма 3DES. Минимальная длина ключа в алгоритме AES — 128 бит, но успешная атака методом дифференциального анализа энергопотребления на незащищенную микросхему, реализующую AES, может быть проведена менее чем за три минуты — от начала измерений до завершения анализа.

Кардинальное решение проблемы побочных каналов возможно при соблюдении следующего основополагающего принципа: данные об операциях, которые можно получить из побочных каналов, должны быть статистически независимы от входных и выходных данных и информации ограниченного доступа. Поскольку защищать от атак с использованием побочных каналов чаще всего приходится системы с весьма ограниченными ресурсами, корректная, полная реализация кардинального принципа — задача весьма непростая. Время операций нормализовать относительно просто, энергопотребление — сложнее, но также возможно (см., например, [24]), ПЭМИН — еще сложнее. На практике системы укрепляются «по мере сил» (что характерно для современной информационной безопасности вообще), а у мотивированных злоумышленников остается масса возможностей для результативных атак.

Об агентах – хороших и ... разных

Агенты и многоагентные системы (МАС) — одно из активно развиваемых направлений современной технологии программирования. Агенты доставляют на удаленные системы код, расширяющий функциональность последних, а многоагентные комплексы позволяют естественным образом распараллелить решение сложных задач. Оба свойства важны, например, для эффективной индексации мультимедийных ресурсов (см. [25]). Файлы мультимедиа велики, и загружать их на центральный сервер для индексации накладно. Проще доставить на целевую систему индексирующий код, а затем получить оттуда лишь результаты индексации, имеющие существенно меньший размер. Подобный подход хорош и с точки зрения информационной безопасности, если подходить к ней с позиций поставщиков информационного наполнения и управления цифровыми правами, поскольку он исключает выгрузку платных ресурсов и, в то же время, способствует распространению сведений о них.

Мобильные агенты опираются на интерфейсы и возможности, присутствующие на большинстве платформ. Кроме того, они способны перемещаться с одной системы на другую, двигаясь по намеченному маршруту, что делает их автономными, не требующими постоянного управления, за счет чего экономятся вычислительные и коммуникационные ресурсы. В контексте информационной безопасности мобильные агенты могут взять на себя роль странствующих рыцарей, контролирующих удаленные системы на предмет своевременного наложения корректирующих заплат и отсутствия признаков вредоносной активности (см. [26]), выявляющих и отражающих вместе с «братьями по оружию» распределенные, скоординированные атаки (см. [27]), внося в защиту элементы динамичности и самоорганизации (см. [28]).

Инкрементальность — важное достоинство мобильных агентов и многоагентных систем. Каждый отдельный агент может решать свою, частную задачу (выделение определенных свойств мультимедийного наполнения, проверка установки определенных программных коррекций, выявление определенных разновидностей вредоносного ПО, реализация определенных положений политики безопасности), но их пополняемая совокупность оказывается адекватным, актуальным отражением и средством проведения в жизнь текущей политики безопасности, меняющейся под влиянием измене-

ний в окружении, при появлении новых рисков и угроз.

Технология мобильных агентов и многоагентных систем — средство сильнодействующее, пользоваться которым следует с осторожностью, осознавая ассоциированные риски. Уязвимость коммуникаций МАС — лишь одна и, вероятно, не самая сложная из проблем. Нужно учитывать (см., например, [29]), что:

- агенты могут атаковать целевые платформы (проблема безопасности платформ);
- агенты сами могут быть атакованы платформами, другими агентами и внешними сущностями, такими как вирусы (проблема безопасности агентов).

Надежная защита и платформ, и агентов может быть построена только с учетом семантики программ (см. [2]), но некоторые частные решения можно получить формальными, криптографическими методами, проводя аутентификацию агентов и их источников, контролируя целостность и обеспечивая конфиденциальность кода и данных агентов (см. [29], [30], [31]).

Особую сложность проблемам безопасности мобильных агентов и целевых платформ придает их взаимное влияние. Предположим, что мобильный агент содержит конфиденциальные данные, лишь часть из которых предназначена для каждой посещаемой им платформы. Эти части шифруют открытыми ключами соответствующих платформ.

В принципе, расшифрование может производиться как агентами, так и платформами, но у обоих подходов есть недостатки (см. [31]). Если расшифрование осуществляет агент, платформа должна передать ему свой закрытый ключ, что (слишком) рискованно. При расшифровании средствами платформы последняя должна знать структуру агента, что противоречит мобильности. Кроме того, следует предупредить попытки расшифрования данных, украденных у агентов. Предложенное в [31] компромиссное решение основано на предоставлении платформой некоторого базового криптографического сервиса, пользоваться которым могут только аутентифицированные агенты. В агентах выделяется ядро «раскрутки», имеющее простую структуру, расшифровывающее тело на конкретной платформе и контролирующее целостность результата.

Согласованность действий МАС — еще одна сложная проблема. Если агенты проводят в жизнь политику безопасности в рамках большой корпоративной сети, то при изменении положений этой политики необходимо отозвать одну совокупность агентов и запустить вместо нее другую. Теоретически все просто, но этому могут помешать асинхронность перемещений мобильных агентов, временное отсутствие связи с удаленными сетевыми сегментами и т.п. В общем, добропорядочное использование МАС — дело непростое, зато для вредоносных агентов открывается масса возможностей.

Как и со скрытыми и потайными каналами, с вредоносным ПО нужно бороться — выявлять и ликвидировать и/или ограничивать. В статье [2] обоснованы преимущества подхода, основанного на ограничении с учетом семантики программ и протоколов. Относительно выявления можно сказать, что для скрытых и потайных каналов оно в общем случае безнадежно, а для вредоносного ПО — еще безнадежнее. Как показывают результаты работы [32], даже лучшие коммерческие антивирусные средства пасуют перед несложными методами обфускации программ, такими, например, как переупорядочение кода, с точки зрения теории информации, факт абсолютно очевидный и естественный. Ситуацию до некоторой степени можно улучшить за счет более изощренного сопоставления с образцами вредоносного поведения, как предлагается в [33], но это «продвижение» никак нельзя назвать решающим.

В таких условиях остается надеяться разве что на детские психологические трюки и пытаться отвлечь хакеров от Интернет, предоставив им отдельный уголок для развлечений и демонстрации своей мощи (см. [34]), или снижать вероятность ненамеренного запуска пользователями вредоносного ПО, распространив на системный загрузчик подход «все, что не разрешено, запрещено» и защитив выполнимые файлы криптографическими контрольными суммами (см. [35]).

О потайных ходах и ремонтных агентах

Если из каких-либо соображений требуется постоянно отслеживать состояние удаленной системы и при необходимости воздействовать на него, конструируют и используют потайные ходы (backdoors). Обычно подобные ходы ассоциируются с вредоносной активностью, но, как показывает проект, развиваемый в университете Rutgers (см. http://discolab.rutgers.edu/bda/), для них имеются и вполне добропорядочные применения, такие как удаленный мониторинг и восстановление работоспособности (удаленное лечение). В таком контексте по-

тайные ходы уместно называть техническими интерфейсами. В статье [36] описан прототип реализации технических интерфейсов для FreeBSD.

Поскольку конечной целью является починка пришедшей в неработоспособное состояние удаленной целевой системы, последняя должна рассматриваться как пассивный объект. Предлагаемый интерфейс с ней сводится к удаленному доступу к памяти, реализуемому при посредничестве программируемой сетевой карты. Целевая система должна поддерживать области сенсоров и внешнего представления, считывание которых позволяет выявлять и диагностировать аномальные ситуации (такие, например, как отсутствие прогресса в выполнении приложений, перерасход или исчерпание ресурсов), а также «ремонтные захваты» — области памяти, запись в которые способна поправить ситуацию (например, таблица процессов или находящаяся в памяти копия суперблока файловой системы).

В принципе, при наличии доступа к памяти целевой системы, возможен перенос реализуемого ею сервиса на другой узел сети (например, в рамках кластерной конфигурации), если ремонт на месте не может быть осуществлен (см. [37]).

Конечно, с точки зрения информационной безопасности, потайные ходы — средство с очень серьезными побочными эффектами. Если система, осуществляющая мониторинг, окажется скомпрометированной, злоумышленник может получить полный контроль над целевой системой. Столь же опасно вмешательство в работу программируемой сетевой карты. В качестве меры противодействия подобные сетевые карты могут быть реализованы в защищенном исполнении, аналогично криптомодулям, а удаленное управление может осуществляться с нескольких машин и только при условии полного согласия в их действиях.

Описанный подход хорош, прежде всего, для восстановления работоспособности системы после непреднамеренных или умышленных атак на доступность (например, при срабатывании fork-бомбы, исчерпания оперативной памяти или порчи файловой системы). Если же система взломана злоумышленниками и поставлена под контроль путем внедрения в нее вредоносного ПО, такого как руткиты, то здравый смысл вроде бы подсказывает, что единственный способ восстановить доверие к ней — полная переустановка с гарантированно безопасных носителей и последующее наложение всех доступных корректирующих заплат, а также восстановление неиспорченных пользовательских данных. Однако в развитой корпоративной сети подобная деятельность может потребовать длительной ручной работы высококвалифицированных специалистов и оказаться экономически нецелесообразной или практически невозможной. Вместо этого можно попытаться реализовать идею автоматического самолечения систем (то есть удаления всего вредоносного без потери доброкачественной информации), встроив в них ремонтных агентов и защитив последних посредством технологии виртуальных машин (см. [38]).

Ремонтные агенты, как и другие средства информационной безопасности, должны удовлетворять следующим проектным принципам:

- простота;
- обособление (агент должен быть защищен от несанкционированного изменения или обхода);
- доверие;
- обозримость (агенту должна быть видна вся система);
- приспособляемость (работа агента и количество потребляемых им ресурсов должны зависеть от состояния контролируемой системы и не мешать ее нормальному функционированию).

Общая схема работы ремонтного агента проста. Он запоминает заведомо безопасное состояние производственной системы, контролирует все вносимые изменения, периодически проверяет наличие признаков аномального поведения и несанкционированных изменений и при необходимости возвращает систему в безопасное состояние. Поскольку производственная система функционирует в рамках виртуальной машины, она не может вмешаться в работу агента, который является доверенным неизменяемым расширением ядра.

Разумеется, на практике все гораздо сложнее. Во-первых, если злоумышленник получит физический доступ к системе, он сможет обойти ремонтного агента; защититься от физических угроз можно только с помощью аппаратной поддержки. Во-вторых, подозрительная активность обнаруживается с некоторым запаздыванием, поэтому критичные данные рискуют оказаться скомпрометированными. В-третьих, «заведомо безопасный» образ системы может оказаться неполным (системный администратор или пользователь могут что-то добавить или изменить в обход агента), и тогда несанкционированные изменения не удастся обнаружить и ликвидировать. Безопасность системы не может быть выше уровня дисциплины, существующей в организации и зафиксированной в ее политике.

Автоматическое восстановление после компрометации должно стать одной из основных целей при проектировании и реализации перспективных информационных систем. С одной стороны, следу-

ет смириться с неизбежностью успешных атак или, по крайней мере, отказов аппаратуры и ошибок администрирования. С другой стороны, стоимость аппаратуры, в том числе, и носителей данных, быстро падает, поэтому имеется техническая и экономическая возможность организовать детальное протоколирование функционирования систем и, в частности, фиксировать все изменения в файловой системе, сохраняя возможность откатки вредоносной или ошибочной деятельности.

Основная проблема состоит в том, чтобы откатить все несанкционированные изменения, не затронув легальных модификаций. В статье [39] описаны возможные подходы к решению этой задачи и прототипная реализация — система восстановления после вторжений Taser. Идея состоит в том, чтобы ассоциировать изменения с процессами, которые их осуществляют, и задать правила, разделяющие процессы на «чистых» и «нечистых». Утверждается, что результаты получаются удовлетворительными и по уровню автоматизации, и по накладным расходам на протоколирование, и по времени восстановления.

О потайных ходах и руткитах

Руткиты, как известно, служат для того, чтобы злоумышленник, взломавший систему и получивший привилегии суперпользователя, мог и в дальнейшем осуществлять к ней скрытный, несанкционированный, суперпользовательский доступ. То есть, руткит — это и потайной ход, и средство маскировки злоумышленной активности.

Руткиты являются разновидностью троянских программ и подразделяются на бинарные и руткиты уровня ядра. Первые подменяют системные утилиты, вторые — функции ядра, реализующие системные вызовы. Методологию классификации руткитов и детальные сведения о механизмах их функционирования можно найти, например, в статье [40].

Для выявления бинарных руткитов достаточно средств контроля целостности (таких, например, как Tripwire) ключевых системных файлов.

С руткитами уровня ядра ситуация существенно сложнее. Сигнатурный подход, разумеется, неэффективен и в этом случае. Если, например, изменен адрес таблицы системных вызовов в обработчике соответствующего прерывания, то какую сигнатуру и где следует искать? Дополнительную техническую проблему при реализации сканирования и проверки целостности файлов составляет отсутствие доверия к результатам системных сервисов.

Если руткит реализован с помощью механизма загружаемых модулей ядра (а это самый распространенный метод применительно к Linux-системам), можно попытаться, как рекомендуют авторы работы [41], перед загрузкой проводить бинарный статический анализ модулей с элементами символьного выполнения на предмет выявления признаков вредоносного поведения, таких как запись в управляющие структуры ядра. Но, по сути, это обобщенный антивирусный подход, сочетающий поиск сигнатур и эвристики, а его ограниченность известна. Правда, руткиты необходимо выявлять не среди произвольных программ, а среди модулей, тяготеющих к определенной внутренней структуре, характерной, например, для драйверов устройств, но методы обфускации программ и здесь оказываются достаточными для сокрытия признаков вредоносности. Точнее, можно прогнозировать «гонку вооружений» между средствами выявления признаков вредоносного поведения и их сокрытия. Согласно опубликованным в статье [41] результатам, предложенные и реализованные ее авторами методы позволили выявить все проверявшиеся руткиты (их было восемь) и не дали ни одного ложного срабатывания на почти пятистах легальных модулях. Время анализа, как правило, не превышало 10 мс, максимум составлял 420 мс (Pentium IV, 2 ГГц, 1 ГБ ОЗУ). Так что, несмотря на теоретические проблемы, наличие и серьезность которых авторы, разумеется, осознают, первые практические результаты оказались обнадеживающими, хотя пока не решались технические проблемы интеграции с ядром и обеспечения невозможности обхода контролирующего загрузчика модулей.

Загружаемые модули — серьезная угроза безопасности монолитных операционных систем, поскольку они имеют неограниченный доступ к коду и структурам данных ядра. На долю подобных модулей приходится до 70% кода ядра и от 70% до 90% ошибок, среднее время жизни которых составляет около 20 месяцев (см. [42]). Даже если отвлечься от злоумышленных руткитов, остаются угрозы, существующие благодаря уязвимостям в поспешно написанных драйверах устройств. Желательно каким-то образом организовать разграничение доступа в ядре, чтобы не допустить эксплуатации уязвимостей.

Работа [42] развивает направление, намеченное в статье [41]. Она предусматривает спецификацию допустимого и недопустимого поведения («белые» и «черные» списки — адреса, по которым можно или нельзя выполнять переходы, данные, к

которым разрешается или запрещается обращаться, области, где нельзя выполнять машинные инструкции, и т.п.). Частично выполнение спецификаций можно проверить статически, остальное контролируется динамически, за счет вставки проверочного кода. Утверждается, что накладные расходы при этом не превышают 23%. Отметим, однако, что будущее не за такими, явно временными, решениями, а за модульными операционными системами, полноценными моделями безопасности для их компонентов и аппаратной поддержкой при проведении политики безопасности в жизнь.

Руткиты можно считать одной из разновидностей скрытного программного обеспечения, включающего, например, средства протоколирования пользовательских сеансов. Скрываться может как исполняемый код, так и используемые им ресурсы и ассоциированная информация. Например, вредоносный код можно поместить во флэш-память видеокарты, а для выполнения «впрыснуть» в существующий процесс. Ресурсы, такие как файлы и процессы, можно скрыть от пользователя, перехватывая системные вызовы по технологии руткитов. Пробовать выявить скрытное ПО можно по крайней мере двумя способами:

- пытаться обнаружить скрывающие механизмы (рассмотренные выше подходы — из этой категории);
- пытаться получить информацию о системе несколькими способами и отыскать различия в выдаваемых результатах (например, сравнить выдачи команд ls и echo * или информацию от ps и из таблицы процессов в ядре, естественно, предварительно приведя результаты к единому формату).

Раз скрытными ресурсами манипулируют, то можно надеяться, что в каком-нибудь (низкоуровневом) представлении они видны. В этом состоит основная идея подхода, предлагаемого в работе [43]. Может показаться, что искать симптомы вместо первопричины болезни — неправильно, но если симптомы выявить проще, то почему бы этого не сделать? «Сканеры сравнения» можно регулярно запускать на всех компьютерах корпоративной сети, на сканирование одного гигабайта дискового пространства, согласно приведенным в [43] данным, уходит порядка полуминуты, так что подобный подход представляется вполне практичным.

(Напомним, что в статье [2] рассматривается применение «дифференциального» метода для выявления потайных каналов.)

Разумеется, желательно не только не допускать установки руткитов или оперативно выявлять таковые, но и проводить самолечение скомпрометированных систем. Последнее — тема статьи [44]. Идея состоит в том, чтобы отслеживать изменения в таблице системных вызовов, появление скрытых файлов, процессов и сетевых взаимодействий, а по выявлении вредоносной активности — ликвидировать ее, восстанавливая корректное состояние таблицы системных вызовов, удаляя скрытые файлы, терминируя скрытые процессы, блокируя скрытые сетевые соединения. По иронии судьбы, прототип данного защитного средства реализован в виде загружаемого модуля ядра и, следовательно, сам может стать объектом атак руткитов, не говоря уже о проблеме полноты диагностики и лечения систем.

Вопрос о том, кто будет охранять охранников, принадлежит к числу вечных и трудноразрешимых. Если защищаемая и защищающая системы совпадают, нет никаких гарантий, что после компрометации защитные средства будут работать корректно, а осуществляемое ими лечение окажется эффективным и полным. Одним из способов изоляции средств защиты является упоминавшаяся выше технология виртуальных машин. Ее применение в контексте выявления подозрительной активности рассмотрено в работе [45]. К сожалению, в сетевой среде эта технология оказывается, с одной стороны, недостаточной, а с другой — накладной (и авторы работы [45], разумеется, осознают серьезность отмеченных проблем). Недостаточность состоит в том, что, несмотря на виртуализацию, система остается одним узлом сети, подверженным атакам на доступность и удаленной эксплуатации уязвимостей сетевых сервисов. Неэффективность объясняется необходимостью частых переключений между контекстами виртуальных машин и контролирующим их монитором.

(Заметим в скобках, что технология виртуальных машин — замечательное средство эффективной, экономически оправданной реализации приманок и ловушек, исчисляемых десятками тысяч, на нескольких аппаратных серверах, см. [46].)

(Также в скобках как курьез упомянем предложение авторов работы [47] учиться методам обфускации у разработчиков руткитов, однако в этой статье есть прекрасная обзорная врезка с изложением основных фактов и положений, относящихся к руткитам.)

Только аппаратная поддержка сулит прорыв в информационной безопасности, реализацию систем, устойчивых к атакам, способных оперативно, автоматически самовосстанавливаться. К сожалению, системы, подобные той, что описана в работе [48], — дело будущего, причем не очень близкого.

Заключение

В статье [2] совершенно справедливо подчеркивается, как важно правильно поставить задачу и рассматривать ее не изолированно, а в реальном окружении. Правильная постановка, связанная с контролируемым выполнением (ограничением) программ с учетом их семантики, важна для всех разновидностей скрытых и потайных каналов.

В то же время, проблему скрытых и потайных каналов с практической точки зрения никак нельзя отнести к числу самых острых. Побочные каналы — гораздо более реальная угроза для встроенных систем. Безопасность мобильных агентов — болевая точка технологии Интернет/Интранет. Наконец, руткиты и вообще скрытное программное обеспечение, — это опасная угроза, противостоять которой обычные пользователи и многие системные администраторы не в состоянии.

Только программными средствами проблем информационной безопасности не решить. На наш взгляд, в настоящее время наметилась тенденция к расширению аппаратной поддержки защитных средств. Когда подобная поддержка обретет реальные очертания — вопрос не одного года. Чтобы он получил реальное решение, нужны экономические и правовые предпосылки, а не только устрашающая статистика злоумышленной активности и оценки потерь от нее.

Первопричину проблем информационной безопасности следует искать в сложности современных систем. Бороться со сложностью — значит делать системы более безопасными. К сожалению, стремление опередить конкурентов, предложить систему с более богатой функциональностью заставляют производителей двигаться в противоположном направлении. В настоящее время не видно причин, способных эту тенденцию изменить. Системным интеграторам и потребителям остается надеяться только на себя, на свое умение выбирать максимально простую, продуманную архитектуру и поддерживать производственные системы в безопасном состоянии техническими и организационными мерами, тратя силы и средства на отражение реальных, а не надуманных угроз.

Литература

1 Тимонина Е.Е. Скрытые каналы (обзор). — Jet Info, 2002, 11.
2 Галатенко А. О скрытых каналах и не только. — Jet Info, 2002, 11.
3 Kemmerer R.A. A Practical Approach to Identifying Storage and Timing Channels: Twenty Years Later. — Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC'02). — IEEE, 2002.
4 Tumoian E., Anikeev M. Network Based Detection of Passive Covert Channels in TCP/IP. — Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05). — IEEE, 2005.
5 Cabuk S., Brodley C.E., Shields C. IP Covert Timing Channels: Design and Detection. — Proceedings of the CCS'04. — ACM, 2004.
6 Karger P.A. Karth H. Increased Information Flow Needs for High-Assurance Composite Evaluations. — Proceedings of the Second IEEE International Information Assurance Workshop (IWIA'04). — IEEE, 2004.
7 Бетелин В.Б., Бобков С.Г., Галатенко В.А., Годунов А.Н., Грюнталь А.И., Кушниренко А.Г., Осипенко П.Н. Анализ тенденций развития аппаратно-программного обеспечения и их влияния на информационную безопасность. — Сб. статей под ред. академика РАН В.Б. Бетелина. — М.: НИИСИ РАН, 2004.
8 Efstathopoulos P., Krohn M., VanDeBogart S., Frey C., Ziegler D., Kohler E., Mazieres D., Kaashoek F., Morris R. Labels and Event Processes in the Asbestos Operating System. — Proceedings of the SOOP'05. — ACM, 2005.
9 Zhu Y., Bettati R. Anonymity v.s. Information Leakage in Anonymity Systems. — Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05). — IEEE, 2005.
10 Graham B., Zhu Y., Fu X., Bettati R. Using Covert Channels to Evaluate the Effectiveness of Flow Confidentiality Measures. — Proceedings of the 2005 11th International Conference on Parallel

and Distributed Systems (ICPADS'05). — IEEE, 2005.
11 Zhu Y., Sivakumar R. Challenges: Communication through Silence in Wireless Sensor Networks. — Proceedings of the MobiCom'05. — ACM, 2005.
12 Browne R. An Entropy Conservation Law for Testing the Completeness of Covert Channel Analysis. — Proceedings of the CCS'94. — ACM, 1994.
13 Weber S., Karger P.A., Paradkar A. A Software Flaw Taxonomy: Aiming Tools At Security. — Proceedings of the Conference on Software Engineering for Secure Systems — Building Trustworthy Applications (SESS'05). — ACM, 2005.
14 Wray J.C. An Analysis of Covert Timing Channels. — IEEE, 1991.
15 Бетелин В.Б., Галатенко В.А., Кобзарь М.Т., Сидак А.А., Трифаленков И.А. Обзор профилей защиты, построенных на основе «Общих критериев». Специфические требования к сервисам безопасности. — «Безопасность информационных технологий», 2003, 3.
16 Loepere K. Resolving Covert Channels withing a B2 Class Secure System. — Honeywell Information Systems.
17 Harmsen J.J., Pearlman W.A. Capacity of Steganographic Channels. — Proceedings of the MM-SEC'05. — ACM, 2005.
18 Moskowitz I.S., Chang L., Newman R. Capacity is the Wrong Paradigm. — Proceedings of the 2002 Workshop on New Security Paradigms. — ACM, 2002.
19 Bauer M. New Covert Channels in HTTP. Adding Unwitting Web Browsers to Anonymity Sets. — Proceedings of the WPES'03. — ACM, 2003.
20 Borders K., Prakash A. Web Tap: Detecting Covert Web Traffic. — Proceedings of the CCS'04. — ACM, 2004.
21 Slater D. A note on the Relationship Between Covert Channels and Application Verification. — Computer Sciences Corporation, 2005.
22 Tiri K., Verbauwhede I. Simulation Models for Side-Channel Information Leaks. — Proceedings of the DAC 2005. — ACM, 2005.
23 Rao J.R., Rohatgi P., Scerzer H, Tinguely S. Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards. — Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02). — IEEE, 2002.
24 Muresan R., Gebotys C. Current Flattening in Software nad Hardware for Security Applications. — Proceedings of the CODES+ISSS'04. — ACM, 2004.
25 Roth V., Pinsdorf U., Peters J. A Distributed Content-Based Search Engine Based on Mobile Code. — Proceedings of the 2005 ACM Symposium on Applied Computing (SAC'05). — ACM, 2005.
26 Carvalho M., Cowin T., Suri N., Breedy M., Ford K. Using Mobile Agents as Roaming Security Guards to Test and Improve Security of Hosts and Networks. — Proceedings of the 2004 ACM Symposium on Applied Computing (SAC'04). — ACM, 2004.
27 Pedireddy T., Vidal J.M. A Prototype MultiAgent Network Security System. — Proceedings of the AAMAS'03. — ACM, 2003.
28 Menezes R. Self-Organization and Computer Security. — Proceedings of the 2005 ACM Symposium on Applied Computing (SAC'05). — ACM, 2005.
29 Page J., Zaslavsky A., Indrawan M. Countering Agent Security Vulnerabilities using an Extended SENSE Schema. — Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04). — IEEE, 2004.
30 Page J., Zaslavsky A., Indrawan M. Countering Security Vulnerabilities in Agent Execution using a Self Sxecuting Security Examination. — Proceedings of the AAMAS'04. — ACM, 2004.
31 Ameiller J., Robles S., Ortega-Ruiz J.A. Self-Protected Mobile Agents. — Proceedings of the AAMAS'04. — ACM, 2004.
32 Christodorescu M., Jha S. Testing Malware Detectors. — Proceedings of the ISSTA'04. — ACM, 2004.
33 Christodorescu M., Jha S., Seshia S.A., Song D., Bryant R.E. Semantics-Aware Malware Detection. — Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P'05). — IEEE, 2005.
34 McHugh J.A.M., Deek F.P. An Incentive System for Reducing Malware Attacks. — Communications of the ACM, 2005, 6.
35 Harrison J.V. Enhancing Network Security By Preventing User-Initiated Malware Execution. — Proceedings of the International Conference on Information Technology Coding and Computing (ITCC'05). — IEEE, 2005.
36 Bohra A., Neamtiu I., Gallard P., Sultan F., Iftode L. Remote Repair of Operating System State Using Backdoors. — Proceedings of the International Conference on Autonomic Computing (ICAC'04). — IEEE, 2004.
37 Sultan F., Bohra A., Smaldone S., Pan Y., Gallard P., Neamtiu I., Iftode L. Recovering Internet Service Sessions from Operating System Failures. — IEEE Internet Computing, 2005, March/April.
38 Grizzard J.B., Krasser S., Owen H.L., Conti G.J., Dodson E.R. Towards an Approach for Automatically Repairing Compromised Network

Systems. — Proceedings of the Third IEEE International Symposium on Network Computing and Applications (NCA'04). — IEEE, 2004.
39 Goel A., Po K., Farhadi K., Li Z., de Lara E. The Taser Intrusion Recovery System. — Proceedings of the SOSP'05. — ACM, 2005.
40 Levine J., Grizzard J., Owen H. A Methodology to Detect and Characterize Kernel Level Rootkit Exploits Involving Redirection of the System Call Table. — Proceedings of the Second IEEE International Information Assurance Workshop (IWIA'04). — IEEE, 2004.
41 Kruegel C., Robesrtson W., Vigna G. Detecting Kernel-Level Rootkits Through Binary Analysis. — Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04). — IEEE, 2004.
42 Xu H., Du W., Chapin S.J. Detecting Exploit Code Execution in Loadable Kernel Modules. — Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04). — IEEE, 2004.
43 Wang Y.-M., Beck D., Vo B., Roussev R., Verbowski C. Detecting Stealth Software with Strider GhostBuster. — Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05). — IEEE, 2005.
44 Ring S., Esler D., Cole E. Self-Healing Mechanisms for Kernel System Compromises. — Proceedings of the WOSS'04. — ACM, 2004.
45 Laureano M., Maziero C., Jamhour E. Intrusion Detection in Virtual Machine Environments. — Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04). — IEEE, 2004.
46 Vrable M., Ma J., Chen J., Moore D., Vandekieft E., Snoeren A.C., Voelker G.M., Savage S. Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm. — Proceedings of the SOSP'05. — ACM, 2005.
47 Ring S., Cole E. Taking a Lesson from Stealthy Rootkits. — IEEE Security & Privacy, 2004, July/August.
48 Shi W., Lee H.-H.S., Gu G., Falk L. An Intrusion-Tolerant and Self-Recoverable Network Service System Using A Security Enhanced Chip Multiprocessor. — Proceedings of the Second International Conference on Autonomic Computing (ICAC'05). — IEEE, 2005.

Jet Info
ИНФОРМАЦИОННЫЙ БЮЛЛЕТЕНЬ
Издается с 1995 года
Издатель: компания Джет Инфо Паблишер

Главный редактор: Дмитриев В.Ю. *(vlad@jet.msk.su)*
Технический редактор: Лапина И.К. *(lapina@jet.msk.su)*
Россия, 127015, Москва, Б. Новодмитровская, 14/1
тел. (495) 411 76 01
факс (495) 411 76 02
email: JetInfo@jet.msk.su http://www.jetinfo.ru

Подписной индекс по каталогу Роспечати 32555

Полное или частичное воспроизведение материалов, содержащихся в настоящем издании, допускается только по согласованию с издателем