Основы iptables для начинающих. Часть 1. Общие вопросы

interface31.ru/tech_it/2020/02/osnovy-iptables-dlya-nachinayushhih-chast-1.html

Когда говорят о брандмауэре в Linux-системах в первую очередь вспоминают **iptables**, однако это не совсем верно, брандмауэром в Linux является **netfilter**, а iptables - одна из утилит для его настройки. Тем не менее именно iptables является стандартом интерфейса брандмауэра в Linux, а также созданных на его основе системах, например, RouterOS от Mikrotik. Поэтому знание iptables является обязательным для Linux-администраторов несмотря на то, что в новых дистрибутивах осуществляется постепенный переход на более новый инструмент управления - **nftables**, который, тем не менее, сохраняет совместимость со своим предшественником.

Изначально iptables может показаться довольно сложным, особенно если браться за него без достаточной теоретической подготовки. Многие начинающие администраторы раз за разом допускают одни и те же типовые ошибки, что и побудило нас к написанию данной статьи.

Если подходить к вопросу упрощенно, то можно рассматривать iptables как брандмауэр - специализированное ПО для фильтрации сетевого трафика и нас прежде всего должна интересовать практическая сторона, а именно сам процесс фильтрации. Поэтому многие моменты мы оставим за кадром, уделив больше внимания практическому применению инструмента. Но любая практика должна основываться на твердой основе теоретических знаний, необходимому минимуму которых мы посвятим эту статью.

iptables структурно состоит из **таблиц**, в которые входят **цепочки**, в свою очередь содержащие наборы **правил**. Существует распространенное заблуждение, что цепочки содержат в себе таблицы, но это неверно и в ряде случаев может привести к ошибочному пониманию принципа действия тех или иных наборов правил. Следует помнить, что верхний уровень иерархии составляют именно таблицы, каждая из которых предназначена для своей цели. Всего имеется пять таблиц, рассмотрим их подробнее:

- **raw** - предназначена для обработки пакетов прежде, чем они будут переданы системе conntrack, которая занимается отслеживанием состояния соединений и принадлежностью пакетов этим соединениям. Содержит встроенные цепочки PREROUTING и OUTPUT.
- **mangle** - используется для модификации некоторых заголовков (TTL, TOS) и маркировке пакетов и соединений, содержит цепочки PREROUTING, INPUT, FORWARD, OUTPUT и POSTROUTING.
- **nat** - предназначен для преобразования адресов и портов источника и назначения пакетов, встроенные цепочки: PREROUTING, OUTPUT, POSTROUTING.
- **filter** - применяется, собственно, для фильтрации пакетов, является таблицей по умолчанию, т.е. если таблица не указана явно, то используется filter, имеет цепочки INPUT, FORWARD и OUTPUT.
- **security** - используется для работы совместно с системами принудительного контроля доступа, такими как SELinux. Встроенные цепочки INPUT, FORWARD, OUTPUT.

Обратите внимание, что названия таблиц всегда пишутся в **нижнем** регистре, а цепочек - в **ВЕРХНЕМ**.

На практике вы обычно будете использовать таблицы **filter** и **nat**, в некоторых случаях - **mangle**, поэтому таблицы **raw** и **security** мы далее рассматривать не будем.

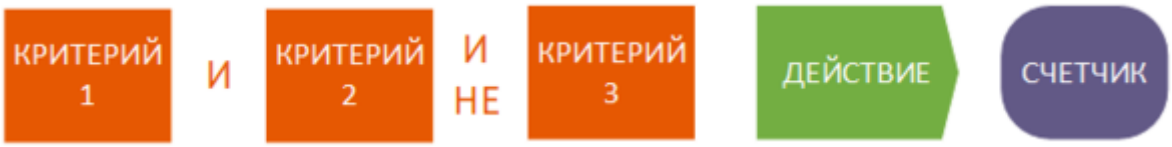
Внутри таблиц находятся цепочки. Существуют встроенные и пользовательские цепочки. Последние создаются непосредственно пользователями и могут применяться для построения сложных конфигураций обработки трафика. В отличие от встроенных цепочек их названия также принято писать в нижнем регистре.

Внутренних цепочек также пять:

- **PREROUTING** - используется для всего входящего трафика до принятия первого решения о маршрутизации
- **INPUT** - применяется для входящего трафика текущего узла
- **FORWARD** - через нее проходит транзитный трафик узла
- **OUTPUT** - применятся для исходящего трафика текущего узла
- **POSTROUTING** - используется для всего исходящего трафика после принятия всех решений о маршрутизации

После инициализации все цепочки пустые и не содержат никаких правил. Но каждая цепочка имеет действие по умолчанию (стандартно - ACCEPT), которое будет применено к пакету, если он прошел всю цепочку и не попал под действие ни одного правила.

Задача администратора - наполнить цепочки правилами. Каждое правило состоит из трех составляющих: **критерия**, **действия** и **счетчика**. Если пакет удовлетворяет критериям, то к нему применяется действие и он учитывается счетчиком. Если не указан критерий, то действие применяется ко всем пакетам. В отсутствии критерия и действия будет работать только счетчик, который считает не только количество пакетов, но и их размер в байтах.

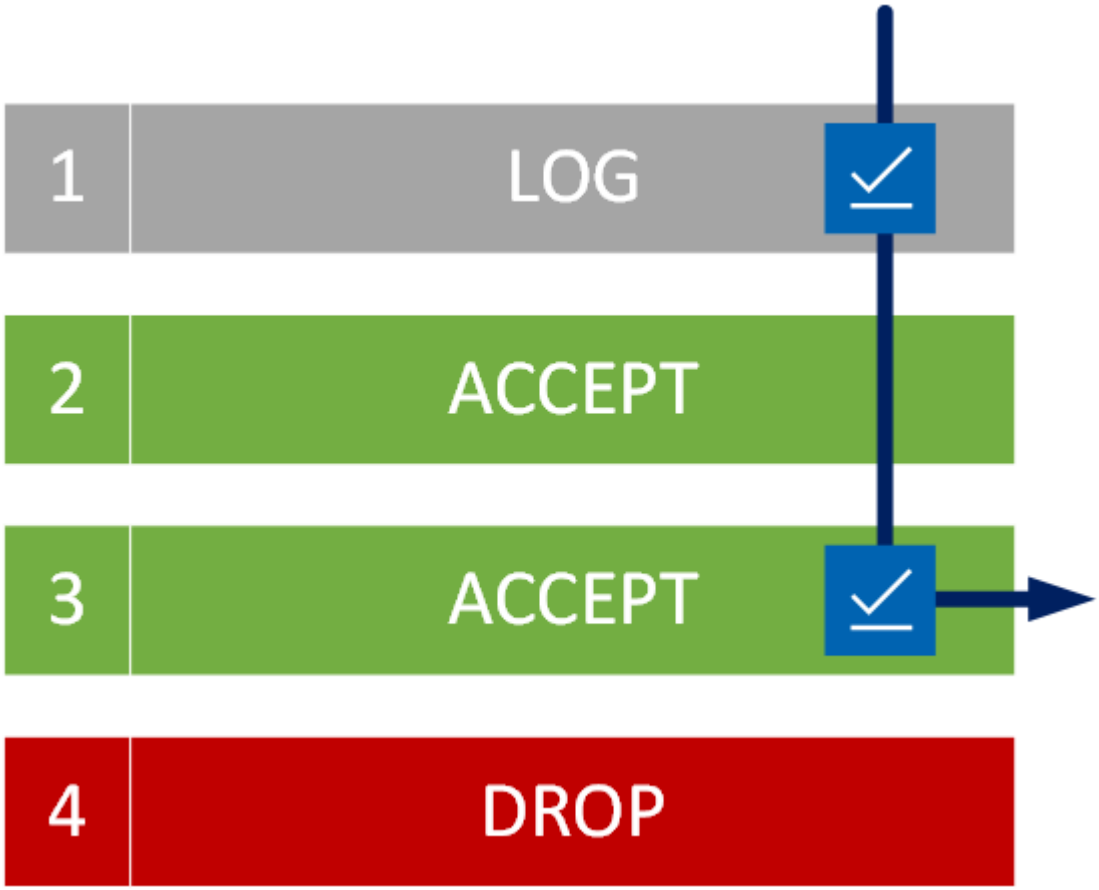


Критериев может быть несколько, они комбинируются по принципу логического И, допускается также условие с отрицанием - И-НЕ, такое выражение будет истинным, если истинными являются все условия. Таким образом правило сработает если пакет удовлетворяет **всем** перечисленным критериям. В качестве критериев используются различные свойства пакетов и/или соединений, наиболее часто используются интерфейсы, IP-адреса источника и назначения, порты.

Действие выполняет какую-либо одну операцию с пакетом, разные таблицы допускают применение разных наборов действий, например, действие **MARK** используется в таблице **mangle**, а **MASQUERADE** - в таблице **nat**. Наиболее общими действиями являются **ACCEPT**, **DROP** и **REJECT**. Первое из них пропускает пакет, а два последних блокируют, при этом **REJECT** сообщает источнику об отказе, а **DROP** нет, поэтому его чаще всего используют для блокировки внешних пакетов, чтобы не давать потенциальному злоумышленнику лишней информации о работе сети.

Все действия делятся на две категории: **терминальные** и **нетерминальные**. **Терминальные** действия прекращают прохождение пакета по цепочке, к ним относятся все специфичные действия для таблиц **filter** и **nat**, **нетерминальными** являются действия специфичные для таблицы **mangle**.

Попав в цепочку пакет последовательно проходит правила **в порядке их перечисления до первого срабатывания**. Дальнейшее его движение зависит от типа действия, если действие нетерминальное - пакет продолжит движение по цепочке, иначе - покидает ее.

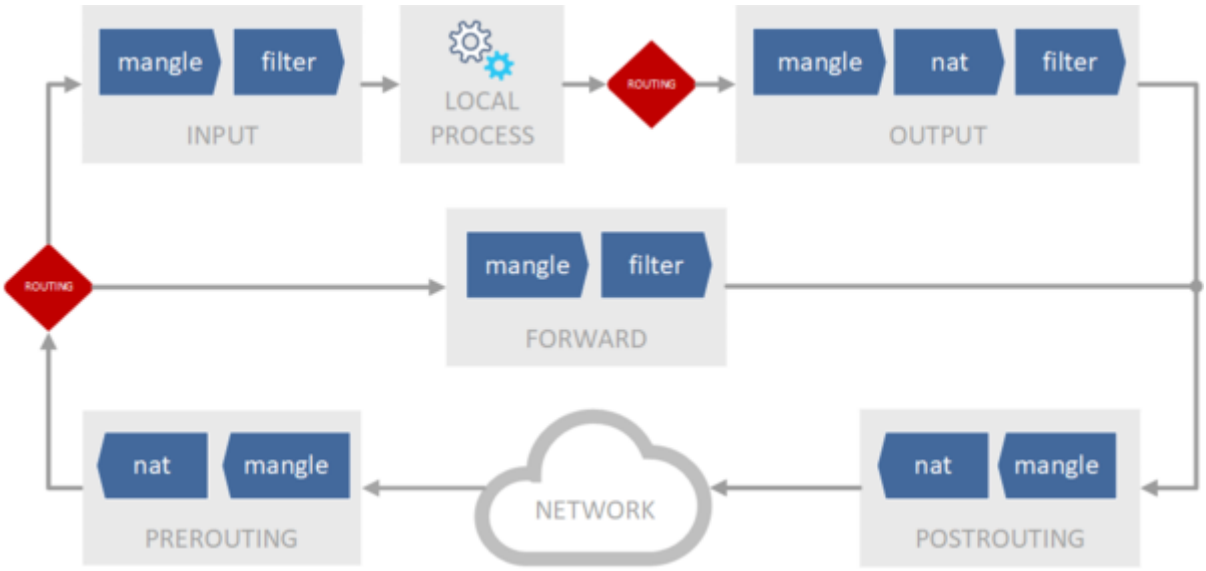


Рассмотрим схему выше. Пакет попадает в цепочку и происходит срабатывание первого правила, но так как действие LOG является нетерминальным, то пакет продолжает движение по цепочке. Следующим срабатывает правило 3, действие ACCEPT терминальное и поэтому пакет покидает цепочку.

При построении последовательности правил важно понимать, что первична именно таблица, внутри которой находится цепочка, в которой мы уже выстраиваем нужную последовательность правил. При этом правила разных таблиц, но одинаковых цепочек могут идти вперемешку, но выполняться они все равно будут в последовательности **таблица - цепочка - набор правил**.

Между тем правилом хорошего тона в администрировании является группировка правил согласно их логическому смыслу и порядку прохождения через брандмауэр, это повышает читабельность правил и позволяет легко разобраться в конфигурации брандмауэра даже увидев ее в первый раз.

Здесь мы подошли к еще одной важной теме - стандартному порядку прохождения пакетов через брандмауэр, именно эта схема часто является источником многих заблуждений, но исторически сложилось изображать ее именно так.



Глядя на нее, можно подумать, что таблицы находятся внутри цепочек, это неверно, но стандартный порядок прохождения таков, что одноименные цепочки таблиц проходятся последовательно, поэтому на схеме их объединяют для лучшего понимания логики процесса.

Итак, входящий пакет прежде всего попадает в цепочку **PREROUTING** таблицы **mangle**, затем передается в **PREROUTING** таблицы **nat**. Здесь мы можем выполнить маркировку пакетов и преобразование адреса назначения - **DNAT**. Затем принимается решение о маршрутизации, в зависимости от того предназначен пакет хосту или является транзитным будут задействованы либо цепочки **INPUT**, либо **FORWARD** таблиц **mangle** и **filter**.

Все основные действия по фильтрации пакетов производятся именно в таблице **filter**, при этом фильтрация локального и транзитного трафика производится отдельно. При составлении правил фильтрации следует учитывать, что у пакета может быть изменен адрес и порт назначения вследствие прохождения цепочки **PREROUTING** таблицы **nat**. Это актуально при пробросе портов на нестандартные номера, скажем, пришедший на адрес внешнего интерфейса и нестандартный порт пакет после выполнения над ним действия DNAT попадет в таблицу filter уже со стандартным портом и адресом внутреннего узла в качестве адреса назначения.

Локальные пакеты после таблицы **filter** передаются локальным процессам, на этом путь входящего пакета в системе заканчивается. В ответ локальный процесс формирует исходящий пакет и после решения о маршрутизации передает его в цепочки **OUTPUT** таблиц **mangle**, **nat** и **filter**. И если таблицы **mangle** и **filter** понятны, то наличие между ними таблицы **nat** требует отдельных пояснений.

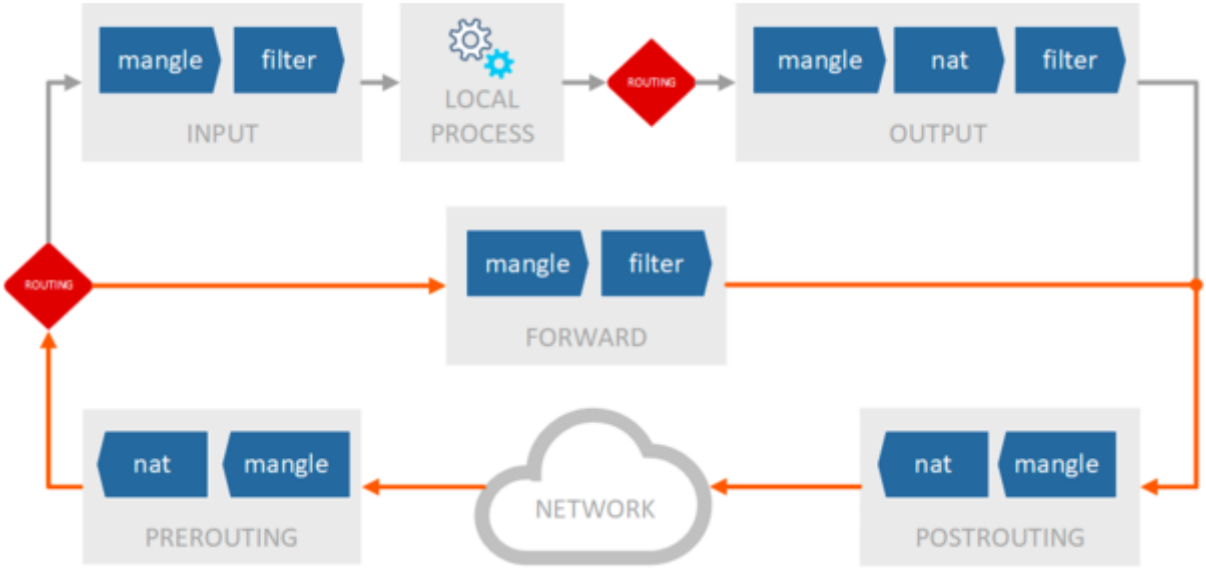
Обычно для преобразования сетевых адресов в таблице **nat** используются цепочки **PREROUTING** и **POSTROUTING**, которые закрывают большинство типовых задач. Но бывают ситуации, когда преобразование сетевых адресов нужно выполнить отдельно для исходящих пакетов по критериям, которым могут удовлетворять и

пакеты транзитные. В этом случае можно и нужно использовать цепочку **OUTPUT** таблицы **nat**. Хотя это достаточно редкий сценарий и в большинстве типовых случаев цепочки **OUTPUT** могут совсем не содержать правил.

После принятия всех решений о маршрутизации как локальные, так и транзитные пакеты попадают в цепочки **POSTROUTING** таблиц **mangle** и **nat**, здесь производятся окончательная обработка исходящих пакетов. Наиболее часто используется таблица **nat** и действия **SNAT** и **MASQUERADE**, используемые для изменения адреса источника пакета. Никакой фильтрации пакеты после этого не проходят. Это также важно понимать при построении правил.

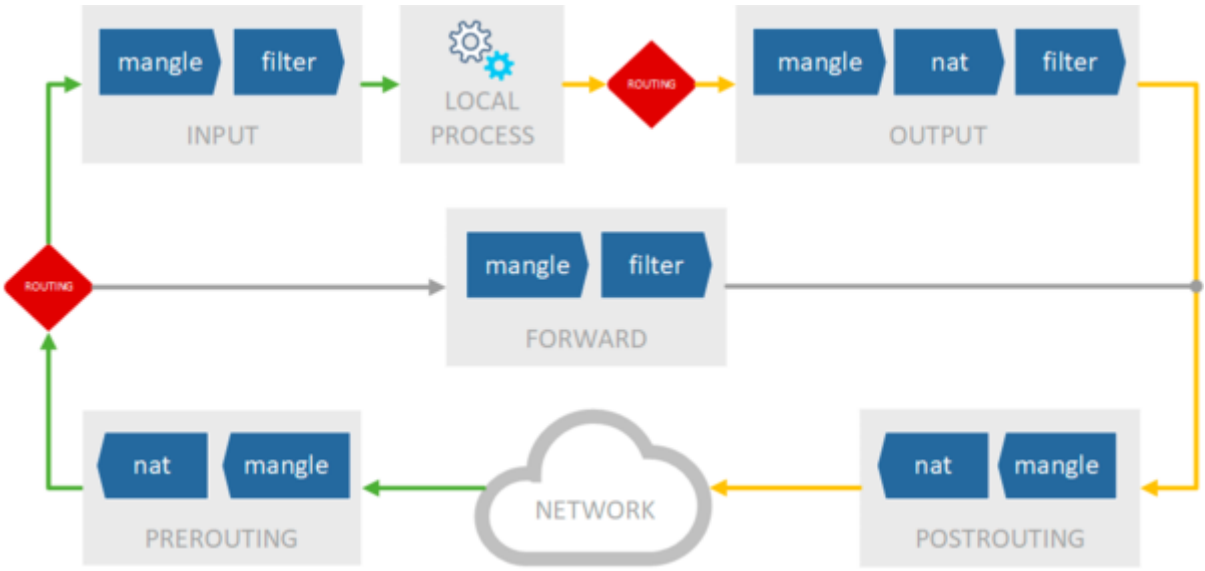
Вне зависимости от типа и направления трафика следует помнить, что преобразование адресов назначения всегда производится до того, как пакет попадает в таблицу **filter** брандмауэра, а преобразование адресов источника после ее прохождения. Непонимание этого момента приводит к использованию неправильных критериев и неработоспособности на первый взгляд "верных" правил.

И еще раз закрепим стандартный порядок прохождения пакетов через брандмауэр, для транзитного трафика он выглядит так:



Самое важное, что следует понять из этой схемы, что транзитный трафик никогда не попадает в цепочки **INPUT** и **OUTPUT**, а преобразование адреса назначения (**DNAT**) выполняется перед фильтрацией.

Для локальных пакетов стандартный порядок будет следующим:



Прежде всего следует запомнить, что локальный трафик никогда не попадает в цепочки **FORWARD**. Также, в отличии от транзитного трафика, локальный представлен двумя видами пакетов: входящими и исходящими. Путь входящего пакета ограничивается цепочками **PREROUTING** и **INPUT**, а исходящего **OUTPUT** и **POSTROUTING**, т.е. если нам нужно выполнить преобразование адреса назначения (DNAT) для локального пакета, то мы должны сделать это в цепочке **OUTPUT**, потому что в **PREROUTING** такой пакет никогда не попадет.

Тоже самое касается и маркировки пакетов, если мы хотим обрабатывать ответы на входящие пакеты по каким-то собственным критериям, то маркировать нужно не пакеты, а соединения. Потому как путь локального пакета заканчивается локальным процессом, однако и входящий и исходящий пакет будут принадлежать одному соединению. Это же касается и транзитного трафика, маркировка пакета будет учитываться только при прохождении в одну сторону, маркировка соединения - для прохождения в обоих направлениях.

Несмотря на то, что приведенная нами схема является упрощенной, надеемся, что она поможет получить базовые знания о работе iptables и не допускать грубых ошибок при составлении правил, а также лучше понимать уже готовые конфигурации брандмауэра, которые мы приводим в наших решениях.