Настройка ядра GNU/Linux для Серверов

rp_filter

— параметр который включает фильтр обратного пути, проще говоря активируется защита от подмены адресов (спуфинга).

Немного подробное описание rp_filter...
Включает/выключает reverse path filter (проверка обратного адреса хотя это слишком вольный перевод термина, но мне он кажется наиболее близким по смыслу. прим. перев.) для заданного интерфейса. Смысл этой переменной достаточно прост все что поступает к нам проходит проверку на соответствие исходящего адреса с нашей таблицей маршрутизации и такая проверка считается успешной, если принятый пакет предполагает передачу ответа через тот же самый интерфейс.

Если вы используете расширенную маршрутизацию тем или иным образом, то вам следует всерьез задуматься о выключении этой переменной, поскольку она может послужить причиной потери пакетов. Например, в случае, когда входящий трафик идет через один маршрутизатор, а исходящий через другой. Так, WEB-сервер, подключенный через один сетевой интерфейс к входному роутеру, а через другой к выходному (в случае, когда включен rp_filter), будет просто терять входящий трафик, поскольку обратный маршрут, в таблице маршрутизации, задан через другой интерфейс.

Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено). Однако в некоторых дистрибутивах по-умолчанию эта переменная включается в стартовых скриптах на этапе загрузки. Поэтому, если у вас эта переменная включена, а вам надо ее выключить просмотрите стартовые скрипты в каталоге rc.d. Более детальную информацию об этой переменной вы найдете в RFC 1812 — Requirements for IP Version 4 Routers на страницах 46-49 (секция 4.2.2.11), странице 55 (секция 4.3.2.7) и странице 90 (секция 5.3.3.3). Если вы всерьез занимаетесь проблемами маршрутизации, то вам определенно придется изучить этот документ.

По умолчанию он отключен:

```
cat /proc/sys/net/ipv4/conf/all/rp_filter
0
```

Рекомендуется включить в «строгий режим» проверки (значение 2 включает «свободный режим» проверки), причем включить его можно на всех интерфейсах:

```
sysctl -w net.ipv4.conf.all.rp_filter=1
```

Так проверку можно включить на определенном интерфейсе:

```
sysctl -w net.ipv4.conf.eth0.rp_filter=1
```

accept_source_route

— запрет маршрутизации от источников.

Немного подробное описание accept_source_route...
Маршрутизация от источника (source routing) позволяет отправителю определить путь, по которому пакет должен пройти по сети Internet, чтобы достигнуть пункта назначения.

Это очень удобно для изучения и отладки
работы сети, но нарушитель получает возможность подмены адресов компьютеров локальной сети. 0 означает, что маршрутизация отключена, 1 — наоборот.

По умолчанию эта опция отключена:

```
cat /proc/sys/net/ipv4/conf/all/accept_source_route
0
```

Но если она у вас почему-то включена — отключите, желательно на всех интерфейсах:

```
sysctl -w net.ipv4.conf.all.accept_source_route=0
sysctl -w net.ipv4.conf.lo.accept_source_route=0
sysctl -w net.ipv4.conf.eth0.accept_source_route=0
sysctl -w net.ipv4.conf.default.accept_source_route=0
```

accept_redirects, secure_redirects, send_redirects

— этими тремя параметрами мы запрещаем принимать и отправлять ICMP пакеты перенаправления. ICMP-перенаправления могут быть использованы злоумышленником для изменения таблиц маршрутизации.

Немного подробное описание... (eng)
accept_redirects — BOOLEAN
Accept Redirects.
Functional default: enabled if local forwarding is disabled.
disabled if local forwarding is enabled.

secure_redirects — BOOLEAN
Accept ICMP redirect messages only to gateways listed in the
interface’s current gateway list. Even if disabled, RFC1122 redirect
rules still apply.
Overridden by shared_media.
secure_redirects for the interface will be enabled if at least one of
conf/{all,interface}/secure_redirects is set to TRUE,
it will be disabled otherwise
default TRUE

send_redirects — BOOLEAN
Send redirects, if router.
send_redirects for the interface will be enabled if at least one of
conf/{all,interface}/send_redirects is set to TRUE,
it will be disabled otherwise
Default: TRUE

По умолчанию все эти параметры включены:

```
cat /proc/sys/net/ipv4/conf/all/accept_redirects
1
cat /proc/sys/net/ipv4/conf/all/secure_redirects
1
cat /proc/sys/net/ipv4/conf/all/send_redirects
1
```

Но, так как наш сервер не маршрутизатор, в них нет необходимости:

```
sysctl -w net.ipv4.conf.all.accept_redirects=0^C
sysctl -w net.ipv4.conf.all.secure_redirects=0^C
sysctl -w net.ipv4.conf.all.send_redirects=0
```

icmp_echo_ignore_broadcasts

— отключаем ответ на ICMP ECHO запросы, переданные широковещательными пакетами.

По умолчанию включено, т.е. broadcast icmp запросы приходить не будут:

```
cat /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
1
```

Так и рекомендуется отставить:

```
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

icmp_ignore_bogus_error_responses

— игнорируем ошибочные ICMP запросы.

По умолчанию:

```
cat /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
1
```

Так и рекомендуется отставить:

```
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1
```

icmp_echo_ignore_all

— отключаем ответ на ICMP запросы (сервер не будет пинговаться)

По умолчанию:

```
cat /proc/sys/net/ipv4/icmp_echo_ignore_all
0
```

На ваше усмотрение, можно отключить:

```
sysctl -w net.ipv4.icmp_echo_ignore_all=1
```

tcp_syncookies

— по умолчанию данный параметр обычно включен. Если количество SYN пакетов забивает всю очередь, включается механизм Syn cookies.

Как проверить, включен ли он у нас:

```
cat /proc/sys/net/ipv4/tcp_syncookies
1
```

Если выдает 1, то включен, 0 — значит отключен. Для отключения «на лету» достаточно воспользоваться следующей командой:

```
sysctl -w net.ipv4.tcp_syncookies=0
```

Естественно поставив в конце «1» — механизм будет снова включен.

Немного подробное описание tcp_syncookies...

Если параметр tcp_syncookies установлен (доступен только когда ядро собрано с CONFIG_SYNCOOKIES), тогда ядро обрабатывает SYN пакеты TCP в обычном режиме до тех пор, пока очередь не заполнится. После заполнения очереди включается механизм SYN cookies.

SYN cookies вообще не использует очередь SYN. Вместо этого ядро отвечает на каждый SYN пакет, как обычно SYN|ACK, но туда будет включено специально сгенерированное число на основе IP адресов и портов источника и получателя, а также времени посылки пакета. Атакующий никогда не получит эти пакеты, а поэтому и не ответит на них. При нормальном соединении, будет послан третий пакет, содержащий число, а сервер проверит был ли это ответ на SYN cookie и, если да, то разрешит соединение даже в том случае, если в очереди SYN нет соответствующей записи.

Включение механизма SYN cookies является очень простым способом борьбы против атаки SYN флудом. При этом немного больше загружается процессор из-за необходимости создавать и сверять cookie. Так как альтернативным решением является отклонять все запросы на соединение, SYN cookies являются хорошим выбором.

Таким образом, как описано выше, мы получаем неплохую защиту от syn флуда и терпим небольшую нагрузку на ЦП..

Но согласно описанию, включать генерацию syncookies на высоконагруженных серверах, для которых этот механизм срабатывает, при большом количестве легальных соединений, не следует. Если в логах есть предупреждения о SYN-флуде, при этом это вполне нормальные соединения, нужно настраивать другие параметры: tcp_max_syn_backlog, tcp_synack_retries, tcp_abort_on_overflow.

tcp_max_syn_backlog

— параметр который определяет максимальное число запоминаемых запросов на соединение, для которых не было получено подтверждения от подключающегося клиента (полуоткрытых соединений). По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_max_syn_backlog
512
```

Если на сервере возникают перегрузки, можно попытаться увеличить это значение, например до 4096:

```
sysctl -w net.ipv4.tcp_max_syn_backlog=4096
```

tcp_synack_retries

— время удержания «полуоткрытых» соединений.

Немного подробное описание tcp_synack_retries...
Целочисленное значение (1 байт) tcp_synack_retries определяет число попыток повтора передачи пакетов SYNACK для пассивных соединений TCP. Число попыток не должно превышать 255. Значение 5 соответствует приблизительно 180 секундам на выполнение попыток организации соединения.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_synack_retries
5
```

Это значение имеет смысл уменьшить, например до 1 (это будет 9 секунд):

```
sysctl -w net.ipv4.tcp_synack_retries=1
```

tcp_max_orphans

— определяет максимальное число «осиротевших» TCP пакетов.

Немного подробное описание tcp_max_orphans...
Целочисленное значение параметра tcp_max_orphans определяет максимальное число допустимых в системе сокетов TCP, не связанных каким-либо идентификатором пользовательского файла (user file handle).

При достижении порогового значения “осиротевшие” (orphan) соединения незамедлительно сбрасываются с выдачей предупреждения. Этот порог помогает предотвращать только простые атаки DoS. Не следует уменьшать пороговое значение (скорее увеличить его в соответствии с требованиями системы – например, после добавления памяти. Каждое orphan-соединение поглощает около 64 Кбайт не сбрасываемой на диск (unswappable) памяти.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_max_orphans
262144
```

Рекомендуется установить 65536, а далее увеличивать, по мере необходимости:

```
sysctl -w net.ipv4.tcp_max_orphans=65536
```

tcp_fin_timeout

— время ожидания приема FIN до полного закрытия сокета.

Немного подробное описание tcp_fin_timeout...
Целое число в файле tcp_fin_timeout определяет время сохранения сокета в состоянии FIN-WAIT-2 после его закрытия локальной стороной. Партнер может не закрыть это соединение никогда, поэтому следует закрыть его по своей инициативе по истечении тайм-аута. По умолчанию тайм-аут составляет 60 секунд.

В ядрах серии 2.2 обычно использовалось значение 180 секунд и вы можете сохранить это значение, но не следует забывать, что на загруженных WEB-серверах вы рискуете израсходовать много памяти на сохранение полуразорванных мертвых соединений. Сокеты в состоянии FIN-WAIT-2 менее опасны, нежели FIN-WAIT-1, поскольку поглощают не более 1,5 Кбайт памяти, но они могут существовать дольше.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_fin_timeout
60
```

Рекомендуется поменять на 10 секунд:

```
sysctl -w net.ipv4.tcp_fin_timeout=10
```

tcp_keepalive_time

— проверять TCP-соединения, с помощью которой можно убедиться в том что на той стороне легальная машина, так как она сразу ответит.

Немного подробное описание tcp_keepalive_time...
Переменная определяет как часто следует проверять соединение, если оно давно не используется. Значение переменной имеет смысл только для тех сокетов, которые были созданы с флагом SO_KEEPALIVE.

По умолчанию 2 часа:

```
cat /proc/sys/net/ipv4/tcp_keepalive_time
7200
```

Рекомендуется каждую минуту:

```
sysctl -w net.ipv4.tcp_keepalive_time=60
```

tcp_keepalive_intvl

— интервал подачи проб…

Немного подробное описание...
Целочисленная переменная tcp_keepalive_intvl определяет интервал передачи проб. Произведение tcp_keepalive_probes * tcp_keepalive_intvl определяет время, по истечении которого соединение будет разорвано при отсутствии откликов. По умолчанию установлен интервал 75 секунд, т.е., время разрыва соединения при отсутствии откликов составит приблизительно 11 минут.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_keepalive_intvl
75
```

Рекомендуется поставить:

```
sysctl -w net.ipv4.tcp_keepalive_intvl=15
```

tcp_keepalive_probes

— Количество проверок перед закрытием соединения.

Немного подробное описание tcp_keepalive_probes...
Целочисленная переменная tcp_keepalive_probes задает число передач проб keepalive, после которого соединение считается разорванным. По умолчанию передается 9 проб.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_keepalive_probes
9
```

Рекомендуется поставить:

```
sysctrl -w net.ipv4.tcp_keepalive_probes=5
```

netdev_max_backlog

— Параметр определяет максимальное количество пакетов в очереди на обработку, если интерфейс получает пакеты быстрее, чем ядро может их обработать.

По умолчанию:

```
cat /proc/sys/net/core/netdev_max_backlog
1000
```

Рекомендуется так и оставить:

```
sysctl -w net.core.netdev_max_backlog=1000
```

somaxconn

— Максимальное число открытых сокетов, ждущих соединения.

По у молчанию:

```
cat /proc/sys/net/core/somaxconn
1024
```

Рекомендуется установить значения в районе 15000-20000:

```
sysctl -w net.core.somaxconn=15000
```

tcp_mem

— векторная (минимум, режим нагрузки, максимум) переменная которая содержит общие настройки потребления памяти для протокола TCP

Немного подробное описание tcp_mem...
Векторная (минимум, режим нагрузки, максимум) переменная в файле tcp_mem содержит общие настройки потребления памяти для протокола TCP. Эта переменная измеряется в страницах (обычно 4Кб), а не байтах.
Минимум: пока общий размер памяти для структур протокола TCP менее этого количества страниц, операционная система ничего не делает.

Режим нагрузки: как только количество страниц памяти, выделенное для работы протокола TCP, достигает этого значения, активируется режим работы под нагрузкой, при котором операционная система старается ограничивать выделение памяти. Этот режим сохраняется до тех пор, пока потребление памяти опять не достигнет минимального уровня.
Максимум: максимальное количество страниц памяти, разрешенное для всех TCP сокетов.
Немного подробное описание tcp_mem... (еще одно)

В этой переменной задаются 3 значения, определяющие объем памяти, который может быть использован стеком TCP. Значения измеряются в страницах памяти. Размер одной страницы зависит от аппаратуры и конфигурации ядра. Для архитектуры i386 размер одной страницы составляет 4Кб, или 4096 байт. Некоторые, более новые аппаратные реализации, имеют размер страницы равный 16, 32 или даже 64 Кб. Все три значения по-умолчанию рассчитываются во время загрузки.

Первое число задает нижний порог. Ниже этого порога, стек TCP вообще никак не беспокоится об управлении памятью, используемой различными TCP сокетами. Когда объем используемой памяти достигает второго предела (числа), то TCP начинает более энергично расталкивать память, стремясь освободить ее как можно быстрее. Этот процесс продолжается до тех пор, пока объем используемой памяти не достигнет нижнего предела. И последнее число максимальный объем памяти, который может использоваться для нужд TCP.

Если используемый объем памяти достигнет этого порога, то TCP просто начинает терятьпакеты и соединения до тех пор, пока объем используемой памяти не уменьшится. Эта переменная может позволить несколько увеличить пропускную способность на толстых каналах, если должным образом настроить переменные tcp_mem, tcp_rmem и tcp_wmem. Впрочем, переменная tcp_rmem не требует особо пристального внимания, поскольку серия ядер 2.4 имеет достаточно хорошие настройки этой переменний, а вот на другие две следует взглянуть поближе. Дополнительную информацию об этом вы найдете в руководстве TCP Tuning Guide.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_mem
96552 128739 193104
```

Можно поставить эти же значения.увеличивать имеет смысл в случае увеличения нагрузки.

tcp_rmem

— векторная (минимум, режим нагрузки, максимум) переменная которая содержит 3 целых числа, определяющих размер приемного буфера сокетов TCP.

Немного подробное описание tcp_rmem...
Векторная (минимум, по умолчанию, максимум) переменная в файле tcp_rmem содержит 3 целых числа, определяющих размер приемного буфера сокетов TCP.

Минимум: каждый сокет TCP имеет право использовать эту память по факту своего создания. Возможность использования такого буфера гарантируется даже при достижении порога ограничения (moderate memory pressure). Размер минимального буфера по умолчанию составляет 8 Кбайт (8192).

Значение по умолчанию: количество памяти, допустимое для буфера передачи сокета TCP по умолчанию. Это значение применяется взамен параметра /proc/sys/net/core/rmem_default, используемого другими протоколами. Значение используемого по умолчанию буфера обычно (по умолчанию) составляет 87830 байт. Это определяет размер окна 65535 с заданным по умолчанию значением tcp_adv_win_scale и tcp_app_win = 0, несколько меньший, нежели определяет принятое по умолчанию значение tcp_app_win.

Максимум: максимальный размер буфера, который может быть автоматически выделен для приема сокету TCP. Это значение не отменяет максимума, заданного в файле /proc/sys/net/core/rmem_max. При «статическом» выделении памяти с помощью SO_RCVBUF этот параметр не имеет значения.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_rmem
4096 87380 4119648
```

Можно поставить эти же значения.увеличивать имеет смысл в случае увеличения нагрузки. В одном из источнике рекомендовали следующие значения:

```
sysctl -w net.ipv4.tcp_rmem="4096 87380 16777216"
```

tcp_wmem

— векторная (минимум, режим нагрузки, максимум) переменная которая содержит 3 целых числа, минимальное, принятое по умолчанию и максимальное количество памяти, резервируемой для буферов передачи сокета TCP.

Немного подробное описание...
Векторная переменная в файле tcp_wmem содержит 3 целочисленных значения, определяющих минимальное, принятое по умолчанию и максимальное количество памяти, резервируемой для буферов передачи сокета TCP.

Минимум: каждый сокет TCP имеет право использовать эту память по факту своего создания. Размер минимального буфера по умолчанию составляет 4 Кбайт (4096)
Значение по умолчанию: количество памяти, допустимое для буфера передачи сокета TCP по умолчанию. Это значение применяется взамен параметра /proc/sys/net/core/wmem_default, используемого другими протоколами и обычно меньше, чем /proc/sys/net/core/wmem_default.

Размер принятого по умолчанию буфера обычно (по умолчанию) составляет 16 Кбайт (16384) Максимум: максимальное количество памяти, которое может быть автоматически выделено для буфера передачи сокета TCP. Это значение не отменяет максимум, заданный в файле /proc/sys/net/core/wmem_max. При «статическом» выделении памяти с помощью SO_SNDBUF этот параметр не имеет значения.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_wmem
4096 16384 4119648
```

Можно оставить эти же значения. Увеличивать их имеет смысл в случае увеличения нагрузки. В одном из источнике рекомендовали следующие значения:

```
sysctl -w net.ipv4.tcp_wmem="4096 65536 16777216"
```

rmem_default, wmem_default

Так же есть «глобальные» параметры размеров буфера, и они не перекрывают значения переменных tcp_rmem и tcp_wmem (читай описание выше)

- **rmem_default** — размер буфера приема данных по умолчанию для всех соединений.
- **wmem_default** — Размер буфера передачи данных по умолчанию для всех соединений.

Их значения по умолчанию:

```
cat /proc/sys/net/core/rmem_default
229376
cat /proc/sys/net/core/wmem_default
229376
```

Можно оставить эти же значения. Увеличивать их имеет смысл в случае увеличения нагрузки. Например, в одном из источнике рекомендовали следующие значения:

```
sysctl -w net.core.rmem_default=65536
sysctl -w net.core.wmem_default=65536
```

rmem_max, wmem_max

а эти «глобальные» параметры перекрывают «максимумы» переменных tcp_rmem и tcp_wmem (опять же, читай описание выше)

- **rmem_max** — максимальный размер буфера приема данных для всех соединений.
- **wmem_max** — максимальный размер буфера передачи данных для всех соединений.

Их значения по умолчанию:

```
cat /proc/sys/net/core/rmem_max
131071
cat /proc/sys/net/core/wmem_max
131071
```

Можно оставить эти же значения. Увеличивать их имеет смысл в случае увеличения нагрузки. Например, в одном из источнике рекомендовали следующие значения:

```
sysctl -w net.core.rmem_max=16777216
sysctl -w net.core.wmem_max=16777216
```

tcp_orphan_retries

— параметр который определяет число неудачных попыток, после которого уничтожается соединение TCP, закрытое на локальной стороне.

Немного подробное описание tcp_orphan_retries...
Целочисленной значение tcp_orphan_retries определяет число неудачных попыток, после которого уничтожается соединение TCP, закрытое на локальной стороне. По умолчанию используется значение 7, соответствующее приблизительно периоду от 50 секунд до 16минут в зависимости от RTO.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_orphan_retries
0
```

Рекомендуется уменьшить значение этого параметра, поскольку закрытые соединения могут поглощать достаточно много ресурсов (т.е. оставляем 0):

```
sysctl -w net.ipv4.tcp_orphan_retries=0
```

ip_conntrack_max

— Максимальное количество соединений для работы механизма connection tracking (используется, например, iptables).

По умолчанию:

```
cat /proc/sys/net/ipv4/netfilter/ip_conntrack_max
65536
```

При слишком маленьких значениях ядро начинает отвергать входящие подключения с соответствующей записью в системном логе:

```
sysctl -w net.ipv4.netfilter.ip_conntrack_max=16777216
```

tcp_timestamps

— включает временные метки протокола TCP, которые позволяют управлять работой протокола в условиях высоких нагрузок (с помощью tcp_congestion_control)

Немного подробное описание...
Разрешает/запрещает использование временных меток (timestamps), в соответствии с RFC 1323. Если коротко, то это расширение TCP используется для расчета Round Trip Measurement (определение времени возврата) лучшим способом, нежели метод Retransmission timeout (RTO). Эта опция должна сохранять обратную совместимость в большинстве случаев, так что лучше оставить ее включенной, особенно если вы работаете в высокоскоростной сети (например LAN или 10mb Интернет).

В случае низкоскоростного оединения (скажем модемное) вы прекрасно обойдетесь и без этой опции, и будет даже лучше, если вы ее отключите. Переменная может принимать два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 1 (включено). Более подробную информацию вы найдете в секции 4 документа RFC 1323 — TCP Extensions for High Performance.

По умолчанию метки включены:

```
cat /proc/sys/net/ipv4/tcp_timestamps
1
```

Кстати, лучше отставить его включенным, иначе не будет работать опция tcp_tw_reuse.

```
sysctl -w net.ipv4.tcp_timestamps=1
```

tcp_sack

— разрешаем выборочные подтверждения протокола TCP. Опция необходима для эффективного использования всей доступной пропускной способности некоторых сетей.

Немного подробное описание...
Разрешает Selective Acknowledgements (SACK Выборочное Подтверждение), детальное описание вы найдете в RFC 2883 — An Extension to Selective Acknowledgement (SACK) Option for TCP и RFC 2883 — An Extension to Selective Acknowledgement (SACK) Option for TCP. Если эта переменная включена (установлена 1), то в TCP-заголовке будет устанавливаться SACK-флаг при передаче SYN-пакета, сообщая тем самым удаленному хосту, что наша система в состоянии обрабатывать SACK, на что удаленный хост может ответить ACK-пакетом с установленным флагом SACK.

Этот режим выборочно подтверждает каждый сегмент в TCP-окне. Это особенно полезно на неустойчивых соединениях, поскольку позволяет производить повторную передачу лишь отдельных, не подтвержденных фрагментов, а не всего TCP-окна, как это диктуется более старыми стандартами. Если какой либо сегмент TCP-окна был утерян, то приемная сторона не пришлет на него SACK-подтверждение о приеме. Отправитель, поняв это, повторит передачу потерявшихся сегментов.

Избыточные данные сохраняются в TCP-заголовке, 40 байт на сегмент. Подтверждение каждого сегмента это два 32-битных беззнаковых целых числа, таким образом в заголовке может разместиться подтверждение 4-х сегментов. Однако, как правило, совместно с опцией SACK используется опция timestamp, которая занимает 10 байт и поэтому в одном пакете может быть подтверждено не более 3 сегментов. Рекомендуется включать эту опцию, если вы имеете неустойчивые соединения. Однако, если вы соединены 1.5-метровым кабелем с другой машиной, то в таком случае, для достижения наивысшей скорости обмена, следует эту опцию отключить.

Обычно эта опция не нужна, но лучше ее включить. Она обеспечивает 100% обратную совместимость, т.е. вы не должны испытывать никаких проблем при соединении с хостами, которые эту опцию не поддерживают. В переменную могут быть записаны два числа 0 (выключено) и 1 (включено). Значение по-умолчанию 1 (включено).

По умолчанию опция включена:

```
cat /proc/sys/net/ipv4/tcp_sack
1
```

Рекомендуется включать эту опцию, если вы имеете неустойчивые соединения. Однако, если вы соединены 1.5-метровым кабелем с другой машиной, то в таком случае, для достижения наивысшей скорости обмена, следует эту опцию отключить:

```
sysctl -w net.ipv4.tcp_sack=1
```

tcp_congestion_control

— протокол, используемый для управления нагрузкой в сетях TCP. **bic** и **cubic** реализации, используемые по умолчанию, содержат баги в большинстве версий ядра RedHat и ее клонов. Рекомендуется использовать htcp.

Немного подробное описание...
Начиная с версии 2.6.13, Linux поддерживает подключаемые алгоритмы управления перегрузкой. Используемый алгоритм управления перегрузки можно задать, используя sysctl переменную net.ipv4.tcp_congestion_control, которая по умолчанию установлена в cubic or reno, в зависимости от версии ядра.

Для получения списка поддерживаемых алгоритмов, выполните: sysctl net.ipv4.tcp_available_congestion_control
Выбор опций контроля за перегрузкой выбирается при сборке ядра. Ниже представлены некоторые из опций, доступных в 2.6.23 ядрах:
* reno: Традиционно используется на большинстве ОС. (default)
* cubic: CUBIC-TCP (Внимание: Есть бага в ядре Linux 2.6.18 Используйте в 2.6.19 или выше!)
* bic:BIC-TCP
* htcp:Hamilton TCP
* vegas:TCP Vegas
* westwood:оптимизирован для сетей с потерями
Для очень длинных и быстрых каналов я предлагаю пробовать cubic или htcp, если использование reno желательно.

По умолчанию:

```
cat /proc/sys/net/ipv4/tcp_congestion_control
cubic
```

Для сервера рекомендуется использовать htcp:

```
sysctl -w net.ipv4.tcp_congestion_control=htcp
```

tcp_no_metrics_save

— данная опция запрещает сохранять результаты изменений TCP соединения в кеше при его закрытии.

По умолчанию опция ничего не запрещает:

```
cat /proc/sys/net/ipv4/tcp_no_metrics_save
0
```

Так как это помогает повысить производительность, рекомендуется включить:

```
sysctl -w net.ipv4.tcp_no_metrics_save=1
```

net.ipv4.route.flush

— актуально для ядер 2.4. По странной причине в ядрах 2.4, если в рамках TCP сессии произошел повтор передачи с уменьшенным размером окна, все соединения с данным хостом в следующие 10 минут будут иметь именно этот уменьшенный размер окна. Данная настройка позволяет этого избежать.

Так как в ядре 3.2 она даже не читается, менять ничего не стал.

ip_local_port_range

— опция, которая содержит диапазон локальных портов, доступных для установки исходящих подключений.

Немного подробное описание...
Содержит два целых числа, которые определяют диапазон локальных портов, которые используются в клиентских соединениях, т.е. для исходящих соединений, которые связывают нашу систему с некоторым узлом сети, где мы выступаем в качестве клиента. Первое число задает нижнюю границу диапазона, второе верхнюю. Значения по-умолчанию зависят от имеющегося объема ОЗУ. Если установлено более чем 128 Мб, то нижняя граница будет 32768, а верхняя 61000.

При меньшем объеме ОЗУ нижняя граница будет 1024 а верхняя 4999 или даже меньше. Этот диапазон определяет количество активных соединений, которые могут быть запущены одновременно, с другой системой, которая не поддерживает TCP-расширение timestamp. Диапазона 1024-4999 вполне достаточно для установки до 2000 соединений в секунду с системами, не поддерживающими timestamp. Проще говоря, этого вполне достаточно для большинства применений.

По умолчанию там такой диапазон:

```
cat /proc/sys/net/ipv4/ip_local_port_range
32768 61000
```

Для тяжелых проектов диапазон рекомендуется увеличить:

```
sysctl -w net.ipv4.ip_local_port_range="1024 65535"
```

tcp_tw_reuse

— опция позволяющая повторное использование TIME-WAIT сокетов в случаях, если протокол считает это безопасным.

Немного подробное описание tcp_tw_reuse (eng)...
tcp_tw_reuse — BOOLEAN
Allow to reuse TIME-WAIT sockets for new connections when it is safe from protocol viewpoint.
Default value is 0.
It should not be changed without advice/request of technical experts.

По умолчанию отключена:

```
cat /proc/sys/net/ipv4/tcp_tw_reuse
0
```

Лучше включить:

```
sysctl -w net.ipv4.tcp_tw_reuse=1
```

tcp_window_scaling

— опция позволяет динамически изменять размер окна TCP стека

Немного подробное описание...
Разрешает/запрещает масштабирование TCP-окна, как определено в RFC 1323. В этом документе описано как производится масштабирование TCP-окна при передаче по Large Fat Pipes (LFP толстый канал).

При передаче TCP-пакетов по толстым каналам возникают существенные потери пропускной способности из-за того, что они не загружены полностью во время ожидания подтверждения о приеме предыдущего TCP-окна. Основная проблема состоит в том, что окно не может иметь размер больше, чем 216 байт (65 Кб). Разрешая масштабирование TCP-окна мы, тем самым, можем увеличить его размер и таким образом уменьшить потери пропускной способности.

Переменная может принимать два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 1 (включено). Дополнительную информацию по этой теме вы найдете в RFC 1323 — TCP Extensions for High Performance.

По умолчанию она включена:

```
cat /proc/sys/net/ipv4/tcp_window_scaling
1
```

Лучше так и оставить:

```
sysctl -w net.ipv4.tcp_window_scaling=1
```

tcp_rfc1337

— с помощью этой опции мы можем защитить себя от TIME_WAIT атак.

Немного подробное описание...
Переменная tcp_rfc1337 является реализацией решения проблемы, описываемой в RFC 1337 — TIME-WAIT Assassination Hazards in TCP. Проблема связана с устаревшими дубликатами пакетов, которые могут вносить помехи во вновь устанавливаемые соединения и порождать три различные проблемы. Первая устаревший дубликат пакета с данными может быть ошибочно воспринят в новом соединении, что приведет к передаче неверных данных.

Вторая соединение может быть десинхронизировано и уйти в ACK-цикл из-за устаревших дубликатов, которые порождают новые соединения (здесь автор имеет ввиду устаревшие дубликаты SYN-пакетов, прим. перев.). И третья, и последняя проблема устаревшие дубликаты могут проникнуть в недавно созданное соединение и ошибочно уничтожить его.

Согласно упомянутому RFC существуют три возможных решения, однако, одно из них решает эту проблему лишь частично, второе требует внесения значительных изменений в протокол TCP. Окончательное решение состоит в том, что RST-пакеты должны просто игнорироваться, пока сокет находится в состоянии TIME_WAIT. Вместе с установкой параметра Maximum Segment Life (MSL максимальное время жизни сегмента) равным 2 мин. такой подход решает все три проблемы, описанные в RFC 1337.

По умолчанию опция отключена:

```
cat /proc/sys/net/ipv4/tcp_rfc1337
0
```

На сервере она точно не помешает:

```
sysctl -w net.ipv4.tcp_rfc1337=1
```

ip_forward

— данная опция управляет переадресацией пакетов. Если этот параметр выключен, ОС считает себя узлом IP сети и дропает все пакеты, предназначенные не ей. Если параметр включен, то ОС считает себя маршрутизатором и действует в соответствии с RFC1812, в том числе пытается переслать адресованные не ей пакеты в соответствии с таблицей маршрутизации.

По умолчанию переадресация включена:

```
cat /proc/sys/net/ipv4/ip_forward
0
```

Если сервер не является маршрутизатором, то включать эту опцию нет необходимости:

```
sysctl -w net.ipv4.ip_forward=0
```

tcp_abort_on_overflow

— Эта переменная заставляет ядро отвергать новые соединения, если их поступает количество, с которым система не в состоянии справиться. **Эту переменную следует использовать как крайнюю меру!**

По умолчанию она отключена:

```
cat /proc/sys/net/ipv4/tcp_abort_on_overflow
0
```

Если ситуация требует, то ее можно включить:

```
sysctl -w net.ipv4.tcp_abort_on_overflow=1
```

Для удобства ниже приведены все параметры разом, если решение вам нужно здесь и сейчас:

```
# включает фильтр обратного пути, она же защита от спуфинга
(включаем строгий режим)
sysctl -w net.ipv4.conf.all.rp_filter=1

# запрет маршрутизации от источников
sysctl -w net.ipv4.conf.all.accept_source_route=0
sysctl -w net.ipv4.conf.lo.accept_source_route=0
sysctl -w net.ipv4.conf.eth0.accept_source_route=0
sysctl -w net.ipv4.conf.default.accept_source_route=0

# запрещаем принимать и отправлять ICMP пакеты перенаправления
sysctl -w net.ipv4.conf.all.accept_redirects=0
sysctl -w net.ipv4.conf.all.secure_redirects=0
sysctl -w net.ipv4.conf.all.send_redirects=0

# отключаем ответ на broadcast ICMP ECHO
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1

# игнорируем ошибочные ICMP запросы
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1

# ответ на ICMP запросы
sysctl -w net.ipv4.icmp_echo_ignore_all=0

# отключаем механизм SYN cookies (лучше отключить на
высоконагруженных серверах)
sysctl -w net.ipv4.tcp_syncookies=0

# максимальное число полуоткрытых соединений
sysctl -w net.ipv4.tcp_max_syn_backlog=4096

# время удержания полуоткрытых соединений
sysctl -w net.ipv4.tcp_synack_retries=1

# определяет максимальное число осиротевших TCP пакетов
sysctl -w net.ipv4.tcp_max_orphans=65536

# время ожидания приема FIN до полного закрытия сокета
sysctl -w net.ipv4.tcp_fin_timeout=10
```

```
# проверять TCP-соединения
sysctl -w net.ipv4.tcp_keepalive_time=60

# интервал передачи проб
sysctl -w net.ipv4.tcp_keepalive_intvl=15

# количество проверок перед закрытием соединения
sysctrl -w net.ipv4.tcp_keepalive_probes=5

# макс. кол-во пакетов в очереди на обработку, если интерфейс
получает пакеты быстрее, чем ядро может их обработать
sysctl -w net.core.netdev_max_backlog=1000

# Максимальное число открытых сокетов, ждущих соединения
sysctl -w net.core.somaxconn=15000

# размер приемного буфера сокетов TCP (мин-нагрузка-макс)
sysctl -w net.ipv4.tcp_rmem="4096 87380 16777216"

# размер передающего буфера сокетов TCP (мин-нагрузка-макс)
sysctl -w net.ipv4.tcp_wmem="4096 65536 16777216"

# размер буфера приема данных по умолчанию для всех соединений
sysctl -w net.core.rmem_default=229376

# Размер буфера передачи данных по умолчанию для всех соединений
sysctl -w net.core.wmem_default=229376

# макс. размер буфера приема данных для всех соединений
sysctl -w net.core.rmem_max=16777216

# макс. размер буфера передачи данных для всех соединений
sysctl -w net.core.wmem_max=16777216

# число неудачных попыток, после которого уничтожается соединение
TCP, закрытое на локальной стороне
sysctl -w net.ipv4.tcp_orphan_retries=0

# макс кол-во соединений для работы механизма connection tracking
sysctl -w net.ipv4.netfilter.ip_conntrack_max=16777216

# включает временные метки протокола TCP
sysctl -w net.ipv4.tcp_timestamps=1

# разрешаем выборочные подтверждения протокола TCP
sysctl -w net.ipv4.tcp_sack=1

# протокол, используемый для управления нагрузкой в сетях TCP
sysctl -w net.ipv4.tcp_congestion_control=htcp
```

```
# запрещаем сохранять результаты изменений TCP соединения в кеше при его закрытии
sysctl -w net.ipv4.tcp_no_metrics_save=1

# диапазон локальных портов, доступных для установки исходящих подключений
sysctl -w net.ipv4.ip_local_port_range="1024 65535"

# включаем повторное использование TIME-WAIT сокетов для безопасных протоколов
sysctl -w net.ipv4.tcp_tw_reuse=1

# разрешаем динамически изменять размер окна TCP стека
sysctl -w net.ipv4.tcp_window_scaling=1

# включаем защиту от TIME_WAIT атак
sysctl -w net.ipv4.tcp_rfc1337=1

# отключаем переадресацию пакетов
sysctl -w net.ipv4.ip_forward=0
```