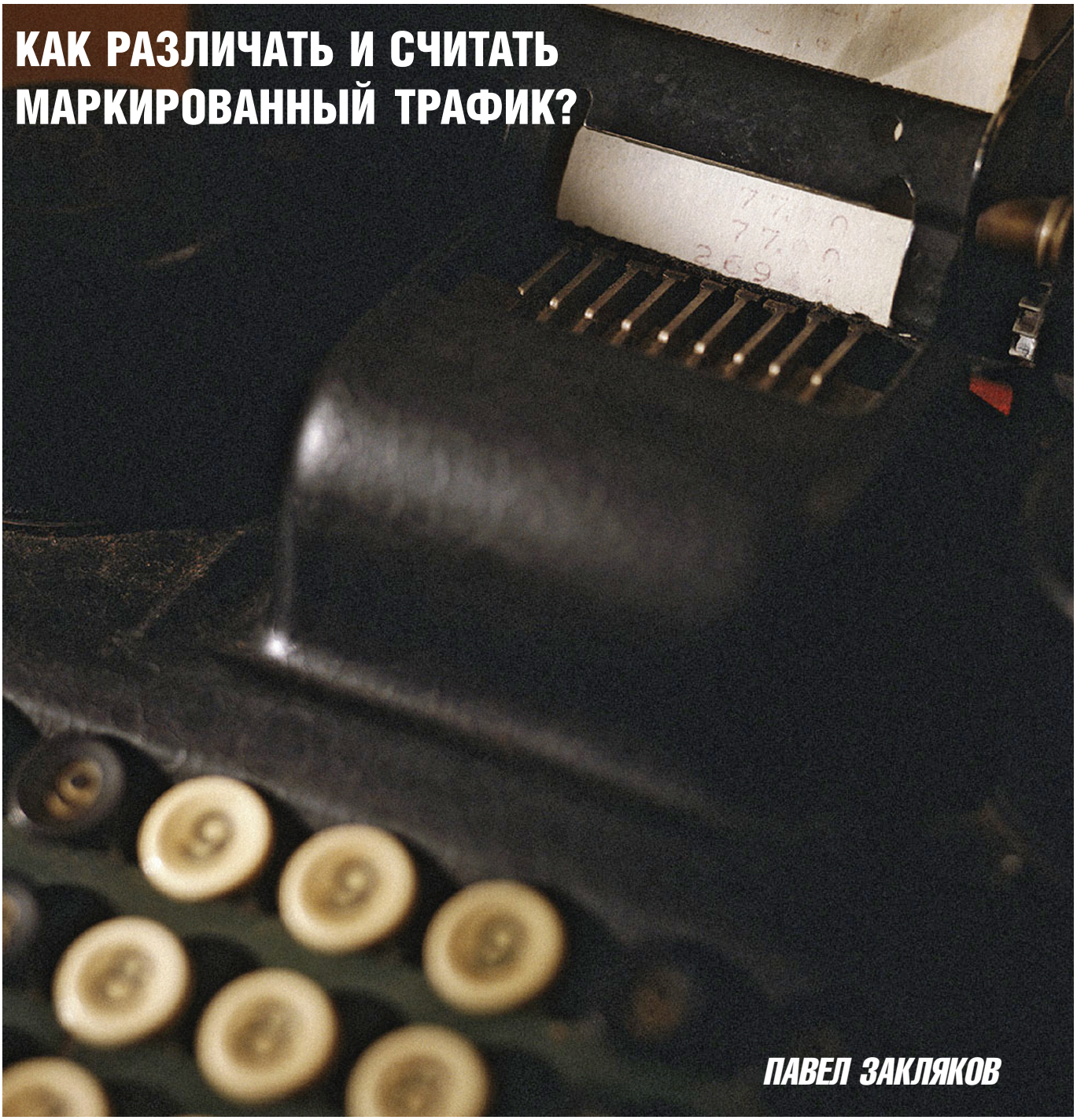
КАК РАЗЛИЧАТЬ И СЧИТАТЬ МАРКИРОВАННЫЙ ТРАФИК?



ПАВЕЛ ЗАКЛЯКОВ

В предыдущей статье о пиринге [2] я специально не затрагивал технических вопросов, неизбежно возникающих при раздельной тарификации. Сейчас я хотел бы в двух словах рассказать, что такое маркировка, зачем она нужна, когда и где может пригодиться. А также поделиться опытом подсчёта маркированного трафика.

Подобно тому, как биологи и химики в своих химических реакциях используют помеченные атомы (нуклиды, а точнее, изотопы[1]) для выяснения в подробностях, какие именно из них участвуют в реакции, и в какие вещества они переходят, в сети используют аналогичную схему с маркировкой с целью выяснения маршрутов и разделения различных видов трафика.

Если выяснение перемещения атомов – задача скорее научная, малоприменимая на практике, то вопрос с трафиком более чем насущен для большинства сетей и на практике очень актуален. С философской точки зрения получается интересная ситуация, когда конечным пользователям не очень интересен вопрос с маркировкой. Что и говорить, если кругозор большинства пользователей обычно не распространяется дальше, чем IP-адрес их компьютера, а их компьютеру, использующему в большинстве случаев ОС Windows, вообще не ведомо такое понятие, как маркировка. Но в то же время, эти же пользователи являются зависимыми от маркировки и маршрутизации, так как большинство из них привыкло экономить, а цена и другие параметры трафика напрямую зависят от маршрута.

Любой уважающий себя владелец сети, желая повысить её надёжность, как только появляется такая возможность, создаёт резервные каналы на ответственных участках. В графе сети появляются петли, а это означает, что на этих участках уже не существует однозначного маршрута между двумя узлами. Проблема появляется в тот момент, когда выясняется, что себестоимость доставки трафика по различным граням графа оказывается разной. Помимо стоимости у граней (маршрутов) имеются и другие важные характеристики. Вот некоторые из них: задержка, надёжность передачи, пропускная способность.

Можно точно сказать, что пиринг неразрывно связан с поставленной проблемой. Допустим, имеется две сети с реальными адресами в двух соседних микрорайонах, обслуживаемых различными провайдерами. Для прохода пакетов из одной сети в другую пакеты уходят к одному провайдеру, передаются другому и попадают в соседнюю сеть. Пользователи двух сетей понимают, что даже если между провайдерами заключены пиринговые соглашения и трафик обходится в копейки, то канал между сетями, какой бы он быстрый не был, так или иначе будет узким местом. В попытках расширить это узкое место создаётся дополнительный канал и с этого момента трафик между сетями может ходить по разным маршрутам.

Если в данном примере между провайдерами нет пиринговых соглашений, то одни и те же пользователи могут быть готовы к обмену файлами через бесплатный канал, а через платные каналы провайдеров – нет. Как же быть в этой ситуации? Когда-то бесплатный шлюз между сетями работает, а когда-то нет. Открыть свой ftp для скачивания и закачивания адресам другой сети – значит привлечь к себе дополнительный трафик в те моменты, когда шлюз между сетями по каким-то причинам не работает или находится на профилактике. Не разрешать доступ к открытому ftp из другой сети – лишить себя возможности получить новые файлы.

На уровне провайдеров проблемы нет, так как его основные маршрутизаторы, принимая трафик, всегда видят откуда (с какого интерфейса) он был получен, поэтому и могут его правильно подсчитывать. Проблемы начинаются у пользователей, которые, наоборот, не знают по какой цене им учитывать те или иные пакеты. Тут однозначно работает придуманное мной правило (из разряда следствий закона Мёрфи): «меньше знаешь – больше платишь».

Узнав, откуда приходит трафик, можно нежелательный ограничить и платить меньше.

Логичным и давно используемым решением данной проблемы служит маркировка трафика. Под маркировкой следует понимать изменение значений определённых битов в пакетах в соответствии с принятыми договорённостями. По идее можно менять что угодно, как угодно и где угодно, однако в большинстве случаев это негативно скажется на правильной работе тех или иных уже существующих протоколов, поэтому согласно RFC791 (Internet Protocol) было решено использовать для целей маркировки только один, второй байт в заголовке IP-пакетов, именуемый ToS, сокращённо от Type of Service, изменения в рамках которого должны пониматься всеми корректно.

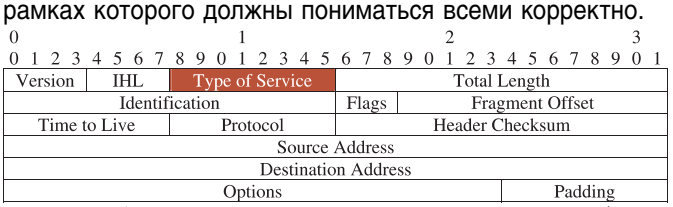


Рисунок 1. Заголовок IP-пакета. Красным выделено поле, обычно используемое для маркировки пакетов

Согласно тому же самому RFC791 выделенное красным цветом поле ToS подразделяется на несколько подполей, которые имеют следующее предназначение для используемых в них битах.

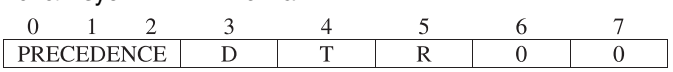


Рисунок 2. Разделение поля ToS на биты согласно RFC791

Первые три бита с 0 по 2 – это поле Precedence. Всего может быть 8 комбинаций значений этих битов:

- 111 – Network Control
- 110 – Internetwork Control
- 101 – CRITIC/ECP
- 100 – Flash Override
- 011 – Flash
- 010 – Immediate
- 001 – Priority
- 000 – Routine

Несмотря на то что согласно рекомендации поле Precedence входит в состав поля ToS, иногда под полем ToS подразумевают всё, что идёт после поля Precedence, не включая его и зарезервированные 2 бита, а встречающиеся далее в нём биты имеют следующее предназначение. Бит 3 (буква D (Delay) на рис. 2) отвечает за задержку трафика и рекомендует в случае, если в этом поле стоит 1 минимизировать задержку (0 = Normal Delay, 1 = Low Delay). Бит 4 (буква T (Throughput) на рис. 2) отвечает за пропускную способность: 0 – обычная (Normal Throughput), 1 – по возможности, повышенная (High Throughput). Бит 5 отвечает за надёжность передачи: 0 – обычная надёжность (Normal Reliability), 1 – повышенная (High Reliability). Биты 6 и

[1] Изотопы: (от изо... и греч. topos – место), разновидности одного химического элемента, занимающее одно место в периодической системе элементов Менделеева, но отличающиеся массами атомов. Химические свойства атомов, т.е. принадлежность атома к тому или иному химическому элементу, зависят от числа электронов и их расположения в электронной оболочке атома. Место химического элемента в периодической системе элементов определяется его порядковым номером Z, равным числу электронов в оболочке атома или, что то же самое, числу протонов, содержащихся в атомном ядре. Кроме протонов, в ядро атома входят нейтроны, масса каждого их которых приблизительно равна массе протона. Количество нейтронов N в ядре атома с данным Z может быть различным, но в определённых пределах... От соотношения чисел протонов и нейтронов в ядре зависят различные свойства атомов. Атомы с одинаковым Z, но с различным числом нейтронов N обладают идентичными химическими свойствами... и также называются изотопами. Большая Советская Энциклопедия, том 10, М.: Издательство «Советская энциклопедия»,1972, стр 109.

7 не используются и зарезервированы для будущих целей.

Замечание. Согласно RFC 792 (Internet Control Message Protocol), так как у ICMP-пакетов используется заголовок, идентичный протоколу IP, то в них также может использоваться маркировка. Программа sing [9] не только показывает значение поля ToS приходящих пакетов, но и позволяет с помощью параметров менять это поле у отправляемых ею пакетов. Использование маркировки IP-пакетов в поле ToS было задумано в 1979 году с самого начала появления этого протокола, определяемого устаревшим на сегодня документом IEN 123.

Появившиеся вместо него документы RFC 760 (1980 г.) и RFC 791 (1981 г.) не изменили ситуацию существенным образом.

Последний документ действует и поныне, однако через 10 лет после появления протокола IP вышел RFC 1122 (Requirements for Internet Hosts – Communication Layers), который включил два зарезервированных бита в TOS, и разделение полей стало выглядеть так:

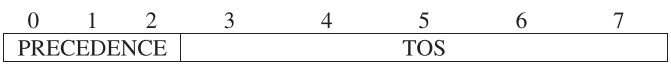


Рисунок 3. Разделение поля ToS на биты согласно RFC 1122

Чуть позже, когда назначение сети Интернет сместилось в коммерческую область, да и многие другие вещи стали замещаться денежными эквивалентами, в 1992 году к вопросу о маркировке пакетов вышло дополнение – RFC 1349 (Type of Service in the Internet Protocol Suite), согласно которому у поля ToS был отобран последний бит, и оно получило следующую трактовку битов.

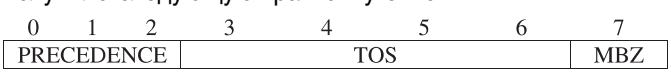


Рисунок 4. Разделение поля ToS на биты согласно RFC 1349

По сравнению с предыдущей редакцией из двух последних неиспользуемых битов (6-го и 7-го) только 6-му биту было дано новое определение – минимизация возможной стоимости передачи трафика. Если 6-й бит равен 1, то следует выбирать более дешёвые маршруты.

В 1993 году вышел RFC 1455, который предлагал вариант использования поля TOS (биты 3-6) для задания гарантированного уровня безопасности соединений.

С появлением IP-телефонии и других сервисов, ставших доступными в сети, в декабре 1998 года вышел RFC 2474 (Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers), согласно которому значение поля ToS целиком должно быть заменено полем DS (for differentiated services). После чего значения поля ToS, отмеченного на рис. 1, приобрели следующие значения.

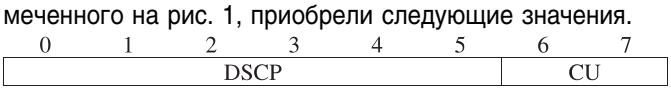


Рисунок 5. Разделение поля DS (бывш. ToS) на биты согласно RFC 2474
DSCP: differentiated services codepoint
CU: currently unused

Свято место пусто не бывает, через месяц, в январе 1999-го, неиспользуемые два бита CU изменили своё название на ECN и получили другое назначение, которое стал определять документ RFC 2481 (A Proposal to add Explicit Congestion Notification (ECN) to IP).

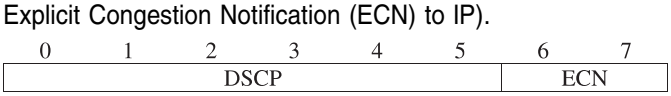


Рисунок 6. Разделение поля DS (бывш. ToS) на биты согласно RFC 2481

В сентябре 2001 вышел RFC 3168 (The Addition of Explicit Congestion Notification (ECN) to IP), который был нацелен на то, чтобы внести стабильность в поле маркировки и закрепить сложившуюся ситуацию, обозначенную в устаревшем RFC 2481. Новая рекомендация заменила старые RFC 2481 и RFC 2474 и внесла поправки в некоторые другие документы.

Такая бурная история изменения рекомендаций по использованию нескольких бит в заголовках IP-пакетов сказалась на том, что используемые оборудование и программы работают каждый по своей рекомендации. Каждая программа или железка считает своё понимание проблемы единственно верным и действует по своему, субъективному наиболее правильному алгоритму.

Администраторы по большей части, если и подозревают о существовании таких скрытых возможностей, вообще с этим полем стараются не связываться без особой на то надобности. Некоторые вообще не видят никакой выгоды от использования этого поля. Всё и так прекрасно работает с их точки зрения.

Маркировка и подсчёт различных видов трафика на практике

Насколько мне известно, провайдеров, использующих раздельную тарификацию, мало [2], а использующих маркировку – вообще один – ИМТЦ «МГУ». Помимо маркировки существуют и другие способы решения проблемы раздельной тарификации. Например, упомянутая мной в [2] сеть gagarino.net, как было недавно выяснено, не использует маркировку трафика для абонентов, а использует списки доступа [3]. При этом договорённость с вышестоящим провайдером у них близка по политике к провайдеру Zenon N.S.P., когда не важно, как пришли пакеты, если они из сети, принадлежащей списку [3], то тогда они считаются российскими. Такая система разделения трафика, с точки зрения конечных пользователей наиболее благоприятна, конечно после полностью безлимитного Интернета. Даже не смотря на частичную безлимитность, пользователь, поставивший скачиваться десяток-другой фильмов на ночь из российского сегмента сети, может спать спокойно. Если вдруг на каких-то участках сети нарушится связанность и пакеты начнут приходить из-за рубежа, то на тарификации это не отразится.

Подсчёт скачанного трафика в этом случае с помощью iptables осуществляется следующим образом. Создаются цепочки, каждая отвечающая за ту или иную российскую сеть и осуществляющая пометку принадлежащего ей трафика как российского, например семёркой.

До прохождения правил весь трафик также метится, допустим шестёркой. Пакет, прошедший через все правила, либо изменит маркировку на российскую, либо так и останется зарубежным.

Например, если российскими из всего списка [3] будут только две сети 217.26.176.0/20 и 217.146.192.0/20, а входящим интерфейсом будет eth0, то вышесказанное на языке iptables будет выглядеть так:

```
iptables -A PREROUTING -t mangle -i eth0 -j MARK --set-mark 6
iptables -A PREROUTING -t mangle -i eth0 -s 217.26.176.0/20 ↵
    -j MARK --set-mark 7
```

```
iptables -A PREROUTING -t mangle -i eth0 -s 217.146.192.0/20 ↵
    -j MARK --set-mark 7
```

Естественно, цель MARK должна поддерживаться ядром.

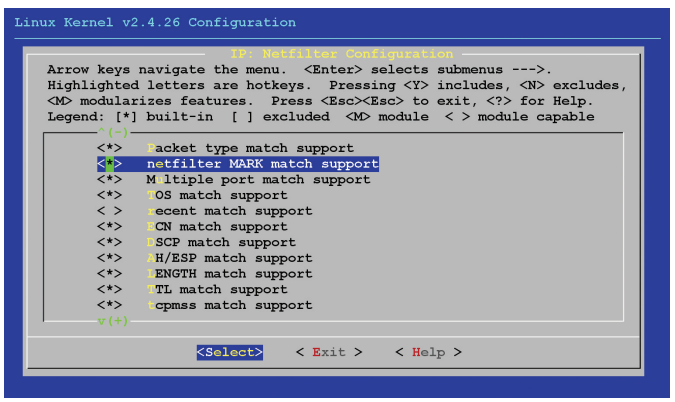


Рисунок 7. Поддержка цели MARK в ядре

После того как пакеты были помечены, их можно уже раздельно учитывать и выполнять над ними различные действия. Например, подсчёт приходящего зарубежного трафика на адрес 192.168.0.5 можно осуществить следующим правилом:

```
iptables -A FORWARD -d 192.168.0.5 -m mark --mark 6
```

а российского:

```
iptables -A FORWARD -d 192.168.0.5 -m mark --mark 7
```

эти же самые метки трафика можно скармливать и другим программам вроде tc для задач выборочного ограничения пропускной способности, например:

```
tc filter add dev eth0 parent 1:0 prio 0 protocol ip ↵
    handle 6 fw flowid 1:6
tc filter add dev eth0 parent 1:0 prio 0 protocol ip ↵
    handle 7 fw flowid 1:7
```

С финансовой точки зрения для провайдера данная схема не очень удобна, особенно если пользователей у провайдера тысячи. В этом случае лучше разграничить ответственность таким образом, чтобы проблемы нарушения связанности каких-то участков сети переложить на плечи пользователей. А именно, провайдер не гарантирует идентичность маршрутов доставки трафика от одних узлов к другим и не занимается вопросами минимизации стоимости. Если пакеты от некоторых российских узлов по каким-то причинам начинают приходить с зарубежного интерфейса и их стоимость значительно увеличивается, то в этом случае затраты пользователей растут. Минимизировать затраты один раз и навсегда невозможно. Пользователи, если желают экономить, должны сами ограничивать нежелательный трафик, выделяя его из общего числа косвенными способами, например, отслеживая маршруты движения пакетов с помощью traceroute, либо анализируя запоздавшую статистику уже пришедших пакетов. Такая схема используется некоторыми провайдерами, упомянутыми мной в [2]. Нельзя сказать, чтобы это было совсем нечестно по отношению к конечному пользователю, тут есть своя логика. Несомненно, более правильно при такой схеме дополнительно для удобства пользователей маркировать, например, зарубежный трафик. ИМТЦ «МГУ» маркирует трафик в поле Precedence установкой 1-го и 2-го битов (Flash)(ToS=0x60) [4].

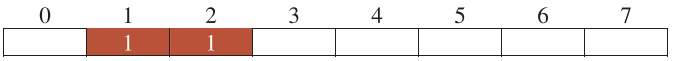


Рисунок 8. Маркировка платного трафика ИМТЦ "МГУ" в поле ToS

Недостатки этого способа в том, что при маркировке трафика провайдером затрачиваются значительные вычислительные ресурсы маршрутизаторов. Возможно, поэтому не все провайдеры используют маркировку, так как просто не могут себе это позволить.

Если вы изменили какие-то биты в пакете, то необходимо пересчитать и изменить контрольную сумму. А теперь представьте, что пакетов миллионы, в этом случае не так-то просто сохранить пропускную способность маршрутизатора, промаркировав при этом несколько терабайт пакетов за месяц.

Попытка выяснить в ИМТЦ «МГУ», кому и почему пришла в голову идея маркировать трафик так, а не по-другому, результатов не дала. «Так исторически сложилось, – был их ответ, – работавшему на то время администратору маркируемые биты, видимо, показались наиболее правильными, а после его ухода за ненадобностью никто ничего не менял».

При маркировке трафика решить проблемы ограничения скачивания чего-то лишнего гораздо проще, чем без таковой. А именно, если поставить скачиваться какой-нибудь фильм на ночь из российской части сети и установить фильтр на запрет прихода любых пакетов из зарубежной части, то можно смело сказать, что вместо скачанных гигабайт в случае изменения маршрутизации вы максимум получите несколько лишних килобайт платного трафика.

При подсчёте внешне маркированного трафика с помощью iptables первое, что приходит в голову – это воспользоваться расширением dscp:

```
iptables -A FORWARD -d 192.168.0.5 -m dscp --dscp 0x18
```

Это правило будет считать все пакеты, у которых установлены два бита маркировки, за исключением тех, у которых почему-то оказались поднятыми и другие биты, кроме отведённых двух.

Например, если поле dscp будет содержать значение 19 (биты 01 1001), то формально биты платности будут в таком пакете установлены, но считаться вышеустановленным правилом он уже не будет. Получится, что для подсчёта всех возможных случаев потребуется 16 правил (оставшихся битов 4, 2^4=16), что не очень-то и удобно, если учесть, что эти правила надо будет создавать на каждый адрес. От того, чтобы не писать так много правил, можно застраховаться. Например, перед проверкой дополнительно потребовать, чтобы пакеты, идущие на правило подсчёта, имели другие биты равными нулю.

Сделать это можно с помощью модуля ToS, реализующего более раннюю рекомендацию, биты которого не пересекаются с битами маркировки. Например:

```
iptables -N user-chain
iptables -A FORWARD -d 192.168.0.5 -m tos ↵
   --tos Normal-Service -j user-chain
iptables -A FORWARD -d 192.168.0.5 -m dscp --dscp 0x18
```

Однако толку от таких конструкций не будет, так как пакеты с ToS, не равным нулю, и так бы не посчитались последним правилом. Чтобы в правилах, описанных выше появился смысл, их надо изменить таким образом, чтобы во всех приходящих пакетах насильно в поле ToS выставлялось нулевое значение (Normal-service). Например, так:

```
iptables -t mangle -A PREROUTING -j TOS --set-tos 0
iptables -t mangle -A FORWARD -j TOS --set-tos 0
```

Тогда подсчитывающему правилу:

```
iptables -A FORWARD -d 192.168.0.5 -m dscp --dscp 0x18
```

не понадобятся никакие другие усложняющие конструкции. Этот метод прост и хорош, и, возможно, даже не потребует перекомпиляции ядра, так как большинство ядер, по умолчанию поставляемых с системами, уже знает про используемые нами модули и осуществляет их поддержку. (Для тех, кто будет компилировать ядро самостоятельно, поддержку DSCP и TOS следует включить, см. рис. 7.) Но в то же время такой способ решения проблемы неудачен, потому что требует очень больших ресурсов. Если пользователь Вася настраивает правила у себя на домашнем компьютере, то скорее всего, как бы он ни старался, у него не получится более 100 правил. Его система заведомо справится со всеми пакетами без потерь и задержек. Но если речь идёт о маршрутизации на уровне небольшого или среднего провайдера, то все ресурсы на счету. Правил может быть больше тысячи, к тому же на сервере могут работать другие задачи. Если в каждом пакете придётся менять какие-то биты, обнулять что-то в поле ToS, то непременно придётся у всех этих пакетов пересчитывать контрольную сумму, что может значительно загрузить систему.

Поэтому лучше и красивее использовать другой способ подсчёта. Наложить на ядро патч u32 [6, 7] от patch-o-matic, написанный Don Cohen (don@isis.cs3-inc.com).

Для этого, как обычно, скачиваем последние версии ядра (http://www.ru.kernel.org/pub/linux/kernel/v2.4/linux-2.4.26.tar.gz), patch-o-matic (http://www.iptables.org/files/patch-o-matic-ng-20040302.tar.bz2) и заодно iptables (http://www.iptables.org/files/iptables-1.2.9.tar.bz2). Затем, распаковав исходники, следует наложить на них патч.

Особенность patch-o-matic состоит в том, что предлагаемые патчи, хотя и были написаны отнюдь не дураками, всё же могут содержать неумышленные ошибки. Также никто не гарантирует совместимость всех патчей между собой. Поэтому, чтобы лишний раз не понижать надёжность системы, следует устанавливать только те патчи, которые вам реально необходимы. В нашем случае это u32. Наложение патча несколько отличается от обычного синтаксиса команды patch, поэтому об этом чуть подробнее. Прочитав прилагающийся файл README, можно узнать, что следует запустить файл runme с ключом extra. Затем программа проверит значения переменных окружения, в которых указано местонахождение исходников ядра и iptables. Если они окажутся пустыми, то нам будут заданы соответствующие вопросы. После ответа на них вам будут предлагаться на выбор различные патчи с их кратким описанием и примерами. Следует выбирать «n» до тех пор, пока у вас на экране не появится запрос на установку u32.

```
Welcome to Patch-o-matic (1.14)!

Kernel:         /usr/src/linux
Iptables:       /usr/src/iptables

Each patch is a new feature: many have minimal impact, some do not.
Almost every one has bugs, so don't apply what you don't need!
-------------------------------------------------------------
Already applied:

Testing u32... not applied
The u32 patch:
   Author: Don Cohen <don@isis.cs3-inc.com>
   Status: Tested locally, seems to work.


   U32 allows you to extract quantities of up to 4 bytes from a packet,
   AND them with specified masks, shift them by specified amounts and
   test whether the results are in any of a set of specified ranges.
   The specification of what to extract is general enough to skip over
   headers with lengths stored in the packet, as in IP or TCP header
   lengths.
   Details and examples are in the kernel module source.
-------------------------------------------------------------------
Do you want to apply this patch [N/y/t/f/a/r/b/w/q/?] t
Patch u32 applies cleanly
-------------------------------------------------------------------
Do you want to apply this patch [N/y/t/f/a/r/b/w/q/?] y
```

Рисунок 9. Приглашение, предлагающее установить патч u32

После этого выбираем «t» (test), и в случае успешного тестирования устанавливаем патч выбором «y». Далее появится предложение установить следующий патч и т. д. Чтобы не выбирать много раз «n», можно сразу выйти с помощью «q» (quit). По окончании вы увидите сообщение о том, что исходники готовы к компиляции.

```
-------------------------------------------------------------
Do you want to apply this patch [N/y/t/f/a/r/b/w/q/?] q

Excellent! Source trees are ready for compilation.

[root@notebook patch-o-matic-ng-20040302]#
```

Рисунок 10. Сообщение: исходники готовы к компиляции

Далее следует выполнить конфигурацию и компиляцию ядра и модулей обычным образом. Следует обратить внимание, что после наложения патча в разделе Networking options IP: Netfilter Configuration должен появиться пункт «U32 match support», который следует выбрать.

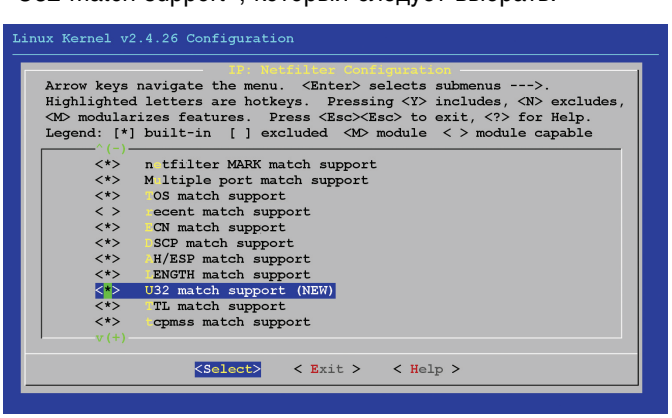


Рисунок 11. Пункт U32 match support

После установки нового ядра, а также компиляции и инсталляции iptables, можно приступать к написанию правил с использованием нового фильтра u32. К сожалению,

документация, подготовленная автором модуля (Don Cohen) и поставляемая в комплекте очень слаба, поэтому лучше воспользоваться руководством, написанным William Steams [7].

Краткий формат написания правил для u32 таков:

```
iptables ... -m u32 --u32 "Start&Mask=Range"
```

где Start – это то, что будет сравниваться, Mask – маска тех битов, которые будут сравниваться, Range – сравниваемое значение, если оно совпадёт с тем что будет в пакете, то правило «сработает».

Из-за особенностей реализации u32 в Start указывается не абсолютное значение байта в пакете, начиная с нулевого, а относительное, смещённое на -3 байта от его реального месторасположения. Например, если вы хотите фильтровать пакеты с TTL<4, то глядя на рис. 1, вы определяете, что полю TTL в заголовке соответствует 8-й байт пакета (вначале пакета идёт заголовок). После чего можно подсчитать (8-3=5), что в поле Start следует указать число 5. Для сравнения следует учитывать все биты, поэтому используемая маска будет 1111 1111 (0xFF). Значения меньше 4 – это те, что принадлежат отрезку значений [0, 3]. Полученное таким образом правило будет выглядеть так:

```
iptables ... -m u32 --u32 "5&0xFF=0:3"
```

при этом оно будет аналогично правилу:

```
iptables ... -m ttl --ttl-lt 4
```

Так как переменная Start не может быть меньше 0, то получить вышеописанным образом доступ к первым трём байтам не получится. Для доступа к ним используется более сложная конструкция со сдвигом. Подробнее об этом можно прочитать в [7]. Так как маркируемые биты находятся во втором байте заголовка (2-3=-1), то простым образом обратиться к ним не получится. Конструкция Start&Mask позволяет считать первые 4 байта сразу без сдвига с помощью «0&0xFFFFFFFF», конструкция, считывающая второй по счёту (первый по нумерации, начиная с нуля) байт выглядит так «0&0x00FF0000». Фактически это то, что нам нужно. Уже сейчас можно написать аналог применяемого нами ранее правила:

```
iptables -A FORWARD -d 192.168.0.5 -m dscp --dscp 0x18
```

который будет выглядеть так:

```
iptables -A FORWARD -m u32 --u32 "0&0x00FF0000=0x00600000" ↲
    -d 192.168.0.5
```

Это правило полностью аналогично, то есть оно содержит все те недостатки, что были обсуждены ранее. Однако теперь их можно исправить, используя для сравнения не все биты, а только те, которые нас интересуют 0110000 (0x60).

```
iptables -A FORWARD -m u32 --u32 "0&0x00600000=0x00600000" ↲
    -d 192.168.0.5
```

Такая конструкция полностью делает всё от неё ожидаемое, хотя и не очень удобна в том плане, что сравниваемое значение приходится окружать большим количеством нулей. Куда разумнее оба сравниваемых значения сдвинуть на 2 байта (16 бит) вправо, тогда лишних нулей можно будет избежать, при этом правило для iptables более элегантно можно будет записать так:

```
iptables -A FORWARD -m u32 --u32 "0&0x00600000>>16=0x60" ↲
    -d 192.168.0.5
```

Логично догадаться, что если у вас маркируются другие биты, то поменять значения в правиле не очень сложно.

Замечание 1. Вадимом Беркгаутом было обнаружено, что маршрутизаторы Cisco (в частности 7206) помечают не все пакеты в поле DSCP. Почему-то не помечаются пакеты с установленными флагами SYN или FIN и иногда некоторые пакеты, идущие перед последним. Также не всегда помечаются и ICMP-пакеты. Если на счёт SYN- и FIN-пакетов есть некоторые соображения, то по поводу случайной не маркировки некоторых ICMP-пакетов сделать выводы сложно. В частности, есть предположение, что существуют те или иные RFC, которые рекомендуют так поступать. Например, RFC 2873 (TCP Processing of the IPv4 Precedence Field), указывающий, что во время установки соединения поле precedence должно игнорироваться обеими сторонами. Это значит, что Cisco может думать так: «а зачем маркировать пакеты, если поле всё равно игнорируется». Возможно, что это особенности конкретного экземпляра, если кто-то из читателей знает, почему так происходит и как ведут себя другие маршрутизаторы, буду рад услышать ваше мнение.

Замечание 2. Если поле ToS (DSCP+ECN по современной трактовке) никак не используется в сети и не обнуляется на межсетевом экране, а при этом есть возможность свободному выходу определённых пакетов из сети (в сеть), то создаётся хорошая лазейка для скрытной передачи информации. Много передать по такому каналу не получится, но можно без особых усилий отправлять небольшие сообщения, не привлекая к происходящему процессу внимания.

Спасибо Павлу Янченко за ответы на некоторые мои вопросы по теме данной статьи.

Литература, ссылки:

1. Список документов RFC: http://www.rfc-editor.org/rfc-index.html
2. Закляков П. Пиринг. – Журнал «Системный администратор» №6(19), июнь 2003 г. – 82-87 с.
3. IP-адреса Российских сетей по данным gagariono.net: http://www.gagarino.net/ips.txt
4. ИМТЦ «МГУ» тарифы и маркировка трафика: http://www.direct.ru/internet/index_line.html
5. Classifying packets with filters: http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.qdisc.filters.html
6. Расширение u32 для iptables: http://www.iptables.org/patch-o-matic/pom-base.html#pom-base-u32
7. William Steams IPTables U32 Match Tutorial: http://www.sans.org/rr/special/iptables_u32.pdf
8. Страница со ссылками для скачивания iptables и patch-o-matic: http://www.iptables.org/downloads.html
9. Сайт проекта sing: http://sourceforge.net/projects/sing