Методы информационно-технического воздействия на киберсистемы и возможные способы противодействия

С. А. Петренко

Арсенал технических приемов, позволяющих различного рода злоумышленникам осуществлять информационно-технические воздействия на киберсистемы, постоянно растет и пополняется. При этом к основным целям нападения относятся:

- перехват (и, возможно, модификация) данных, передаваемых через сеть от одного узла другому;
- имперсонация (узел злоумышленника выдает себя за другой узел, чтобы воспользоваться какими-либо привилегиями имитируемого узла);
- несанкционированное подключение к сети;
- несанкционированная передача данных (обход правил фильтрации IP-трафика в сетях, защищенных брандмауэрами);
- принуждение узла к передаче данных на завышенной скорости;
- приведение узла в состояние, когда он не может нормально функционировать, передавать и принимать данные (так называемый DoS — denial of service, отказ в обслуживании).

Для достижения своих целей злоумышленники используют *прослушивание* (*sniffing*), *сканирование сети* и *генерацию пакетов*. Давайте рассмотрим названные методы информационно-технического воздействия на компьютерные сети критически важных объектов и возможные способы противодействия подробнее.

1. Прослушивание сети

Прослушивание сети Ethernet (а подавляющее большинство локальных и федеральных сетей используют именно эту технологию) является тривиальной задачей: для этого нужно просто перевести интер-

фейс в режим прослушивания (*promiscuous mode*). Легко доступны программы, не только записывающие весь трафик в сегменте Ethernet, но и выполняющие его отбор по установленным критериям: например, свободно распространяемая программа *tcpdump* или входящая в поставку ОС Solaris программа *snoop*.

Среди других сетевых технологий подвержены прослушиванию сети FDDI и радиосети (например, Radio Ethernet). Несколько сложнее для злоумышленника извлечь трафик из выделенных и телефонных линий — главным образом из-за сложности физического доступа и подключения к таким линиям. Однако следует помнить, что злоумышленник может атаковать и оккупировать промежуточный маршрутизатор и таким образом получить доступ ко всему транзитному трафику, независимо от используемых технологий на уровне доступа к сети.

Ограничить область прослушивания в сети Ethernet можно разбиением сети на сегменты с помощью коммутаторов. В этом случае злоумышленник, не прибегая к активным действиям, может перехватить только кадры, получаемые или отправляемые узлами сегмента, к которому он подключен. Единственным способом борьбы с прослушиванием сегмента Ethernet является шифрование данных.

Злоумышленник, прослушивающий сеть, может быть обнаружен с помощью системной утилиты *AntiSniff*, которая выявляет в сети узлы, чьи интерфейсы переведены в режим прослушивания.

Отметим, что представление о прослушивании, как о безопасной деятельности, которую нельзя обнаружить, не соответствует действительности.

2. Сканирование сети

Сканирование сети имеет своей целью выявление подключенных к сети компьютеров и определение работающих на них сетевых сервисов (открытых портов TCP или UDP). Первая задача выполняется посылкой ICMP-сообщений «Echo» с помощью программы *ping* с последовательным перебором адресов узлов в сети.

Для большей скрытности злоумышленник может существенно растянуть процесс сканирования во времени («медленное сканирование») — это же касается и сканирования портов TCP/UDP. Также злоумышленник может применить «обратное сканирование» (*inverse mapping*): в этом случае на тестируемые адреса посылаются не сообщения ICMP «Echo», а другие сообщения, например RST-сегменты TCP, ответы на несущест-

вующие DNS-запросы и т. п. Если тестируемый узел не существует (выключен), злоумышленник получит в ответ ICMP-сообщение «Destination Unreachable: Host Unreachable». Следовательно, если сообщение не было получено, то соответствующий узел подключен к сети и работает.

Программа *traceroute* поможет в определении топологии сети и обнаружении маршрутизаторов.

Для определения того, какие UDP- или TCP-приложения запущены на обнаруженных компьютерах, используются программы-сканеры, например, программа *nmap*. Поскольку номера портов всех основных сервисов стандартизованы, то, определив, например, что порт 25/TCP открыт, можно сделать вывод о том, что данный хост является сервером электронной почты, и т. д. Полученную информацию злоумышленник может использовать для развертывания атаки на уровне приложения.

Сканирование TCP-портов хоста производится несколькими способами. Наиболее простой способ — установление TCP-соединения с тестируемым портом с помощью функции *connect*(). Если соединение удалось установить, значит, порт открыт и к нему подсоединено серверное приложение. Достоинством этого способа является возможность выполнения сканирования любым пользователем и даже без специального программного обеспечения: стандартная программа *telnet* позволяет указать произвольный номер порта для установления соединения. Существенный недостаток — возможность отслеживания и регистрации такого сканирования: при анализе системного журнала сканируемого хоста будут обнаружены многочисленные открытые и сразу же прерванные соединения, в результате чего могут быть приняты меры по повышению уровня безопасности.

Сканирование в режиме половинного открытия (*half-open scanning*) не имеет описанного недостатка, но требует от злоумышленника возможности формировать одиночные TCP-сегменты в обход стандартного модуля TCP (или, при использовании уже написанных программ, как минимум — прав суперпользователя). В этом режиме злоумышленник направляет на сканируемый порт SYN-сегмент и ожидает ответа. Получение ответного сегмента с битами SYN и ACK означает, что порт открыт; получение сегмента с битом RST означает, что порт закрыт. Получив SYN+ACK, злоумышленник немедленно отправляет на обнаруженный порт сегмент с битом RST, таким образом ликвидируя попытку соединения. Так как соединение так и не было открыто (ACK от злоумышленника не был получен), то зарегистрировать такое сканирование гораздо сложнее.

Третий способ — сканирование с помощью FIN-сегментов. В этом случае на сканируемый порт посылается сегмент с установленным битом FIN. Хост должен ответить RST-сегментом, если FIN-сегмент адресован закрытому порту. FIN-сегменты, направленные на порт, находящийся в состоянии LISTEN, многими реализациями TCP/IP игнорируются (стандарт требует в состоянии LISTEN посылать RST-сегменты в ответ на сегменты, имеющие неприемлемый ACK SN; про сегменты, имеющие только флаг FIN, ничего не говорится). Таким образом, отсутствие отклика говорит о том, что порт открыт. Варианты этого способа сканирования — посылка сегментов с флагами FIN, PSH, URG («*Xmas scan*») или вообще без всяких флагов («*Null scan*»).

Конечно, сканирование SYN-сегментами дает более надежные результаты, однако многие брандмауэры могут не пропускать SYN-сегменты из Интернет во внутреннюю сеть, в то же время пропуская любые сегменты без флага SYN (так запрещаются соединения хостов Интернет с внутренними хостами, инициируемые из Интернет, но разрешаются соединения, инициируемые изнутри).

Программа *tcplogd* может зарегистрировать попытки сканирования в различных режимах.

Для определения открытых портов UDP злоумышленник может отправить на тестируемый порт UDP-сообщение. Получение в ответ ICMP-сообщения «Port Unreachable» (тип 3, код 3) говорит о том, что порт закрыт.

Программа-сканер может также определить операционную систему сканируемого узла по тому, как узел реагирует на специальным образом сконструированные, нестандартные пакеты: например, TCP-сегменты с бессмысленными сочетаниями флагов или ICMP-сообщения некоторых типов, и по другим признакам.

Отметим, что для определения адресов работающих в сети компьютеров и запущенных на них UDP- или TCP-сервисах злоумышленник, непосредственно подключенный к сегменту сети, может использовать простое прослушивание. Такая форма сканирования сети является более скрытной, чем рассылка тестирующих дейтаграмм.

3. Генерация пакетов

Генерация дейтаграмм или кадров произвольного формата и содержания производится не менее просто, чем прослушивание сети Ethernet. Библиотека *libnet* обеспечит программиста всем необходимым для ре-

шения этой задачи. Библиотека *libpcap* предоставляет инструментарий для обратного действия — извлечения пакетов из сети и их анализа.

На многочисленных сайтах Интернет имеются уже готовые программы, генерирующие пакеты целенаправленно для выполнения какой-либо атаки или сканирования сети (например, программа *nmap*, упомянутая выше). Применение таких программ часто не требует от злоумышленника ни квалификации программиста, ни понимания принципов работы сети, что делает многие из описанных атак, особенно атаки типа «отказ в обслуживании», широко доступными для исполнения.

4. Перехват данных

Простейшей формой перехвата данных является прослушивание сети. В этом случае злоумышленник может получить полезную информацию: имена пользователей и пароли (многие приложения передают их в открытом виде), адреса компьютеров в сети, в том числе адреса серверов и запущенные на них приложения, адрес маршрутизатора, собственно передаваемые данные, которые могут быть конфиденциальными (например, тексты электронных писем) и т. п.

Однако если сеть разбита на сегменты с помощью коммутаторов, то злоумышленник может перехватить только кадры, получаемые или отправляемые узлами сегмента, к которому он подключен. Простое прослушивание также не позволяет злоумышленнику модифицировать передаваемые между двумя другими узлами данные. Для решения этих задач злоумышленник должен перейти к активным действиям, чтобы внедрить себя в тракт передачи данных в качестве промежуточного узла.

Ложные ARP-ответы

Для перехвата трафика между узлами A и B, расположенными в одной IP-сети, злоумышленник использует протокол ARP. Он рассылает сфальсифицированные ARP-сообщения так, что каждый из атакуемых узлов считает MAC-адрес злоумышленника адресом своего собеседника (рис. 1).

Возможный план компьютерной атаки:

1. Злоумышленник определяет MAC-адреса узлов A и B:

   ping IP(A); ping IP(B); arp –a

2. Злоумышленник конфигурирует дополнительные логические IP-интерфейсы на своем компьютере, присвоив им адреса A и B и отклю-

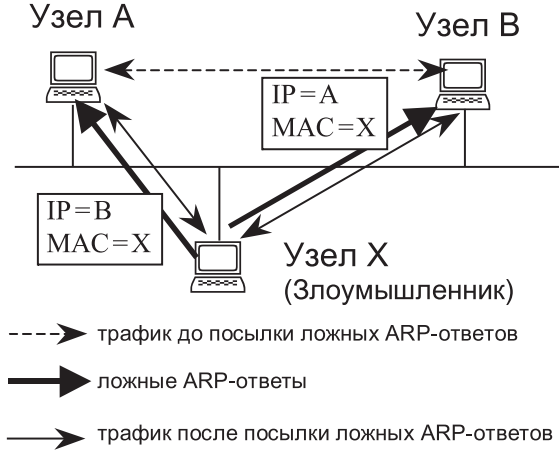

Рис. 1. Схема ARP-атаки

чив протокол ARP для этих интерфейсов, чтобы их не было «слышно» в сети, например:

ifconfig hme0:1 inet IP(A) –arp
ifconfig hme0:2 inet IP(B) –arp,

устанавливает статическую ARP-таблицу:

arp –s IP(A) MAC(A)
arp –s IP(B) MAC(B)

и разрешает ретрансляцию IP-дейтаграмм (переводит компьютер в режим маршрутизатора).

3. Злоумышленник отправляет на MAC-адрес узла A кадр со сфабрикованным ARP-ответом, в котором сообщается, что IP-адресу B соответствует MAC-адрес X. Аналогично узлу B сообщается, что IP-адресу A соответствует тот же MAC-адрес X (рис. 1.; для формирования сообщений можно использовать библиотеку libnet). Так как ARP — протокол без сохранения состояния, то узлы A и B примут ARP-ответы, даже если они не посылали запросов. Далее злоумышленник периодически (достаточно это делать раз в 40 секунд) повторяет посылку сфальсифицированных ARP-ответов для обновления сфабрикованных записей в ARP-таблицах узлов A и B, чтобы удерживать эти узлы в неведении относительно истинных адресов и не дать им сформировать соответствующие ARP-запросы.
4. Введенные в заблуждение узлы A и B пересылают свой трафик через узел X, полагая, что сообщаются друг с другом непосредст-

венно. Злоумышленник может просто прослушивать трафик или изменять передаваемые данные в своих интересах.

Узел B может быть шлюзом сети, в которой находится узел A. В этом случае злоумышленник может перехватывать весь трафик между узлом A и Интернет.

Также злоумышленник может вообще не передавать кадры узла A узлу B, а выдавать себя за узел B, фабрикуя ответы от его имени и отсылая их в A.

Разделение сети на сегменты с помощью коммутаторов, очевидно, не является препятствием для описанной ARP-атаки.

Отметим, что в сфабрикованных ARP-ответах злоумышленник может вместо своего MAC-адреса указать несуществующий адрес Ethernet. Впоследствии он может извлекать из сети кадры, направленные на этот адрес путем прослушивания сети или перепрограммирования MAC-адреса своей сетевой карты. Это потребует несколько бóльших усилий, но взамен злоумышленник сможет практически полностью замести следы, поскольку его собственный MAC-адрес уже нигде не фигурирует.

Для обнаружения ARP-атак администратор должен вести базу данных соответствия MAC- и IP-адресов всех узлов сети и использовать программу *arpwatch*, которая прослушивает сеть и уведомляет администратора о замеченных нарушениях. Если сеть разделена на сегменты коммутаторами, то администратор должен настроить их таким образом, чтобы в сегмент, где находится станция администратора, перенаправлялись кадры из всех сегментов сети вне зависимости от того, кому они предназначены.

Использование статических ARP-таблиц, по крайней мере — на ключевых узлах (серверах, маршрутизаторах), защитит их от ARP-атаки, правда, за счет накладных расходов на поддержку этих таблиц в актуальном состоянии.

Навязывание ложного маршрутизатора

Для перехвата трафика, направленного от некоторого узла A в другую сеть, злоумышленник может навязать хосту свой адрес в качестве адреса маршрутизатора, которому должны быть переданы отправляемые узлом A данные. В этом случае узел A будет направлять трафик на узел злоумышленника, который после анализа и, возможно, модификации данных, отправит их далее настоящему маршрутизатору.

Как правило, навязывание ложного маршрутизатора выполняется с помощью фальсифицированных ICMP-сообщений «Redirect». В под-

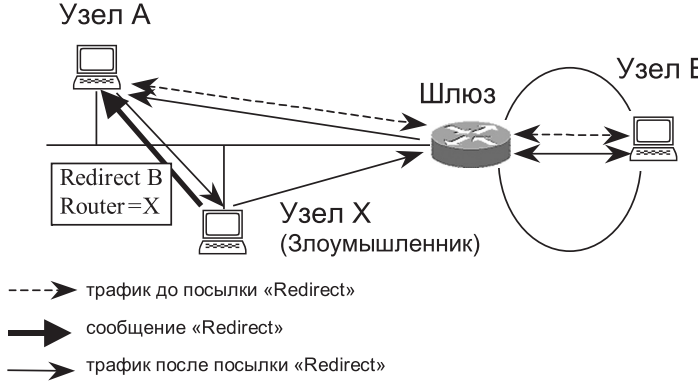

Рис. 2. Навязывание ложного маршрутизатора с помощью ICMP «Redirect»

ложном сообщении злоумышленник объявляет свой собственный адрес в качестве адреса маршрутизатора (рис. 2).

Для устранения возможности названной атаки необходимо отключить на хосте обработку сообщений «Redirect». Несмотря на то что это действие противоречит требованиям к хостам, оно выглядит совершенно разумным, особенно для хостов в сетях с единственным шлюзом, однако не все операционные системы могут поддерживать такое отключение.

Если злоумышленник хочет перехватить трафик между узлами сети P и узлами сети Q, и при этом не находится ни в одной из сетей P или Q, но расположен на пути между ними, он может попытаться ввести в заблуждение маршрутизаторы. Маршрутизаторы не реагируют на сообщения ICMP «Redirect», поэтому для успешной атаки необходимо, чтобы они использовали какой-либо протокол маршрутизации. В этом случае злоумышленник может сформировать подложные сообщения протокола маршрутизации с целью переключения требуемых маршрутов на себя. Например (рис. 3) узел X, приняв широковещательные RIP-сообщения, рассылаемые узлами A (вектор P = 3) и B (вектор Q = 2), отправляет сообщение с вектором Q =1 на индивидуальный адрес маршрутизатора A, а сообщение P = 2 — на индивидуальный адрес B.

Возможна ситуация, когда значение вектора, объявляемого, например, маршрутизатором B: Q = 1. В этом случае X не может немедленно предложить лучшего маршрута, но он может применить следующий прием. Сначала, выбрав паузу в рассылке RIP-сообщений маршрутизатором B, X от имени B отправляет в A вектор Q = 16, что заставит

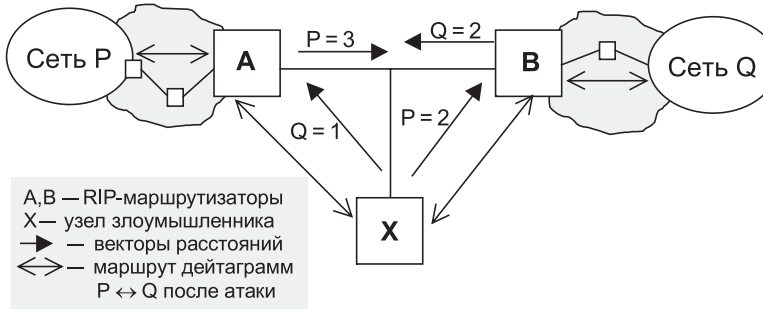

Рис. 3. Навязывание ложного RIP-маршрутизатора X для перехвата трафика между сетями P и Q

маршрутизатор A удалить из своей таблицы маршрут в сеть Q, так как до этого A отправлял дейтаграммы в Q через B. Сразу же вслед за этим X отправляет вектор Q = 1 от своего имени и A устанавливает маршрут в сеть Q через X. последующие векторы Q = 1 от B будут проигнорированы, поскольку они не предлагают лучшего маршрута.

По аналогичной схеме можно провести атаку на протоколы EIGRP и OSPF.

Отметим, что для осуществления этой атаки узел X должен находиться в одной сети с маршрутизаторами A и B, поэтому следует воздержаться от размещения хостов в транзитных сетях. Впрочем, узел X может быть оккупированным злоумышленником маршрутизатором, поэтому отсутствие в транзитной сети хостов не означает невозможности провести описанную атаку.

Защитой от подобных атак является аутентификация сообщений протокола маршрутизации, например с помощью отечественного алгоритма аналогичного MD5.

Особенностью протокола BGP в контексте обсуждаемой атаки является использование протокола TCP и указание IP-адресов BGP-соседей. Таким образом, для внесения ложной маршрутной информации в базу данных протокола BGP маршрутизатора A злоумышленник не может действовать ни от своего имени, ни от имени несуществующего узла: он должен имитировать маршрутизатор B (последний, соответственно, должен быть каким-то образом устранен из процесса). Отметим две возможные схемы атаки на BGP.

BGP-соседи из разных автономных систем должны быть доступны друг другу непосредственно, то есть находиться одной IP-сети. Следовательно, для осуществления атаки злоумышленник также должен на-

ходиться в той же сети. Технически такая ситуация возможна, но практически маловероятна, так как связи между автономными системами — это скорее всего выделенные каналы, физически недоступные никому, кроме сетевых администраторов. Говоря здесь «выделенные каналы», мы имеем в виду не используемую сетевую технологию (это может быть и сеть Ethernet, как во многих точках обмена трафиком), а физическую изолированность сети и невозможность несанкционированного подключения к ней.

В противоположность внешним BGP-соединениям, BGP-соседи в одной автономной системе не обязательно должны находиться в одной сети. Это открывает несколько больший простор для деятельности злоумышленника. В зависимости от того, где он находится по отношению к BGP-маршрутизаторам, злоумышленник может применять различные методы имперсонации. Например, учитывая, что IBGP-соседи для пересылки дейтаграмм друг другу могут пользоваться результатами работы внутреннего протокола маршрутизации, злоумышленник может предварительно атаковать протокол внутренней маршрутизации, замкнув на себя трафик между сетями, в которых находятся BGP-маршрутизаторы (например, это сети P и Q на рис. 3) и модифицируя данные BGP-соединения в своих целях.

Атака злоумышленника на протокол BGP выглядит трудноосуществимой, но тем не менее такие атаки возможны. Аутентификация TCP-сегментов поможет избежать неприятностей.

5. Имперсонация

Предположим, что узел A обменивается IP-дейтаграммами с узлом B, при этом узлы идентифицируют друг друга по IP-адресам, указываемым в дейтаграммах. Предположим далее, что узел B имеет особые привилегии при взаимодействии с A: то есть, A предоставляет B некоторый сервис, недоступный для других хостов Интернет. Злоумышленник на узле X, желающий получить такой сервис, должен имитировать узел B — такие действия называются имперсонацией узла B узлом X.

Говоря о сервисах, мы имеем в виду приложения UDP или TCP, то есть, речь идет об имперсонации UDP-сообщений или TCP-соединений. Часто одновременно с имперсонацией злоумышленник предпринимает атаки типа «отказ в обслуживании» против узла B для исключения последнего из процесса сетевого взаимодействия.

Хосты A, B и X могут располагаться друг относительно друга различным образом, от этого зависит, какие методы имперсонации применит злоумышленник.

Если злоумышленник может перехватить трафик идущий от узла A к B, то задача имперсонации практически выполнена. Получая от узла A UDP-сообщения или TCP-сегменты, адресованные узлу B, злоумышленник не передает их по назначению, а вместо этого сам формирует пакеты от имени B. При этом узел B даже не подозревает о происходящем, поэтому атаковать его на предмет отказа в обслуживании не имеет смысла.

Напомним, что перехват трафика возможен, когда

1) A, B и X находятся в одной IP-сети (ARP-атака), или
2) A и X находятся в одной сети, а B — в другой (навязывание ложного маршрутизатора), или
3) A и B находятся в разных сетях, а X находится на пути между ними (или включает себя в маршрут путем атаки на протокол маршрутизации).

В остальных случаях злоумышленник не может перехватить данные, передаваемые из A в B. По-прежнему ничто не мешает ему отправлять в адрес A сфальсифицированные дейтаграммы от имени B, но ответные пакеты A будет отправлять узлу B, минуя злоумышленника. Важным обстоятельством в этих условиях является то, имеет ли узел X возможность подслушивать эти ответные пакеты, или же злоумышленник вынужден работать вслепую.

Рассмотрим самый сложный случай: перехват и прослушивание данных, отправляемых из A в B невозможны. Этот случай является наиболее общим: узел X находится в сети, не имеющей никакого отношения к узлам A и B и не лежащей между ними (A и B могут находиться как в одной, так и в разных сетях).

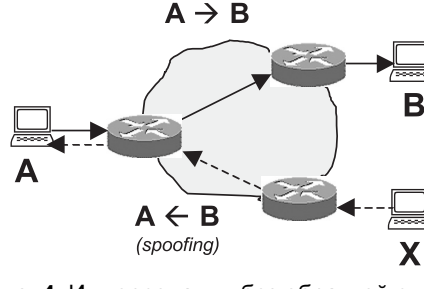

Рис. 4. Имперсонация без обратной связи

Подчеркнем, что имперсонация без обратной связи имеет смысл лишь тогда, когда злоумышленнику для достижения своей цели достаточно только передать данные на узел A от имени узла B, и последующий ответ узла A уже не имеет значения. Классическим примером такой атаки является отправка злоумышленником TCP-сегмента на порт программы *rlogin*, содержащего какую-либо команду операционной системе узла A. Узел A выполняет эту команду, полагая, что она поступила с узла B.

Имперсонация TCP-соединения без обратной связи

Если имперсонация UDP-сообщений без обратной связи остается тривиальной: злоумышленник должен только сфабриковать дейтаграмму, адресованную от узла B узлу A, и отправить ее по назначению, то в случае с TCP все обстоит не так просто. Прежде чем отправить узлу A сегмент с данными, узел X должен установить с ним соединение от имени узла B. Напомним, как происходит установление соединения: узел X от имени B отправляет в A сегмент с битом SYN, где указывает начальный номер ISN(B). Узел A отвечает узлу B SYN-сегментом, в котором подтверждает получение предыдущего сегмента, и устанавливает свой начальный номер ISN(A). Этот сегмент злоумышленник никогда не получит.

Здесь возникает две проблемы: во-первых, узел B, получив от A ответ на SYN-сегмент, который он никогда не посылал, отправит узлу A сегмент с битом RST, тем самым сводя к нулю усилия злоумышленника. Во-вторых, узел X все равно не сможет отправить в A следующий сегмент (как раз это должен быть сегмент с данными), потому что в этом сегменте узел X должен подтвердить получение SYN-сегмента от A, то есть поместить в поле «ACK SN» заголовка своего сегмента значение ISN(A)+1. Но злоумышленник не знает номера ISN(A), потому что соответствующий сегмент ушел к узлу B.

Первая проблема решается относительно просто: злоумышленник проводит против узла B атаку типа «отказ в обслуживании» с тем расчетом, узел B не был способен обрабатывать сегменты, приходящие из A. Например, можно поразить узел B шквалом SYN-сегментов от несуществующих узлов. Впрочем, к облегчению злоумышленника, может оказаться, что узел B просто выключен.

Для решения второй проблемы злоумышленник должен уметь предсказывать значения ISN(A). Если операционная система узла A использует какую-либо функцию для генерации значений ISN (например, линейную зависимость от показания системных часов), то последовательно

открыв несколько пробных соединений с узлом A и проанализировав присылаемые в SYN-сегментах от A значения ISN, злоумышленник может попытаться установить эту зависимость опытным путем.

Хорошая реализация TCP должна использовать случайные числа для значений ISN (более подробное обсуждение этого вопроса можно найти в RFC-1948). Несмотря на кажущуюся простоту этого требования проблема с угадыванием номеров ISN остается актуальной и по сей день.

Итак, приведем схему атаки для имперсонации TCP-соединения без обратной связи (рис. 5).

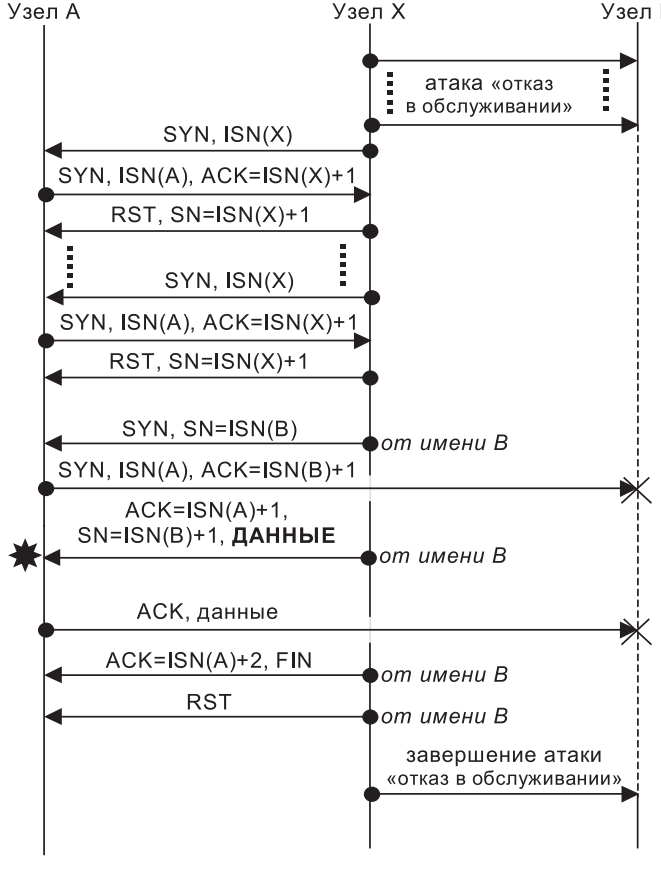

Рис. 5. Схема атаки с имперсонацией TCP-соединения без обратной связи

1. Злоумышленник выводит из строя узел B.
2. Злоумышленник делает несколько пробных попыток установить соединения с узлом A с целью получить от A последовательность значений ISN(A). Сразу после поступления SYN-сегмента от A злоумышленник разрывает наполовину установленное соединение посылкой сегмента с флагом RST. Проанализировав полученные значения ISN(A), злоумышленник определяет закон формирования этих значений.
3. Злоумышленник отправляет в A SYN-сегмент от имени B.
4. Узел A отвечает узлу B свои SYN-сегментом, подтверждающим получение SYN-сегмента от B, и указывает значение ISN(A) для этого соединения. Злоумышленник не видит этого сегмента.
5. На основе ранее полученных данных злоумышленник предсказывает значение ISN(A) и отправляет в A сегмент от имени B, содержащий подтверждение ISN(A)+1 и данные для прикладного процесса. Получив этот сегмент, узел A считает соединение с B установленным и передает поступившие данные прикладному процессу. Цель атаки достигнута. Данные могут быть, например, командой, которую узел A выполняет, потому что она поступила от доверенного узла B.
6. Узел A отправляет подтверждение получения данных и, возможно, свои данные в узел B. Злоумышленник этих сегментов не получит, но они его (по условиям задачи) и не интересуют. Чтобы корректно закрыть соединение, злоумышленник может вслепую отправить в узел A от имени узла B подтверждение получения одного октета (ACK SN=ISN(A)+2), а следом выслать сегмент с флагом FIN. Таким образом канал передачи данных от узла X (он же B) к узлу A корректно закрыт. Для полного закрытия соединения злоумышленник должен подтвердить получение FIN-сегмента от A (равно как и всех данных, которые этому сегменту предшествовали) — разумеется, он этого сделать не может, потому что в общем случае ни объем этих данных, ни время отправки FIN-сегмента из A ему не известны. Но поскольку данные, передаваемые из A в B для злоумышленника ценности не имеют, он просто отправляет в A сегмент с флагом RST, тем самым полностью ликвидируя соединение.
7. Злоумышленник завершает атаку «отказ в обслуживании» против узла B.

Десинхронизация TCP-соединения

Злоумышленник X, находящийся в одном сегменте сети с узлами A и B или на пути между A и B, может произвести десинхронизацию TCP-соединения между A и B для установления полного контроля над соединением, то есть, злоумышленник получит возможность действовать как от имени A, так и от имени B.

Определим, что такое десинхронизация TCP-соединения. При установленном соединении каждый из узлов A и B знает, октеты с какими номерами может прислать ему собеседник в данный момент: если последнее подтверждение, высланное узлом A, было $ACK_{A\to B}$ и при этом узел A объявил окно $W_{A\to B}$, то A ожидает от B октетов с номерами $SN_{B\to A}$, попадающими в объявленное окно, то есть:

$$ACK_{A\to B} \le SN_{B\to A} \le ACK_{A\to B} + W_{A\to B}$$

Аналогично в узле B ожидается от A:

$$ACK_{B\to A} \le SN_{A\to B} \le ACK_{B\to A} + W_{B\to A}$$

Если, например, узел A по какой-то причине получает от B сегмент с номером $SN_{B\to A}$, не попадающим в окно, то этот сегмент уничтожается, а в ответ A отправляет в B сегмент с $SN_{A\to B}$, $ACK_{A\to B}$, $W_{A\to B}$, чтобы указать узлу B, какие именно октеты ожидает получить A. Отметим, что скорее всего этот сегмент не содержит данных, но номер $SN_{A\to B}$ в этом сегменте тем не менее должен быть указан, где $SN_{A\to B}$ — номер следующего октета данных, который A когда-либо вышлет в B.

Предположим, злоумышленнику каким-то образом удалось сбить показатели счетчиков узлов A и (или) B так, что вышеприведенные неравенства больше не выполняются (как это можно сделать, мы обсудим ниже). Далее мы будем использовать обозначения вида $SN_{A\to B(B)}$, что означает «приемлемый $SN_{A\to B}$ с точки зрения B».

Теперь, если B посылает в A сегмент с неким номером $SN_{B\to A(B)}$, адекватным с точки зрения B, но уже не попадающим в окно в узле A, то A возвращает узлу B подтверждение со своим значением $ACK_{A\to B} = SN_{B\to A(A)}$. Однако в этом же сегменте имеется номер $SN_{A\to B(A)}$, который теперь уже B рассматривает, как не попадающий в свое окно, и отправляет в A подтверждение $SN_{B\to A(B)}$, $ACK_{B\to A} = SN_{A\to B(A)}$. Номер $SN_{B\to A(B)}$, как и раньше, неприемлем для A, и узел A вновь отправляет в B подтверждение, и этот цикл, называемый *ACK-шторм*, теоретически продолжается до бесконечности, а практически — до тех пор, пока один

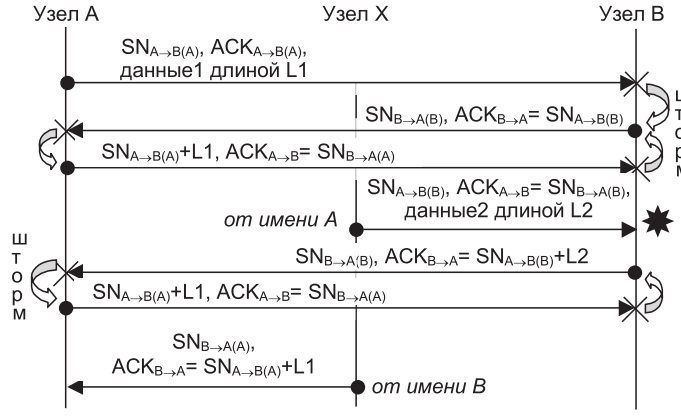

Рис. 6. Действия злоумышленника-посредника в десинхронизированном соединении между узлами A и B (L1, L2 — объем данных в пересылаемых сегментах)

из ACK-сегментов не потеряется в сети. Чем сильнее шторм, тем больше загрузка сети, тем выше процент потерь, следовательно, тем быстрее шторм прекратится.

Итак, в десинхронизированном состоянии любая попытка обмена данными вызывает только ACK-шторм, а сами сегменты с данными участниками соединения уничтожаются.

В это время злоумышленник, знающий «правильные» номера с точки зрения обоих узлов, берет на себя функции посредника (рис. 6). Он прослушивает сеть, обнаруживает сегмент с данными длиной L октетов, направленный, например, из A в B, меняет в нем номер $SN_{A\to B(A)}$ на ожидаемый узлом B номер $SN_{A\to B(B)}$, и, пересчитав контрольную сумму, отправляет сегмент в B от имени A. После этого в узел A от имени B злоумышленник отправляет подтверждение на этот сегмент, содержащее правильный с точки зрения A номер $SN_{B\to A(A)}$. Во время этого обмена в виде побочного явления возникают два ACK-шторма: первый инициирует узел B, получивший из A оригинальный сегмент с $SN_{A\to B(A)} \neq SN_{A\to B(B)}$, а второй шторм возникает, когда узел A получает от B сегмент, подтверждающий получение данных от X, и в этом сегменте $SN_{B\to A(B)} \neq SN_{B\to A(A)}$.

Разумеется, злоумышленник затевает атаку не для того, чтобы просто ретранслировать сегменты, которые он и так может подслушать. Ничто не мешает ему изменять содержащиеся в сегментах данные или добавлять свои: на рис. 6 это отображено в виде «данных2»,

имеющих длину L2 октетов, в то время как оригинальные данные обозначены «данные1» длиной L1 октетов. Например, если имеет место сессия программы *telnet* и А посылает в В некоторую команду, то злоумышленник может вставить в этот сегмент еще одну команду. Результат выполнения своей команды он получит, подслушав ответный сегмент $SN_{B\to A(B)}$, направленный из В в А, который узел А не воспримет из-за несовпадения порядковых номеров, так как $SN_{B\to A(B)} \neq SN_{B\to A(A)}$. Зато злоумышленник, удалив из этого сегмента результат выполнения своей команды, отправит то, что осталось (то есть результат оригинальной команды) в А от имени В уже с приемлемым порядковым номером $SN_{B\to A(A)}$.

Злоумышленник может вообще игнорировать сегменты, посылаемые узлом А и отправлять в В только свои данные, получая ответы прослушиванием сети и соответственно реагируя на них, но в этом случае узел А заметит, что В не отвечает на его команды, и может забеспокоиться.

Рассмотрим, каким образом злоумышленник может перевести TCP-соединение в десинхронизированное состояние.

А. Ранняя десинхронизация (рис. 7): злоумышленник, прослушивая сеть, обнаруживает момент установления соединения между А и В, от имени А сбрасывает соединение RST-сегментом и тут же открывает его заново, но уже с новыми номерами ISN. Разберем эту процедуру в деталях.

1. Сначала А посылает в В сегмент $SYN_{A\to B}$, $ISN_{A\to B(A)}$, потом В отвечает сегментом $SYN_{B\to A}$, $ISN_{B\to A(1)}$, $ACK_{B\to A}(ISN_{A\to B(A)}+1)$. По получении этого сегмента А переходит в состояние ESTABLISHED и посылает в В подтверждающий сегмент $ACK_{A\to B}(ISN_{B\to A(1)}+1)$.
2. В этот момент злоумышленник от имени А отправляет в В сегмент $RST_{A\to B}$ и следом за ним сегмент $SYN_{A\to B}$, $ISN_{A\to B(X)}$, содержащий те же номера портов, но другой номер $ISN_{A\to B} = ISN_{A\to B(X)}$, неприемлемый для А.
3. При получении этих сегментов узел В закрывает установленное соединение с А, а затем тут же вновь отрывает его и отправляет в А сегмент $SYN_{B\to A}$, $ISN_{B\to A(2)}$, $ACK_{B\to A}(ISN_{A\to B(X)}+1)$, где $ISN_{B\to A(2)}$ — новый начальный порядковый номер, $ISN_{B\to A(1)} \neq ISN_{B\to A(2)}$.
4. Узел А не воспринимает этот сегмент из-за несовпадения порядковых номеров, но злоумышленник от имени А посылает в В сегмент $SN_{A\to B} = ISN_{A\to B(X)}+1$, $ACK_{A\to B}(ISN_{B\to A(2)}+1)$.

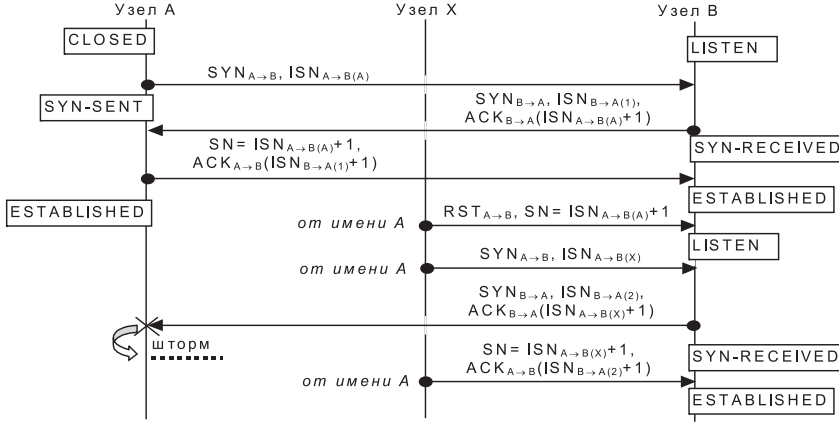

Рис. 7. Ранняя десинхронизация TCP-соединения (сегменты ACK-шторма не показаны)

После этого оба узла A и B находятся в состоянии ESTABLISHED, но соединение десинхронизировано:

	Следующий $SN_{A\to B}$	**Следующий $SN_{B\to A}$**
С точки зрения A	$ISN_{A\to B(A)}$+1 (A будет отправлять)	$ISN_{B\to A(1)}$+1 (A ожидает получить)
С точки зрения B	$ISN_{A\to B(X)}$+1 (B ожидает получить)	$ISN_{B\to A(2)}$+1 (B будет отправлять)

Отметим, что некоторые реализации TCP в нарушение стандарта в ответ на получение RST-сегмента сами отравляют RST-сегмент. В этом случае десинхронизация описанным способом невозможна.

Б. Десинхронизация нулевыми данными (рис. 8): злоумышленник, дожидаясь момента, когда соединение находится в неактивном состоянии (данные не передаются), посылает узлу A от имени B и узлу B от имени A фальсифицированные сегменты с данными, вызывая тем самым десинхронизацию. Посылаемые данные должны быть «нулевыми» — то есть приложение-получатель должно их молча игнорировать и не посылать никаких данных в ответ. Этот метод десинхронизации подходит для Telnet-соединений, которые, во-первых, часто находятся в неактивном состоянии, а во-вторых, в протоколе Telnet имеется команда «нет операции». Сегмент, содержащий произвольное число таких команд, будет принят приложением и полностью проигнорирован.

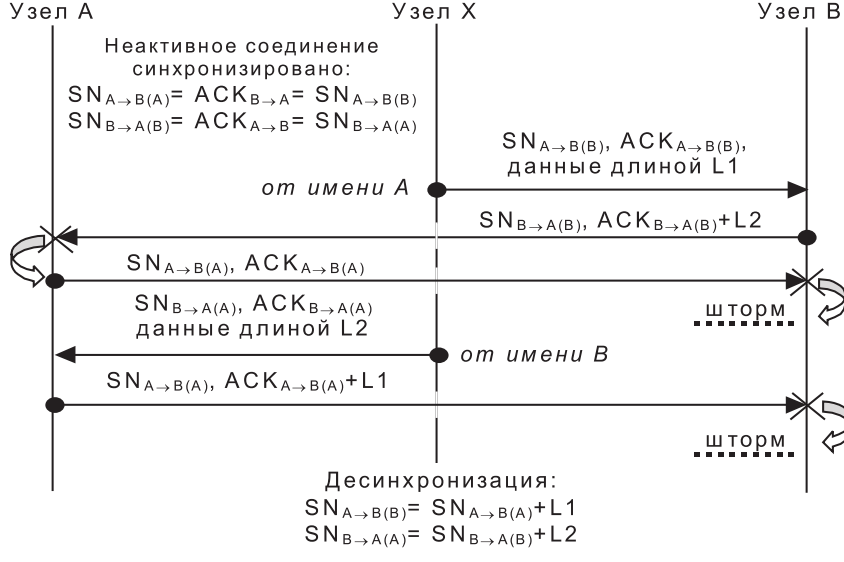

Рис. 8. Десинхронизация TCP-соединения нулевыми данными (сегменты ACK-шторма не показаны)

После описанной процедуры имеет место следующая ситуация:

	Следующий $SN_{A\to B}$	**Следующий $SN_{B\to A}$**
С точки зрения A	$SN_{A\to B(A)}$ (A будет отправлять)	$SN_{B\to A(B)}$+L2 (A ожидает получить)
С точки зрения B	$SN_{A\to B(A)}$+L1 (B ожидает получить)	$ISN_{B\to A(B)}$ (B будет отправлять)

Имперсонация с помощью десинхронизации является сравнительно простой и очень эффективной атакой. Она позволяет злоумышленнику установить полный контроль над TCP-соединением без использования ложных сообщений ARP, ICMP или протоколов маршрутизации, без атак типа «отказ в обслуживании», которые могут быть обнаружены администратором сети или атакуемого узла. Обнаружить такие атаки можно, прослушивая сеть на предмет ACK-штормов.

Для защиты от описанных атак маршрутизатор (шлюз, брандмауэр), соединяющий сеть с внешней сетью должен быть настроен на запрет пропуска пакетов:

а) приходящих на внешний интерфейс, но имеющих адрес отправителя из внутренней сети;

б) приходящих на внутренний интерфейс, но имеющих адрес отправителя из внешней сети.

Случай (а) соответствует ситуации, когда узлы А и В находятся во внутренней сети, а злоумышленник расположен снаружи и пытается послать узлу А дейтаграмму якобы от узла В. Случай (б) соответствует ситуации, когда злоумышленник находится во внутренней сети, а узлы А и В — снаружи. Подчеркнем, что предложенные меры не защитят от всех разновидностей имперсонации: например, когда узел Х находится в одной сети с узлом А или В или, естественно, когда все три узла расположены в одной сети.

Хороший алгоритм генерации случайных номеров ISN защитит от атаки в случае отсутствия обратной связи, но бесполезен, если злоумышленник может видеть сегменты, передаваемые из А в В.

В общем случае только шифрование данных или аутентификация сегментов могут гарантировать защиту от имперсонации.

6. Несанкционированное подключение к сети

Злоумышленник для несанкционированного подключения к сети должен иметь физическую возможность такого подключения. Следующим шагом для злоумышленника является конфигурирование параметров стека TCP/IP его компьютера.

Прослушивание сети (сегмента сети) даст злоумышленнику много полезной информации. В частности, он может определить, какие IP-адреса имеют узлы сети, и с помощью ICMP «Echo»-запросов (программа *ping*) определить, какие адреса не используются (или компьютеры выключены). После этого злоумышленник может присвоить себе неиспользуемый адрес.

Найти IP-адрес маршрутизатора по умолчанию можно, подслушав кадры с дейтаграммами, направленными на IP-адреса, не принадлежащие сети. Эти кадры направлены на MAC-адрес маршрутизатора. Очевидно, что узлы сети время от времени генерируют ARP-запросы о MAC-адресе маршрутизатора; ответы на эти запросы, посылаемые маршрутизатором, содержат как его MAC-адрес, так и IP-адрес. Зная MAC-адрес маршрутизатора и подслушав такие ответы, злоумышленник определит искомый IP-адрес.

Для обнаружения маршрутизатора злоумышленник может использовать также сообщения ICMP «Router Advertisement/Solicitation».

Для определения маски сети злоумышленник может послать сообщение ICMP «Address Mask Request». В ответ маршрутизатор должен выслать маску сети в сообщении «Address Mask Reply».

Если маршрутизатор не поддерживает сообщения «Address Mask Request/Reply», злоумышленник может применить следующий простой метод. Путем арифметических вычислений он определяет минимальную сеть, включающую его собственный адрес и найденный адрес маршрутизатора, и назначает себе маску этой сети. Например, пусть адрес, присвоенный злоумышленником, X = 10.0.0.57, а адрес маршрутизатора G = 10.0.0.1, то есть, расписывая последний октет в двоичном виде:

$$X = 10.0.0.\mathit{00111001},$$

$$G = 10.0.0.\mathit{00000001}.$$

Максимальная общая часть обоих адресов (то есть, искомая минимальная сеть, включающая оба адреса):

$$N = 10.0.0.\mathit{00XXXXXX}.$$

Значит, N = 10.0.0.0/26, а маска — 255.255.255.192.

Все дейтаграммы, направленные за пределы этой минимальной сети будут переданы маршрутизатору. Если маска определена неправильно и на самом деле злоумышленник находится в сети, например 10.0.0.0/16, и посылает дейтаграмму узлу 10.0.1.1, маршрутизатор примет эту дейтаграмму от злоумышленника и просто передаст ее узлу назначения в этой же самой сети.

Конечно, существует вероятность, что злоумышленник неправильно определит IP-адрес для присвоения и окажется за пределами сети. Кроме того, возможная конфигурация нескольких IP-сетей в одном сегменте Ethernet усложнит задачу злоумышленника. Однако после периода проб и ошибок злоумышленник имеет все шансы определить необходимые параметры для конфигурирования своего хоста.

Отметим, что если в сети имеется сервер DHCP, который предоставляет IP-адреса всем желающим, то он полностью сконфигурирует узел злоумышленника без всяких усилий со стороны последнего. Это событие будет зарегистрировано в журнале сервера.

Для предотвращения несанкционированного подключения к сети администратор должен использовать статическую ARP-таблицу на маршрутизаторе (и ключевых хостах-серверах) и программу *arpwatch*. Статическая *arp*-таблица не позволит злоумышленнику получить ни одну дейтаграмму от узла, который ее использует, поскольку MAC-адрес зло-

умышленника, естественно, не значится в таблице. Программа *arpwatch* уведомит администратора о появлении узла с неизвестным MAC-адресом.

Однако, если злоумышленник, определив IP- и MAC-адреса какого-либо компьютера в своей сети, дождется его выключения (или проведет против него атаку «отказ в обслуживании», приводящую к неспособности атакуемого хоста работать в сети), а потом присвоит себе его MAC- и IP-адреса, то обнаружить такого злоумышленника будет невозможно и все его действия будут приписаны атакованному хосту.

7. Несанкционированный обмен данными

С целью обеспечения безопасности внутренней (корпоративной) сети на шлюзе могут использоваться фильтры, препятствующие прохождению определенных типов дейтаграмм. Дейтаграммы могут фильтроваться по IP-адресам отправителя или получателя, по протоколу (поле «Protocol» IP-дейтаграммы), по номеру порта TCP или UDP, по другим параметрам, а также по комбинации этих параметров.

Рассмотрим два приема, которые может использовать злоумышленник для проникновения через некоторые фильтры.

Туннелирование

Предположим, злоумышленник хочет отправить данные с узла X узлу A, находящемуся за пределами его сети, однако правила фильтрации на маршрутизаторе запрещают отправку дейтаграмм узлу A (рис. 9). В то же время разрешена отправка дейтаграмм узлу B, также находящемуся за пределами защищаемой сети.

Злоумышленник использует узел B как ретранслятор дейтаграмм, направленных в A. Для этого он создает дейтаграмму, направленную из X в B, в поле «Protocol» которой помещается значение 4 («IP»), а в качестве данных эта дейтаграмма несет другую IP-дейтаграмму, направленную из X в A. Фильтрующий маршрутизатор пропускает сформированную дейтаграмму, поскольку она адресована разрешенному узлу B, а IP-модуль узла B извлекает из нее вложенную дейтаграмму. Видя, что вложенная дейтаграмма адресована не ему, узел B отправляет ее по назначению, то есть узлу A (рис. 9).

Описанная операция называется туннелированием. Для ее осуществления злоумышленник может иметь, а может и не иметь сговора с узлом B — в зависимости от того, что представляет собой этот узел и как он сконфигурирован.

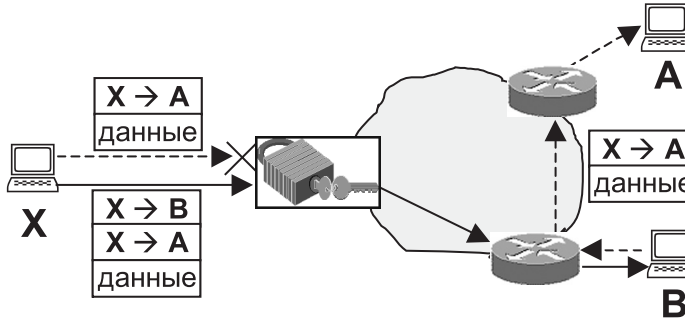

Рис. 9. Туннелирование сквозь фильтрующий маршрутизатор

Того же самого результата можно добиться, применив IP-опцию «Loose/Strict Source Routing».

Отметим, что адрес отправителя дейтаграммы скрыть нельзя, поэтому если маршрутизатор не пропускает также дейтаграммы, идущие *из* A, то есть осуществляет фильтрацию по адресу отравителя, то обмануть его вышеописанным способом невозможно. В этом случае злоумышленник имеет только одностороннюю связь с узлом A.

Очевидно, что туннелирование может использоваться и в обратном направлении, то есть для проникновения из Интернет внутрь охраняемой сети. При этом узел X находится в Интернет, а узлы A и B — в охраняемой сети и узлу B разрешено получение дейтаграмм из внешних сетей.

Для защиты от туннелирования следует запретить маршрутизатору транслировать во внешнюю сеть дейтаграммы с полем «Protocol» = 4 и дейтаграммы с опциями. Напомним, что туннелирование используется для подключения сетей, поддерживающих групповую рассылку, к сети MBONE через сети, не поддерживающие групповую маршрутизацию. В этом случае все маршрутизаторы защищаемой (внутренней) системы сетей должны поддерживать групповую маршрутизацию, а инкапсуляция и извлечение групповых дейтаграмм должны выполняться непосредственно на фильтрующем маршрутизаторе.

Tiny Fragment Attack

В случае, когда на вход фильтрующего маршрутизатора поступает фрагментированная дейтаграмма, маршрутизатор производит досмотр только первого фрагмента дейтаграммы (первый фрагмент определяется по значению поля IP-заголовка «Fragment Offset» = 0). Если пер-

вый фрагмент не удовлетворяет условиям пропуска, он уничтожается. Остальные фрагменты можно безболезненно пропустить, не затрачивая на них вычислительные ресурсы фильтра, поскольку без первого фрагмента дейтаграмма все равно не может быть собрана на узле назначения.

При конфигурировании фильтра перед сетевым администратором часто стоит задача: разрешить соединения с TCP-сервисами, инициируемые компьютерами внутренней сети, но запретить установление соединений внутренних компьютеров с внешними по инициативе последних. Для решения поставленной задачи фильтр конфигурируется на запрет пропуска TCP-сегментов, поступающих из внешней сети и имеющих установленный бит SYN; сегменты без этого бита беспрепятственно пропускаются в охраняемую сеть, поскольку они могут относиться к соединению, уже установленному ранее по инициативе внутреннего компьютера.

Рассмотрим, как злоумышленник может использовать фрагментацию, чтобы обойти это ограничение, то есть, передать сегмент с битом SYN из внешней сети во внутреннюю.

Злоумышленник формирует искусственно фрагментированную дейтаграмму с TCP-сегментом, при этом первый фрагмент дейтаграммы имеет минимальный размер поля данных — 8 октетов (напомним, что размеры фрагментов указываются в 8-октетных блоках). В поле данных дейтаграммы находится TCP-сегмент, начинающийся с TCP-заголовка. В первых 8 октетах TCP-заголовка находятся номера портов отправителя и получателя и поле «Sequence Number», но значения флагов не попадут в первый фрагмент. Следовательно, фильтр пропустит первый фрагмент дейтаграммы, а остальные фрагменты он проверять не будет. Таким образом, дейтаграмма с SYN-сегментом будет успешно доставлена на узел назначения и после сборки передана модулю TCP.

Пример дейтаграммы из 2 фрагментов. Фрагмент 1 (IP-заголовок выделен серым, в поле данных — 8 октетов TCP-заголовка):

IP-заголовок: MF = 1, Fragment Offset = 0							
	Source	Port			Destination	Port	
			Sequence	Number (SN)			

Фрагмент 2 (в поле данных — остальная часть TCP-заголовка с установленным флагом SYN):

IP-заголовок: MF = 0, Fragment Offset = 1																				
			Acknowledgment							Sequence Number (ACK SN)=0										
Data Offset			reserved			–	–	–	–	**SYN**	–		Window							
				Checksum									Urgent		Pointer = 0					
					Options													Padding		

Описанный выше прием проникновения сквозь фильтр называется «Tiny Fragment Attack» (RFC-1858). Использование его в других случаях (для обхода других условий фильтрации) не имеет смысла, так как все остальные «интересные» поля в заголовке TCP и других протоколов находятся в первых 8 октетах заголовка и, следовательно, не могут быть перемещены во второй фрагмент.

Для защиты от этой атаки фильтрующему маршрутизатору, естественно, не следует инспектировать содержимое не первых фрагментов дейтаграмм — это было бы равносильно сборке дейтаграмм на промежуточном узле, что быстро поглотит все вычислительные ресурсы маршрутизатора. Достаточно реализовать один из двух следующих подходов:

1) не пропускать дейтаграммы с «Fragment Offset» = 0 и «Protocol» = 6 (TCP), размер поля данных которых меньше определенной величины, достаточной, чтобы вместить все «интересные поля» (например, 20); или
2) не пропускать дейтаграммы с «Fragment Offset» = 1 и «Protocol» = 6 (TCP): наличие такой дейтаграммы означает, что TCP-сегмент был фрагментирован с целью скрыть определенные поля заголовка и что где-то существует первый фрагмент с 8 октетами данных. Несмотря на то что в данном случае первый фрагмент будет пропущен, узел назначения не сможет собрать дейтаграмму, так как фильтр уничтожил второй фрагмент.

Второй аспект фрагментации, интересный с точки зрения безопасности, — накладывающиеся (*overlapping*) фрагменты. Рассмотрим пример дейтаграммы, несущей TCP-сегмент и состоящей из двух фрагментов. Первый фрагмент (IP-заголовок выделен серым цветом, в поле данных — полный TCP-заголовок, без опций, дополненный нулями до размера, кратного 8 октетам):

IP-заголовок: MF = 1, Fragment Offset = 0											
	Source	Port			Destination	Port					
			Sequence	Number (SN)							
			Acknowledgment	Sequence Number (ACK SN)							
Data Offset	reserved	–	ACK	–	–	–	–		Window		
			Checksum			Urgent	Pointer = 0				
			0								

Фрагмент 2 (в поле данных — часть другого TCP-заголовка, начиная с девятого по порядку октета, в котором установлен флаг SYN):

IP-заголовок: MF = 0, Fragment Offset = 1											
			Acknowledgment	Sequence Number (ACK SN) = 0							
Data Offset	reserved	–	–	–	–	**SYN**	–		Window		
			Checksum			Urgent	Pointer = 0				
		Options					Padding				

Видно, что второй фрагмент накладывается на первый (первый фрагмент содержит октеты 0–23 данных исходной дейтагаммы, а второй фрагмент начинается с октета 8, потому что его «Fragment Offset» = 1). Поведение узла назначения, получившего такую дейтграмму зависит от реализации модуля IP. Часто при сборке дейтаграммы данные второго, накладывающегося фрагмента записываются поверх предыдущего фрагмента. Таким образом, при сборке приведенной в примере дейтаграммы в TCP-заголовке переписываются поля, начиная с «ACK SN», в соответствии со значениями из второго фрагмента, и в итоге получается SYN-сегмент.

Если для защиты от Tiny Fragment Attack применяется подход (1) из описанных выше (инспекция первого фрагмента дейтаграммы), то с помощью накладывающихся фрагментов злоумышленник может обойти эту защиту.

Маршрутизатор, применяющий второй подход, будет успешно противостоять Tiny Fragment Attack с накладывающимися фрагментами.

8. Принуждение к ускоренной передаче данных

Механизм реагирования на заторы сети (*congestion control*), реализуемый протоколом TCP, позволяет злоумышленнику — получателю данных принудить отправителя высылать данные с многократно увеличенной скоростью. В результате злоумышленник отбирает для своих нужд ресурсы сервера-отправителя и компьютерной сети, замедляя или блокируя соединения прочих участников сетевого взаимодействия.

Атаки выполняются путем специально организованной посылки злоумышленником подтверждений приема данных (ACK-сегментов). Все описываемые в этом пункте атаки эксплуатируют следующее неявное допущение, заложенное в протокол TCP: один участник TCP-соединения полностью доверяет другому участнику в том, что тот действует в строгом соответствии с теми же спецификациями протокола, что и первый.

В следующих подпунктах мы будем считать, что узел A отправляет данные узлу B, а последний пытается принудить узел A передавать данные на завышенной скорости.

Расщепление подтверждений

Пусть узел A, после установления соединения, находится в режиме медленного старта. Окно перегрузки *cwnd* равно одному полноразмерному сегменту, который и высылается в B (рис. 10). Однако B, вместо того чтобы ответить одним подтверждением о получении всего сегмента (ACK SN = 1001), высылает несколько подтверждений с возрастающими номерами ACK SN (например, 300, 600 и, наконец, 1001), как бы подтверждая получение сегмента по частям.

Отметим, что стандарт TCP при этом не нарушается, потому что в TCP подтверждается получение октетов, а не сегментов. Однако алгоритм медленного старта неявно подразумевает, что получатель посылает не более одного подтверждения при получении каждого очередного сегмента, и, как следствие, окно *cwnd* увеличивается на один *полноразмерный* сегмент с приходом каждого подтверждения, независимо от того, получение какого числа октетов оно подтверждает.

В итоге, если при нормальном поведении узла B, отправитель на следующем шаге увеличил бы окно *cwnd* только на 1 сегмент и отправил бы в B два следующих полноразмерных сегмента (SN=1001 и 2001). Но узел B, выслав три подтверждения вместо одного, вынудил узел A увеличить значение cwnd до 4 и отправить 4 сегмента вместо двух.

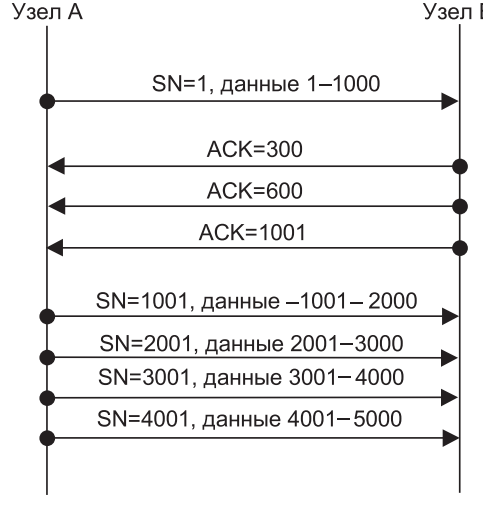

Рис. 10. Расщепление подтверждений

В общем случае, если полноразмерный сегмент содержит N октетов, то узел B может отправить $M \le N$ подтверждений. Простые арифметические подсчеты показывают, что тогда на i-м шаге (то есть через время $RTT \times i$, где RTT — время обращения) узел A будет отправлять не $N \times 2^i$ октетов, как это было бы при нормальном поведении получателя, а порядка $N \times M^i$ октетов, если, конечно, узел B позаботится об объявлении подобающего размера окна получателя. Типичный размер поля данных сегмента — 1460 октетов. Наиболее агрессивное поведение получателя (1460 подтверждений на каждый сегмент) теоретически приведет к тому, что уже после третьего шага узел B может получить 2,9 Гбайт данных. Конечно, скорость не может расти бесконечно — она будет ограничена возможностями сети и узла-отправителя.

Ложные дубликаты подтверждений

Опять рассмотрим ситуацию, когда узел A находится в медленном старте после установления соединения. A отправляет в B один сегмент (SN = 1–1000), но узел B, вместо того чтобы ответить подтверждением ACK SN = 1001, отвечает серией сфабрикованных подтверждений ACK SN = 1.

Получение дубликатов подтверждений включает на узле A алгоритмы быстрой повторной передачи (fast retransmit) и быстрого восстановления (fast recovery). Последний из них представляет в данном

случае особый интерес. Напомним, что алгоритм быстрого восстановления базируется на предположении, что каждый полученный дубликат предыдущего подтверждения, кроме того, что говорит о потере сегмента, также подразумевает, что какой-то другой *полноразмерный* сегмент покинул сеть. Вследствие этого окно *cwnd*, измеряемое в полноразмерных сегментах, временно увеличивается на число полученных дубликатов, пока не придет подтверждение, отличное от предыдущих.

Поскольку сам дубликат подтверждения не несет никакой информации о том, получение какого именно сегмента вызвало отправку данного дубликата, ничто не может уберечь отправителя от ложных дубликатов подтверждений, сгенерированных получателем. Таким образом, получатель заставляет отправителя необоснованно увеличить окно *cwnd* и ускорить отправку данных.

Фактически узел А будет отправлять данные со скоростью, с которой В генерирует дубликаты подтверждений. В какой-то момент в узле А сработает таймер повторной передачи для сегмента SN = 1–1000, поскольку он так и не был подтвержден. Узел В отреагирует на это посыл-

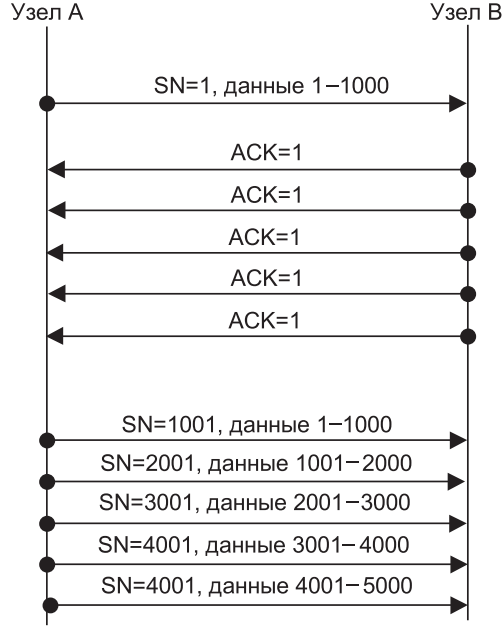

Рис. 11. Ложные дубликаты подтверждений

кой кумулятивного подтверждения на все полученные к этому моменту данные и переключится на генерацию ложных дубликатов этого последнего подтверждения, снова вынуждая отправителя необоснованно увеличивать *cwnd*.

Преждевременные подтверждения

Еще одна разновидность атаки строится на том, что получатель может заранее высылать подтверждения еще не принятых им, находящихся в пути сегментов, заставляя отправителя поверить, что данные уже доставлены, результатом чего будет увеличение *cwnd* и преждевременная отправка новых данных.

Отметим, что хотя получатель и может предсказать границы сегментов отправителя (при передаче большого объема данных отправитель, как правило, помещает данные в полноразмерные сегменты, имеющие одинаковый объем) и соответственно формировать преждевременные подтверждения, большая точность при этом не требуется. Подтверждение любого блока данных, как мы уже замечали выше, приводит к увеличению окна *cwnd* и отправке новых полноразмерных сегментов.

Более того, если получатель «перестарается» и подтвердит то, что еще не выслано, подтверждение будет просто проигнорировано отправителем и не приведет ни к каким неприятным последствиям.

В отличие от двух предыдущих атак атака преждевременными подтверждениями разрушает механизм обеспечения надежности передачи данных: если какой-либо из сегментов с данными, отправленный из A в B, потеряется в пути, повторной передачи этого сегмента не будет, поскольку он был уже заранее подтвержден получателем. Однако прикладные протоколы HTTP и FTP, с помощью которых и передаются большинство данных в сетях TCP/IP, предоставляют возможность запрашивать у сервера не весь файл, а его определенные части (большинство серверов поддерживают эту возможность). Поэтому, применив описанную атаку и получив основной объем данных с HTTP- или FTP-сервера на завышенной скорости, злоумышленник может впоследствии с помощью запросов на частичную передачу «залатать дыры», образовавшиеся из-за потерянных сегментов.

Описанные атаки особенно эффективны при передаче сравнительно небольших объемов данных (файлов), когда весь файл может быть передан взрывным образом за одно время обращения. Эксперименты показали, что скорость загрузки файла увеличивается в несколько раз. Работа конкурирующих TCP-соединений (имеющихся в том же коммуникационном канале) практически блокируется, поскольку из-за резко возросшей

интенсивности трафика другие соединения диагностируют состояние затора и принимают соответствующие меры по уменьшению скорости передачи данных, фактически освобождая канал для злоумышленника.

Для реализации описанных атак требуется сравнительно небольшая (несколько десятков строк) модификация модуля TCP на компьютере B. Для защиты от расщепления подтверждений достаточно доработать реализацию модуля TCP: разрешить увеличивать *cwnd* только после получения подтверждения, охватывающего *целый* сегмент. Адекватной защиты от двух других атак не существует. Чтобы решить эту проблему, авторам протокола TCP необходимо было внести в заголовок TCP дополнительное поле *cumulative nonce*, которое играло бы роль идентификатора сегмента; используя это поле, легитимные подтверждения должны были бы ссылаться на сегмент (сегменты), получение которых вызвало отправку данного подтверждения. Это предотвратило бы отправку ложных дубликатов и подтверждений еще не полученных сегментов. Однако маловероятно, чтобы дизайн протокола TCP был изменен.

9. Отказ в обслуживании

Атаки «отказ в обслуживании» (*DoS, denial of service*), по-видимому, являются наиболее распространенными и простыми в исполнении. Целью атаки является приведение атакуемого узла или сети в такое состояние, когда передача данных другому узлу (или передача данных вообще) становится невозможна или крайне затруднена. Вследствие этого пользователи сетевых приложений, работающих на атакуемом узле, не могут быть обслужены — отсюда название этого типа атак. Атаки DoS используются как в комплексе с другими (имперсонация), так и сами по себе.

DoS-атаки можно условно поделить на три группы:

- атаки большим числом формально корректных, но, возможно, сфальсифицированных пакетов, направленные на истощение ресурсов узла или сети;
- атаки специально сконструированными пакетами, вызывающие общий сбой системы из-за ошибок в программах;
- атаки сфальсифицированными пакетами, вызывающими изменения в конфигурации или состоянии системы, что приводит к невозможности передачи данных, сбросу соединения или резкому снижению его эффективности.

Истощение ресурсов узла или сети

Smurf. Атака состоит в генерации шквала ICMP «Echo»-ответов, направленных на атакуемый узел. Для создания шквала злоумышленник направляет несколько сфальсифицированных «Echo»-запросов от имени жертвы на широковещательные адреса нескольких сетей, которые выступят в роли усилителей. Потенциально большое число узлов, находящихся в сетях-усилителях и поддерживающих обработку широковещательных «Echo»-запросов, одновременно отправляет ответы на атакуемый узел. В результате атаки сеть, в которой находится жертва, сам атакуемый узел, а также и сети-усилители могут быть временно заблокированы шквалом ответных сообщений. Более того, если атакуемая организация оплачивает услуги провайдера Интернет пропорционально полученному трафику, ее расходы могут существенно возрасти.

Для атакуемого узла и его сети не существует адекватных способов защиты от этой атаки. Очевидно, что блокирование ICMP-сообщений маршрутизатором на входе в атакуемую сеть не является удовлетворительным решением проблемы, поскольку при этом канал, соединяющий организацию с провайдером Интернет, остается подверженным атаке, а именно он, как правило, является наиболее узким местом при работе организации с Интернет. И, поскольку ICMP-сообщения были доставлены провайдером на маршрутизатор организации, они подлежат оплате.

Атаку *smurf* можно обнаружить путем анализа трафика на маршрутизаторе или в сети. Признаком атаки является так же полная загрузка внешнего канала и сбои в работе хостов внутри сети. При обнаружении атаки следует определить адреса отправителей сообщений «Echo Reply» (это сети-усилители), установить в регистратуре Интернет их административную принадлежность и обратиться к администраторам с просьбой принять меры защиты для усилителей, описываемые ниже. Администратор атакуемой сети также должен обратиться к своему провайдеру с извещением об атаке; провайдер может заблокировать передачу сообщений «Echo Reply» в канал атакуемой организации.

Для устранения атак *smurf* защитные меры могут быть предприняты как потенциальными усилителями, так и администраторами сетей, в которых может находиться злоумышленник. Это:

- запрет на маршрутизацию дейтаграмм с широковещательным адресом назначения между сетью организации и Интернет;

- запрет на обработку узлами «Echo»-запросов, направленных на широковещательный адрес;
- запрет на маршрутизацию дейтаграмм, направленных из внутренней сети (сети организации) в Интернет, но имеющих внешний адрес отправителя.

В этой связи отметим, что каждая сеть может оказаться в любой из трех ролей: сети злоумышленника, усилителя или жертвы, поэтому, принимая меры по защите других сетей, вы можете надеяться, что администраторы других сетей достаточно квалифицированы и принимают те же самые меры, которые могут защитить вас.

SYN flood (Neptune) и Naptha. Распространенная атака SYN flood состоит в посылке злоумышленником SYN-сегментов TCP на атакуемый узел в количестве большем, чем тот может обработать одновременно (это число не велико — обычно несколько десятков).

При получении каждого SYN-сегмента модуль TCP создает блок TCB, то есть, выделяет определенные ресурсы для обслуживания будущего соединения, и отправляет свой SYN-сегмент. Ответа на него он никогда не получит. (Чтобы замести следы и не затруднять себя игнорированием ответных SYN-сегментов, злоумышленник будет посылать свои SYN-сегменты от имени несуществующего отправителя или нескольких случайно выбранных несуществующих отправителей.) Через несколько минут модуль TCP ликвидирует так и не открытое соединение, но если одновременно злоумышленник сгенерирует большое число SYN-сегментов, то он заполнит все ресурсы, выделенные для обслуживания открываемых соединений, и модуль TCP не сможет обрабатывать новые SYN-сегменты, пока не освободится от запросов злоумышленника. Постоянно посылая новые запросы, злоумышленник может продолжительно удерживать жертву в блокированном состоянии. Чтобы снять воздействие атаки, злоумышленник посылает серию сегментов с флагом RST, которые ликвидируют полуоткрытые соединения и освобождают ресурсы атакуемого узла.

Целью атаки является приведение узла (сервера) в состояние, когда он не может принимать запросы на открытие соединений. Однако некоторые недостаточно хорошо спроектированные системы в результате атаки не только перестают открывать новые соединения, но и не могут поддерживать уже установленные, а в худшем случае — зависают.

Атаку SYN flood можно определить как удерживание большого числа соединений на атакуемом узле в состоянии SYN-RECEIVED.

В последнее время было показано, что большое число соединений в состояниях ESTABLISHED и FIN-WAIT-1 так же вызывает отказы в обслуживании на сервере. Эти отказы выражаются по-разному в различных системах: переполнение таблиц процессов («*process table attack*»)и файловых дескрипторов, невозможность открытия новых и обрыв установленных соединений, зависание серверных процессов или всей системы.

Подобные уязвимости стеков TCP/IP получили собирательное название Naptha. Подчеркнем, что выполнение атаки Naptha не требует от злоумышленника тех же затрат по поддержанию TCP-соединений, что и от атакуемого узла. Злоумышленник не создает блоков TCB, не отслеживает состояния соединений и не запускает прикладных процессов; при получении TCP-сегмента от атакуемого узла злоумышленник просто генерирует приемлемый ответ, основываясь на флагах и значениях полей заголовка принятого сегмента, чтобы перевести или удержать TCP-соединение на атакуемом узле в нужном состоянии. Разумеется, злоумышленник, чтобы скрыть себя, будет посылать сегменты от имени несуществующего (выключенного) узла и принимать ответные сегменты от атакуемого узла методом прослушивания.

Полной защиты от описанных атак не существует. Чтобы уменьшить подверженность узла атаке администратор должен использовать программное обеспечение, позволяющее установить максимальное число открываемых соединений, а также список разрешенных клиентов (если это применимо). Только необходимые порты должны быть открыты (находиться в состоянии LISTEN), остальные сервисы следует отключить. Операционная система должна иметь устойчивость к атакам Naptha — при проведении атаки не должно возникать отказа в обслуживании пользователей и сервисов, не имеющих отношения к атакуемым.

Должен также проводиться анализ трафика в сети для выявления начавшейся атаки, признаком чего является большое число однотипных сегментов, и блокирование сегментов злоумышленника фильтром на маршрутизаторе. Маршрутизаторы Cisco предлагают механизм TCP Intercept, который служит посредником между внешним TCP-клиентом и TCP-сервером, находящимся в защищаемой сети. При получении SYN-сегмента из Интернет маршрутизатор не ретранслирует его во внутреннюю сеть, а сам отвечает на этот сегмент от имени сервера. Если соединение с клиентом устанавливается, то маршрутизатор также

устанавливает соединение с сервером от имени клиента и при дальнейшей передаче сегментов в этом соединении действует как прозрачный посредник, о котором ни клиент, ни сервер не подозревают. Если ответ от клиента за определенное время так и не поступил, то оригинальный SYN-сегмент не будет передан отправителю.

Если SYN-сегменты начинают поступать в большом количестве и на большой скорости, то маршрутизатор переходит в «агрессивный» режим: время ожидания ответа от клиента резко сокращается, а каждый вновь прибывающий SYN-сегмент приводит к исключению из очереди одного из ранее полученных SYN-сегментов. При снижении интенсивности потока запросов на соединения, маршрутизатор возвращается в обычный, «дежурный» режим.

Отметим, что применение TCP Intercept фактически переносит нагрузку по борьбе с SYN-атакой с атакуемого хоста на маршрутизатор, который просто лучше подготовлен для этого.

UDP flood. Атака состоит в затоплении атакуемой сети шквалом UDP-сообщений. Для генерации шквала злоумышленник использует сервисы UDP, посылающие сообщение в ответ на любое сообщение. Примеры таких сервисов: echo (порт 7) и chargen (порт 19). От имени узла A (порт отправителя — 7) злоумышленник посылает сообщение узлу B (порт получателя — 19). Узел B отвечает сообщением на порт 7 узла A, который возвращает сообщение на порт 19 узла B, и так далее до бесконечности (на самом деле, конечно, до тех пор, пока сообщение не потеряется в сети). Интенсивный UDP-трафик затрудняет работу узлов A и B и может создать затор в сети.

UDP-сервис echo может быть также использован для выполнения атаки *fraggle*. Эта атака полностью аналогична *smurf*, однако менее популярна у злоумышленников из-за меньшей эффективности.

Для защиты от атак типа UDP flood следует отключить на узлах сети все неиспользуемые сервисы UDP (отметим, что вам вряд ли когда-нибудь вообще понадобятся сервисы echo и chargen). Фильтр на маршрутизаторе-шлюзе должен блокировать все UDP-сообщения кроме тех, что следуют на разрешенные порты (например, порт 53 — DNS).

Отметим, что использование промежуточных систем для реализации атак, связанных с генерацией направленного шквала пакетов (типа *smurf*), оказалось весьма плодотворной идеей для злоумышленников. Такие атаки называются *распределенными* (*Distributed DoS, DDoS*). Для их реализации злоумышленниками создаются программы-демоны, захватывающие промежуточные системы и впоследствии координи-

рующие создание направленных на атакуемый узел шквалов пакетов (ICMP «Echo», UDP, TCP SYN). Захват систем производится путем использования ошибок в программах различных сетевых сервисов. Примеры таких распределенных систем — TFN, trin00.

Ложные DHCP-клиенты. Атака состоит в создании злоумышленником большого числа сфальсифицированных запросов от различных несуществующих DHCP-клиентов. Если DHCP-сервер выделяет адреса динамически из некоторого пула, то все адресные ресурсы могут быть истрачены на фиктивных клиентов, вследствие чего легитимные хосты не смогут сконфигурироваться и лишатся доступа в сеть.

Для защиты от этой атаки администратору следует поддерживать на DHCP-сервере таблицу соответствия MAC- и IP-адресов (если это возможно); сервер должен выдавать IP-адреса в соответствии с этой таблицей.

Сбой системы

DoS-атаки этой группы не связаны с какими-либо проблемами протоколов стека TCP/IP, а используют ошибки в их программной реализации. Эти ошибки сравнительно просто исправить. Мы дадим краткий обзор наиболее известных атак, имея в виду, что в скором времени они, видимо, будут представлять лишь исторический интерес. С другой стороны нет никаких гарантий того, что не будут обнаружены новые ошибки. Все описываемые ниже атаки приводят к общему сбою уязвимой системы. Для защиты от них необходимо использовать последние версии операционных систем и следить за обновлениями к ним.

Ping of death. Атака состоит в посылке на атакуемый узел фрагментированной дейтаграммы, размер которой после сборки превысит 65 535 октетов. Напомним, что длина поля «Fragment Offset» заголовка IP-дейтаграммы — 13 бит (то есть, максимальное значение равно 8192), а измеряются смещения фрагментов в восьмерках октетов. Если последний фрагмент сконструированной злоумышленником дейтаграммы имеет, например, смещение «Fragment Offset» = 8190, а длину — 100, то его последний октет находится в собираемой дейтаграмме на позиции 8190*8 + 100 = 65620 (плюс как минимум 20 октетов IP-заголовка), что превышает максимальный разрешенный размер дейтаграммы.

Teardrop. Атака использует ошибку, возникающую при подсчете длины фрагмента во время сборки дейтаграммы. Предположим, фраг-

мент А имеет смещение 40 («Fragment Offset» = 5) и длину 120, а фрагмент В — смещение 80 и длину 160, то есть фрагменты накладываются, что, в общем-то, разрешено. Модуль IP вычисляет длину той части фрагмента В, которая не накладывается на А: (80 + 160) - (40 + 120) = 80, и копирует последние 80 октетов фрагмента В в буфер сборки. Предположим, злоумышленник сконструировал фрагмент В так, что тот имеет смещение 80, а длину 72. Вычисление длины перекрытия дает отрицательный результат: (80+72) - (40+120) = -8, что с учетом представления отрицательных чисел в машинной арифметике интерпретируется как 65 528, и программа начинает записывать 65 528 байт в буфер сборки, переполняя его и затирая соседнюю область памяти.

Land. Атака состоит в посылке на атакуемый узел SYN-сегмента TCP, у которого IP-адрес и порт отправителя совпадают с адресом и портом получателя.

Nuke. Атака состоит в посылке на атакуемый TCP-порт срочных данных (задействовано поле Urgent Pointer). Атака поражала Windows-системы через порт 139, где в прикладном процессе не была предусмотрена возможность приема срочных данных, что приводило к краху системы.

Изменение конфигурации или состояния узла

Перенаправление трафика на несуществующий узел. Ранее были рассмотрены методы различные методы перехвата трафика злоумышленником. Любой из этих методов может быть использован также и для перенаправления посылаемых атакуемым узлом (или целой сетью) данных «в никуда», результатом чего будет потеря коннективности жертвы с выбранными злоумышленником узлами или сетями.

В дополнение отметим, что при использовании в сети протокола маршрутизации злоумышленник может генерировать сфальсифицированные сообщения протокола, содержащие некорректные или предельные значения некоторых полей (порядковых номеров, возраста записи, метрики), что приведет к нарушениям в системе маршрутизации).

Навязывание длинной сетевой маски. Если хост получает маску для своей сети через ICMP-сообщение «Address Mask Reply», то, сформировав ложное сообщение с длинной маской (например, 255.255.255.252), злоумышленник существенно ограничит коннектив-

ность атакуемого хоста. Если в «суженной» злоумышленником сети не окажется маршрутизатора по умолчанию, то жертва сможет посылать данные только узлам, попадающим под навязанную маску.

В настоящее время хосты динамически конфигурируются с помощью протокола DHCP (что, впрочем, открывает перед злоумышленником более широкие возможности: выдав себя за DHCP-сервер, злоумышленник может полностью исказить настройки стека TCP/IP атакуемого хоста). Тем не менее следует проверить, как система отреагирует на получение ICMP «Address Mask Reply».

Десинхронизация TCP-соединения была рассмотрена ранее. Очевидно, что если после выполнения десинхронизации злоумышленник не будет функционировать в качестве посредника, то передача данных по атакованному TCP-соединению будет невозможна.

Сброс TCP-соединения. Если узел A установил соединение с узлом B, то злоумышленник может принудить узел A преждевременно закрыть это соединение. Для этого злоумышленник может послать узлу A от имени B сфальсифицированный RST-сегмент или ICMP-сообщение «Destination Unreachable».

Для формирования приемлемого для узла A RST-сегмента злоумышленник должен подслушивать соединение для того, чтобы поместить в RST-сегмент номер SN, находящийся в рамках доступного окна узла B, иначе узел A сфабрикованный сегмент проигнорирует.

Реакция TCP-модуля узла A на получение ICMP-сообщений «Destination Unreachable» четко не определена стандартами, хотя документ RFC-1122 утверждает, что уровень IP должен передать соответствующее сообщение модулю TCP, а тот в свою очередь обязан на него както прореагировать. Также RFC-1122 говорит, что при получении такого сообщения с кодом 2 («Protocol Unreachable») или 3 («Port Unreachable») модулю TCP следует ликвидировать соединение, к которому это сообщение относится.

Принуждение к передаче данных на заниженной скорости. Злоумышленник может вынудить модуль TCP узла A снизить скорость передачи данных узлу B в TCP-соединении следующими способами.

- Отправка узлу A от имени промежуточного маршрутизатора ложного сообщения ICMP «Source Quench». RFC-1122 утверждает, что уровень IP должен проинформировать модуль TCP о получении

этого сообщения, а последний обязан отреагировать уменьшением скорости отправки данных, причем рекомендуется, чтобы модуль TCP перешел в режим медленного старта как после срабатывания таймера повторной передачи.

- Отправка узлу A от имени промежуточного маршрутизатора ложных сообщений ICMP «Destination Unreachable: Datagram Too Big». При использовании алгоритма Path MTU Discovery модуль TCP, получая такие сообщения, будет уменьшать размер отправляемых сегментов до числа, указанного в ICMP-сообщении, или пока не достигнет установленного минимума. Если строго следовать стандарту, то минимальный размер дейтаграммы, которую любой модуль IP обязан передать без фрагментации — 68 октетов. В этом случае, если в IP- и TCP-заголовках не используется опций, полезная нагрузка составит 28 октетов на сегмент. Кроме того, при получении сообщения «Datagram Too Big» включается режим медленного старта, хотя и без уменьшения значения окна *cwnd*.
- Отправка узлу A от имени узла B ложных дубликатов TCP-подтверждений. Таким образом злоумышленник вынудит узел A, во-первых, включить алгоритм быстрой повторной передачи и посылать заново уже полученные узлом B сегменты, во-вторых, перейти в режим устранения затора, уменьшив скорость отправки данных

10. Возможные способы противодействия

В заключении укажем возможные способы противодействия описанным информационно-техническим воздействиям. Ответственные лица, исходя из политики сетевой безопасности, и имея четкое представление о возможных инцидентах и их последствиях, смогут определить, какие меры являются необходимыми и приемлемыми в каждом конкретном случае.

Фильтрация на маршрутизаторе

Фильтры на маршрутизаторе применяются для запрета пропуска дейтаграмм, которые могут быть использованы для атак злоумышленника.

- Запретить пропуск дейтаграмм с широковещательным адресом назначения между сетью организации и Интернетом.
- Запретить пропуск дейтаграмм, направленных из внутренней сети (сети организации) в Интернет, но имеющих внешний адрес отправителя.
- Запретить пропуск дейтаграмм, прибывающих из Интернета, но имеющих внутренний адрес отправителя.
- Запретить пропуск дейтаграмм с опцией «Source Route» и инкапсулированных дейтаграмм (IP-дейтаграмма внутри IP-дейтаграммы).
- Запретить пропуск дейтаграмм с ICMP-сообщениями между сетью организации и Интернетом, кроме необходимых («Destination Unreachable: Datagram Too Big» — для алгоритма Path MTU Discovery; также «Echo», «Echo Reply», «Destination Unreachable: Network Unreachable», «Destination Unreachable: Host Unreachable», «TTL exceeded»).
- На сервере доступа клиентов по коммутируемой линии — разрешить пропуск дейтаграмм, направленных только с/на IP-адрес, назначенный клиенту.
- Запретить пропуск дейтаграмм с UDP-сообщениями, направленными с или на порты echo и chargen, либо на все порты, кроме используемых (часто используется только порт 53 для службы DNS).
- Использование TCP Intercept для защиты от атак SYN flood.
- Фильтрация TCP-сегментов выполняется в соответствии с политикой безопасности: разрешаются все сервисы, кроме запрещенных, или запрещаются все сервисы, кроме разрешенных. Если во внутренней сети нет хостов, к которым предполагается доступ из Интернет, но разрешен доступ внутренних хостов в Интернет, то следует запретить пропуск TCP SYN-сегментов из Интернет во внутреннюю сеть, а также запретить пропуск дейтаграмм с «Fragment Offset» = 1 и «Protocol» = 6 (TCP).

Анализ сетевого трафика

Анализ сетевого трафика проводится для обнаружения атак, предпринятых злоумышленниками, находящимися как в сети организации, так и в Интернете.

- Сохранять и анализировать статистику работы маршрутизаторов, особенно — частоту срабатывания фильтров.

- Применять специализированное программное обеспечение для анализа трафика для выявления выполняемых атак (*NIDS — Network Intrusion Detection System*). Выявлять узлы, занимающие ненормально большую долю полосы пропускания, и другие аномалии в поведении сети.
- Применять программы типа *arpwatch* для выявления узлов, использующих нелегальные IP- или MAC-адреса.
- Применять программы типа *Antisniff* для выявления узлов, находящихся в режиме прослушивания сети.

Защита маршрутизатора

Мероприятия по защите маршрутизатора проводятся с целью предотвращения атак, направленных на нарушение схему маршрутизации дейтаграмм или на захват маршрутизатора злоумышленником.

- Использовать аутентификацию сообщений протоколов маршрутизации с помощью алгоритма MD5.
- Осуществлять фильтрацию маршрутов, объявляемых сетями-клиентами, провайдером или другими автономными системами. Фильтрация выполняется в соответствии с маршрутной политикой организации; маршруты, не соответствующие политике, игнорируются.
- Использовать на маршрутизаторе, а также на коммутаторах статическую ARP-таблицу узлов сети организации.
- Отключить на маршрутизаторе все ненужные сервисы (особенно так называемые «диагностические» или «малые» сервисы TCP: echo, chargen, daytime, discard, и UDP: echo, chargen, discard).
- Ограничить доступ к маршрутизатору консолью или выделенной рабочей станцией администратора, использовать парольную защиту; не использовать telnet для доступа к маршрутизатору в сети, которая может быть прослушана.
- Использовать последние версии и обновления программного обеспечения, следить за бюллетенями по безопасности, выпускаемыми производителем.

Защита хоста

Мероприятия по защите хоста проводятся с целью предотвращения атак, имеющих целью перехват данных, отказ в обслуживании, или проникновение злоумышленника в операционную систему.

- Запретить обработку ICMP «Echo»-запросов, направленных на широковещательный адрес.
- Запретить обработку ICMP-сообщений «Redirect», «Address Mask Reply», «Router Advertisement», «Source Quench».
- Если хосты локальной сети конфигурируются динамически сервером DHCP, использовать на DHCP-сервере таблицу соответствия MAC- и IP-адресов и выдавать хостам заранее определенные IP-адреса.
- Отключить все ненужные сервисы TCP и UDP (перевести порты из состояния LISTEN в CLOSED).
- Для обслуживания входящих соединений использовать супердемон типа *tcpserver* или *xinetd*, позволяющий устанавливать максимальное число одновременных соединений, список разрешенных адресов клиентов, выполнять проверку легальности адреса через DNS и регистрировать соединения в лог-файле.
- Использовать программу типа *tcplogd*, позволяющую отследить попытки скрытного сканирования (например, полутокрытыми соединениями).
- Использовать статическую ARP-таблицу узлов локальной сети.
- Применять средства безопасности используемых на хосте прикладных сервисов.
- Использовать последние версии и обновления программного обеспечения, следить за бюллетенями по безопасности, выпускаемыми производителем.

Превентивное сканирование

Администратор безопасности должен знать и использовать методы и инструменты злоумышленника и проводить превентивное сканирование сети своего объекта с целью обнаружения слабых мест в безопасности до того, как это сделает злоумышленник. Для этой цели имеется ряд сертифицированных ФСТЭК и ФСБ России сканеров безопасности, *network security scanners*, например *Nessus и Xspider и др.*

Литература

1. Постановление Правительства Российской Федерации от 23 марта 2006 г. № 411–рс «Об утверждении перечня критически важных и опасных объектов Российской Федерации».

2. *Петренко С. А., Курбатов В. А.* Политики информационной безопасности. М.: ДМК Пресс, 2006. 400 с.: ил. (Информационные технологии для инженеров).
3 *Петренко С. А., Симонов С. В.* Управление информационными рисками. Экономически оправданная безопасность. М.: ДМК Пресс, 2005. 384 с.: ил. (Информационные технологии для инженеров).
4. *Петренко С. А., Петренко А. А.* Аудит безопасности Intranet. М.: ДМК Пресс, 2002. 416 с.: ил. (Информационные технологии для инженеров).
5. *Мамаев М. А., Петренко С. А.* Технологии защиты информации в Интернете. Специальный справочник. СПб: Изд-во Питер, 2002. 848 с.: ил. (Специальный справочник).