Linux IPCHAINS-HOWTO

Автор Paul Russell, *ipchains@rustcorp.com*
Русский перевод Ilgiz Kalmetev, *ilgiz@mail.rb.ru* v1.0.7, 12 March 1999

Этот документ рассказывает как получить, установить и сконфигурировать программное обеспечение для IP firewalling цепочек(chains) в Linux и предлагает некоторые приемы их использования.

Содержание

Примечание переводчика
Ниже указаны англо-русские переводы, используемые в данном переводе, как мне кажется более подходящие в контексте предложений данного документа:

```
chains        -- цепочки (цепочки правил)
target        -- действие (действие, совершаемое над пакетом)
policy        -- политика (политика фильтрации)
to resolve    -- преобразование имен в адреса (или наоборот)
MTU discovery -- проверка MTU
redirection   -- перенаправление
forwarding    -- форвардинг
wildcards     -- уайлдкарты
patch         -- патч
```

1 Введение

Это - Linux IPCHAINS-HOWTO; в “Где?"указан исходный сайт, который содержит последнюю копию этого документа. Вы должны также прочесть Linux NET-3-HOWTO. Также неплохо бы ознакомиться с IP-Masqurading HOWTO, PPP-HOWTO, Ethernet-HOWTO и Firewall HOWTO.
Если вам подходит пакетная фильтрация, читайте разделы “Почему?", “Как? "и далее по заголовкам раздела “IP Firewalling цепочки".
Если вы переходите из ipfwadm, прчтите разделы “Введение", “Как?"и в приложениях разделы “Различия между ipchains и ipfwadm"и “Использование сценария ipfwadm-wrapper".

1.1 Что?

Linux ipchains заменяет Linux IPV4 firewalling код (который был главным образом перенесен из BSD) и заменяет ipfwadm, который был потомком ipfw из BSD. Он нужен для администрирования фильтрацией IP пакетов в ядрах Linux версии 2.1.102 и выше.

1.2 Почему?

Старый Linux firewalling код не работает с фрагментами, имеет 32-разрядные счетчики (на Intel по крайней мере), не позволяет работать с протоколами отличными от TCP, UDP или ICMP, и не может делать большие изменения atomically, не может определять обратные правила, имеет некоторые причуды, и может быть жестковат в управлении (что приводит к ошибкам пользователя).

1.3 Как?

В настоящее время код находится в ядрах от 2.1.102. Для ядер 2.0, вам нужно будет загрузить ядерный патч из сети. Если ваше ядро 2.0 более новое, чем предлагаемый патч, то патч должен подойти; эта часть ядер 2.0 довольно устойчива (напр. патч для ядра 2.0.34 работает только лучше на ядрах 2.0.35). Так как патч 2.0 несовместим с патчами ipportfw и ipautofw, я не рекомендую применять его, если вам действительно не нужны некоторые функциональных возможности, предлагаемые ipchains.

1.4 Где?

Официальная страница - *Linux IP Firewall Chains Page*
Для сообщений об ошибках, обсуждения, разработки и использования существует список почтовой рассылки. Подписаться на него можно послав сообщение, содержащее слово “subscribe"в ipchains-request на rustcorp.com. Чтобы отправить письмо в список рассылки укажите "ipchains' вместо "ipchains-requetst".

2 Основы фильтрации пакетов

2.1 Что?

Весь трафик в сети производится в виде пакетов. Например, при скачивании этого пакета (примерно 50КБ) вы, возможно, приняли 36 или около этого пакетов длиной 1460 байт каждый.
Начало каждого пакета сообщает, куда он направляется, откуда он исходит, тип пакета и другие административные подробности. Это начало пакета называется заголовком. Остальная часть пакета, содержащая фактически передаваемые данные, обычно называется телом.
Некоторые протоколы, такие как TCP, которые используются для web, электронной почты и удаленных регистраций в системах, используют понятие "соединения перед посылкой любых пакетов с данными производится обмен различными настроечными пакетами (со специальными заголовками), говорящими "Я хочу соединиться", "Хорошо, соединяйся"и "Спасибо". Затем идут обычные пакеты.
Пакетный фильтр - кусок программного обеспечения, которое рассматривает заголовок курсирующих пакетов и решает что делать со всем пакетом. Оно может отвергнуть пакет (то есть отбросить пакет, как будто никогда не получало его), принять пакет (то есть позволить пакету пройти) или отклонить пакет (не только отвергнуть, но и сообщить об этом отправителю пакета).
Под Linux, фильтрация пакетов встроена в ядро, и хотя есть способы, которыми мы можем как-то оперировать пакетами, но общий принцип просмотра заголовков и решения судьбы пакета остается в коде ядра.

2.2 Почему?

Управление. Защита. Осторожность.

Управление:

когда вы используете Linux-машину для подключения вашей внутренней сети к другой сети (скажем, к Internet), то вы имеете возможность разрешить какой-то тип трафика и запретить какой-то другой тип. Например, заголовок пакета содержит адрес назначения пакета, так что вы можете предотвратить выход некоторых пакетов во внешнюю сеть. Еще пример: я использую `Netscape`, чтобы обратиться к архивам `Dilbert`. На странице размещены баннеры от `doubleclick.net`, и `Netscape` будет тратить мое время на их скачивание. Эту проблему можно решить сообщив пакетному фильтру, чтобы он не пропускал любые пакеты в или из адресов, принадлежащих `doubleclick.net` (хотя для этого есть лучшие способы).

Защита:

когда ваша Linux-машина - единственый заслон между хаосом Internet и вашей красивой, организованной сетью, хорошо знать, что вы можете ограничить то, что пытается пройти в вашу дверь. Например, вы могли бы позволить чему-нибудь выйти из вашей сети, но вас может беспокоить известный "Пинг Смерти", посылаемый подлыми злоумышленниками. Вот другой пример: вы не хотите пускать посторонних на telnet-порт вашей Linux-машины, даже если все ваши аккаунты имеют пароли; возможно вы хотите (как большинство людей) только просматривать ресурсы Internet, а не предоставлять сервис (так или иначе) - просто не позволяйте никому подсоединяться к порту; при наличии пакетного фильтра входящие пакеты, используемые для установки соединения, будут отклоняться.

Осторожность:

иногда неправильно настроенная машина в локальной сети может отправлять пакеты во внешний мир. Хорошо если бы фильтр пакетов дал вам знать, что происходит нечто аварийное; возможно вы захотите принять какие-то меры или только поинтересуетесь природой событий.

2.3 Как?

2.3.1 Ядро с фильтрацией пакетов

Вам нужно ядро, в которое встроен новый код IP firewall chains (цепочки). Вы можете проверить, если этот код в ядре, поискав файл "/proc/net/ip_fwchains'. Если он существует, то все в порядке. Если нет, то вы должны сделать ядро, в котором есть IP firewall цепочки. Сначала, скачайте исходники ядра. Если у вас есть ядро 2.1.102 или выше, то вам не надо будет это патчить (сейчас этот код встроен в ядро основной ветки). Иначе, вам придется найти в web патч, указанный выше, и установить конфигурацию как разъяснено ниже. Если вы не знаете, как это сделать, не паникуйте - прочтите Kernel-HOWTO.
Опции настроек, которые вы должны будете установить для ядер 2.0:

```
CONFIG_EXPERIMENTAL=y
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
CONFIG_IP_FIREWALL_CHAINS=y
```

Для ядер 2.1 или 2.2 :

```
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
```

ipchains общается с ядром и сообщает ему какие пакеты фильтровать. Если вы не программист, или очень любопытный человек, это как вы будете управлять пакетной фильтрацией.

2.3.2 ipchains

Ipchains вставляет и удаляет правила из раздела пакетной фильтрации ядра. Это означает, что все, что вы устанавливаете будет потеряно после перезагрузки; как этого избежать написано в “Создание постоянных правил".
Ipchains заменяет ipfwadm, который использовался в старом коде IP Firewall. Есть набор полезных скриптов, доступных на ftp-сайте ipchains:

```
ftp://ftp.rustcorp.com/ipchains/ipchains-scripts-1.1.2.tar.gz
```

Это скрипт shell, называемый ipfwadm-wrapper, который позволяет вам сделать такую же пакетную фильтрацию, как это было выполнено прежде. Вам не стоит использовать этот скрипт, если вы не хотите быстро перейти на новую систему со старого ipfwadm (это медленнее, отсутствует настройка параметров и т.д).
В этом случае, вам также не нужен этот HOWTO.
Подробности о проблемах ipfwadm см. в приложениях “Различия между ipchains и ipfwadm"и “Использование скрипта ipfwadm-wrapper'.

2.3.3 Создание постоянных правил

Ваша текущая настройка firewall хранится в ядре, и таким образом будет потеряна на перезагрузке. Я рекомендую использовать скрипты ‘ipchains-save' и ‘ipchainsrestore', чтобы ваши правила действовали постоянные. Чтобы сделать это, настройте ваши правила, затем выполните (от root):

```
# ipchains-save > /etc/ipchains.rules
#
```

Создайте примерно такой скрипт:

```
#! /bin/sh
# Скрипт управления пакетной фильтрацией.

# Если правил нет, то ничего не делать.
[ -f /etc/ipchains.rules ] || exit 0

case "$1" in

  start)
    echo -n "Включение пакетной фильтрации:"
    /sbin/ipchains-restore < /etc/ipchains.rules || exit 1
    echo 1 > /proc/sys/net/ipv4/ip_forward
    echo "." ;;

  stop)
    echo -n "Отключение пакетной фильтрации:"
    echo 0 > /proc/sys/net/ipv4/ip_forward
    /sbin/ipchains -X
    /sbin/ipchains -F
    /sbin/ipchains -P input ACCEPT
    /sbin/ipchains -P output ACCEPT
    /sbin/ipchains -P forward ACCEPT
    echo "." ;;

  *)
    echo "Использование: /etc/init.d/packetfilter {start|stop}"
    exit 1 ;;
```

```
esac

exit 0
```

Убедитесь, что этот скрипт запускается при загрузке. В моем случае (Debian 2.1), я делаю символическую связь с именем "S39packetfilter"в каталоге "/etc/rcS.d' (он будет выполняться перед S40network).

3 Я запутался! Маршрутизация, маскарадинг, форвардинг портов, ipautofw ...

Этот HOWTO рассказывает о фильтрации пакетов. Это значит, что принимается решение, нужно ли пакету позволить пройти или нет. Однако, так как Linux является детской песочницей для хакера, то вы вероятно захотите сделать больше, чем только это.
Одна проблема состоит в том, что один и тот же инструмент (“ipchains ") используется для управления и маскарадингом, и прозрачным прокси, хотя они и не имеют к пакетной фильтрации совершенно никакого отношения (текущая реализация Linux создает впечатление, что они близко связаны).
Маскарадинг и прокси объясняются в отдельных HOWTO, а автоматический форвардинг и форвардинг портов управляются отдельными инструментальными средствами, но так как много людей спрашивает меня о них, я включу набор общих сценариев и укажу, когда каждый из них должен примениться. Качество защиты каждой установки здесь не обсуждаются.

3.1 Трехстрочное руководство Русти по маскарадингу

Подразумевается, что ваш внешний интерфейс называется ppp0. Проверьте это утилитой ifconfig и отредактируйте по вкусу:

```
# ipchains -P forward DENY
# ipchains -A forward -i ppp0 -j MASQ
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

3.2 Безвозмездная поддержка: правила WatchGuard

Вы можете купить off-the-shelf файерволы. Отличный выбор - один из WatchGuard FireBox. Отличный потому, что он мне нравится, он безопасен, основан на Linux, и потому что они финансируют сопровождение ipchains, также как и нового файерволльного кода (в 2.3). Короче говоря, WatchGuard оплачивает за меня еду, в то время как я работаю для вас. Так что пожалуйста посмотрите, что они предлагают.

http://www.watchguard.com

3.3 Общие Firewall-ные установки

У вас есть домен littlecorp.com. У вас есть внутренняя сеть, и одно подключение к Internet по коммутируемому каналу (PPP) (firewall.littlecorp.com с адресом 1.2.3.4). Ваша локальная сеть построена на ethernet, а ваша персональная машина называется "myhost".
В этом разделе приводятся различные соглашения, которые являются общими. Тщательно изучите их, потому что они имеют тонкие различия.

3.3.1 Частная сеть: традиционные прокси

В этом сценарии, пакеты из частной сети никогда не выходят в Internet, и наоборот. Адреса IP частной сети должны быть назначены по RFC1597 Private Network Allocations (то есть, 10.*.*.*, 172.16.*.* или 192.168.*.*).
Единственный способ подсоединиться к Internet - через firewall, который является единственной машиной в обеих сетях, которая перераспределяет соединения. Вы запускаете программу (на firewall), называемую proxy, которая все это делает (имеются прокси для FTP, web, telnet, RealAudio, Usenet и других услуг). См. Firewall HOWTO.
Любые услуги Интернет, которые вам потребовались, должны быть на firewall.
(Однако см. "Ограниченные внутренние услуги"ниже).
Пример: Разрешить доступ из частной сети к web-сервису Интернет.

1. Частной сети назначены адреса 192.168.1.*, myhost имеет адрес 192.168.1.100, а ethernet интерфейс firewall'а 192.168.1.1.
2. Web proxy (напр. "Squid") установлен и сконфигурирован на firewall, скажем, на порту 8080.
3. Netscape в частной сети сконфигурирован для использования firewall порта 8080 в качестве прокси.
4. DNS в частной сети не нужно настраивать.
5. DNS должен быть настроен на firewall.
6. В частной сети маршрут по умолчанию (он же - гейт) не нужно настраивать.

Netscape на myhost обращается к http://slashdot.org.

1. Netscape соединяется с firewall портом 8080, используя порт 1050 на myhost. Он запрашивает страницу web "http://slashdot.org".
2. Proxy переводит имя "slashdot.org"в IP адрес, и получает 207.218.152.131. Затем он открывает соединение с этим адресом IP (используя порт 1025 на внешнем интерфейсе firewall'а), и запрашивает у web-сервера (порт 80) страницу web.
3. Как только он получит страницу web через соединение с сервером web, он скопирует данные в соединение с Netscape.
4. Netscape отображает страницу.

То есть, с точки зрения slashdot.org, было создано соединение между адресом 1.2.3.4 (интерфейсом PPP firewall'а) порт 1025 и адресом 207.218.152.131 (slashdot.org) порт 80. С точки зрения myhost, соединение создано с 192.168.1.100 (myhost) порт 1050 и 192.168.1.1 (ethernet интерфейс firewall'а) порт 8080.

3.3.2 Частная сеть: прозрачные прокси

В этом сценарии, пакеты из частной сети никогда не выходят в Internet, и наоборот. Адреса IP частной сети должны быть назначены по RFC1597 Private Network Allocations (то есть, 10.*.*.*, 172.16.*.* или 192.168.*.*).
Единственный способ подсоединиться к Internet - через firewall, который является единственной машиной в обеих сетях, которая перераспределяет соединения. Вы запускаете программу (на firewall), называемую прозрачный proxy, которая все это делает; ядро пересылает пакеты прокси вместо того, чтобы отправить их наружу (то есть, это похоже на маршрутизацию).
Прозрачный прокси означает, что клиенты не должны знать, что в сети работает прокси.
Любые услуги Интернет, которые вам потребовались, должны быть на firewall.
(Однако см. "Ограниченные внутренние услуги"ниже).
Пример: Разрешить доступ из частной сети к web-сервису Интернет.

1. Частной сети назначены адреса 192.168.1.*, myhost имеет адрес 192.168.1.100, а ethernet интерфейс firewall'а 192.168.1.1.
2. Прозрачный прокси (я полагаю, что это патчи к squid, или "transproxy") установлен и настроен на firewall, скажем, на порту 8080.
3. Ядру сообщают, что надо переназначать соединения с портом 80 на прокси, используя ipchains.
4. Netscape в частной сети настроен на прямое подключение.
5. В частной сети должен быть настроен DNS (то есть вы должны запустить DNS сервер как прокси на firewall).
6. В частной сети должен быть настроен маршрут по умолчанию (aka гейт), чтобы пакеты посылались на firewall.

Netscape на myhost обращается к http://slashdot.org.

1. Netscape запрашивает страницу web "http://slashdot.org"и получает 207.218.152.131. Он открывает соединение с этим IP адресом, используя локальный порт 1050 и запрашивает на web-сервере (порт 80) страницу web.
2. Поскольку пакеты из myhost (порт 1050) к slashdot.org (порт 80) проходят через firewall, они переназначаются ждущему прозрачному прокси на порту 8080. Прозрачный прокси открывает соединение (используя локальный порт 1025) с 207.218.152.131 порт 80 (которому предназначались первоначальные пакеты).
3. Как только прокси получит страницу web из соединения с сервером web, он копирует данные на соединение с Netscape.
4. Netscape отображает страницу.

То есть с точки зрения slashdot.org, соединение установлено между 1.2.3.4 (интерфейс PPP firewall'а) порт 1025 и 207.218.152.131 (slashdot.org) порт 80. С точки зрения myhost соединение установлено между 192.168.1.100 (myhost) порт 1050 и 207.218.152.131 (slashdot.org) порт 80, но фактически обмен информацией совершает прозрачный прокси.

3.3.3 Частная сеть: маскарадинг

В этом сценарии, пакеты из частной сети никогда не выходят в Internet без специальной обработки, и наоборот. Адреса IP частной сети должны быть назначены по RFC1597 Private Network Allocations (то есть, 10.*.*.*, 172.16.*.* или 192.168.*.*).
Вместо использования прокси, мы используем специальное средство ядра, называемое "маскарадинг". Маскарадинг перезаписывает пакеты, когда они проходят через firewall, так, чтобы казалось, что они всегда исходят от firewall непосредственно. Затем он перезаписывает ответы так, чтобы было похоже, что они пришли от первоначального получателя.
Маскарадинг имеет отдельные модули для обработки "сложных"протоколов, типа FTP, RealAudio, Quake и т.д. Для действительно тяжелых в обработке протоколов применяется "автофорвардинг", который может обработать некоторые из них автоматической установкой форвардинга портов для соответствующих номеров портов: см. “ ipportfw"(ядра 2.0) или “ipmasqadm"(ядра 2.1).
Любые услуги Интернет, которые вам нужны, должны быть на firewall.
(Однако см. “Ограниченные внутренние услуги"ниже).
Пример: Разрешить доступ из частной сети к web-сервису Интернет.

1. Частной сети назначены адреса 192.168.1.*, myhost имеет адрес 192.168.1.100, а ethernet интерфейс firewall'а 192.168.1.1.
2. Firewall установлен на подмену любых пакетов, исходящих из частной сети и идущих на порт 80 хоста в Интернет.

3. Netscape сконфигурирован для непосредственного соединения.
4. DNS в частной сети должен быть правильно сконфигурирован.
5. Firewall должен быть маршрутом заданным по умолчанию (aka гейтом) для частной сети.

Netscape на myhost обращается к http://slashdot.org.

1. Netscape запрашивает страницу web "http://slashdot.org"и получает 207.218.152.131. Он открывает соединение с этим IP адресом, используя локальный порт 1050 и запрашивает на web-сервере (порт 80) страницу web.
2. Поскольку пакеты от myhost (порт 1050) к slashdot.org (порт 80) проходят через firewall, они перезаписываются так, чтобы выходить с интерфейса PPP firewall порт 65000. Firewall имеет допустимый Internet адрес (1.2.3.4), так что ответные пакеты от slashdot.org нормально приходят на firewall.
3. По прибытию пакеты от slashdot.org (порт 80) на firewall.littlecorp.com (порт 65000) перезаписываются так, чтобы идти на myhost порт 1050. В этом и состоит фокус маскарадинга: он помнит, когда он перезаписал исходящие пакеты, и может переписывать их обратно, когда приходят ответы на них.
4. Netscape отображает страницу.

то есть с точки зрения slashdot.org соединение было сделано между 1.2.3.4 (интерфейс PPP firewall'a) порт 65000 и 207.218.152.131 (slashdot.org) порт 80. С точки зрения myhost соединение было сделано между 192.168.1.100 (myhost) порт 1050 и 207.218.152.131 (slashdot.org) порт 80.

3.3.4 Общедоступная сеть

В этом сценарии, ваша персональная сеть - часть Internet: пакеты могут течь без изменения из одной сети в другую. Адреса IP внутренней сети должны быть назначены из выделенного блока адресов IP, так что остальная часть сети будет знать, как получить пакеты, адресованные вам. Подразумевается постоянное соединение.
В этом случае, фильтрация пакетов используется для ограничений пакетов, пересылаемых между вашей сетью и остальной частью Internet, т.е. ограничить только доступ остальной части Internet к вашим внутренним серверам web.
Пример: Разрешить доступ из частной сети к web-сервису Интернет.

1. В вашей внутренней сети адреса назначены согласно выделенному блоку IP адресов (скажем 1.2.3.*).
2. Установки firewall позволяют весь трафик.
3. Netscape настроен для непосредственного соединения.
4. В вашей сети должен быть правильно настроен DNS.
5. Firewall должен быть маршрутом заданным по умолчанию (aka гейтом) для частной сети.

Netscape на myhost обращается к http://slashdot.org.

1. Netscape запрашивает страницу web "http://slashdot.org"и получает 207.218.152.131. Он открывает соединение с этим IP адресом, используя локальный порт 1050 и запрашивает на web-сервере (порт 80) страницу web.
2. Пакеты проходят через ваш firewall также, как они проходят через несколько маршрутизаторов между вами и slashdot.org.
3. Netscape отображает страницу.

то есть имеется только одно соединение: между 1.2.3.100 (myhost) порт 1050 и 207.218.152.131 (slashdot.org) порт 80.

3.3.5 Ограничение внутренних услуг

Кроме firewall существуют и другие способы обеспечить доступ из Интернет к вашим внутренним ресурсам. Эти способы основаны на принципах проксирования или маскарадинга для внешних соединений.
Самый простой подход состоит в запуске "перенаправителя(redirector)", который является подвидом прокси, который ждет соединение на данном порте, и затем открывает соединение на фиксированном внутреннем хосте и порту, и копирует данные между двумя соединениями. Пример такой программы - "redir". С точки зрения Internet соединение установлено с вашим firewall. С точки зрения вашего внутреннего сервера, соединение установлено от внутреннего интерфейса firewall к серверу.
Другой подход (для него требуется ядро 2.0, с пропатченным ipportfw, или версии ядра 2.1 или более поздние) состоит в использовании форвардинга портов в ядре. Он делает ту же самую работу, что и "redir", но другим способом: ядро перезаписывает проходящие пакеты, заменяя их адрес и порт назначения на адрес и порт внутреннего хоста. С точки зрения Internet, соединение установлено с вашим firewall. С точки зрения вашего внутреннего сервера, это прямое соединение от хоста Internet до сервера.

3.4 Подробная информация о маскарадинге

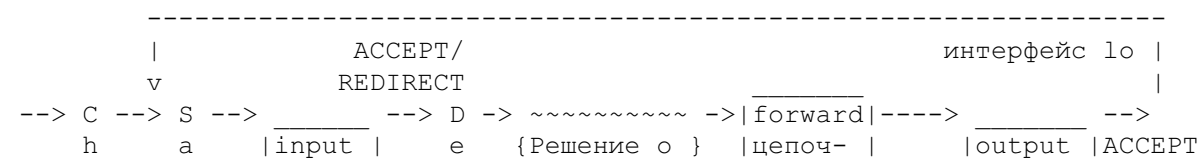
Дэвид Ранч написал превосходный новый HOWTO о маскарадинге, который во многом пересекается с этим HOWTO. Вы можете найти его на *http://www.ecst.csuchico.edu/~dranch/LINUX/index-LINUX.html#ipmasq* Подробности о маскарадинге.
Я думаю, что скоро этот документ будет входить в состав Linux Documentation Project на *http://www.metalab.unc.edu/LDP*
Официальная страница маскарадинга - *http://ipmasq.cjb.net*

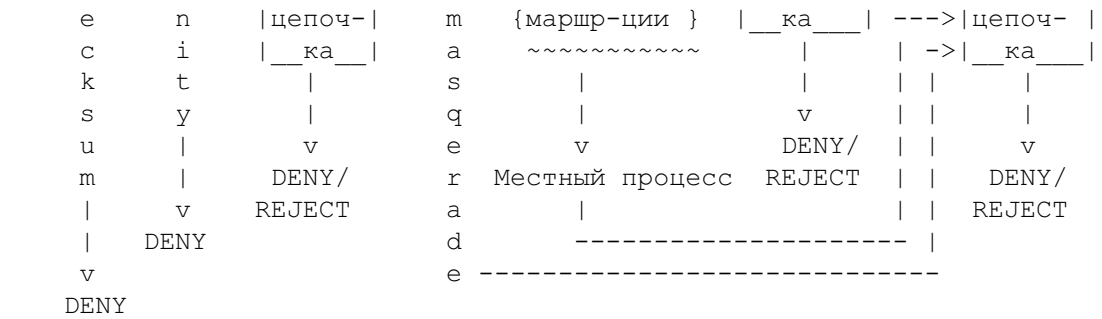
4 IP Firewalling цепочки

Этот раздел описывает все, что вы действительно должны знать для построения пакетного фильтра, удовлетворяющего вашим потребностям.

4.1 Как фильтры отсеивают пакеты

Ядро стартует с тремя списками правил; эти списки называются firewall-цепочками или просто цепочками. Три цепочки называются input, output и forward. Когда приходит пакет (скажем, через плату ethernet) ядро использует цепочку input, чтобы решить судьбу пакета. Если пакет фильтр пропустит пакет, то ядро решает, куда послать пакет дальше (это называется маршрутизацией). Если пакет предназначен для другой машины, ядро консультируется с цепочкой forward. И наконец, прежде, чем пакет выйдет через сетевой интерфейс, ядро консультируется с цепочкой output.
Цепочка - контрольный список правил. Каждое правило говорит "если заголовок пакета походит на это, то с пакетом поступить так-то'. Если правило не применимо к пакету, то рассматривается следующе правило в цепочке. Наконец, если никакие правил не подошли, то ядро обращается к стратегии цепочек, чтобы решить, что делать. В системе с повышенными требованиями к безопасности, эта стратегия обычно заставляет ядро отклонить или отвергнуть пакет.
Для любителей ASCII-картинок, нарисован полный путь пакета, входящего в машину.





Вот методическое описание каждой стадии:

Контрольная сумма(Checksum):

Проверка целостности пакета. Если контрольная сумма не совпадает, то пакет отбрасывается(DENY).

Здравомыслие(Sanity):

На самом деле проверка на правильность формата пакета проводится перед каждой цепочкой, но цепочка input наиболее важна. Некоторые пакеты неправильного формата могли бы запутать код, обеспечивающий проверку правила, вот такие пакеты здесь и отбрасываются (DENY) (если такое случилось, то в syslog помещается сообщение).

Цепочка input:

Это - первая firewall цепочка, проверяющая пакет. Если цепочки на приняла решение об отбрасывании(DENY) или отклонении(REJECT) пакета, пакет идет дальше.

Демаскарадинг(Demasquerade):

Если пакет является ответом на замаскараденный пакет, он демаскарадируется, и перебрасывается прямо на цепочку output. Если вы не используете IP маскарадинг, то можете мысленно стереть это место в диаграмме.

Решение о маршрутизации:

Поле адреса назначения исследуется кодом маршрутизации, чтобы решить, не направлен ли этот пакет локальному процессу (см. "Локальный процесс"ниже) или послан удаленной машине (см. "цепочку forward"ниже).

Локальный процесс:

процесс, выполняющийся на машине может получать пакеты после шага "Решение о маршрутизации", и может отправлять пакеты (которые проходят шаг "Решения о маршрутизации"и затем пересекают цепочку output).

Интерфейс lo:

Если пакеты из локального процесса предназначены локальному процессу, они пройдут цепочку output с интерфейсом, установленным в "lo", и возвратятся через входную цепочку с интерфейсом тоже "lo". Интерфейс lo обычно называется петлевым интерфейсом (loopback).

Локальный(local):

Если пакет не был создан локальным процессом, то цепочка forward проверяет его, иначе пакет идет на цепочку output.

Цепочка forward:

Эту цепочку проходят любые пакеты, которые пытаются уйти на другую машину.

Цепочка output:

Эта цепочка проверяет все пакеты прежде, чем выпустить их наружу.

4.1.1 Использование ipchains

Сначала убедитесь, что ваша версия ipchains соответствует той версии, о которой рассказывается в этом документе:

```
$ ipchains --version
ipchains 1.3.9, 17-Mar-1999
```

Обратите внимание, что я рекомендую 1.3.4 (у которого нет никаких длинных опций, типа '–sport'), или 1.3.8 или выше; они очень устойчивы.
Ipchains имеет довольно подробный man (`man ipchains`), а если вам нужны подробности, вы можете проверить программный интерфейс (`man 4 ipfw`), или файл `net/ipv4/ip_fw.c` в исходных текстах ядра 2.1.x, который является (очевидно) авторитарным.
Имеется также превосходный краткий справочник Скотта Бронсона в пакете с исходными текстами в форматах A4 и US Letter Postscript(TM).
С ipchains можно делать множество вещей. Во-первых, управлять целыми цепочками. Изначально у вас есть три встроенных цепочки: input, forward и output, которые вы не можете удалять.

1. Создать новую цепочку (**-N**).
2. Удалить пустую цепочку (**-X**).
3. Изменить стратегию для встроенной цепочки (**-P**).
4. Выдать список правил цепочки (**-L**).
5. Удалить правила из цепочки (**-F**).
6. Обнулить счетчики пакетов и байтов во всех правилах цепочки (**-Z**).

Имеется несколько способов управлять правилами внутри цепочки:

1. Добавить новое правило к цепочке (**-A**)
2. Вставить новое правило в определенную позицию цепочки (**-I**)
3. Заменить правило в определенной позиции цепочки (**-R**)
4. Удалить правило в определенной позиции цепочки (**-D**)
5. Удалить первое правило в цепочке, удовлетворяющее условию (**-D**)

Есть несколько операций для маскарадинга, которые находятся в ipchains из-за отсутствия более подходящего места для их размещения:

1. Вывести текущие параметры маскарадинга (**-M -L**)
2. Установить значения таймаутов для маскарадинга (**-M -S**) (Но см. "Я не могу установить таймауты маскарадинга!").

Последняя (и, возможно, наиболее полезная) функция позволяет вам проверить, что случилось бы с данным пакетом, если бы он должен был пересечь данную цепочку.

4.1.2 Обработка одним правилом

Это - "бутерброд"ipchains; правила управления. Наиболее часто вы вероятно используете команды добавления (-A) и удаления (-D). Другие (-I для вставки и -R для замены) - простые расширения этих концепций.
Каждое правило определяет набор условий отбора (например по полю "адрес назначения") и обработки пакетов. Например, вы можете захотеть отвергать все ICMP пакеты, исходящие с IP адреса 127.0.0.1. В этом случае наши условия заключаются в том, что протокол должен быть ICMP и исходный адрес 127.0.0.1. Наше действие - "DENY"(отбросить).
127.0.0.1 - это "петлевой"интерфейс, который у вас будет существовать, даже если у вас отсутствует реальное сетевое соединение. Вы можете использовать программу "ping", чтобы сгенерировать такие пакеты (она просто посылает тип ICMP 8 (запрос ECHO), на которые все кооперативные хосты должны из вежливости ответить пакетами с типом ICMP 0 (ответ ECHO)). Это полезно для тестирования.

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
```

Вы видите, что первый ping проходит нормально (опция с 1' означает, что надо послать один пакет). Затем мы прибавляем (-A) к цепочке "input"правило, определяющее, что пакеты от 127.0.0.1 (s 127.0.0.1') протокола ICMP (p ICMP') мы должны отбросить(DENY) (j DENY').
Затем мы проверяем наше правило, вторично запустив ping. Программа попытается дождаться ответа, который так и не придет.
Мы можем удалить правило одним из двух способов. Во-первых, так как мы знаем, что это - единственое правило во входной цепочке, то мы можем использовать удаление по номеру. Удаляем правило 1 в цепочке:

```
# ipchains -D input 1
#
```

Второй способ состоит в применении зеркальную команде -A команду -D. Это полезно, когда у вас сложная цепочка правил, и вы не хотите выяснять номер правила. В этом случае, мы действовали бы так:

```
# ipchains -D input -s 127.0.0.1 -p icmp -j DENY
#
```

Синтаксис -D должен иметь такие же опции, что и -A (или -I или -R). Если в той же самой цепочке имеются несколько идентичных правил, то будет удалено только первое из них.

4.1.3 Указание правил фильтрации

Мы видели, что p"применяется для определения протокола, a s' для определения исходного адреса, но имеются и другие опции, которые мы можем использовать для определения характеристик пакета. Ниже такие опции поясняются очень подробно.

Определение адресов назначения и источника IP адреса источника (-s) и назначения (-d) могут быть определены четырьмя способами. Наиболее общий способ состоит в том, чтобы использовать полное имя, типа "localhost"или "www.linuxhq.com". Второй способ состоит в том, чтобы указать IP адрес типа "127.0.0.1".
Третий и четвертый способ позволяют указать группы адресов IP, типа "199.95.207.0/24"или "199.95.207.0/255.255.255.0".
Они оба определяют любой адрес IP от 192.95.207.0 до 192.95.207.255 включительно; цифры после "/"сообщают, какая часть IP адреса является значительной. "/32"или "/255.255.255.255 значение по умолчанию (соответствует всем IP адресам). Чтобы указать любой адрес, можно использовать маску "/0":

```
# ipchains -A input -s 0/0 -j DENY
#
```

Однако, такая запись используется крайне редко, потому что тот же эффект достигается, если в правиле совсем не указывать адрес.

Указание инверсии Многие флажки, включая s′ и d′ могут иметь параметры, которым предшествует "!′ (произносится "не′) для выделения адресов, НЕ равных этим данным. Например, s! localhost′ соответствует любому пакету, не исходящему из localhost.

Указание протокола Протокол может быть определен флажком p". Протокол может быть числом (если вы знаете числовые значения протокола для IP) или именем типа "TCP", "UDP"или "ICMP". Регистр букв не имеет значения, так что "tcp"понимается эквивалентно "TCP".
Имени протокола может предшествовать "!"для инверсии заданных параметров, напр., p! TCP′.

Указание портов UDP и TCP В специальных случаях, когда определен протокол TCP или UDP, может указываться дополнительный параметр, указывающий порт TCP или UDP, или (включительно) диапазон портов (но см. “Обработка фрагментов"ниже). Диапазон определяется с помощью символа ":′, типа "6000:6010′, что означает 11 портов с номерами от 6000 до 6010. Если нижняя граница не указана, то значение по умолчанию 0. Если верхний предел опущен, это значение по умолчанию 65535. Напр., для указания TCP соединений, исходящих из первых 1024 портов, пишем так: p TCP -s 0.0.0.0/0:1023′. Номера порта могут быть определены именами, напр. "www′.
Обратите внимание, что спецификации порта можно предшествовать "!", который инвертирует это условие. Вот так можно определить любой TCP пакет, кроме www:

```
-p TCP -d 0.0.0.0/0 ! www
```

Важно понять что спецификация

```
-p TCP -d ! 192.168.1.1 www
```

очень отличается от

```
-p TCP -d 192.168.1.1 ! www
```

Первое определяет любой TCP пакет на WWW порту любой машины, кроме 192.168.1.1. Второй определяет любое TCP соединение на любом порту на 192.168.1.1, кроме WWW порта.
В заключение, случай для не WWW порта и не 192.168.1.1:

```
-p TCP -d ! 192.168.1.1 ! www
```

Указание типа и кода ICMP ICMP также может принять необязательный параметр, но поскольку ICMP не имеет портов, (ICMP имеет тип и код) они имеют другое значение.
Вы можете указать их как ICMP имена (используйте `ipchains -h icmp` для вывода списка имен) после опции s', или как числовой тип и код ICMP, где тип следует за опцией s', а код следуют за опцией d".
ICMP имена довольно длинны: вам только нужно использовать достаточно символов, чтобы сделать имя, отличное от других имени.
Вот маленькая таблица некоторых из наиболее общих ICMP пакетов:

```
Номер   Имя                     Требуется для
0       echo-reply              ping
3       destination-unreachable любого TCP/UDP трафика
5       redirect                маршрутизации, если не запущен демон маршрутизации
8       echo-request            ping
11      time-exceeded           traceroute
```

Обратите внимание, что именам ICMP в настоящее время не могут предшествовать "!".
ВНИМАНИЕ! Ни в коем случае не запрещайте передачу ICMP-пакетов типа 3 ! (См. " ICMP пакеты "ниже).

Указание интерфейса Опция i"определяет имя интерфейса. Интерфейс - физическое устройство, на(из) который(ого) приходит(уходит) пакет . Вы можете использовать команду `ifconfig`, чтобы посмотреть какие интерфейсы в системе "подняты"(то есть работают в настоящий момент).
Интерфейс для входящих пакетов (то есть пакетов, проходящих цепочку input) является интерфейсом, на который они пришли. Логически следует, что интерфейс для исходящих пакетов (пакеты, проходящие цепочку output) - интерфейс, через который они выйдут. Интерфейс для пакетов, проходящих цепочку forward - тоже интерфейс, через который они выйдут; довольно спорное решение, как мне кажется.
Совершенно допустимо определить интерфейс, который в настоящее время не существует; правило ничему не будет соответствовать до тех пор, пока не появится интерфейс.
Это чрезвычайно полезно для коммутируемой PPP связи (обычно интерфейс ppp0) и т.п..
Как специальный случай, имя интерфейса, оканчивающееся на "+"будет соответствовать всем интерфейсам (существуют ли они в настоящее время или нет), которые начинаются с той же последовательности символов. Например, чтобы определить правило, которое соответствует всем интерфейсам PPP, использовалось бы опция i ppp+'.
Имени интерфейса может предшествовать "!"для соответствия пакету, который не соответствует определенному интерфейсу(ам).

Указание только пакетов TCP SYN Иногда полезно разрешить TCP соединения в одном направлении. Например, вы могли бы захотеть позволить соединение с внешним WWW сервером, но не соединения от этого сервера.
Наивно было бы блокировать пакеты TCP, исходящие от сервера. К сожалению, TCP соединения для своей работы требуют, чтобы пакеты ходили в обоих направлениях.
Решение состоит в блокировании только пакетов, используемых для запроса соединения. Эти пакеты называются SYN пакетами (ok, технически это пакеты с установленным флажком SYN, и опущенными флажками ACK и FIN, но мы называем их SYN пакетами). Отвергая только эти пакеты, мы можем прервать попытки инициировать соединение.
Для этого используется флажок y": он допустим только для правил, которые относятся к TCP протоколу. Например, вот как указываются попытки TCP соединения от 192.168.1.1:

```
-p TCP -s 192.168.1.1 -y
```

Аналогично, этот флажок может быть инвертирован, с помощью "!", что означает все пакеты, кроме пакетов инициирования соединения.

Обработка фрагментов Иногда пакет слишком большой, чтобы пересылаться одним куском. Когда это случается, пакет делится на фрагменты, и посылается как несколько пакетов. Другой конец собирает несколько фрагментов в один целый пакет.
Проблема с фрагментами состоит в том, что некоторые из спецификаций, перечисленных выше (в частности порта источника, порта назначения, тип ICMP, код ICMP, или SYN флажок TCP) требуют, чтобы ядро анализировало начало пакета, которое содержится только в первом фрагменте.
Если ваша машина - единственое соединение со внешней сетью, то вы можете сообщить Linux ядру, что надо собирать все фрагменты, которые проходят через него, компилируя ядро с установкой `IP: always defragment "Y"`. Это аккуратное решение проблемы.
В любом случае, важно понять, как фрагменты обрабатываются правилами фильтрации. Любое правило фильтрации, которое запрашивает информацию, не будет срабатывать, если не будет соответствия. Это означает, что первый фрагмент обработается подобно любому другому пакету. Второй и далее фрагментя не будут обрабатываться. Таким образом правило `p TCP -s 192.168.1.1 www'` (указание исходного порта "www") никогда не будет соответствовать фрагменту (не первому фрагменту). Обратное правило `p TCP -s 192.168.1.1 ! www'` тоже не сработает.
Однако, вы можете определить правило специально для второго и далее фрагментов, используя флажок f". Ясно-понятно, что его нельзя применять при указании TCP или UDP порта, типа ICMP, кода ICMP или TCP SYN, так как эта информация во вторых и последующих фрагментах отсутствует.
Можно также указать, что правило не применяется ко второму и последующим фрагментам, указав '!' перед '-f'.
Обычно это считается безопасным для получения второго и дальнейших фрагментов, так как фильтр отбросит при необходимости первый фрагмент, и таким образом все остальные фрагменты, однако, сейчас стало известно об ошибках, которые могут повесить машину простой посылкой фрагментов. Учтите это.
Обратите внимание: пакеты некорректного формата (TCP, UDP и ICMP пакеты, слишком короткие для обработки firewalling кодом, настолько короткие, что из них нельзя получить информацию о портах или коде и типе ICMP) тоже обрабатываются как фрагменты. Только TCP фрагменты, начинающиеся с позиции 8 явно отбрасываются firewall кодом (в syslog отправляется соответствующее сообщение).
Например, следующее правило отбросит любые фрагменты, приходящие на 192.168.1.1:

```
# ipchains -A output -f -d 192.168.1.1 -j DENY
#
```

4.1.4 Фильтрация побочных эффектов

Отлично, теперь мы знаем все способы, которыми мы можем определять соответствия, используемые в правилах, для пакетов.
Если пакет соответствует правилу, происходят следующие вещи:

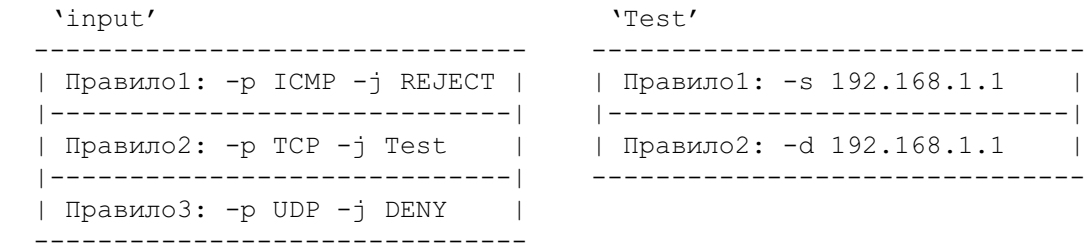
1. Счетчик байтов для того правила увеличивается на размер пакета (заголовок и все остальное).
2. Счетчик пакетов для того правила увеличивается.
3. Если правило предусматривает это, то пакет рагистрируется в журнале.
4. Если правило предусматривает это, изменяется поле Type Of Service.
5. Если правило предусматривает это, пакет помечается (не для ядер 2.0)
6. Анализируется, какое действие над пакетом указывается в правиле, чтобы решить дальнейшую судьбу пакета.

Для разнообразия, я буду описывать их в порядке важности.

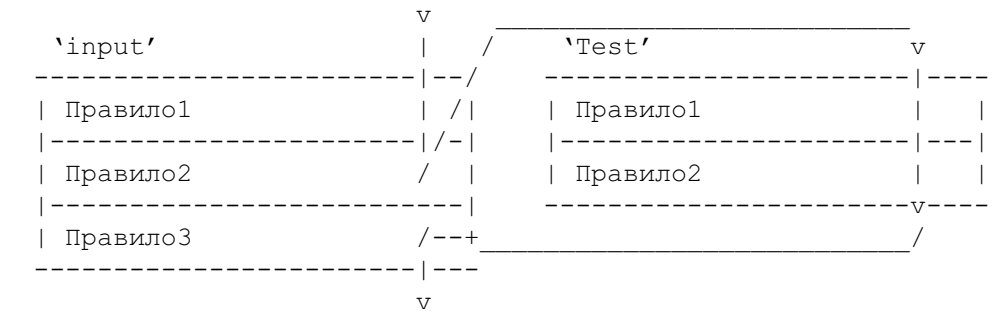
Указание действия Описание действия сообщает ядру, что делать с пакетом, который соответствует правилу. Ipchains для определения действия использует флаг j"(наверное "jump-to"). Выходное имя должно быть менее 8 символов длиной. Имена регистрозависимы, варианты "RETURN"и "return"совершенно различны.
Самый простой случай - когда действие не определено. Этот тип правила (часто называемый правилом "учета") полезен для просто подсчета пакетов некоторых типов. Соответствуют ли эти правила или нет, ядро просто переходит к следующему правилу в цепочке. Например, чтобы подсчитать число пакетов от 192.168.1.1, мы могли бы сделать так:

```
# ipchains -A input -s 192.168.1.1
#
```

(Используя "ipchains -L -v' мы можем увидеть счетчики байтов и пакетов, связанные с каждым правилом).
Есть шесть специальных действий. Первые три, ACCEPT, REJECT и DENY довольно просты. ACCEPT позволяет пропустить пакет. DENY отбрасывает пакет, как будто он никогда не приходил. REJECT тоже отбрасывает пакет, но (если это не ICMP пакет) генерирует ICMP-ответ источнику, сообщая, что адресат недостижим.
Следующий, MASQ, говорит ядру, что надо подменить пакет. Для этого нужно, чтобы ваше ядро скомпилировалось с разрешенным IP маскарадингом. Подробности см. в Masquerading-HOWTO и приложении “Различия между ipchains и ipfwadm". Это действие допустимо только для пакетов, проходящих через цепочку forward.
Еще одно важное действие - REDIRECT, которое сообщает, что пакет надо переадресовать на локальный порт машины вместо того места, куда он действительно направляется.
Это может быть определено только для правил, с указанными TCP или UDP протоколами. Необязательно можно определить порт (имя или номер) после j REDIRECT", что заставит ядро перенаправить пакет на этот порт, даже если пакет был адресован другому порту.
Это действие допустимо только для пакетов, проходящих через цепочку forward.
Последнее действие - RETURN приводит к немедленному прекращению дальнейших проверок в цепочке. (См. ниже “Установка политики”).
Любое другое действие определяет пользовательскую цепочку (как описано в “Обработка всей цепочки"ниже). Пакет начнет просматриваться правилами в этой цепочке. Если эта цепочка не решает судьбу пакета, то, как только обход этой цепочке закончится, продолжится проверка правил со следующего правила текущей цепочки.
Подошло время для еще одной ASCII-картинки. Рассмотрите две (глупых) цепочки: input (встроенная цепочка) и Test (пользовательская цепочка).



Рассмотрим TCP пакет идущий из 192.168.1.1 на 1.2.3.4. Он поступает на цепочку input и не соответствует правилу Правило1.
Правило2 подходит, и его действие - Test, так что проверка следующего правило начнется с первого правила Test. Правило1 в Test соответствует, но действие в ней не указано, поэтому осуществляется переход к Правило2. Оно не соответствует, так что мы достигли конца цепочки. Мы возвращаемся к цепочке input, где мы дошли только до Правило2, так что мы теперь проверяем Правило3, который тоже не соответствует.
Так что путь пакета:



В разделе “Как организовать ваши Firewall правила"описано как сохдавать ваши собственные пользовательские цепочки.

Регистрация пакетов Это полезный побочный эффект; вы можете отметить в журнале соответствующий условию правила пакет, используя флажок l". Для стандартных пакетов это обычно не нужн, но это весьма полезная возможность в тех случаях, когда вы хотите отловить какие-то особые события.
Ядро записывает в журнал записи типа:

```
Packet log: input DENY eth0 PROTO=17 192.168.2.1:53 192.168.1.1:1025
  L=34 S=0x00 I=18 F=0x0000 T=254
```

Эта журнальная запись создается с таким расчетом, чтобы быть краткой, но и содержать техническую информацию, полезную только для гуру-сетевиков, но она может быть полезной и для остальной части пользователей. Журнальная запись расшифровывается так:

1. ‘input' - цепочка, которая содержала правило, чье соответствие пакету вызывает запись сообщения в журнал.
2. ‘DENY' - то, что правило будет делать с пакетом. Если это – “-", то правило не воздействует на пакет (учетное правило).
3. "eth0 имя интерфейса. Поскольку это была цепочка input, то значит пакет пришел на "eth0".
4. ‘PROTO=17' означает, что это пакет протокола номер 17. Список номеров протоколов задан в "`/etc/protocols`'. Наиболее часто используемые протоколы 1 (ICMP), 6 (TCP) и 17 (UDP).
5. "192.168.2.1 это IP адрес, с которого пришел пакет
6. ":53"означает, что порт источника имеет номер 53. В "/etc/services' это описано как порт ‘domain' (то есть это - вероятно ответ DNS). Для UDP и TCP, этот номер порта источника. Для ICMP, это - тип ICMP. Для других он будет 65535.
7. "192.168.1.1 IP адрес назначения.
8. ":1025"означает, что порт назначения имеет номер 1025. Для UDP и TCP этот номер - порт назначения. Для ICMP это - код ICMP. Для других он будет 65535.
9. "L=34' означает, что пакет был общей длиной 34 байта.
10. "S=0x00' означает поле Type of Service (делится 4, чтобы получить Type of Service как используется ipchains).
11. ‘I=18' идентификатор IP.
12. "F=0x0000' является 16-разрядным смещением фрагмента плюс флажки. Значение, начинающееся с "0x4"или "0x5", означает, что установлен бит Don't Fragment. "0x2"или "0x3"означает установку бита ‘More Fragments'; после этого ожидается прибытие следующих фрагментов. Остальная часть числа - смещение этого фрагмента, деленное на 8.

13. "T=254' является временем жизни (Time To Live) пакета. При каждом переходе от маршрутизатора к маршрутизатору это значение уменьшается, и оно обычно начинается с 15 или 255.

14. '(#5)' может вставляться на более современных ядрах (возможно после 2.2.9). Это - номер правила, которое зарегистрировало пакет в журнале.

На стандартной Linux системе, этот вывод ядра фиксируется klogd (kernel logging daemon) который поручает это syslogd (system logging daemon). "`/etc/syslog.conf`' управляет поведением syslogd, определяя адресата для каждого "facility"(в нашем случае, facility - "ядро") и "уровень"(для ipchains, используемый уровень - "info").
Например, мой (Debian) `/etc/syslog.conf` содержит две строки, которые соответствуют "kern.info":

```
kern.*                          -/var/log/kern.log
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none;\
        mail,news.none          -/var/log/messages
```

Этот означает, что сообщения дублируются в "`/var/log/kern.log`' и "`/var/log/messages`'. Подробности см. в "`man syslog.conf`'.

Управление Type Of Service В IP заголовке имеются четыре редко-используемых бита, называемых Type of Service (TOS) битами. Они управляют обработкой пакетов; четыре бита - "Минимальная задержка", "Максимальная производительность", "Максимальная надежность"и "Минимальная стоимость". Установлен может быть только один из этих битов. Rob van Nieuwkerk, автор TOS-кода, разъясняет их смысл следующим образом:

> "Минимальная задержка"кажется мне наиболее важной установкой. Я включаю ее для "интерактивных"пакетов на моем upstream маршрутизаторе (Linux). Передо мною находится 33kб модемная связь. Linux располагает по приоритетам пакеты из 3 очередей. Таким образом я получаю приемлемую интерактивную эффективность при одновременном выполнении объемных загрузок (Если бы не было такой большой очереди в драйвере последовательной линии, то результат был бы даже лучше, но и сейчас время ожидания составляет менее 1.5 секунды).

Обратите внимание: вы в явном виде не имеете никакого контроля над входящими пакетами; вы можете только управлять приоритетами пакетов, уходящих с вашей машины. Чтобы договориться о приоритетах с другим концом линии, должен использоваться протокол типа RSVP (о котором я не знаю ничего, так что меня не спрашивайте).
В наиболее общем случае надо установить для управляющих соединений telnet и ftp "Минимальную задержку", а для данных FTP - "Максимальную производительность". Это можно выполнить следующим образом:

```
ipchains -A output -p tcp -d 0.0.0.0/0 telnet   -t 0x01 0x10
ipchains -A output -p tcp -d 0.0.0.0/0 ftp      -t 0x01 0x10
ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

Флажок t"берет два дополнительных параметра в hex-виде. С их помощью можно производить сложные установки битов TOS: первая маска - операция AND над текущим значением TOS пакета, а вторая маска - XOR над тем же самым значением.
Если вам это все равно не понятно, используйте следующую таблицу:

```
TOS имя                         Значение        Типичное применение
```

```
Минимальная задержка              0x01 0x10       ftp, telnet
Максимальная производительность   0x01 0x08       ftp-data
Максимальная надежность           0x01 0x04       snmp
Минимальная стоимость             0x01 0x02       nntp
```

Andi Kleen указывает на следующее (немного подредактировано):

> Возможно было бы полезно добавить ссылку на txqueuelen параметр ifconfig для обсуждения битов TOS. Заданная по умолчанию длина очереди устройства настроена для ethernet-карточек, для модемов она является слишком длинной и в результате 3-полосный планировщик (на котором основаны очереди TOS) работает неоптимально. Было бы лучше установить это значение между 4-10 для модема или одного B-канала ISDN связи: на коммутируемых устройствах необходима более длинная очередь.

Это проблема в ядрах 2.0 и 2.1, но в 2.1 есть флажок ifconfig (в недавних nettools), в то время как ядро 2.0 требуется патчить.
Итак, чтобы увидеть максимальную пользу от манипулирования TOS для модемной PPP связи, внесите строку "ifconfig $1 txqueuelen' в ваш сценарий /etc/ppp/ip-up. Числовой параметр зависит от скорости модема и объема буфера модема; далее опять говорит Andi:

> Наилучшее значение для данной конфигурации требуется подобрать экспериментально. Если очереди на маршрутизаторе слишком короткие, то пакеты станут пропадать. Также конечно польза будет даже без перезаписи TOS, перезапись TOS может вам помочь для некооперативнях программ. (Но все стандартные программы linux кооперативные).

Маркировка пакета Это позволяет осуществлять сложные и мощные взаимодействия с новой реализацией Качества Сервиса (Quality of Service) Алексея Кузнецова, для основанного на метках форвардинга в ядрах серии 2.1. Более пока ничего не известно. В ядрах 2.0 эта опция игнорируется.

Обработка всей цепочки Очень полезная возможность ipchains - способность группировать связанные правила в цепочки. Вы можете нызывать цепочки как вам хочется при условии, что имена не совпадают с именами встроенных цепочек (input, output и forward) или действий (MASQ, REDIRECT, ACCEPT, DENY, REJECT или RETURN).
Я предлагаю вам избегать полностью использования верхнего регистра меток, так как я могу использовать их в будущих расширениях. Имя цепочки может быть длиной до 8 символов.

Создание новой цепочки Давайте создадим новую цепочку. Я назову ее test – вот такое у меня богатое воображение!

```
# ipchains -N test
#
```

Это просто пример. Теперь вы можете помещать правила в нее как было описано выше.

Удаление цепочки Удаление цепочки также просто.

```
# ipchains -X test
#
```

Почему X"? Ну, это тоже хорошая буква.
Есть пара ограничений на удаление цепочки: они должны быть пустыми (см. "Очистка цепочки"ниже) и они не должны быть действием любого правила. Вы не можете удалить ни одну из трех встроенных цепочек.

Очистка цепочки Есть простой способ освобождения всех правил в цепочке использованием команды F".

```
# ipchains -F forward
#
```

Если вы не указали цепочку, то будут очищены все цепочки.

Просмотр цепочки Вы можете просмотреть все правила в цепочке, используя команду L".

```
# ipchains -L input
Chain input (refcnt = 1): (policy ACCEPT)
target     prot opt    source               destination          ports
ACCEPT     icmp -----  anywhere             anywhere             any
# ipchains -L test
Chain test (refcnt = 0):
target     prot opt    source               destination          ports
DENY       icmp -----  localnet/24          anywhere             any
#
```

"refcnt"в test - это число правил, в которых test назначен как действие. Это значение должно быть нулем (а цепочка пустой) перед удалением цепочки.
Если имя цепочки опущено, распечатываются все цепочки, даже пустые.
Есть три опции, которые могут сопровождать опцию L". n"(numeric) опция очень полезна, поскольку не дает ipchains пытаться переводить адреса IP в доменные имена, что (если вы, как и многие, используете DNS) приведет к большим задержкам, если ваш DNS неправильно установлен, или вы фильтруете наружние DNS запросы. Также эта опция указывает, что порты должны быть распечатаны как числа, а не как имена.
Опция v"показывают вам все подробности правил, такие как пакетные и байтовые счетчики, маски TOS, интерфейс и метку пакета. Иначе эти значения будут опущены. Например:

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target     prot opt   tosa tosx  ifname    mark        source
   10   840 ACCEPT     icmp ----- 0xFF 0x00  lo                    anywhere
```

Обратите внимание, что пакетные и байтовые счетчики распечатаны, с использованием суффиксов "K", "M"или "G"для 1000, 1,000,000 и 1,000,000,000 соответственно. Использование флажка x"(expand numbers) также распечатывает числа полностью, независимо от их значений.

Сброс (обнуление) счетчиков Полезно иметь возможность сбросить счетчики. Это делается опцией Z". Например:

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target     prot opt   tosa tosx  ifname    mark        source
   10   840 ACCEPT     icmp ----- 0xFF 0x00  lo                    anywhere
# ipchains -Z input
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target     prot opt   tosa tosx  ifname    mark        source
    0     0 ACCEPT     icmp ----- 0xFF 0x00  lo                    anywhere
#
```

Проблема с этим подходом в том, что иногда вы должны знать значения счетчика до того, как они сброшены. В вышеупомянутом примере, некоторые пакеты могли бы пройти во время выполнения команд L"и Z". По этой причине вы можете использовать L"и Z"вместе, сбрасывать счетчики при их просмотре. К сожалению, если вы делаете это, вы не можете обрабатывать одну цепочку: вы просматриваете и обнуляете все цепочки сразу.

```
# ipchains -L -v -Z
Chain input (policy ACCEPT):
 pkts bytes target     prot opt   tosa tosx  ifname    mark        source
   10   840 ACCEPT     icmp ----- 0xFF 0x00  lo                    anywhere

Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
    0     0 DENY       icmp ----- 0xFF 0x00  ppp0                  localnet/24
# ipchains -L -v
Chain input (policy ACCEPT):
 pkts bytes target     prot opt   tosa tosx  ifname    mark        source
   10   840 ACCEPT     icmp ----- 0xFF 0x00  lo                    anywhere

Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
    0     0 DENY       icmp ----- 0xFF 0x00  ppp0                  localnet/24
#
```

Установка политики Мы упоминали о том, что случается, когда пакет доходит до конца встроенной цепочки, когда обсуждали, как пакет идет по цепочкам в “Указание действия"выше. В этом случае, судьбу пакета определяет политика цепочки. Только встроенные цепочки (input, output и forward) имеют политику, потому что, если пакет достигает конца пользовательской цепочки, обход возвращается к предыдущей цепочке.
Политика может быть любой из первых четырех специальных действий: ACCEPT, DENY, REJECT или MASQ. MASQ допустим только для цепочки "forward".
Также важно обратить внимание, что действие RETURN в правиле в одной из встроенных цепочек полезно для явного назначения политики цепочки, когда пакет соответствует правилу.

4.1.5 Операции по маскарадингу

Есть несколько параметров, которые вы можете применить для IP Маскарадинга. Они связаны с ipchains, потому что отдельный инструмента для них не написан (хотя это изменится в будущем).
Команда IP Masquerading - M", и она может быть объединена с L", чтобы отобразить текущие masqueraded соединения, или с S', чтобы установить параметры маскарадинга.
Команда L"может сопровождаться n"(чтобы показать числа вместо имен хостов и портов) или -v"(показывать deltas в порядковых числах для masqueraded соединений, если вас это заботит).
Команда S' должна сопровождаться тремя значениями таймаутов, в секундах: для TCP сеансов, для TCP сеансов после пакета FIN и для UDP пакетов. Если вы не хотите изменять одно из этих значений, просто дайте значение "0'.
Значения по умолчанию перечислены в "`/usr/src/linux/include/net/ip_masq.h`', в настоящее время это 15 минут, 2 минуты и 5 минут соответственно.
Наиболее общее значение для изменения - первое, для FTP (см. “Кошмары FTP"ниже).
Обратите внимание на проблемы с установкой таймаутов, перечисленные в “Я не могу установить таймауты для маскарадинга!".

4.1.6 Проверка пакета

Иногда вы хотите знать, что происходит, когда некоторый пакет входит на вашу машину, напр., для отладки ваших firewall цепочек. В ipchains есть команда C", которая использует те же самые подпрограммы, которые ядро использует для диагностики реальных пакетов.
Вы указываете, какой цепочке проверить пакет, указывая имя после параметра C". В то время как ядро всегда начинает с цепочек input, output или forward, вы можете начать тестирование с любой цепочки.
Подробности "пакета"определяются по тому же самому синтаксису, который используется для определения правил firewall. В частности протокол (p"), исходный адрес (s′), адрес назначения (d′) и интерфейс (i′) обязательны. Если протокол - TCP или UDP, то должны быть указаны одиночные порты источника и адресата, а для ICMP протокола должны быть определены тип и код ICMP (если не определен флажок f′ для указания правил фрагментов, в этом случае опции запрещены).
Если протокол - TCP (и флажок f"не определен), можно определять флажок y", чтобы указать, что тестовый пакет должен иметь установленным бит SYN.
Вот пример тестировочного TCP SYN пакета с хоста 192.168.1.1 порт 60000 к хосту 192.168.1.2 порт www, приходящий на интерфейс eth0 в цепочку "input". (Это - классическая инициация входящего WWW соединения):

```
# ipchains -C input -p tcp -y -i eth0 -s 192.168.1.1 60000 -d 192.168.1.2 www
packet accepted
#
```

4.1.7 Несколько правил одновременно и наблюдение за происходящим

Иногда одна командная строка может воздействовать на несколько правил сразу. Это происходит по двум причинам. Во-первых, если вы определяете имя хоста, которое преобразуется (используя DNS) в несколько IP адресов, ipchains будет действовать так, как будто вы ввели несколько команд для каждой комбинации адресов.
Так, если имя хоста "www.foo.com"указывает на три IP адреса, а имя хоста "www.bar.com на два IP адреса, то команда "`ipchains -A input -j REJECT -s www.bar.com -d www.foo.com`′ добавила бы 6 правил ко цепочке input.
Другая причина, заставляющая ipchains выполнить несколько действий, состоит в использовании флажка (“-b" – bidirectional). Этот флажок заставляет ipchains вести себя так, словно вы ввели команду дважды, поменяв во второй команде местами значения параметров “-s" и “-d". Так, чтобы запретить обмен с хостом 192.168.1.1, вы могли бы напечатать следующее:

```
# ipchains -b -A forward -j REJECT -s 192.168.1.1
#
```

Лично мне опция b"не слишком нравится; если вам хочется удобства, см. “Использование ipchains-save"ниже.
Опция b"может использоваться с командами вставки (I"), удаления (D") (но без использования номера правила), добавления (A") и проверки (C").
Другой полезный флажок v"(verbose), который выводит информацию о том, что ipchains делает с вашими командами. Это полезно, если вы имеете дело с командами, которые могут производить множественные правила. Например, здесь мы проверяем поведение фрагментов между 192.168.1.1 и 192.168.1.2.

```
# ipchains -v -b -C input -p tcp -f -s 192.168.1.1 -d 192.168.1.2 -i lo
  tcp opt   ---f- tos 0xFF 0x00  via lo    192.168.1.1  -> 192.168.1.2    * ->    *
packet accepted
  tcp opt   ---f- tos 0xFF 0x00  via lo    192.168.1.2  -> 192.168.1.1    * ->    *
packet accepted
#
```

4.2 Полезные примеры

У меня есть коммутируемое PPP соединение (`-i ppp0`). При каждой PPP-сессии я скачиваю новости (`-p TCP -s news.virtual.net.au nntp`) и почту (`-p TCP -s mail.virtual.net.au pop-3`). Для обновления пакетов Debian я регулярно использую FTP-метод (`-p TCP -y -s ftp.debian.org.au ftp-data`). По web я брожу через прокси моего ISP (`-p TCP -d proxy.virtual.net.au 8080`), но ненавижу `doubleclick.net` на Dilbert Archive (`-p TCP -y -d 199.95.207.0/24` и `-p TCP -y -d 199.95.208.0/24`).
Меня не беспокоят люди, заходящие на мой ftp пока я в Сети (`-p TCP -d $LOCALIP ftp`), но мне не нравится, когда кто-то притворяется, что он из моей внутренней сети (`-s 192.168.1.0/24`). Это обычно называется IP spoofing (спуфингом), и для защиты от него в ядрах 2.1.x и выше существуют лучшие способы: см. "Как мне установить защиту от IP спуфинга?".
Эта установка довольно проста, потому что в моей внутренней сети в настоящее время нет никаких других машин.
Я не хочу, чтобы любой локальный процесс (то есть Netscape, lynx и т.д.) подсоединялся к `doubleclick.net`:

```
# ipchains -A output -d 199.95.207.0/24 -j REJECT
# ipchains -A output -d 199.95.208.0/24 -j REJECT
#
```

Теперь я хочу установить приоритеты для различных исходящих пакетах (для входящих пакетов это делать бессмысленно). Так как этих правил много, то имеет смысл поместить их всех в одну цепочку по имени ppp-out.

```
# ipchains -N ppp-out
# ipchains -A output -i ppp0 -j ppp-out
#
```

Минимальная задержка для web и telnet.

```
# ipchains -A ppp-out -p TCP -d proxy.virtual.net.au 8080 -t 0x01 0x10
# ipchains -A ppp-out -p TCP -d 0.0.0.0 telnet -t 0x01 0x10
#
```

Низкая цена для ftp-data, nntp, pop-3:

```
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 ftp-data -t 0x01 0x02
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 nntp -t 0x01 0x02
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 pop-3 -t 0x01 0x02
#
```

Имеется несколько ограничений на пакеты, приходящие на интерфейс ppp0: давайте создадим цепочку "ppp-in":

```
# ipchains -N ppp-in
# ipchains -A input -i ppp0 -j ppp-in
#
```

Теперь, никакие пакеты, приходящие на ppp0 не должны быть из сети 192.168.1.*, так что мы регистрируем их и отбрасываем:

```
# ipchains -A ppp-in -s 192.168.1.0/24 -l -j DENY
#
```

Я разрешаю входящие UDP пакеты для DNS (у меня стоит кэширующий сервер имен, который пересылает все запросы к 203.29.16.1, так что я ожидаю ответы DNS только от них), входящие пакеты ftp, и только ответные пакеты ftp-data (которые должны идти только на порту с номером больше 1023, и не на портах X11 в районе примерно 6000).

```
# ipchains -A ppp-in -p UDP -s 203.29.16.1 -d $LOCALIP dns -j ACCEPT
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 1024:5999 -j ACCEPT
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 6010: -j ACCEPT
# ipchains -A ppp-in -p TCP -d $LOCALIP ftp -j ACCEPT
#
```

В заключение, даем добро для передачи от локального хоста к локальному хосту:

```
# ipchains -A input -i lo -j ACCEPT
#
```

Теперь, моя заданная по умолчанию сполитика на цепочке input - DENY, так что все неподошедшее под условия правил цепочки выбрасывается:

```
# ipchains -P input DENY
#
```

ОБРАТИТЕ ВНИМАНИЕ: я не располагаю мои цепочки в этом порядке, поскольку в то время, пока я их настраиваю, могли прийти пакеты. Самая надежная политика - сперва назначить DENY, а затем вставлять правила. Конечно, если ваши правила требуют поиска DNS для преобразования имен, то вы можете столкнуться с проблемами.

4.2.1 Использование ipchains-save

После настройки firewall цепочек вам надо где-то сохранить их, чтобы эта настройка не потерялась при перезагрузке машины.
Итак, ipchains-save - скрипт, который читает вашу текущую настройку цепочек и сохраняет ее в файле. Я немного расскажу, что делает этот скрипт.
ipchains-save может сохранять одну или все цепочки (если имя цепочки не определено). В настоящее время единственая опция скрипта - v", которая печатает правила (на stderr) в том виде, как они записаны. поскольку они сохранены. Также сохраняется политика цепочек input, output и forward

```
# ipchains-save > my_firewall
Saving ‘input’.
Saving ‘output’.
Saving ‘forward’.
Saving ‘ppp-in’.
Saving ‘ppp-out’.
#
```

4.2.2 Использование ipchains-restore

ipchains-restore восстанавливает цепочки, сохраненные ipchains-save. У скрипта две опции: v", которая описывает каждое добавленное правило, и f"которая, вынуждает очистить пользовательские цепочки, если они существуют, как описано ниже.
Если пользовательская цепочка найдена в input, ipchains-restore проверяет, не существует ли уже цепочка. Если да, то вас спросят, нужно ли цепочки очистить (от всех правил) или не восстанавливать эту цепочку. Если вы определили f"в командной строке, то запроса не будет; цепочка будет очищена.
Например:

```
# ipchains-restore < my_firewall
Restoring ‘input’.
Restoring ‘output’.
Restoring ‘forward’.
Restoring ‘ppp-in’.
```

```
Chain ‘ppp-in’ already exists. Skip or flush? [S/f]? s Skipping ‘ppp-in’.
Restoring ‘ppp-out’.
Chain ‘ppp-out’ already exists. Skip or flush? [S/f]? f Flushing ‘ppp-out’.
#
```

5 Разное.

Этот раздел содержит всю информацию и FAQи, который я не мог вставить в приведенную выше структуру.

5.1 Как организовать ваши Firewall правила

Этот вопрос требует некоторых размышлений. Вы можете попробовать организовать их, чтобы оптимизировать быстродействие (минимизировать число правил-проверок для большинства пакетов) или увеличить управляемость.
Если у вас непостоянная связь, скажем связь PPP, то вы могли бы захотеть установить первое правило в цепочке input i ppp0 -j DENY' во время загрузки системы, тогда помещаем нечто такое в ваш скрипт ip-up:

```
# Создать цепочку ‘ppp-in’ заново.
ipchains-restore -f < ppp-in.firewall
# Заместить правило DENY на цепочку ppp-обработки.
ipchains -R input 1 -i ppp0 -j ppp-in
```

А в скрипт ip-down:

```
ipchains -R input 1 -i ppp0 -j DENY
```

5.2 Что не нужно отфильтровывать

Прежде, чем вы начинете фильтровать наружний трафик, вам надо знать несколько вещей.

5.2.1 ICMP пакеты

ICMP пакеты используются (помимо прочего) для индикации состояний других протоколов (типа TCP и UDP). Пакеты "destination-unreachable"в частности. Блокирование этих пакетов означает, что вы никогда не получите сообщений об ошибках "Host unreachable' или "No route to host'; любые соединения будут только ожидать ответ, который никогда не придет. Это раздражает, но несмертельно.
Более коварная проблема - использование ICMP пакетов в проверках MTU. Все хорошие TCP реализации (включая Linux) используют проверки MTU, чтобы выяснить максимальный размер нефрагментированного пакета, который может принять адресат (фрагментация, снижает эффективность, особенно, когда при потере некоторых фрагментов). Проверка MTU осуществляется посылкой пакетов с установленным битом "Don't Fragment". Если в ответ на эти пакеты приходит ICMP-ответ “Fragmentation needed but DF set” – то есть “необходима фрагментация, но установлен флаг DF". Это пакеты типа “destination unreachable", и если они не получены, локальный хост не будет уменьшать MTU, и эффективность будет крайне низкой или нулевой.
Также обратите внимание на ICMP сообщения перенаправления (тип 5); они могут использоваться для управления маршрутизацией (хотя хорошие IP стеки защищены), и считаются немного опасными.

5.2.2 TCP соединения с DNS (сервером имен)

Если вы попробуете заблокировать выходящие TCP соединения, не забудьте, что DNS не всегда использует UDP; если ответный пакет от сервера превышает 512 байтов, клиент использует TCP соединение (также на порту номер 53).
DNS в этом случае будет "в общем-то работать'; вы можете обнаружить странные длинные задержки и другие случайные DNS проблемы.
Если ваши DNS запросы всегда направляются на один и тот же самый внешний источник (либо непосредственно, с использованием строки nameserver в /etc/resolv.conf, либо с использованием forward в кеширующем сереверe имен), то вам нужно всего лишь позволить TCP соединения между портом domain на этом сервере имен и локальным портом domain (при использовании кэширующего сервера имен) или с портом с большим номером (>1023) при использовании /etc/resolv.conf.

5.2.3 Кошмары FTP

Классическая проблема при пакетной фильтрации - FTP. FTP имеет два режима; традиционный вызывается активным режимом и более современный называется пассивным режимом. Web-браузеры обычно по умолчанию работают в пассивном режиме, но консольные программы FTP обычно по умолчанию работают в активном режиме.
В активном режиме, когда удаленная сторона хочет послать файл (или даже результаты команд ls или dir), она пробует открыть TCP соединение с локальной машиной. Это означает, что вы не можете отфильтровывать эти TCP соединения без прерывания активного FTP.
Если у вас есть опция использования пассивного режима, то все прекрасно; пассивный режим создает соединения от клиента к серверу, даже для входящих данных. Вам рекомендуется разрешить только TCP соединения с номерами портов более 1024 и не 6000..6010 (используются для XWINDOWS).

5.3 Фильтрация Пинга Смерти (DeathPing)

Linux-машины теперь невосприимчивы к известному Пингу Смерти, который заключается в посылке чересчур большого ICMP пакета, который переполняет буфера TCP-стека на машине-приемнике и приводит к хаосу.
Если вы защищаете машины, которые могли бы быть уязвимы для "пинга смерти", то вы могжете просто фрагментировать блоки ICMP. Нормальный ICMP пакеты не настолько велики, чтобы требовать фрагментации, так что фрагментироваться будут только очень большие пакеты - такие как "пинги смерти". Я слышал (по непроверенным данным), что некоторые системы могут пострадать и от последних фрагментов больших пакетов ICMP, поэтому рекомендуется фрагментировать не только первый фрагмент.
Хотя все известные программы, использующие эту атаку, генерируют ICMP-пакеты, для той же цели можно использовать TCP или UDP-пакеты (или неизвестный протокол), так что эта защита может служить лишь как временная мера.

5.4 Фильтрация Teardrop и Bonk

Teardrop и Bonk - две атаки (главным образом против машин Windows NT Microsoft), которые полагаются на накладывающиеся фрагменты. Решается это дефрагментацией пакетов на маршрутизаторе, либо запретом приема фрагментированных пакетов на уязвимых машинах.

5.5 Фильтрация фрагментированных бомб

Говорят, в некоторых "менее надежных"TCP стеках имеются проблемы, возникающие при большом количестве фрагментов пакетов, когда на машину приходят не все фрагменты. В Linux нет этой проблемы. Вы можете отфильтровывать фрагменты (что может отразиться и на нормальных

пакетах) или скомпилировать ваше ядро с опцией "IP: always defragment "Y (только в том случае, если ваша машина Linux - единственный маршрут для этих пакетов).

5.6 Изменение Firewall правил

Бывают ситуации, когда надо на лету поменять правила файервола. По неосторожности, вы можете пропустить некоторые пакеты во время изменений правил. Упрощенный подход заключается в следующем:

```
# ipchains -I input 1 -j DENY
# ipchains -I output 1 -j DENY
# ipchains -I forward 1 -j DENY
... вносим изменения ...
# ipchains -D input 1
# ipchains -D output 1
# ipchains -D forward 1
#
```

На время внесения изменений пакеты будут отбрасываться.
Если ваши изменения касаются только одной цепочки, то можно создать новую цепочку с новыми правилами, и затем заменить (R") правило, которое указывает на старую цепочку, на то, которое указывает на новую цепочку; затем вы можете удалить старую цепочку. Эта замена произойдет атомарно.

5.7 Как установить защиту от IP спуфинга?

IP спуфинг - отправка пакетов с поддельным IP адресом источника. Так как фильтрация пакета принимает решения на основани адреса источника, то IP спуфинг используется, чтобы ввести пакетный фильтр в заблуждение.
Также он используется при комплексных атаках, с использованием SYN, Teardrop, "пинга смерти"и т.п.(если не знаете что это такое - не волнуйтесь), чтобы атакуемый хост не знал, что все это валится с одного адреса .
Самый лучший способ защититься от IP spoofing называется Source Address Verification (проверка адреса источника). Эта проверка выполняется программами маршрутизации, в не программами фильтрации. Поищите файл `/proc/sys/net/ipv4/conf/all/rp_filter`.
Если он существует, то можно включать Source Address Verification при загрузке. Чтобы сделать это, вставьте следующие строки в скрипты инициализации перед инициализацией сетевых интерфейсов:

```
# Наилучший способ: включить Source Address Verification и защитить
# от спуфинга все текущие и будущие интерфейсы.
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
  echo -n "Установка защиты от спуфинга... "
  for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
      echo 1 > $f
  done
  echo "готово."
else
  echo ПРОБЛЕМЫ ПРИ ПОПЫТКЕ ВКЛЮЧИТЬ ЗАЩИТУ ОТ СПУФИНГА.
  echo "Нажмите CONTROL-D для выхода в shell и продолжения системной загрузки."
  echo
  # Запуск однопользовательского shell на консоли
  /sbin/sulogin $CONSOLE
fi
```

Если вы не можете сделать это, то вы можете вручную вставлять правила для защитыь каждого интерфейса. Это требует знаний о каждом интерфейсе. Ядра 2.1 автоматически отклоняют пакеты, приходящие с адресов 127.* (зарезервированных для локального петлевого интерфейса, lo).
Например, скажем, мы имеем три интерфейса: eth0, eth1 и ppp0. Мы можем использовать ifconfig, чтобы узнать адреса и сетевые маски интерфейсов.
Допустим, что eth0 присоединен к сети 192.168.1.0 с сетевой маской 255.255.255.0, eth1 присоединен к сети 10.0.0.0 с сетевой маской 255.0.0.0, а ppp0 соединен с Internet (где разрешены любые адреса за исключением зарезервированных частных адресов IP), мы вставили бы следующие правила:

```
# ipchains -A input -i eth0 -s ! 192.168.1.0/255.255.255.0 -j DENY
# ipchains -A input -i ! eth0 -s 192.168.1.0/255.255.255.0 -j DENY
# ipchains -A input -i eth1 -s ! 10.0.0.0/255.0.0.0 -j DENY
# ipchains -A input -i ! eth1 -s 10.0.0.0/255.0.0.0 -j DENY
#
```

Такой подход не столь хорош как Source Address Verification, потому что если ваши сетевые настройки изменились, вы должны изменить и правила фильтра.
Для ядер 2.0 вам желательно защитить и петлевой интерфейс:

```
# ipchains -A input -i ! lo -s 127.0.0.0/255.0.0.0 -j DENY
#
```

5.8 Продвинутые проекты

Существует userspace библиотека, написанная мною, которая включена в исходный дистрибутив, называемая "libfw". Она использует возможность IP цепочек 1.3 и выше копировать пакет в userspace (используя опцию ядра IP_FIREWALL_NETLINK).
Значение метки может использоваться для определения параметров Quality of Service пакетов, или определения, на какой порт надо отфорвардить пакеты. Я никогда пользовался ни тем, ни другим, но если вы хотите написать об этом, пожалуйста, свяжитесь со мной.
Используя эту библиотеку могут быть выполнены в userspace вещи типа stateful проверки (я предпочитают термин динамический firewalling). Другие красивые идеи заключаются в управлении пакетами на основании пользователя, с помощью userspace daemon. Это должно быть довольно просто.

5.8.1 SPF: Stateful фильтрация пакетов

<ftp://ftp.interlinx.bc.ca/pub/spf> - это сайт SPF проекта Brian Murrell'a, который осуществляет трекинг соединения в userspace. Это значительно улучшает защиту низкопроизводительных сайтов.
В настоящее время документация довольно скупая, но имеется список почтовой рассылки, в котором Брайен ответил на некоторые вопросы:

> Я полагаю, это делает в точности то, что мне надо: установка временного
> "обратногоправила, чтобы позволить приходить пакетам как бы ответом на
> исходящий запрос. Да, именно это он делает. Большинство протоколов его понимают, больше "обратные"правила it gets right. Прямо сейчас он поддерживается для (говорю навскидку, заранее извиняюсь за ошибки или пропуски) FTP (и активного, и пассивного и входящего и исходящего), некоторых RealAudio, traceroute, ICMP и основного ICQ (входящего от ICQ сервера и прямых TCP соединений, однако вторичные прямые TCP соединения для операций типа передач файлов и т.д. пока нет)

> Это замена ipchains или добавление?

Это - добавление. ipchains - это средство для ограничения прохождения пакетов через Linux машину. SPF - драйвер, постоянно контролирующий трафик и сообщающий ipchains как изменять политику фильтрации для отображения изменений в шаблонах трафика.

5.8.2 Хак Michael Hasenstein'a для ftp-data

Michael Hasenstein из SuSE написал патч для ядра, который добавляет в ipchains трекинг ftp-соединений. Сейчас его можно найти на <http://www.csn.tu-chemnitz.de/~mha/patch.ftp-data-2.gz>

5.9 Будущие расширения

В ядрах 2.3 firewalling и NAT разрабатываются повторно. Планы и обсуждения можно найти в архиве netdev и списке рассылки ipchains-dev. Эти расширения должны прояснить много проблем применения (firewalling и маскарадинг не должны быть такими жесткими) и позволить создать гораздо более гибкий firewalling.

6 Общие проблемы

6.1 ipchains -L замирает!

Вы вероятно блокируете DNS; это будет приводить к таймаутам.
Пробуйте использовать опцию n"(numeric) для ipchains, который подавляет преобразование имен.

6.2 Маскарадинг/форвардинг не работают!

Удостоверьтесь, что форвардинг пакетов разрешен (в последних версиях ядра, он не блокируется по умолчанию, что означает, что пакеты не приходят в цепочку "forward"). Вы можете отменить этот (от root), введя команду

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
#
```

Если это у вас сработает, то вы можете поместть эту строку куда-нибудь в ваши загрузочные сценарии, чтобы она выполнялась при загрузке; firewalling должен устанавливаться прежде, чем выполнится эта команда.

6.3 -j REDIR не работает!

Вы должны разрешить форвардинг пакетов (см. выше) для того, чтобы переназначение работало; иначе код маршрутизации отбрасывает пакет. Так, если вы используете только перенаправление, и вообще не используете форвардинг, то вы должны знать об этом.
Обратите внимание, что REDIR (находящийся во цепочке input) не воздействует на соединения из локального процесса.

6.4 Уайлдкарты интерфейсов не работают!

В версиях ядра 2.1.102 и 2.1.103 имелась ошибка (и некоторых старых моих патчах), из-за которой не работали команды ipchains, в которых используется уайлдкарты интерфейсов (например, -i ppp +).
Это исправлено в последних ядрах, и в сети есть патч для ядра 2.0.34. Вы можете также установить его вручную на исходники ядра, заменив строку 63 или около того в include/linux/ip_fw.h:

```
#define IP_FW_F_MASK    0x002F  /* All possible flag bits mask   */
```

Здесь должно стоять “0x003F". Исправьте это, и пересоберите ядро.

6.5 TOS не работает!

Это было моей ошибкой: установка поля Type of Service фактически не устанавливала Type of Service в ядрах версий 2.1.102 - 2.1.111. Эта проблема была исправлена в 2.1.112.

6.6 ipautofw and ipportfw не работают!

Для 2.0.x это так; у меня нет времени на создание и поддержку гигантского патча для ipchains и ipautofw/ipportfw.
Для 2.1.x скачайте ipmasqadm Juan Ciarlante'a с <http://juanjox.linuxhq.com/> и используйте его точно так же, как вы использовали бы ipautofw или ipportfw, за исключением того, что вместо ipportfw вы печатаете ipmasqadm portfw, а вместо ipautofw печатаете ipmasqadm autofw.

6.7 xosview падает!!

Возьмите версию 1.6.0 или выше, которым для ядер 2.1.x не требуются никаких правил firewall. Эта ошибка автора, кажется, снова появилась в версии 1.6.1 (это не моя ошибка!).

6.8 Segmentation Fault при '-j REDIRECT'!

Это была ошибка в ipchains версии 1.3.3. Пожалуйста обновите.

6.9 Я не могу установить таймауты маскарадинга!

Это было так (для ядер 2.1.x) до 2.1.123. В 2.1.124 попытка установить таймаут для маскарадинга вызывает зависание ядра (изменить return на ret = в строке 1328 net/ipv4/ip_fw.c). В 2.1.125 все прекрасно работает.

6.10 Я хочу файерволлить IPX!

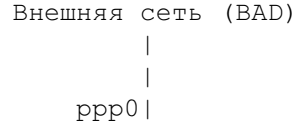
Этим, как мне кажется, занимаются другие. К сожалению, мой код относится только к IP. On the good side, all the hooks are there to firewall IPX! Вам только надо написать код; я счастлив буду помочь там, где возможно.

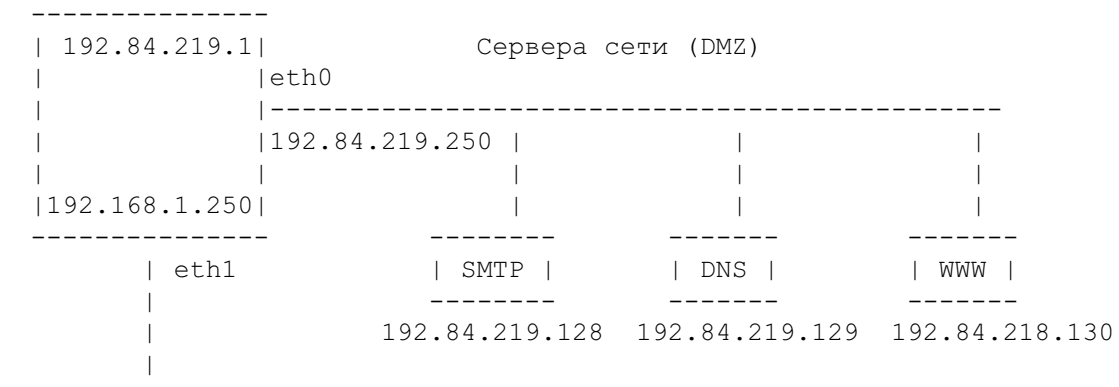
7 Серьезный пример.

Это пример от Michael Neuling и моего LinuxWorld Tutorial за Март 1999; это не единственый способ решить данную проблему, но вероятно самый простой. Я надеюсь, что вы найдете его информативным.

7.1 Соглашения

- Внутренняя сеть за маскарадом (различные операционные системы), которую мы называем "GOOD"("ХОРОШАЯ").
- Выставленные серверы в отдельной сети (называемой "DMZ" – Demilitarized Zone – нейтральная зона).
- PPP соединение с Internet (называемым "BAD" – "ПЛОХИМ").





7.2 Цели

- Машина пакетной фильтрации:

  PING любой сети
  Полезно для определения, действует ли машина.

  TRACEROUTE любой сети
  Аналогично, полезно для диагностики.

  DNS доступ
  Чтобы ping и DNS были полезнее.

- Внутри DMZ:
  - Почтовый сервер
    - SMTP ко внешним соединениям
    - Прием SMTP от внутренних и внешних соединений
    - Прием Pop-3 от внутренних соединений
  - Сервер имен
    - Отправка DNS во внешнюю сеть
    - Прием DNS от внутренней и внешней сети и машины пакетной фильтрации
  - Web сервер
    - Прием HTTP от внутренней и внешней сетей
    - Rsync доступ из внутренней сети

- Внутренняя сеть:

  Разрешенные WWW, ftp, traceroute, ssh доступы во внешнюю сеть.
  Это достаточно стандартные вещи, чтобы разрешить их использование: здесь мы ограничиваемся только этими сервисами, а не всеми доступными сервисами Интернет.

  Разрешить SMTP на почтовом сервере
  То есть позволить отправку почты наружу.

  Разрешить POP-3 на почтовом сервере
  То есть позволить пользователям читать их почту

  Разрешить DNS на серевере имен
  Это нужно для работы с внешними именами WWW, ftp, traceroute и ssh.

Разрешить rsync на веб сервер

Для синхронизации внешнего веб сервера со внутренним.

Разрешить WWW на веб сервере

Очевидно, что пользователи должны иметь доступ к своему внешнему веб серверу.

Разрешить ping на машине пакетной фильтрации

Это вежливость по отношению к пользователям. Они могут проверить, работает ли машина пакетной фильтрации (чтобы не наезжали на нас, если на самом деле не работает внешний сайт).

7.3 Перед фильтрацией пакетов

- Антиспуфинг

  Так как у нас не асимметричная маршрутизация, мы можем просто включить антиспуфинг для всех интерфейсов.

```
# for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 1 > $f; done
#
```

- Установка правил фильтрации на отбрасывание (DENY) всего: Запрещаем все, кроме loopback трафика.

```
# ipchains -A input -i ! lo -j DENY
# ipchains -A output -i ! lo -j DENY
# ipchains -A forward -j DENY
#
```

- Установка интерфейсов Это обычно делается в сценариях начальной загрузки. Удостоверьтесь, что вышеупомянутые шаги выполнены прежде, чем интерфейсы настроены, чтобы не пропустить пакеты до того, как будут настроены правила.

- Вставка модулей маскарадинга по-протоколам. Мы должны вставить модуль маскарадинга для FTP, так, чтобы активный и пассивный FTP "работал только' из внутренней сети.

```
# insmod ip_masq_ftp
#
```

7.4 Фильтрация проходящих пакетов

При маскарадинге самое лучшее – фильтр в цепочке forward.
Разбейте цепочку forward на несколько пользовательских цепочек в зависимости от интерфейсов источника/приемника; проблема разделяется на более простые в управлении части.

```
ipchains -N good-dmz
ipchains -N bad-dmz
ipchains -N good-bad
ipchains -N dmz-good
ipchains -N dmz-bad
ipchains -N bad-good
```

ACCEPT'ие стандартных ICMP сообщений об ошибках – достаточно общая вещь, поэтому создадим для нее цепочку.

```
ipchains -N icmp-acc
```

7.4.1 Установка переходов из цепочки forward

К сожалению, мы знаем (в цепочке forward) только исходящий интерфейс. Таким образом, чтобы выяснить с какого интерфейса пришел пакет, мы используем адрес источника (подделку этого адреса предотвращает антиспуфинг).
Обратите внимание, что в журнал регистрации пойдет все, что не подпадает ни под одно из этих правил (очевидно, что такого никогда не случится).

```
ipchains -A forward -s 192.168.1.0/24 -i eth0 -j good-dmz
ipchains -A forward -s 192.168.1.0/24 -i ppp0 -j good-bad
ipchains -A forward -s 192.84.219.0/24 -i ppp0 -j dmz-bad
ipchains -A forward -s 192.84.219.0/24 -i eth1 -j dmz-good
ipchains -A forward -i eth0 -j bad-dmz
ipchains -A forward -i eth1 -j bad-good
ipchains -A forward -j DENY -l
```

7.4.2 Определим icmp-acc цепочку

Пакеты, которые являются одним из ICMP сообщений об ошибке ACCEPT'ся, иначе управление перейдет обратно к вызывающей цепочке.

```
ipchains -A icmp-acc -p icmp --icmp-type destination-unreachable -j ACCEPT
ipchains -A icmp-acc -p icmp --icmp-type source-quench -j ACCEPT
ipchains -A icmp-acc -p icmp --icmp-type time-exceeded -j ACCEPT
ipchains -A icmp-acc -p icmp --icmp-type parameter-problem -j ACCEPT
```

7.4.3 От Good (внутренняя сеть) к DMZ (сервера)

Внутренние ограничения:

- Позволить WWW, ftp, traceroute, ssh к внешней сети
- Позволить SMTP к почтовому серверу
- Позволить POP-3 к почтовому серверу
- Позволить DNS к сереверу имен
- Позволить rsync к веб серверу
- Позволить WWW к веб серверу
- Позволить ping к машине пакетной фильтрации

Можно было бы сделать маскарадинг от внутренней сети к DMZ, но здесь мы это не делаем. Так как кто-то из внутренней сети может попробовать напакостить, мы регистрируем в журнале все отклоненные пакеты.

```
ipchains -A good-dmz -p tcp -d 192.84.219.128 smtp -j ACCEPT
ipchains -A good-dmz -p tcp -d 192.84.219.128 pop-3 -j ACCEPT
ipchains -A good-dmz -p udp -d 192.84.219.129 domain -j ACCEPT
ipchains -A good-dmz -p tcp -d 192.84.219.129 domain -j ACCEPT
ipchains -A good-dmz -p tcp -d 192.84.218.130 www -j ACCEPT
ipchains -A good-dmz -p tcp -d 192.84.218.130 rsync -j ACCEPT
ipchains -A good-dmz -p icmp -j icmp-acc
ipchains -A good-dmz -j DENY -l
```

7.4.4 От BAD (внешняя сеть) к DMZ (сервера).

- DMZ ограничения:
  - Почтовый сервер
    - SMTP во внешнюю сеть
    - Прием SMTP от внутренней и внешней сетей
    - Прием POP-3 от внутренней сети
  - Сервер имен
    - Посылка DNS во внешнюю сеть
    - Прием DNS от внутренней и внешней сетей и машины пакетной фильтрации
  - Web сервер
    - Прием HTTP от внутренней и внешней сети
    - Rsync доступ из внутренней сети
- Вещи, которые мы разрешаем от внешней сети к DMZ.
  - Не регистрировать нарушения, поскольку они могут случаться.

```
ipchains -A bad-dmz -p tcp -d 192.84.219.128 smtp -j ACCEPT
ipchains -A bad-dmz -p udp -d 192.84.219.129 domain -j ACCEPT
ipchains -A bad-dmz -p tcp -d 192.84.219.129 domain -j ACCEPT
ipchains -A bad-dmz -p tcp -d 192.84.218.130 www -j ACCEPT
ipchains -A bad-dmz -p icmp -j icmp-acc
ipchains -A bad-dmz -j DENY
```

7.4.5 От Good (внутренняя сеть) к Bad (внешняя сеть).

Внутренние ограничения:

- Позволить WWW, ftp, traceroute, ssh ко внешней сети
- Позволить SMTP к почтовому серверу
- Позволить POP-3 к почтовому серверу
- Позволить DNS к серверу имен
- Позволить rsync к веб серверу
- Позволить WWW к веб серверу
- Позволить ping к машине пакетной фильтрации
- Многие люди позволяют все из внутренней сети ко внешней сети, и затем добавляют ограничения. Мы – тираны.
- Регистрационные нарушения.
- Пассивный FTP, обработанный модулем masq.

```
ipchains -A good-bad -p tcp --dport www -j MASQ
ipchains -A good-bad -p tcp --dport ssh -j MASQ
ipchains -A good-bad -p udp --dport 33434:33500 -j MASQ
ipchains -A good-bad -p tcp --dport ftp --j MASQ
ipchains -A good-bad -p icmp --icmp-type ping -j MASQ
ipchains -A good-bad -j REJECT -l
```

7.4.6 От DMZ к Good (внутренняя сеть).

Внутренние ограничения:

- Позволить WWW, ftp, traceroute, ssh к внешней сети
- Позволить SMTP к почтовому серверу
- Позволить POP-3 к почтовому серверу
- Позволить DNS к серверу имен
- Позволить rsync к веб серверу
- Позволить WWW к веб серверу
- Позволить ping к машине пакетной фильтрации
- Многие люди позволяют доступ ко всем сервисам из внутренней сети ко внешней сети, и затем добавляют ограничения. Мы – бессовестные тираны.
- Регистрационные нарушения.
- Пассивный FTP, обработанный модулем masq.
- Если бы мы были замаскарадены от внутренней сети к DMZ, то просто отказывались бы от всех пакетов, приходящих другим путем. Сейчас позволяем только пакеты, которые могли бы быть часть установленного соединения.

```
ipchains -A dmz-good -p tcp ! -y -s 192.84.219.128 smtp -j ACCEPT
ipchains -A dmz-good -p udp -s 192.84.219.129 domain -j ACCEPT
ipchains -A dmz-good -p tcp ! -y -s 192.84.219.129 domain -j ACCEPT
ipchains -A dmz-good -p tcp ! -y -s 192.84.218.130 www -j ACCEPT
ipchains -A dmz-good -p tcp ! -y -s 192.84.218.130 rsync -j ACCEPT
ipchains -A dmz-good -p icmp -j icmp-acc
ipchains -A dmz-bad -j DENY -l
```

7.4.7 От DMZ к bad (внешняя сеть).

DMZ ограничения:

- Почтовый сервер
  - SMTP во внешнюю сеть
  - Прием SMTP от внутренней и внешней сетей
  - Прием POP-3 от внутренней сети
- Сервер имен
  - Посылка DNS во внешнюю сеть
  - Прием DNS от внутренней и внешней сетей и машины пакетной фильтрации
- Web сервер
  - Прием HTTP от внутренней и внешней сети
  - Rsync доступ из внутренней сети

```
ipchains -A dmz-bad -p tcp -s 192.84.219.128 smtp -j ACCEPT
ipchains -A dmz-bad -p udp -s 192.84.219.129 domain -j ACCEPT
ipchains -A dmz-bad -p tcp -s 192.84.219.129 domain -j ACCEPT
ipchains -A dmz-bad -p tcp ! -y -s 192.84.218.130 www -j ACCEPT
ipchains -A dmz-bad -p icmp -j icmp-acc
ipchains -A dmz-bad -j DENY -l
```

7.4.8 От Bad (внешняя сеть) к Good (внутренняя сеть).

Мы не позволяем ничего (не-маскарадное) от внешней сети к внутренней сети

```
ipchains -A bad-good -j REJECT
```

7.4.9 Фильтрация пакетов непосредственно для Linux машины

Если мы хотим использовать фильтрацию пакета на пакетах, приходящих непосредственно к linux машине, то мы должны настроить фильтрацию в цепочке input. Мы создаем одну цепочку для каждого интерфейса адресата:

```
ipchains -N bad-if
ipchains -N dmz-if
ipchains -N good-if
```

Создаем переходы на них:

```
ipchains -A input -d 192.84.219.1 -j bad-if
ipchains -A input -d 192.84.219.250 -j dmz-if
ipchains -A input -d 192.168.1.250 -j good-if
```

Интерфейс Bad (внешняя сеть). Машина пакетной фильтрации:

- PING любой сети
- TRACEROUTE любой сети
- Доступ к DNS
- Внешний интерфейс также получает ответы на маскараденные пакеты и ICMP сообщения об ошибках на них и ping ответы.

```
ipchains -A bad-if -i ! ppp0 -j DENY -l
ipchains -A bad-if -p TCP --dport 61000:65096 -j ACCEPT
ipchains -A bad-if -p UDP --dport 61000:65096 -j ACCEPT
ipchains -A bad-if -p ICMP --icmp-type pong -j ACCEPT
ipchains -A bad-if -j icmp-acc
ipchains -A bad-if -j DENY
```

Интерфейс DMZ. Ограничения машины пакетной фильтрации:

- PING любой сети
- TRACEROUTE любой сети
- Доступ к DNS
- DMZ интерфейс получает ответы DNS, ответы ping и ICMP сообщения об ошибках.

```
ipchains -A dmz-if -i ! eth0 -j DENY
ipchains -A dmz-if -p TCP ! -y -s 192.84.219.129 53 -j ACCEPT
ipchains -A dmz-if -p UDP -s 192.84.219.129 53 -j ACCEPT
ipchains -A dmz-if -p ICMP --icmp-type pong -j ACCEPT
ipchains -A dmz-if -j icmp-acc
ipchains -A dmz-if -j DENY -l
```

Интерфейс Good (внутренний).

- Ограничения машины пакетной фильтрации:
  - PING любой сети
  - TRACEROUTE любой сети
  - Доступ к DNS
  - DMZ интерфейс получает ответы DNS, ответы ping и ICMP сообщения об ошибках.
- Ограничения внутренней сети:
  - Разрешить WWW, ftp, traceroute, ssh ко внешней сети
  - Позволить SMTP к почтовому серверу
  - Позволить POP-3 к почтовому серверу
  - Позволить DNS к серверу имен
  - Позволить rsync к веб серверу
  - Позволить WWW к веб серверу
  - Позволить ping к машине пакетной фильтрации
  - Внутренний интерфейс получает ответы DNS, ответы ping и ICMP сообщения об ошибках.

```
ipchains -A good-if -i ! eth1 -j DENY
ipchains -A good-if -p ICMP --icmp-type ping -j ACCEPT
ipchains -A good-if -p ICMP --icmp-type pong -j ACCEPT
ipchains -A good-if -j icmp-acc
ipchains -A good-if -j DENY -l
```

7.5 В заключение

Удаление правил блокировки:

```
ipchains -D input 1
ipchains -D forward 1
ipchains -D output 1
```

8 Приложение. Различия между ipchains и ipfwadm.

Некоторые из этих изменений - результат изменений в ядре, и поэтому в ipchains есть отличия от ipfwadm.

1. Многие параметры были повторно переопределены: заглавные буквы теперь указывают команду, а строчные буквы теперь указывают опцию.
2. Поддерживаются произвольные цепочки, так даже встроенные цепочки обозначаются именами вместо флажков (напр. "input"вместо I").
3. k"опция исчезла: используйте "! -y'.
4. b"опция фактически вставляет/добавляет/удаляет два правила, скорее чем одно правило ‘bidirectional'.
5. b"опцию можно использовать с C", чтобы сделать две проверки (в каждом направлении).
6. x"опция в l"была заменена на v".

7. Больше не поддерживаются множественные порты источника и адресата. Однако можно воспользоваться возможностью отрицать диапазоны портов.
8. Интерфейсы могут быть определены только именем (не адресом). Старая семантика заменена в ядре 2.1.
9. Фрагменты проверяются, не разрешены автоматически.
10. Explicit accounting chains have been done away with.
11. Могут быть проверены произвольные протоколы над IP.
12. Изменено старое поведение соответствий SYN и ACK (который предварительно игнорировались для не-TCP пакетов); опция SYN не допустима для не-TCP-специфичных правил.
13. Счетчики на 32-разрядных машинах теперь 64-разрядные, а не 32-разрядные.
14. Поддерживаются обратные опции.
15. Поддерживаются ICMP коды.
16. Поддерживаются уайлдкарты интерфейсов.
17. Исправлены манипуляции с TOS: старый код ядра просто зависал при (неправильном) управлении битом 'Must Be Zero' TOS; теперь в этом и аналогичных случаях ipchains возвращает ошибку.

8.1 Краткая таблица перекрестных ссылок.

[В основном, аргументы команд в ВЕРХНЕМ РЕГИСТРЕ, а параметры опций - в нижнем]
Обратите внимание на такую вещь: маскарадинг определяется j MASQ'; это полностью отличается j ACCEPT', и не обрабатыватся как просто побочный эффект, как это делает ipfwadm.

ipfwadm	ipchains	Notes
-A [both]	-N acct	Создает 'acct' цепочку
	& -I 1 input -j acct	для входящих с исходящих
	& -I 1 output -j acct	пакетов.
	& acct	
-A in	input	Правило без действия
-A out	output	Правило без действия
-F	forward	Использ. это как [цепочка].
-I	input	Использ. это как [цепочка].
-O	output	Использ. это как [цепочка].
-M -l	-M -L	
-M -s	-M -S	
-a policy	-A [chain] -j POLICY	(но см. -r и -m).

-d policy	-D [chain] -j POLICY	(но см. -r и -m).
-i policy	-I 1 [chain] -j POLICY	(но см. -r и -m).
-l	-L	
-z	-Z	
-f	-F	
-p	-P	
-c	-C	
-P	-p	
-S	-s	Можно указывать только один порт, не несколько.
-D	-d	Можно указывать только один порт, не несколько.
-V	<нету>	Использ. -i [имя].
-W	-i	
-b	-b	Факт-ки создает 2 правила.
-e	-v	
-k	! -y	Не работает, если нет опции -p tcp.
-m	-j MASQ	
-n	-n	
-o	-l	
-r [redirpt]	-j REDIRECT [redirpt]	
-t	-t	
-v	-v	
-x	-x	
-y	-y	Не работает, если нет опции -p tcp.

8.2 Примеры транслируемых команд ipfwadm

```
Старая команда: ipfwadm -F -p deny
Новая команда: ipchains -P forward DENY

Старая команда: ipfwadm -F -a m -S 192.168.0.0/24 -D 0.0.0.0/0
Новая команда: ipchains -A forward -j MASQ -s 192.168.0.0/24 -d 0.0.0.0/0

Старая команда: ipfwadm -I -a accept -V 10.1.2.1 -S 10.0.0.0/8 -D 0.0.0.0/0
Новая команда: ipchains -A input -j ACCEPT -i eth0 -s 10.0.0.0/8 -d 0.0.0.0/0
```

(Обратите внимание, что интерфейсы адресами указать нельзя: используйте имя интерфейса. На этой машине, 10.1.2.1 соответствует eth0).

9 Приложение. Использование скрипта ipfwadm-wrapper.

Скрипт оболочки ipfwadm-wrapper предназначен для обратной совместимости с ipfwadm 2.3a.
Единственая возможность, которую он фактически не может обрабатать - опция V". Когда она используется, выдается предупреждение. Если также используется опция W", опция V"игнорируется. Иначе, сценарий попробует найти имя интерфейса, связанное с этим адресом, используя ifconfig. Если это не удастся (например для интерфейса, который не поднят), то враппер завершится с сообщением об ошибке.
Это предупреждение может быть подавлено или изменением V"на W", или направлением стандартного вывода скрипта в /dev/null.
Если вы обнаружили какие-то ошибки в этом скрипте или какие-то изменения между реальным ipfwadm и этим скриптом, пожалуйста сообщите об этом мне: пошлите EMail на *ipchains@rustcorp.com* с сабжектом "BUG-REPORT".
Пожалуйста, укажите вашу старую версию ipfwadm (ipfwadm -h), вашу версию ipchains (ipchains –version), версию скрипта ipfwadm-wrapper (ipfwadm-wrapper –version). Также пошлите вывод ipchainssave. Заранее спасибо.
Вы применяете ipchains со скриптом ipfwadm-wrapper на ваш собственный страх и риск.

10 Приложение. Благодарности.

Большое спасибо Michael Neuling, который написал первый отрывок о коде ipchains Приношу публичные извинения за отклонение его result-caching, которую позже выдвинул Алан Кокс, а я в конце концов начал выполнять, увидев ошибку в собственных действиях.
Благодарю Алана Кокса за его 24-часовую EMail техническую поддержку, и ободрения.
Благодарю всех авторов кода ipfw и ipfwadm, особенно Jos Vos. Standing on the shoulders of giants and all that... Это относится к Linus Torvalds, а также всем хакерам ядра и пользовательского ПО.
Спасибо старательным бета-тестерам и ловцам ошибок, особенно Jordan Mendelson, Shaw Carruthers, Kevin Moule, Dr. Liviu Daia, Helmut Adams, Franck Sicard, Kevin Littlejohn, Matt Kemner, John D. Hardin, Alexey Kuznetsov, Leos Bitto, Jim Kunzman, Gerard Gerritsen, Serge Sivkov, Andrew Burgess, Steve Schmidtke, Richard Offer, Bernhard Weisshuhn, Larry Auton, Ambrose Li, Pavel Krauz, Steve Chadsey, Francesco Potorti‘ и Alain Knaff.