Автор: Administrator
18.03.10 13:35 - Последнее обновление 18.03.10 13:39

Описание некоторых sysctl переменных ядра Linux

Немного информации о настройки не маловажного файла /etc/sysctl.conf:

Оригинал: http://debian.telenet.ru/doc/sysctl.conf

> net.ipv4.conf.default.forwarding=1

Включение форвардинга пакетов. Иными словами, необходимо разрешить ядру операционной системы осущетсвлять проброс трафика с одного интерфейса на другой.

> fs.file-max = 64000

Максимальное значение открытых файлов. Что бы его узнать, выполните команду cat /proc/sys/fs/file-max

> net.ipv4.conf.all.rp_filter = 1

Эта переменная сообщает ядру о необходимости фильтрации пакетов по их исходящему адресу.

> kernel.sysrq = 0

Отключается комбинация клавиш sysrq, которая используется при крахе системы.

> net.ipv4.conf.default.rp_filter=1

Фильтр обратного пути (когда пакет приходит с одного интерфейса, а уходит на другой)

> net.ipv4.conf.all.accept_source_route = 0

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.

> net.ipv4.tcp_syncookies=1

Разрешает/запрещает передачу так называемых syncookies вызывающему хосту в случае переполнения очереди SYN-пакетов для заданного сокета. Когда в систему поступает слишком много запросов на соединение, то очередь может переполниться и тогда запускается передача syncookies в ответ на каждый SYN-запрос. Эта переменная используется для предотвращения syn-flood атак. Переменная может принимать два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено). Эта функция будет работать только в том случае, если ядро собрано с опцией CONFIG_SYN_COOKIES.

(прим. перев.) В прошлом было много дискуссий по поводу проблем и недостатков, связанных с syncookies. Лично я рассматриваю эту возможность как достаточно удобную и безопасную. Однако, в некоторых случаях, использование этой опции может представлять некоторую угрозу, см. ниже. Опция tcp_syncookies подразумевает, что на системах с высокой нагрузкой новые соединения будут устаналиваться без таких фишек, как ECN и SACK. Если передача syncookies срабатывет при невысоких нагрузках, то вам следует подкорректировать параметр, задающий длину очереди. Не следует использовать эту возможность на системах с высокой нагрузкой. Если в системный журнал поступают предупреждения о syn flood для вполне законных соединений, то можете попробовать подстроить переменные tcp_max_syn_backlog, tcp_synack_retries и tcp_abort_on_overflow.

> net.ipv4.ip_default_ttl = 64

Устанавливает значение по-умолчанию для величины Time To Live исходящих пакетов. Это число определяет продолжительность жизни пакета в Internet. Каждый раз, когда пакет попадает на очередной роутер, брандмауэр и т.п., величина TTL пакета уменьшается на 1. Значение по-умолчанию 64. Это достаточно приемлемое значение. Очень трудно представить себе ситуацию, когда это число оказалось бы недостаточным. Переменная принимает целое число без знака в диапазоне от 0 до 255 включительно, однако, значение 255 слишком велико, а если поставить 0 то вы вообще лишитесь выхода в сеть. Число 64 достаточно хорошее, если вы не пытаетесь установить соединение с компьютером, отстоящим от вас на значительном растоянии, измеряемом в переходах (hops). Тогда этой вличины TTL может и не хватить. Однако, я в своей практике еще не встречался с компьютерами, отстоящими от меня более чем на 30 переходов (hops) и я не думаю, что вам придется увеличивать значение этой переменной. Увеличение TTL пакета до 255 может иметь свои негативные последствия. Если представить себе, что где-то произошел сбой в 2-х маршрутизаторах и пакет зациклился между

ними, тогда пакет начнет бегать туда-сюда между маршрутизаторами, поглощая пропускную способность канала, до тех пор, пока TTL пакета не истечет. Как правило, величину TTL не устанавливают больше 100.

> net.ipv4.ip_dynaddr = 0

Переменная используется для разрешения некоторых проблем, связанных с динамической адресацией. Позволяет демону diald одновременно устанавливать соединение и изменять исходящий адрес в пакетах (и сокетах для локальных процессов). Эта возможность была реализованв для поддержки TCP по коммутируемым соединениям и соединениям с маскарадингом (masqueradig). Эта опция позволяет маскарадить исходящий адрес пакета при изменении динамического IP-адреса.В переменную можно записать одно из 3-х значений: 0, 1 или 2. (0 опция выключена, это значение по-умолчанию. 1 опция включена.) Любое другое значение, отличающееся от 0 и 1 подразумевает включение этой опции в многословном (verbose) режиме, что приводит к записи в системный журнал отладочных сообщений. Что происходит, если изменяется адрес исходящего интерфейса при включенной опции ip_dynaddr: Исходящий адрес пакета и сокета изменяется ПРИ ПОВТОРНОЙ ПЕРЕДАЧЕ, когда он находится в состоянии SYN_SENT. Это касается локальных процессов. Для транзитных пакетов исходящий адрес пакета изменяется на выходе, когда внутренний хост выполняет ПОВТОРНУЮ ПЕРЕДАЧУ, пока не будет принят внешний пакет. Эта опция особенно полезна для случаев автодозвона, когда в момент установления связи исходящий адрес еще не известен. Любой запрос на соединение заставляет diald поднять исходящий интерфейс, после чего пакеты отправляются через него. Это означает, что нет необходимости сначала выдавать запрос который поднимет исходящий интерфейс, а затем выдавать реальный запрос на соединение, вместо этого мы просто устанавливаем соединение.

> net.ipv4.ip_local_port_range = 32768 61000

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.

> net.ipv4.ip_no_pmtu_disc = 0

Переменная ip_no_pmtu_disc запрещает поиск PMTU (от англ. Path Maximum Transfer Unit Максимальный Размер Пакета для выбранного Пути). Для большинства случаев лучше установить в эту переменную значение FALSE, или 0 (т.е. система будет пытаться определить максимальный размер пакета, при котором не потребуется выполнять их фрагментацию, для передачи на заданный хост). Но иногда, в отдельных случаях, такое поведение системы может приводить к срывам соединений. Если у вас возникают такие проблемы, то вам следует попробовать отключить эту опцию (т.е. записать в переменную число 1) и установить нужное значение MTU. Хочу обратить ваше внимание на то, что MTU и PMTU это не одно и то же! MTU (от англ. Maximum Transfer Unit максимальный размер пакета) определяет максимальный размер пакета для наших сетевых интерфейсов, но не для сетевых интерфейсов на другом конце. PMTU опция, которая заставляет систему вычислять максимальный размер пакета, при котором не потребуется фрагментация пакетов, для маршрута к заданному хосту, включая все промежуточные переходы. Значение по-умолчанию FALSE (0), т.е. функция определения разрешена. Если записать число 1 в эту переменную, то функция определения PMTU будет запрещена. Переменная ip_no_pmtu_disc может принимать значение 0 или 1.

> net.ipv4.ip_nonlocal_bind = 0

Установка этой переменной позволяет отдельным локальным процессам выступать от имени внешнего (чужого) IP адреса. Это может оказаться полезным в некоторых случаях, когда необходимо прослушивать внешние (чужие) IP адреса, например сниффинг чужого траффика. Однако, эта опция может оказывать отрицательное влияние на работоспособность отдельных приложений. Эта переменная может иметь два значения 0 или 1. Если установлено значение 0, то опция отключена, 1 включена. Значение по-умолчанию 0.

> net.ipv4.ipfrag_low_thresh = 196608

Переменная задает максимальный объем памяти, выделяемый под очередь фрагментированных пакетов. Когда длина очереди достигает этого порога, то обработчик фрагментов будет отвергать все фрагментированные пакеты до тех пор, пока длина очереди не уменьшится до значения переменной ipfrag_low_thresh. Это означает, что все отвергнутые фрагментированные пакеты должны быть повторно переданы узлом-отправителем. Пакеты фрагментируются в том случае, если их размер слишком велик, чтобы быть переданными по данному каналу. Узел-отправитель, в этом случае, режет пакеты на более мелкие части и затем передает их одну за другой. На узле-получателе эти фрагменты опять собираются в полноценный пакет. Примечательно, что идея фрагментации пакетов, сама по себе, вещь

замечательная, но, к сожалению, может быть использована в весьма неблаговидных целях. Переменная содержит целое число в диапазоне 0 .. 2147483647 и означает верхний порог длины очереди фрагментов в байтах. Значение по-умолчанию 262144 байт, или 256 Кб. Этого количества, как правило, вполне достаточно даже для самых крайних случаев.

> net.ipv4.ipfrag_time = 30

Эта переменная определяет максимальное время хранения фрагментов в секундах. Это относится только к тем фрагментам, которые пока невозможно собрать, поскольку собранные пакеты к этому сроку скорее всего уже будут переданы дальше (на следующий уровень или в сеть). Переменная принимает целое значение и определяет предельное время хранения фрагментов в секундах. Если записать туда число 5, то это будет означать 5 секунд.

> net.ipv4.inet_peer_gc_maxtime = 120

Переменная inet_peer_gc_maxtime определяет частоту сборки мусора при незначительном объеме данных. Эта переменная имеет то же предназначение, что и inet_peer_gc_mintime, только выбор, какой переменной пользоваться, производится в зависимости от величины нагрузки на систему. Переменная принимает целое число, измеряемое в тиках. Значение по-умолчанию 120 тиков, чего вполне достаточно для большинства серверов и рабочих станций.

> net.ipv4.inet_peer_gc_mintime = 10

Переменная inet_peer_gc_maxtime определяет частоту сборки мусора при незначительном объеме данных. Эта переменная имеет то же предназначение, что и inet_peer_gc_mintime, только выбор, какой переменной пользоваться, производится в зависимости от величины нагрузки на систему. Переменная принимает целое число, измеряемое в тиках. Значение по-умолчанию 120 тиков, чего вполне достаточно для большинства серверов и рабочих станций.

> net.ipv4.inet_peer_maxttl = 600

Это максимальное время хранения записей. При незначительных нагрузках на систему неиспользуемые записи будут удаляться через данный промежуток времени.

> net.ipv4.inet_peer_minttl = 120

Переменная определяет минимальное время хранения данных в inet peer storage. Это время должно быть достаточно большим, чтобы перекрыть время сборки фрагментов на противоположной стороне. Минимальное время хранения

является гарантией того, что размер пула будет меньше чем величина inet_peer_threshold.

> net.ipv4.inet_peer_threshold = 65664

Переменная определяет минимальное время хранения данных в inet peer storage. Это время должно быть достаточно большим, чтобы перекрыть время сборки фрагментов на противоположной стороне. Минимальное время хранения является гарантией того, что размер пула будет меньше чем величина inet_peer_threshold.

> net.ipv4.tcp_abort_on_overflow = 0

Эта переменная заставляет ядро отвергать новые соединения, если их поступает такое количество, что система не в состоянии справиться с таким потоком. Что это означает? Допустим, что на систему обрушивается шквал запросов на соединение, тогда они могут быть просто отвергнуты, если переменная будет включена, поскольку система не в состоянии обработать их все. Если переменная не установлена, то система будет пытаться обслужить все запросы. Переменная может иметь два значение 0(выключено) или 1(включено). Значение по-умолчанию 0. Включение этой опции следует расценивать как крайнюю меру. Перед этим необходимо попытаться поднять производительность сервисов.

> net.ipv4.tcp_adv_win_scale = 2

Эта переменная влияет на вычисление объема памяти в буфере сокета, выделяемой под размер TCP-окна и под буфер приложения. Если величина tcp_adv_win_scale отрицательная, то для вычисления размера используется следующее выражение: Где bytes это размер окна в байтах. Если величина tcp_adv_win_scale положительная, то для определения размера используется следующее выражение: Переменная принимает целое значение. Значение по-умолчанию 2, т.е. под буфер приложения отводится 1/4 часть объема, определяемого переменной tcp_rmem.

> net.ipv4.tcp_app_win = 31

Эта переменная определяет количество байт, резервируемых под TCP-окно в приемном буфере сокета. Выражение, согласно которому делается расчет, выглядит следующим образом: Как видно из выражения чем больше значение переменной, тем меньше будет размер буфера окна. Исключение составляет значение 0. В этом случае резервирование не производится. Значение по-умолчанию 31. Не изменяйте эту переменную, если вы не уверены в том, что делаете.

> net.ipv4.tcp_dsack = 1

Эта опция необходима для выполнения передачи дублирующих SACK-пакетов (от англ. Selective ACKnowledgement Выборочное Подтверждение Приема, прим. перев.). Техника выборочного подтверждения кратко описана в разделе, посвященном переменной tcp_sack. Детальное описание можно найти в RFC 2883. Здесь подробно описываются действия по обработке ситуаций, когда получен дубликат пакета, пришедшего ранее, или когда пакеты поступают не в той очередности. D-SACK это расширение стандарта SACK и используется для уведомления хоста-отправителя о том, что пакет был получен дважды (т.е. продублирован). Эту информацию хост-отправитель может использовать для корректировки сетевых настроек и т.п.. Эта опция гарантирует 100%-ную обратную совместимость с устаревшими реализациями TCP, если в них техника SACK не реализована каким-либо нестандартным образом. Но это чрезвычайно редкий случай, а посему у вас едва ли будут возникать проблемы с этим. Переменная может иметь два состояния 1 (включено) и 0 (выключено). Значение по-умолчанию 1 (включено). И, само собой разумеется, установка этой переменной имеет смысл только если включена переменная tcp_sack, поскольку tcp_dsack очень тесно связана с ней. В большинстве случаев разумным будет оставить переменную включенной.

> net.ipv4.tcp_ecn = 0

Переменная tcp_ecn отвечает за работу Explicit Congestion Notification (Явное Уведомление о Перегруженности) в TCP-соединениях. Используется для уведомления о возникновении затора на маршруте к заданному хосту или сети. Может использоваться для извещения хоста-отправителя о необходимости снизить скорость передачи пакетов через конкретный маршрутизатор или брандмауэр. Explicit Congestion Notification (ECN) детально описано в RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP. Дополнительную информацию о ECN вы найдете в RFC 2884 - Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks. Коротко: этот документ подробно описывает как должно передаваться уведомление о перегруженности хосту-отправителю, который, в свою очередь, может выбрать другой маршрут или начать снижать скорость передачи до тех пор, пока к нему не перестанут поступать ECN-пакеты. В Интернете еще встречаются маршрутизаторы и брандмауэры, которые не пропускают пакеты с установленным флагом ECN и, если вам не повезет, то вы можете столкнуться с ними. При возникновении проблем с прохождением ECN попробуйте отключить эту опцию. Если вам встретится подобный маршрутизатор, то можете попробовать вступить в контакт с администратором и предупредить его об имеющихся проблемах. Подробное описание проблемы и общий список аппаратуры, являющейся причиной этой неприятности, вы найдете в приложении Ссылки в пункте ECN-under-Linux Unofficial Vendor Support Page. Переменная может иметь два состояния 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено).

> net.ipv4.tcp_fack = 1

Переменная tcp_fack ответственна за систему Forward Acknowledgement (Упреждающее Подтверждение) в Linux. Forward Acknowledgement это специальный алгоритм, который работает поверх SACK, и предназначен для контроля заторов. Главная идея алгоритма FACK состоит в отслеживании наибольшего номера выборочно подтвержденной последовательности как признака того, что все предыдущие не подтвержденные (выборочно) сегменты были потеряны. Это позволяет оптимизировать восстановление потерь. Однако, этот алгоритм становится неработоспособным в случае неупорядоченного потока данных, тогда он (алгоритм) автоматически отключается для данного конкретного соединения. Изначально алгоритм FACK был разработан Мэттью Матисом (Matthew Mathis) с соавторами. Документ, описывающий алгоритм, можно найти на http://www.psc.edu/~mathis/. Переменная может принимать два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 1 (включено). Эта опция используется только тогда, когда включена переменная tcp_sack.

> net.ipv4.tcp_fin_timeout = 60

Переменная задает максимальное время пребывания сокета в состоянии FIN-WAIT-2. Используется в тех случаях, когда другая сторона по тем или иным причинам не закрыла соединение со своей стороны. Каждый сокет занимает в памяти порядка 1.5 Кб, что может привести к значительным утечкам памяти в некоторых случаях. Переменная принимает целое число. Значение по-умолчанию 60 секунд. В ядрах серии 2.2 это значение было равно 180 секундам, но было уменьшено, поскольку иногда возникали проблемы, связанные с нехваткой памяти, на web-серверах, которые, как правило, обслуживают огромное количество подключений. За дополнительной информацией обращайтесь к описанию переменных tcp_max_orphans и tcp_orphan_retries.

> net.ipv4.tcp_keepalive_intvl = 75

Переменная определяет интервал проверки жизнеспособность сокета. Это значение учитывается при подсчете времени, которое должно пройти перед тем как соединение будет разорвано. Переменная принимает целое число. Значение по-умолчанию 75 секунд. Это достаточно высокое значение, чтобы рассматривать его как нормальное. Значения переменных tcp_keepalive_probes и tcp_keepalive_intvl могут использоваться для определения времени, через которое соединение будет разорвано. Со значениями по-умолчанию (9 попыток с интервалом 75 секунд) это займет примерно 11 минут. Попытки определения жизнеспособности, в свою очередь, начнутся через 2 часа после того, как через данное соединение проследовал последний пакет.

> net.ipv4.tcp_keepalive_probes = 9

Переменная определяет количество попыток проверки жизнеспособности прежде, чем будет принято решении о разрыве соединения. Переменная принимает целое число, которое не следует устанавливать больше 50-ти. Значение по-умолчанию 9. Это означает, что будет выполнено 9 попыток проверки соединения, чтобы убедиться в том, что соединение разорвано.

> net.ipv4.tcp_keepalive_time = 7200

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. Переменная принимает целое число секунд. Значение по-умолчанию 7200 секунд, или 2 часа. Не уменьшайте это число без необходимости, поскольку это может привести к увеличению излишнего трафика.

> net.ipv4.tcp_max_orphans = 65536

Переменная задает максимальное число осиротевших (не связанных ни с одним процессом) сокетов. Если это число будет превышено, то такие соединения разрываются, а в системный журнал пишется предупреждение. Это ограничение существует исключительно ради предотвращения простейших разновидностей DoS-атак. Вообще вы не должны полагаться на эту переменную! Не рекомендуется уменьшать это число. Сетевая среда может потребовать увеличение этого порога, однако, такое увеличение может привести к необходимости увеличения объема ОЗУ в системе. Прежде чем поднимать этот предел попробуйте перенастроить сетевые сервисы на более агрессивное поведение по отношению к осиротевшим сокетам. Переменная принимает целое число. Значение по-умолчанию 8192, однако оно очень сильно зависит от объема памяти в системе. Каждый осиротевший сокет съедает примерно 64 Кб памяти, которая не может быть сброшена в своп (swap). При возникновении проблем, связанных с этим ограничением в системный журнал будет записано сообщение, подобное этому: TCP: too many of orphaned sockets Это может служить поводом к тому, чтобы пересмотреть значения переменных tcp_fin_timeout или tcp_orphans_retries.

> net.ipv4.tcp_max_syn_backlog = 1024

Переменная определяет максимальное время хранения SYN-запросов в памяти до момента получения третьего, завершающего установление соединения, пакета. Эта опция работает только тогда, когда включена переменная tcp_syncookies. Если сервер испытывает серьезные нагрузки, то можно попробовать немного увеличить этот параметр. Переменная принимает целое число. Значение по-умолчанию зависит от количества памяти, имеющейся в системе. Если объем памяти менее 128 Мб, то значение по-умолчанию равно

128, если больше, то значение по-умолчанию равно 1024. Если вы увеличиваете эту переменную до величины более чем 1024, то было бы неплохо изменить величину TCP_SYNQ_HSIZE и пересобрать ядро. TCP_SYNQ_HSIZE находится в файле linux/include/tcp.h. Эта величина рассчитывается по формуле: TCP_SYNQ_HSIZE*16 tcp_max_syn_backlog

> net.ipv4.tcp_max_tw_buckets = 180000

Максимальное число сокетов, находящихся в состоянии TIME-WAIT одновременно. При превышении этого порога лишний сокет разрушается и пишется сообщение в системный журнал. Цель этой переменной предотвращение простейших разновидностей DoS-атак. Переменная принимает целое число. Значение по-умолчанию 180000. На первый взгляд может показаться, что это очень много, но на самом деле это не так. Если у вас начинают возникать ошибки, связанные с этим параметром, то попробуйте увеличить его. Вам не следует уменьшать значение этой переменной. Вместо этого, если начали поступать сообщения в системный журнал, ее следует увеличить, однако, это может потребовать наращивания памяти в системе.

> net.ipv4.tcp_mem = 196608 262144 393216

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.

> net.ipv4.tcp_orphan_retries = 0

Количество попыток закрыть соединение перед тем как оно будет разорвано принудительно. Если вы администрируете http-сервер, который испытывает большие нагрузки, то стоит подумать об уменьшении этого значения. Переменная принимает целое число. Значение по-умолчанию 7, что соответствует, примерно, от 50 секунд до 16 минут, в зависимости от величины Retransmission Timeout (RTO Таймаут для Повторной Передачи. прим. перев.). Детальное описание RTO вы найдете в разделе 3.7. Data Communication RFC 793 - Transmission Control Protocol. Кроме того, посмотрите описание переменной tcp_max_orphans.

> net.ipv4.tcp_reordering = 3

Максимальное количество пакетов, пришедших в потоке не по-порядку, прежде чем будет сделано предположение о том, что пакет был потерян где-то на маршруте. Когда делается такое предположение, то стек TCP переходит обратно в режим slow start (медленный старт), поскольку пакет действительно мог быть утерян из-за перегруженности сети. Кроме того, стек TCP откажется от дальнейшего использования алгоритма FACK при обмене с этим хостом. Переменная принимает целое число. Значение по-умолчанию 3. Это достаточно хорошее значение и не следует его изменять. Если этот параметр уменьшить, то это может значительно ухудшить работу сетевой подсистемы особенно, если пакеты часто переупорядочиваются.

> net.ipv4.tcp_retrans_collapse = 1

Включает/выключает эмуляцию ошибки протокола TCP, делая возможным сетевое взаимодействие с некоторыми устройствами, в которых реализация стека TCP имеет эту ошибку. Без ее эмуляции было бы невозможным работать с отдельными моделями принтеров. Ошибка заключается в отправке полноразмерных пакетов при повторной передаче. Переменная может принимать два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 1 (включено). Эмуляция ошибки никак не повредит взаимодействию с другими узлами сети, но при этом позволит общаться с устройствами, имеющими ошибку в реализации стека TCP. Вообще эта опция совершенно безопасна, однако, если в журнал пишутся непонятные сообщения можете попробовать отключить ее.

> net.ipv4.tcp_retries1 = 3

Максимальное количество попыток повторной передачи пакетов по установленному соединению прежде, чем сообщение об ошибке будет передано сетевому уровню, в результате чего может быть выбран другой маршрут для отправки последующих пакетов. Минимальное значение этого параметра определяется в RFC ???? и равно 3. Это число является значением по-умолчанию, что соответствует интервалу времени от 3 секунд до 8

минут, в зависимости от величины Retransmission timeout (RTO). Детальное описание RTO вы найдете в разделе 3.7. Data Communication RFC 793 - Transmission Control Protocol. Переменная принимает целое число. Значение по-умолчанию 3. Стандарты определяют диапазон изменения этого параметра от 3 до 100.

> net.ipv4.tcp_retries2 = 15

Максимальное количество попыток повторной передачи пакетов, до того как соединение будет считаться разорванным. Это ограничение определено в RFC 1122 и равно 100, но обычно его уменьшают. Переменная принимает целое число. Значение по-умолчанию 15, что соответствует примерно 13-30 минутам в зависимости от величины Retransmission timeout (RTO). При желании можете попробовать уменьшить этот параметр.

> net.ipv4.tcp_rfc1337 = 0

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.

> net.ipv4.tcp_rmem = 4096 87380 4194304

Переменная содержит три числа, которые используются при управлении размерами приемных буферов. Первое число минимальный размер приемного буфера, который выделяется для каждого сокета. Этот размер буфера выделяется всегда, даже при очень высоких нагрузках на систему. Значение по-умолчанию 4096 байт, или 4 Кб. В предыдущих версиях ядра Linux это значение было 8192 байт. Это достаточно большой размер и не следует увеличивать его, иначе, при взрывных нагрузках на сеть, вы можете столкнуть с очень серьезными проблемами. Второе число размер приемного

буфера по-умолчанию, который выделяется для каждого сокета. Это значение перекрывает значение переменной /proc/sys/net/core/rmem_default, которая используется другими протоколами. Значение по-умолчанию 87380 байт, или 85 Кб. Совместно с переменными tcp_adv_win_scale и tcp_app_win эта переменная используется при расчете размера TCP-окна. Это значение используется при незначительных нагрузках. Как и в случае с первым значением не следует без особой нужды изменять этот параметр. Эта переменная может дать некоторый прирост производительности на толстых каналах, если достаточно корректно используется вместе с переменными tcp_mem и tcp_wmem. tcp_rmem не требует вмешательства извне, поскольку ядра серии 2.4 достаточно хорошо настраивают ее автоматически, а вот на tcp_mem и tcp_wmem можно взглянуть попристальнее. Дополнительную информацию, по настройке переменных, вы найдете в TCP Tuning Guide. Третье, и последнее, число максимально возможный размер приемного буфера, который может быть размещен для каждого сокета. Этот параметр перекрывается переменной /proc/sys/net/core/rmem_max, если значение в ipv4 оказывается больше. Перед изменением этого параметра вам следует сначала взглянуть на значение /proc/sys/net/core/rmem_max. Значение по-умолчанию удвоенное значение второго параметра, т.е. 87380 * 2 bytes, или 174760 байт (170 Кб). Как правило этот параметр не нуждается в корректировке.

> net.ipv4.tcp_sack = 1

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).

> net.ipv4.tcp_stdurg = 0

Разрешает/запрещает соответствие стандарту RFC 1122. Поведение по-умолчанию соответствует стандарту использования флага URG BSD 4.2, описанному в RFC 793. Если эта переменная включена, то возможны сбои при работе с отдельными узлами Интернета, точнее с узлами, которые придерживаются стандарта BSD 4.2. За дополнительной информацией обращайтесь к RFC 1122 - Requirements for Internet Hosts Communication Layers секция 4.2.2.4 Urgent Pointer: RFC 793 Section 3.1 explanation, где имеется ссылка на RFC 793 - Transmission Control Protocol. Переменная принимает два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено).

> net.ipv4.tcp_syn_retries = 5

Количество попыток передачи SYN-пакета при установлении нового соединения. Переменная принимает целое число, которое не должно устанавливаться больше чем 255, поскольку каждая повторная попытка отнимает значительное время. На каждую попытку отводится примерно 30-40 секунд. Значение по-умолчанию 5, что соответствует, примерно, 180 секундам.

> net.ipv4.tcp_synack_retries = 5

Количество попыток передачи SYN,ACK-пакета в ответ на SYN-запрос. Другими словами максимальное число попыток установить пассивное TCP-соединение, инициированное другим хостом. Переменная принимает целое число, которое не должно устанавливаться больше чем 255 по тем же причинам, что и в случае с переменной tcp_syn_retries. Значение по-умолчанию 5.

> net.ipv4.tcp_timestamps = 1

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.

> net.ipv4.tcp_tw_recycle = 0

Разрешает/запрещает быструю утилизацию сокетов, находящихся в состоянии TIME-WAIT. Если вы не уверены в своих действиях, то вам лучше не трогать эту переменную. Переменная принимает целое число (а не булевское из моего опыта и моего понимания исходных текстов ядра следует, что описание переменной в linux/Documentation/ip-sysctl.txt содержит ошибку, если я не ошибаюсь). Значение по-умолчанию 0. Не изменяйте эту переменную, если вы не уверены в своих действиях или не получили совет от опытных экспертов по ядру.

> net.ipv4.tcp_window_scaling = 1

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.

> net.ipv4.tcp_wmem = 4096 16384 4194304

Переменная содержит три числа, которые используются при управлении размерами буферов передачи, выделяемых для каждого сокета. Каждое из значений используется при определенных условиях. Первое значение минимальный размер буфера передачи для каждого сокета. Системой гарантируется выделение этого пространства при открытии сокета. Обычно

это значение равно 4096 байт. Второе значение размер передающего буфера по-умолчанию. При попытке увеличить размер буфера передачи больше этого ограничения, приложение может столкнуться с нежеланием системы выделения большего объема памяти при тяжелых нагрузках. Это может даже привести к потере пакетов при очень высоких нагрузках. Значение по-умолчанию 16384 байт, или 16 Кб. Будет неразумным попытаться увеличить это значение. Этот параметр перекрывается переменной /proc/sys/net/core/wmem_default, которая используется другими протоколами и, как правило, tcp_wmem должна быть меньше чем /proc/sys/net/core/wmem_default. Третье значение максимальный размер буфера передачи для отдельного сокета. По-умолчанию 131072 байт, или 128 Кб. Это достаточно разумное значение и в большинстве случаев вам едва ли придется корректировать его. Однако, если вам придется его увеличивать помните о существовании переменной /proc/sys/net/core/wmem_max, которая должна быть всегда больше чем tcp_wmem. Эта переменная может дать некоторый прирост производительности на толстых каналах, если достаточно корректно используется вместе с переменными tcp_mem и tcp_rmem. tcp_wmem дает наибольший прирост производительности из этих трех переменных. Замечу при этом, что вы практически не получите никакого выигрыша в сетях со скоростью передачи менее 1 Гб. Дополнительную информацию, по настройке переменных, вы найдете в TCP Tuning Guide.

> net.ipv4.icmp_echo_ignore_broadcasts = 1

Эта переменная очень близка по смыслу к icmp_echo_ignore_all, только в данном случае будут игнорироваться ICMP сообщения, отправленные на широковещательный или групповой адрес. Вполне очевидно, почему полезно включить этот параметр защита от smurf атак. Переменная может принимать два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено).

> net.ipv4.icmp_ignore_bogus_error_responses = 1

Отдельные маршрутизаторы, вопреки стандартам, описанным в RFC 1122, отправляют фиктивные ответы в широковещательном диапазоне. Обычно эти ошибки заносятся в системный журнал. Если вы не желаете регистрировать их, то включите этот параметр и тем самым сбережете некоторый объем дискового пространства в своей системе. Переменная может принимать два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено).

> net.ipv4.icmp_ratelimit = 250

Максимальная частота генерации ICMP-пакетов с типом, указанным в icmp_ratemask (см. icmp_ratemask). Это значение задается в тиках и устанавливает временной интервал между ICMP-посылками. Таким образом,

значение 0 означает отсутствие ограничений. Обычно 1 тик равен 0.01 секунды, так значение 1 в этой переменной ограничивает скорость передачи не более 100 посылок в секунду, а значение 100 не более 1 посылки в секунду. Значение по-умолчанию 100, что означает не более 1 ICMP посылки за интервал в 100 тиков.

> net.ipv4.icmp_ratemask = 6168

Маска ICMP типов, на которые накладывается ограничение по частоте генерации переменной icmp_ratelimit. Каждый из ICMP типов маскируется своим битом. icmp_ratemask это битовая маска, где каждый ICMP тип представлен своим битом. Соответствие между символическим названием ICMP типа и его порядковым номером вы найдете в заголовочном файле netinet/ip_icmp.h (обычно это /usr/include/netinet/ip_icmp.h). За дополнительной информацией обращайтесь к RFC 792 - Internet Control Message Protocol. Математически маска определяется так: где n принимает значения всех типов ICMP, которые должны быть ограничены. Например: Ограничим передачу ICMP Destination Unreachable сообщений. В /usr/include/netinet/ip_icmp.h этому типу соответствует число 3. Подсчитаем значение 23, что равно 8. Теперь нужно прибавить это число к имеющейся битовой маске. Допустим, что маска уже равна числу 6160 (в двоичном виде 0001100000010000), тогда результирующая маска получится: 6160 + 8 = 6168 (в двоичном виде 0001100000011000). Перед добавлением маски подобным образом убедитесь сначала, что нужный вам бит сброшен, иначе вы получите неверную маску! К примеру, если у вас маска является числом 256 и вы еще добавите число 256, то это приведет к размаскированию ICMP Echo Request и маскировке 9-го отсутствующего типа ICMP. Значение по-умолчанию 6168 (в двоичном виде 0001100000011000), что подразумевает наложение ограничений на ICMP Destination Unreachable, ICMP Source Quench, ICMP Time Exceeded и ICMP Parameter Problem, где ICMP Destination Unreachable = 3, ICMP Source Quench = 4, ICMP Time Exceeded = 11 и ICMP Parameter Problem = 12. Таким образом, значение по-умолчанию соответствует выражению: Злоумышленник может заставить некоторый маршрутизатор или хост затопить жертву ICMP-посылками, передаваемыми в ответ на поддельный ICMP пакет с обратным адресом жертвы. Поэтому очень важно ограничить частоту генерации отдельных видов ICMP-сообщений. На сайте http://www.frozentux.net вы сможете найти небольшую программу ratemask, которая может оказаться полезной при создании маски для переменной icmp_ratemasks или для дешифрации существующей маски.

> net.ipv4.igmp_max_memberships = 20

Максимальное число групп на каждый сокет. Значение по-умолчанию 20 и может быть изменено по мере необходимости.

> net.ipv4.conf.all.accept_redirects = 0

Переменная управляет приемом ICMP-сообщений о переадресации. Сообщения ICMP Redirect используются для уведомления маршрутизаторов или хостов о существовании лучшего маршрута движения пакетов к заданному хосту, который (маршрут) может быть быстрее или менее загружен. Переменная может иметь два значения 0 (выключено сообщения о переадресации игнорируются) и 1 (включено сообщения о переадресации принимаются). Значение по-умолчанию 1 (включено), однако я посоветовал бы отключать эту опцию, поскольку она далеко небезопасна. В подавляющем большинстве случаев необходимость в переадресации отсутствует, поэтому лучше держать эту переменную выключенной, если конечно вы на 100% не уверены в обратном.

> net.ipv4.conf.all.accept_source_route = 0

Переменная разрешает/запрещает маршрутизацию от источника. Маршрутизация от источника весьма небезопасна. Дополнительную информацию по этой темы вы сможете почерпнуть из ip-param.txt. Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 1 (включено).

> net.ipv4.conf.all.arp_filter = 0

Включает/выключает связывание IP-адреса с ARP-адресом. Если эта опция включена, то ответ будет передаваться через тот интерфейс, через который поступил запрос. В принципе, было бы не плохо, если бы ответы исходили через тот же интерфейс, через который был получен запрос, однако, в отдельных случаях, это может стать причиной ошибок. Обычно включение этой опции необходимо только в том случае, если на вашем хосте производится управление распределением нагрузки между сетевыми интерфейсами. Значение по-умолчанию 0 (выключено), поскольку эта опция идет немного вразрез с современным пониманием принципов IP-адресации. Ранее IP-адреса рассматривались как путь к некоторому устройству, в смысле аппаратуры, теперь же их следует рассматривать как отдельную службу доставки, которая должна выдавать ответы на запросы вне зависимости от того на какой интерфейс эти запросы были получены. Дополнительную информацию по данной тематике вы найдете в Guide to IP Layer Network Administration with Linux.

> net.ipv4.conf.all.bootp_relay = 0

Переменная разрешает/запрещает форвардинг пакетов с исходящими адресами 0.b.c.d. Демон BOOTP relay должен перенаправлять эти пакеты на корректный адрес. Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено). Обработка переменной

bootp_relay еще не реализована. Если вы желаете предложить свою реализацию милости просим! С этой целью можете вступить в контакт с командой разработчиков netdev mailinglist.

> net.ipv4.conf.all.log_martians = 0

Переменная включает/выключает функцию журналирования всех пакетов, которые содержат неправильные (невозможные) адреса (так называемые martians марсианские пакеты). Под невозможными адресами, в данном случае, следует понимать такие адреса, которые отсутствуют в таблице маршрутизации. В некоторых ситуациях эта опция позволит получить дополнительную информацию, но вы должны понимать, что эта информация не так подробна как можно было бы подумать. Основными причинами, порождающими записи в системном журнале, могут быть: невозможность переадресации; плохая классификация; ограничения на широковещательные сообщения, или несоответствия в Forwarding Information Base (FIB). Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено).

> net.ipv4.conf.all.mc_forwarding = 0

Включает/выключает поддержку маршрутизации групповых рассылок для заданного интерфейса. Кроме того, чтобы иметь поддержку маршрутизации групповых рассылок, необходимо собрать ядро с включенной опцией CONFIG_MROUTE. Дополнительно в системе должен иметься демон, осуществляющий групповую маршрутизацию. Его вы можете получить с FTP-сайта AT&T Research.. Этот демон реализует протокол DVMRP (от англ. Distance Vector Multicast Routing Protocol протокол маршрутизации групповых рассылок типа вектор-расстояние). Еще один демон маршрутизации групповых рассылок доступен на сайте PIMd. Это реализация разреженного протокола PIM (от англ. Protocol Independent Multicast протокол маршрутизации групповых рассылок, независимый от используемого протокола маршрутизации), или PIM-SM. Там же вы найдете ссылки на другие реализации протоколов PIM-DM (Protocol Independent Multicast-Dense Mode протокол маршрутизации групповых рассылок, независимый от используемого протокола маршрутизации, уплотненного режима) и PIM-SM (Protocol Independant Multicast-Sparse Mode протокол маршрутизации групповых рассылок, независимый от используемого протокола маршрутизации, разреженного режима). Дополнительную информацию по групповой адресации вы найдете в Multicast HOWTO. Групповая адресация используется в тех случаях, когда необходимо выполнить доставку информации сразу к нескольким пунктам назначения. Например, WEB-страничка передается отдельно каждому, кто ее запросил, а если несколько человек решили принять участие в видеоконференции? Есть два пути реализации доставки. Либо каждому участнику отдавать отдельный поток данных (тогда трафик будет таким огромным, что для него может просто не хватить пропускной

способности канала), либо использовать групповую рассылку. В этом случае отправитель создает одну дейтаграмму с групповым адресом назначения, по мере продвижения через сеть дейтаграмма будет дублироваться только на развилках маршрутов от отправителя к получателям. Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено). Обратите внимание нет никакой необходимости включать эту опцию, если вы желаете лишь получать групповые пакеты. Она необходима только если вы собираетесь перенаправлять групповой трафик через вашу систему.

> net.ipv4.conf.all.proxy_arp = 0

Включает/выключает проксирование arp-запросов для заданного интерфейса. ARP-прокси позволяет маршрутизатору отвечать на ARP запросы в одну сеть, в то время как запрашиваемый хост находится в другой сети. С помощью этого средства происходит обман отправителя, который отправил ARP запрос, после чего он думает, что маршрутизатор является хостом назначения, тогда как в действительности хост назначения находится на другой стороне маршрутизатора. Маршрутизатор выступает в роли уполномоченного агента хоста назначения, перекладывая пакеты от другого хоста. Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено). Дополнительную информацию вы найдете в Proxy-ARP mini HOWTO.

> net.ipv4.conf.all.rp_filter = 1

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.

> net.ipv4.conf.all.secure_redirects = 1

Включает/выключает режим безопасной переадресации. Если переменная выключена, то будут приниматься любые сообщения ICMP Redirect от любого хоста из любого места. Если переменная включена, то сообщения о переадресации будут восприниматься только от тех шлюзов (gateways), которые имеются в списке шлюзов по-умолчанию. С помощью этой опции можно избежать большинства ложных переадресаций, которые могут быть использованы для перехвата трафика. Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 1 (включено). Обратите внимание действие этой переменной отменяется переменной shared_media, так что, если вы включаете secure_redirects, то необходимо включить и shared_media.

> net.ipv4.conf.all.send_redirects = 1

Включает/выключает выдачу ICMP Redirect другим хостам. Эта опция обязательно должна быть включена, если хост выступает в роли маршрутизатора любого рода. Как правило ICMP-сообщения о переадресации отправляются в том случае, когда необходимо сообщить хосту о том, что он должен вступить в контакт с другим сервером. Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 1 (включено). Если компьютер не выступает в роли маршрутизатора, то эту переменную можно отключить.

> net.ipv4.conf.all.shared_media = 1

Включает/выключает признак того, что физическая сеть является носителем нескольких логических подсетей, например, когда на одном физическом кабеле организовано несколько подсетей с различными сетевыми масками. Этот признак используется ядром при принятии решения о необходимости выдачи ICMP-сообщений о переадресации. Переменная может иметь два значения 0 (выключено) и 1 (включено). Значение по-умолчанию 0 (выключено). Эта переменная перекрывает действие переменной secure_redirects.

> net.ipv4.route.error_burst = 1250

Переменная используется в паре с error_cost для ограничения количества генерируемых сообщений ICMP Unreachable. Эта переменная несет в себе смысл верхнего предела стоимости передачи сообщений, в то

время как error_cost обозначает цену одного сообщения. Когда error_burst опустошается, то передача сообщений ICMP Unreachable прекращается. Сообщения ICMP Unreachable обычно отсылаются тогда, когда невозможно определить дальнейший маршрут движения пакета. Этому могут быть три причины: 1. Невозможно выполнить передачу хосту. 2. Не известен маршрут к заданному сегменту сети или хосту. 3.Если данный маршрут запрещен набором правил маршрутизации. В этих трех случаях сетевая подсистема генерирует сообщение ICMP Unreachable, свое для каждого случая: 1. ICMP Host Unreachable когда хост, находящийся в той же сети, что и наш роутер недоступен (т.е. не отвечает на ARP-запросы. прим. перев.). 2. ICMP Network Unreachable когда в таблице маршрутизации роутера нет ни одного маршрута, по которому пакет мог бы быть отправлен дальше. 3. ICMP Communication Administratively Prohibited By Filtering когда пакет не может быть переправлен из-за наличия правил маршрутизации явно запрещающих передачу. Значение по-умолчанию 500. Учитывая значение по-умолчанию переменной error_cost (100) это соответствует 5-ти сообщениям ICMP Destination Unreachables в секунду.

> net.ipv4.route.error_cost = 250

Более подробное описание см. выше error_burst. Основной смысл этой переменной цена одного сообщения ICMP Destination Unreachable. Значение по-умолчанию 100, что подразумевает передачу не более 5-ти сообщений ICMP Destination Unreachable за 1 секунду (если расчет производить с учетом значения по-умолчанию переменной error_burst).

> net.ipv4.ipfrag_high_thresh = 262144

Переменная задает максимальный объем памяти, выделяемый под очередь фрагментированных пакетов. Когда длина очереди достигает этого порога, то обработчик фрагментов будет отвергать все фрагментированные пакеты до тех пор, пока длина очереди не уменьшится до значения переменной ipfrag_low_thresh. Это означает, что все отвергнутые фрагментированные пакеты должны быть повторно переданы узлом-отправителем. Пакеты фрагментируются в том случае, если их размер слишком велик, чтобы быть переданными по данному каналу. Узел-отправитель, в этом случае, режет пакеты на более мелкие части и затем передает их одну за другой. На узле-получателе эти фрагменты опять собираются в полноценный пакет. Примечательно, что идея фрагментации пакетов, сама по себе, вещь замечательная, но, к сожалению, может быть использована в весьма неблаговидных целях. Переменная содержит целое число в диапазоне 0 .. 2147483647 и означает верхний порог длины очереди фрагментов в байтах. Значение по-умолчанию 262144 байт, или 256 Кб. Этого количества, как правило, вполне достаточно даже для самых крайних случаев.

оригинал: http://linuxopen.ru/2008/10/31/opisanie-nekotorykh-sysctl-peremennykh.html

{jcomments on}