Пакет tcpdump

http://www.tcpdump.org

Данный документ соответствует tcpdump версии 4.9.3 и libpcap версии 1.9.1.

Синтаксис

```
tcpdump [ -AbdDefhHIJKlLnNOpqStuUvxX# ] [ -B buffer_size ]
        [ -c count ]
        [ -C file_size ] [ -G rotate_seconds ] [ -F file ]
        [ -i interface ] [ -j tstamp_type ] [ -m module ] [ -M secret ]
        [ --number ] [ -Q in|out|inout ]
        [ -r file ] [ -V file ] [ -s snaplen ] [ -T type ] [ -w file ]
        [ -W filecount ]
        [ -E spi@ipaddr algo:secret,...  ]
        [ -y datalinktype ] [ -z postrotate-command ] [ -Z user ]
        [ --time-stamp-precision=tstamp_precision ]
        [ --immediate-mode ] [ --version ]
        [ expression ]
```

Программа **tcpdump**, включаемая во все дистрибутивы UNIX, выводит заголовки пакетов для указанных сетевых интерфейсов в соответствии с заданным логическим выражением. Программа также допускает использование с флагом **-w** для записи пакетов данных в файл, которым может впоследствии использоваться для анализа. Возможен и просмотр заголовков из таких файлов с помощью флага **-r**. Во всех случаях tcpdump имеет дело только с пакетами, соответствующими заданному логическому выражению (фильтру).

Tcpdump (если в команде не был указан флаг **-c**) продолжает собирать пакеты до тех пор, пока процесс не будет прерван сигналом SIGINT (например, при нажатии клавиш control-C) или SIGTERM (например, в результате команды kill(1)). Если команда используется с флагом **-c**, сбор пакетов кроме описанных выше способов может быть прекращён также после обработки определённого числа пакетов.

При завершении работы tcpdump выводит значения счётчиков:

- собранных (captured) пакетов (число пакетов, полученных и обработанных tcpdump);
- полученных фильтром (received by filter) пакетов; толкование этого значения зависит от ОС, под управлением которой работала программа tcpdump (в некоторых ОС указывается число пакетов независимо от числа совпадений с условиями фильтрации, а в других - число пакетов, соответствующих фильтрам);
- отброшенных ядром (dropped by kernel) пакетов (число пакетов, отброшенных ядром по причине нехватки ресурсов или фильтрации внутри ядра).

На платформах, поддерживающих сигналы SIGINFO (например, BSD), могут выводиться значения перечисленных выше счётчиков по сигналу SIGINFO (этот сигнал может быть подан обычно с помощью клавиш control-T) без прерывания работы команды.

Отметим, что чтение пакетов из сетевого интерфейса может потребовать от пользователя специальных привилегий в зависимости от используемой ОС:

- **SunOS 3.x** или **4.x** с NIT или BPF
  требуется доступ для чтения к файлам устройств **/dev/nit** или **/dev/bpf***.
- **Solaris** с DLPI
  требуется доступ для чтения и записи к сетевому псевдоустройству (например, **/dev/le**). На некоторых версиях Solaris таких прав недостаточно для работы **tcpdump** в режиме захвата[1]; в таких ситуациях для использования tcpdump требуются полномочия root или установка для tcpdump флага **SUID**. Отметим, что на многих (возможно, на всех) системах при работе устройства в обычном режиме вы не сможете видеть никаких исходящих пакетов, поэтому сбор данных в таком режиме может оказаться практически бесполезным.
- **HP-UX** с DLPI
  требуются полномочия **root** или установка для **tcpdump** флага **SUID**.
- **IRIX** с snoop
  требуются полномочия **root** или установка для **tcpdump** флага **SUID**.
- **Linux**
  требуются полномочия **root** или установка для **tcpdump** флага **SUID**, если ваша система не использует ядро с поддержкой битов возможностей[2] (таких, как CAP_NET_RAW). В последнем случае для вам потребуется установка бита **CAP_NET_RAW** для захвата пакетов и бита **CAP_NET_ADMIN** для просмотра списка устройств помощью опции **-D**. Для просмотра текущего состояния битов возможностей служит функция **getcap**, а для управления этими битами - **setcap** из библиотеки **libcap**. Дополнительную информацию о поддерживаемых битах возможностей вы найдёте, воспользовавшись командой **man capabilities**.
- **ULTRIX** и **Digital UNIX/Tru64 UNIX**
  всем пользователям разрешено использование программы tcpdump. Однако никому из пользователей не разрешено использовать режим захвата пакетов, пока администратор (super-user) не разрешит этот режим для данного интерфейса с помощью команды **pfconfig**. Захват принимаемых или передаваемых интерфейсом unicast-пакетов не будет возможен до тех пор, пока администратор (super-user) не включит для этого интерфейса режим

[1]**Promiscuous -** “Неразборчивый” режим, при котором драйвер устройства захватывает все передаваемые через среду пакеты. В нормальном режиме драйвер обычно читает из среды лишь пакеты, адресованные данному устройству.
[2]Поддержка “битов возможностей” (capability bit) обеспечивается в ядре Linux начиная с версии 2.2.

copy-all с помощью команды **pfconfig**. Поскольку сбор пакетов обычно требует включения обоих упомянутых режимов, реальное использование tcpdump возможно только с позволения администратора.

- **BSD** и **Mac OS X**
  требуется доступ для чтения к устройству **/dev/bpf***. На системах BSD с поддержкой **devfs** (сюда относятся и системы Mac OS X) кроме установки принадлежности и прав доступа к устройствам BPF может потребоваться настройка конфигурации **devfs**, позволяющая задавать принадлежность и права доступа всякий раз при перезагрузке системы.

Чтение собранных пакетов из файла не требует специальных привилегий.

Опции tcpdump

Программа tcpdump позволяет в командной строке задать все опции сбора и отображения пакетов, а также спецификацию фильтров захвата, описанных ниже (параграф на стр.). Таблица содержит список опций tcpdump и описание каждой их них. Отметим, что некоторые опции поддерживаются не всеми платформами.

Таблица 1. Опции командной строки tcpdump.

Опция	***Описание***
-A	Задаёт вывод каждого пакета (без заголовков канального уровня) в формате ASCII. Этот режим удобен для сбора трафика HTTP.
-b	Задаёт вывод номера автономной системы (AS) из пакетов протокола BGP в формате ASDOT вместо ASPLAIN.
-B <размер буфера> **--buffer-size=<размер буфера>**	Задаёт размер буфера захвата пакетов для операционной системы в килобайтах (1024 байта).
-c <число пакетов>	Задаёт завершение работы программы после захвата заданного числа пакетов.
-C <размер файла>	Задаёт необходимость проверки размера файла захвата перед записью в него каждого нового пакета. Если размер файла превышает указанное значение параметра, прежний файл закрывается и создаётся новый файл для записи в него пакетов. Для файлов захвата используется имя, заданное параметром **-w** и, начиная со второго файла, к имени добавляется в качестве суффикса порядковый номер файла. Параметр задаёт размер файла в миллионах байтов (не в мегабайтах = 1 048 576 байт).
-d	Задаёт вывод дампа скомпилированного кода сопоставления пакетов в понятном человеку формате и завершение работы программы.
-dd	Выводит дамп кода сопоставления в виде фрагмента C-программы.
-ddd	Выводит дамп кода сопоставления в виде строки десятичных значений с префиксом в форме значения счётчика.
-D	Выводит список сетевых интерфейсов системы, с которых tcpdump может собирать пакеты. Для каждого сетевого интерфейса указывается имя и номер, за которыми может следовать текстовое описание интерфейса. Имя и номер интерфейса могут использоваться с флагом **-i** для задания сбора пакетов с одного интерфейса. Эта опция может быть весьма полезна для систем, не дающих информации об имеющихся сетевых интерфейсах[1]. Флаг **-D** не поддерживается, если программа tcpdump была скомпилирована со старой версией libpcap, которая не поддерживает функцию pcap_findalldevs().
-e	Выводит заголовок канального уровня в каждой строке дампа.
-E <algo:secret>	Задаёт использование алгоритма и секрета spi@ipaddr для расшифровки пакетов IPsec ESP, направленных по адресу ipaddr и содержащих в поле Security Parameter Index значение spi. Комбинация spi и адреса может быть повторена с использованием в качестве разделителя запятой или новой строки. Отметим, что установка секрета для пакетов IPv4 ESP в настоящее время поддерживается. В качестве алгоритмов могут использоваться **des-cbc**, **3des-cbc**, **blowfish-cbc**, **rc3-cbc**, **cast128-cbc** или **none**. По умолчанию применяется алгоритм **des-cbc**. Возможность дешифровки пакетов обеспечивается только в тех случаях, когда при компиляции **tcpdump** были включены опции поддержки криптографии. Параметр **secret** содержит ASCII-текст секретного ключа ESP. Если секрет начинается с символов 0x, будет считываться шестнадцатеричное значение. Опция предполагает использование ESP в соответствии с RFC 2406, а не RFC 1827. Эта опция поддерживается только для отладки и использовать её с реальными секретными ключами не следует, поскольку введённый в командной строке ключ IPsec доступен другим пользователям системы[2]. Кроме явного указания параметров в командной строке их можно задать в файле опций, который tcpdump будет читать при получении первого пакета ESP.

[1]Например, Windows или UNIX-системы, в которых не поддерживается команда **ifconfig -a**
[2]Например, его можно увидеть с помощью команды **ps**.

Опция	*Описание*
`-f`	Задаёт вывод чужих адресов IPv4 в числовом формате. Использование этой опции позволяет избавиться от проблем, возникающих на серверах Sun NIS при попытках трансляции нелокальных адресов. Проверка чужеродности адреса IPv4 осуществляется с использованием адреса и маски принявшего пакет интерфейса. Если адрес и маска интерфейса недоступны (например, при использовании unnumbered-интерфейсов или при захвате пакетов со всех адресов в Linux с использованием фиктивного интерфейса **any**), эта опция будет работать некорректно.
`-F <файл>`	задаёт использование фильтров, содержащихся в указанном файле. В этом случае заданные в командной строке фильтры игнорируются.
`-G rotate`	Задаёт смену файла, заданного опцией -w каждые rotate секунд. Файлы будут именоваться в соответствии с опцией -w с добавлением временной метки в формате strtime. Если формат временных меток не задан, новые файлы будут записываться вместо предшествующего. При использовании с опцией -C имена файлов будут иметь вид file<count>
`-h` `--help`	Выводит информацию о текущих версиях tcpdump и libpcap, а также краткую справку о программе tcpdump, после чего завершает заботу.
`--vervion`	Выводит информацию о текущих версиях tcpdump и libpcap, после чего завершает заботу.
`-H`	Задаёт попытку декодировать заголовки 802.11s.
`-i <интерфейс>` `--interface=<интерфейс>`	Задаёт сбор пакетов с указанного интерфейса. Если интерфейс не задан, tcpdump ищет в системе список доступных интерфейсов и выбирает в нем активное устройство с минимальным номером (исключая loopback). В системах Linux, начиная с ядра 2.2 поддерживается фиктивный интерфейс с именем **any**, обеспечивающий сбор пакетов со всех активных интерфейсов системы. Отметим, что сбор пакетов с устройства **any** осуществляется в обычном (не promiscuous) режиме. Если в системе поддерживается флаг -D, можно в качестве аргумента задавать номер интерфейса, выводимый при использовании этого флага.
`-I` `--monitor-mode`	Переводит интерфейс в режим мониторинга, доступный лишь для IEEE 802.11 Wi-Fi в некоторых операционных системах. Следует отметить, что в этом режиме сетевой интерфейс может потерять установленную связь с беспроводной сетью и использование этой сети станет невозможным. Эта опция влияет на вывод при использовании опции -L. При выключенной опции -I будут отображаться только те типы кадров канального уровня, которые не доступны в режиме мониторинга, а при включённой опции - только доступные в режиме мониторинга.
`--immediate-mode`	Задаёт режим непосредственного захвата (immediate), когда пакеты доставляются в tcpdump без буферизации. Такое поведение принято по умолчанию при отображении пакетов без записи в файл.
`-j <тип метки>` `--time-stamp-type=<тип метки>`	Задаёт тип временных меток, отображаемых с пакетами. Имена меток задаются pcap-tstamp, но некоторые типы меток пригодны не для всех интерфейсов.
`-J` `--list-time-stamp-types`	Выводит список поддерживаемых для интерфейсов типов временных меток и на этом завершает работу.
`--time-stamp-precision=<точность меток>`	Задаёт точность временных меток при захвате пакетов, которая зависит от используемого оборудования. Следует отметить, что при записи меток с наносекундной точностью в файл не все программы смогут считывать такие метки и файлы. При чтении данных из файла эта опция задаёт преобразование использованной при записи файла точности в указанное опцией разрешение, при котором может происходить снижение точности. Поддерживаются значения micro (мискросекундная точность, принята по умолчанию) и nano (наносекундная точность).
`-K` `--dont-verify-checksums`	Отключает проверку контрольных сумм IP, TCP и UDP. Это полезно для интерфейсов, полностью или частично вычисляющих контрольные суммы на аппаратном уровне, поскольку без опции контрольные суммы TCP могут оказаться в таких случаях неверными.

`-l`	Задаёт буферизацию строк **stdout**. Эта опция полезна в тех случаях, когда нужно просматривать данные во время сбора пакетов. Например, команды
	
	`tcpdump -l | tee dat`
	или
	
	`tcpdump -l > dat & tail -f dat`
	обеспечивают запись пакетов в файл **dat** и одновременный вывод на консоль.

`-L` `--list-data-link-types`	Задаёт вывод списка известных типов канального уровня и завершение работы программы. Список может зависеть от режима и , например, на некоторых платформах часть типов Wi-Fi может не поддерживаться в режиме мониторинга, а другие могут поддерживаться лишь в этом режиме.
`-m <файл>`	Загружает модуль определений SMI MIB из указанного файла. Эта опция может использоваться неоднократно для загрузки нескольких модулей MIB.

Опция	*Описание*
-M <секрет>	Задаёт использование общего секрета для проверки цифровых подписей в сегментах TCP с опцией TCP-MD5 (RFC 2385).
-n	Отключает преобразование адресов и номеров портов в символьные имена.
-N	Задаёт использование лишь хостовой части доменного имени вместо FQDN.
-# **--number**	Задаёт вывод номера пакета в начале строки.
-O **--no-optimize**	Отключает оптимизатор кода сопоставления пакетов с фильтрами (стр. 5). Опция полезна лишь при наличии ошибок в оптимизаторе.
-p **--no-promiscuous-mode**	Указывает программе, что интерфейс не нужно переводить в неразборчивый режим[1]. Опцию **-p** нельзя использовать в сокращённой форме вместе с фильтром **ether host {local-hw-addr}** или **ether broadcast**.
-Q direction **--direction=direction**	Выбирает направление приёма или передачи для захвата пакетов и может принимать значение in, out или inout. Поддерживается не всеми системами.
-q	Задаёт вывод минимального объёма информации.
-r <файл>	задаёт чтение данных из файла, созданного ранее с использованием команды **tcpdump -w** или с помощью другой программы, поддерживающей формат tcpdump (например, Ethereal). Если в качестве имени файла задан символ **-**, используется поток данных от стандартного устройства ввода (**stdin**).
-S **--absolute-tcp-sequence-numbers**	Задаёт вывод абсолютных порядковых номеров TCP взамен относительных.
-s <snaplen> **--snapshot-length=snaplen**	Задаёт захват из каждого пакета **snaplen** байтов вместо отбираемых по умолчанию 68 байтов[1]. Значение 68 подходит для протоколов IP, ICMP, TCP и UDP, но может приводить к потере протокольной информации для некоторых пакетов DNS (стр. 15) и NFS (стр. 15). Потеря части пакетов по причине малого размера кадра захвата (snapshot) указывается в выходных данных полями вида **[|proto]'**, где **proto** - имя протокольного уровня, на котором произошла отсечка части пакета[2]. Отметим, что увеличение кадра захвата приведёт к дополнительным временным затратам на обработку пакетов и уменьшению числа буферизуемых пакетов, что может привести к потере части пакетов. Следует указывать минимальное значение **snaplen**, которое позволит обойтись без потери информации об интересующем протоколе. Установка **snaplen = 0** приведёт к захвату до 262144 байтов.
-T <тип>	Задаёт интерпретацию пакетов, выбранных с помощью фильтра (см. параграф на стр.) как пакетов указанного параметром типа. В настоящее время поддерживаются типы **aodv**[3], **carp**[4], **cnfp**[5], **lmp**[6], **pgm**[7], **pgm_zmtp1**[8], **resp**[9], **radius**, **rpc**[10], **rtp**[11], **rtcp**[12], **snmp**[13], **tftp**[14], **vat**[15], **wb**[16], **zmtp1**[17] и **vxlan**[18].
-t	Отключает вывод временных меток в каждой строке дампа.
-tt	Задаёт вывод в каждой строке дампа неформатированных временных меток.
-ttt	Задаёт вывод временных интервалов (в микросекундах) между захватом предыдущего и данного пакетов в каждой строке дампа.
-tttt	Задаёт вывод временных меток в принятом по умолчанию формате для каждой строки дампа.
-u	Задаёт вывод манипуляторов (handle) NFS без декодирования.
-U **--packet-buffered**	Задаёт режим буферизации на уровне пакетов для файлов, когда содержимое пакета отображается на стандартном устройстве вывода (и записывается в файл при наличии опции -w) без ожидания заполнения буфера. Флаг **-U** не будет поддерживаться, если программа **tcpdump** была скомпилирована со старой опцией **libpcap**, не поддерживающей функцию **pcap_dump_flush()**.

[1]Интерфейс может быть переведён в неразборчивый режим другими программами, поэтому использование флага **-p** отнюдь не гарантирует работу интерфейса в обычном режиме - программа просто не будет переводить этот интерфейс в неразборчивый режим. Кроме того, даже в обычном режиме захватываться будут не только пакеты, адресованные этому интерфейсу, поскольку в сети всегда присутствуют широковещательные пакеты и могут использоваться пакеты с групповыми адресами (multicast).
[1]В SunOS NIT минимум составляет 96 байтов.
[2]Например, при захвате пакетов в сети Ethernet с помощью команда **tcpdump -s 12** на выходе будут появляться строки вида **22:31:43.385357 [|ether]**, показывающие, что отсечка пакетов произошла на уровне Ethernet.
[3]Протокол Ad-hoc On-demand Distance Vector.
[4]Common Address Redundancy Protocol - базовый протокол для избыточных (резервных) адресов.
[5]Протокол Cisco NetFlow.
[6]Link Management Protocol - протокол управления каналом.
[7]Pragmatic General Multicast.
[8]ZMTP/1.0 внутри PGM/EPGM.
[9]REdis Serialization Protocol.
[10]Протокол Remote Procedure Call (удалённый вызов процедур).
[11]Протокол Real-Time Applications (приложения в реальном масштабе времени).
[12]Протокол управления приложениями реального времени (Real-Time Applications control protocol).
[13]Простой протокол сетевого управления (Simple Network Management Protocol).
[14]Тривиальный протокол обмена файлами (Trivial File Transfer Protocol).
[15]Visual Audio Tool.
[16]Распределенные доски White Board.
[17]ZeroMQ Message Transport Protocol 1.0.
[18]Virtual eXtensible Local Area Network.

Опция	*Описание*
`-v`	Задаёт вывод дополнительной информации из пакета. К такой информации может относиться значение TTL (время жизни), идентификация, общий размер, опции IP и т. п. При использовании этого флага также выполняется дополнительная проверка целостности пакетов с помощью контрольных сумм (например, для протоколов IP и ICMP).
`-vv`	Задаёт дополнительное увеличение объёма выводимой информации (например, полное декодирование пакетов SMB, вывод дополнительных полей откликов NFS и т. п.).
`-vvv`	Задаёт максимальный объем выводимой информации (например, полностью выводятся опции **telnet SB ... SE**). При использовании вместе с ключом **-X** опции Telnet выводятся также в шестнадцатеричном представлении.
`-V <файл>`	Задаёт считывание списка имён из файла или со стандартного ввода (-).
`-w <файл>`	Задаёт запись необработанных (raw) пакетов в файл. Собранные в файл пакеты можно впоследствии просматривать с использованием опции **-r** или передавать для анализа другим программам. Если в качестве имени файла указан символ **-**, запись осуществляется на стандартное устройство вывода (**stdout**). Вывод буферизуется при записи в файл или канал, поэтому читающая файл или канал программа не будет видеть пакеты сразу (отключить буферизацию можно опцией -U). Для файлов pcap агентством IANA зарегистрирован тип MIME application/vnd.tcpdump.pcap.
`-W`	При использовании с опцией -C будет ограничивать число записываемых файлов заданным значением, по достижении которого самый старый файл будет перезаписываться. При именовании файлов номера будут дополняться слева нужным числом нулей для корректной сортировки по росту номеров. При использовании с опцией -G будет ограничивать число создаваемых при ротации по времени файлов, возвращая статус 0 при достижении предела. При использовании с опциями -C и -G будет зацикливать ротацию файлов.
`-x`	Задаёт вывод шестнадцатеричного дампа (без заголовка канального уровня) для каждого захваченного пакета. Объем выводимой информации определяется меньшим из двух значений - размер пакета и значение параметра **snaplen** (см. стр. 4). Отметим, что при захвате полных кадров канального уровня дамп может включать также байты заполнения, если пакет сетевого уровня имеет малый размер.
`-xx`	Задаёт вывод шестнадцатеричного дампа для каждого пакета с включением заголовков канального уровня.
`-X`	Задаёт вывод дампа в шестнадцатеричном и ASCII-формате без заголовков канального уровня. Эта опция может быть очень удобна при анализе новых протоколов.
`-XX`	Задаёт вывод дампа в шестнадцатеричном и ASCII-формате с включением заголовков канального уровня.
`-y <тип>` `--linktype=datalinktype`	Задаёт тип канального уровня, используемый при захвате пакетов. Поддерживаемые значения можно посмотреть с помощью опции **-L**.
`-z <команда>`	При использовании с опцией -C или -G будет вызывать указанную команду после закрытия каждого записываемого файла. Команда запускается параллельно tcpdump (с низким приоритетом) и не прерывает захват пакетов. Если нужно использовать команду, включающую флаги и/или аргументы, можно поместить эту команду в файл сценария, а сценарий вызывать данной опцией.
`-Z user` `--relinquish-privileges=user`	При работе tcpdump от имени пользователя root эта опция задаёт смену идентификатора пользователя и группы на идентификатор основного пользователя и группы после открытия устройства захвата или входного файла, но до открытия выходных файлов.
`expression`	Задаёт фильтр отбора пакетов. Синтаксис фильтров описан ниже.

Фильтрация при отборе пакетов

В командной строке tcpdump наряду с опциями могут задаваться выражения, определяющие фильтрацию пакетов на этапе их отбора. Если никакого фильтра не задано, программа будет собирать все пакеты.

Каждое выражение, задающее фильтр, включает один или несколько примитивов, состоящих обычно из одного или множества идентификаторов объектов и предшествующих им классификаторов. Идентификатором объекта может служить его имя или номер. Классификаторы объектов могут относиться к одному из трёх видов.

type - тип

Указывает тип объекта, заданного идентификатором. В качестве типа объектов могут указываться значения **host** (хост), **net** (сеть) и **port** (порт). Если тип объекта не указан, предполагается значение **host**.

dir - направление

Задаёт направление по отношению к объекту. Для этого классификатора поддерживаются значения **src** (объект является отправителем), **dst** (объект является получателем), **src or dst** (отправитель или получатель) и **src and dst** (отправитель и получатель), **ra**, **ta**, **addr1**, **addr2**, **addr3**, **addr4**. Например, **src foo** указывает на пакеты,

отправленные с хоста **foo**, **dst net 128.3** - пакеты, адресованные в сеть **128.3.0.0/16**, а **src or dst port ftp-data** - пакеты данных протокола FTP (порт **ftp-data**), передаваемые в обоих направлениях. Если классификатор **dir** не задан, предполагается значение **src or dst**. Для некоторых типов соединений (например, SLIP) и режимов отбора (например, отбор с фиктивного интерфейса **any** в Linux-системах) могут использоваться классификаторы **inbound** и **outbound**. Значения ra, ta, addr1, addr2, addr3, addr4 применимы только для беспроводных интерфейсов IEEE 802.11.

proto - протокол

Задаёт протокол, к которому должны относиться пакеты. Этот классификатор может принимать значения **ether**, **fddi**[1], **tr**[2], **wlan**[3], **ip**, **ip6**, **arp**, **rarp**, **decnet**, **tcp** и **udp**. Если примитив не содержит классификатора протокола, предполагается, что данному фильтру удовлетворяют все протоколы, совместимые с типом объекта[4].

Кроме объектов и классификаторов примитивы могут содержать ключевые слова **gateway** (шлюз), **broadcast** (широковещательный), **less** (меньше), **greater** (больше) и арифметические выражения.

Сложные фильтры могут содержать множество примитивов, связанных между собой с использованием логических операторов **and**, **or** и **not** (например, **host foo and not port ftp and not port ftp-data**). Для сокращения задающих фильтры выражений можно опускать идентичные списки классификаторов. Например, выражение **tcp dst port ftp or ftp-data or domain** будет краткой формой выражения

```
tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain
```

Допустимые примитивы фильтрации пакетов

Ниже приводится список допустимых примитивов с краткими комментариями для каждого из них.

Таблица 2. Примитивы фильтров tcpdump.

Примитив	***Описание***
`dst host <хост>`	Будет отбирать пакеты, в которых поле адреса получателя IPv4/v6 содержит адрес хоста, заданного в примитиве.
`src host <хост>`	Будет выбирать все пакеты, в которых поле отправителя содержит адрес указанного хоста.

`host <хост>`	Будет отбирать все пакеты, для которых адрес хоста указан в поле получателя или отправителя.
	
	Все три приведённых выше выражения могут содержать идентификаторы протоколов **ip**, **arp**, **rarp** или **ip6**, как в выражении
	
	`ip host <хост>`
	эквивалентном фильтру:
	
	`ether proto \ip and host <хост>`
	Если именем задан хост, с которым связано несколько адресов IP, фильтру будут соответствовать пакеты с любым из этих адресов в заголовках пакетов.

`ether dst <ehost>`	Будет выбирать все кадры, в которых поле MAC-адреса получателя содержит значение ehost (имя хоста из файла /etc/ethers или шестнадцатеричное представление MAC-адреса[5]).
`ether src <ehost>`	Будет выбирать все кадры, в которых поле MAC-адреса отправителя содержит значение ehost.
`ether host <ehost>`	Будет отбирать все пакеты с адресом, указанным значением **ehost** в поле отправителя или получателя.
`gateway <шлюз>`	Будет отбирать все пакеты, использующие указанный именем хост в качестве шлюза[6]. Указанное параметром имя хоста должно преобразовываться в IP-адрес механизмами преобразования имён, доступными локальному компьютеру (/etc/hosts, DNS, NIS и т. п.), а также механизмами определения MAC-адреса по имени хоста (/etc/ethers и т. п.). Эквивалентное выражение **ether host ehost and not host <хост>** позволяет указывать хост по имени или адресу, заданному в файле host/ehost. Отметим, что данный примитив пока не поддерживается для конфигураций IPv6.
`dst net <сеть>`	Отбирает все пакеты IPv4/v6, направленные в указанную сеть. Для указания сети можно использовать имя из файла **/etc/networks** или номер сети. Сети IPv4 можно задавать в сокращённой форме (например, 192.168 или 10), сети IPv6 должны указываться полностью.
`src net <сеть>`	Выбирает все пакеты Ipv4/v6, отправленные из указанной сети.
`net <сеть>`	Выбирает все пакеты IPv4/v6, содержащие адреса из указанной сети в поле отправителя или получателя.
`net <сеть> mask <маска>`	Будет отбирать все пакеты IPv4, содержащие в поле отправителя или получателя адреса из сети, указанной с использованием маски.
`net <сеть/размер маски>`	Будет отбирать все пакеты IPv4/v6, содержащие в поле отправителя или получателя адреса из сети, указанной с использованием маски.

[1]**fddi** в действительности является псевдонимом **ether** и при анализе примитива оба классификатора трактуются как “канальный уровень, используемый указанным интерфейсом”. Заголовки FDDI содержат адреса отправителя и получателя, подобные адресам Ethernet, поля типа также зачастую содержат значения, подобные используемым для Ethernet, поэтому можно фильтровать эти поля в кадрах FDDI как и аналогичные поля кадров Ethernet. Заголовки FDDI содержат и другие поля, но их нельзя указать в фильтрах.

[2]**tr** является псевдонимом **ether**, поскольку оба типа кадров используют весьма похожую структуру заголовков.

[3]Идентификатор протокола **wlan** (беспроводные сети 802.11) является псевдонимом **ether**. В заголовках 802.11 адрес получателя содержится в поле DA, отправителя - в поле SA, а поля BSSID, RA и TA не проверяются фильтрами.

[4]Например, примитиву **src foo** будут соответствовать все пакеты **ip**, **arp** и **rarp**, исходящие от хоста foo.

[5]MAC-адрес записывается в формате **xx:xx:xx:xx:xx:xx**

[6]Т. е., адрес отправителя или получателя на канальном уровне (например, ethernet) соответствует адресу хоста, заданному значением <шлюз>, но IP-адреса отправителя и получателя в заголовке пакета не совпадают с IP-адресом указанного шлюза.

Примитив	*Описание*
`dst port <порт>`	Отберёт все пакеты **ip/tcp**, **ip/udp**, **ip6/tcp** и **ip6/udp**, направленные в указанный порт. Номера портов могут задаваться номерами или именами из файла **/etc/services**. При указании имени (протокол/порт) проверяется как порт, так и протокол. Если примитив содержит неоднозначный номер или имя порта, проверяется только порт (без протокола) и фильтру будут соответствовать пакеты обоих протоколов (tcp и udp). Например, фильтру **dst port 513** будут соответствовать пакеты **tcp/login** и **udp/who**, а фильтру **port domain** - **tcp/domain** и **udp/domain**.
`src port <порт>`	Отбирает все пакеты, отправленные из указанного порта.
`port <порт>`	Отбирает все пакеты, содержащие указанный номер порта в поле отправителя или получателя. Любое из трёх перечисленных правил для портов может включать в качестве префикса идентификатор протокола **tcp** или **udp** (например, **tcp src port <порт>**, будет отбирать пакеты tcp, отправленные из указанного порта).
`dst portrange port1-port2`	Отберёт пакеты **ip/tcp**, **ip/udp**, **ip6/tcp** и **ip6/udp**, направленные в указанный диапазон портов. Интерпретация параметров port такая же, как для одного порта.
`src portrange port1-port2`	Отберёт пакеты, направленные из указанного диапазона портов.
`portrange port1-port2`	Отберёт пакеты, в которых порт отправителя или получателя попадает в указанный диапазон.
`less <размер>`	Отберёт пакеты, размер которых не больше указанного. Эквивалентно len <= length.
`greater <размер>`	Отберёт пакеты, размер которых не меньше указанного. Эквивалентно len >= length.
`ip proto <протокол>`	Отбирает пакеты IPv4, содержащие заданный идентификатор типа в поле протокола. Типы протоколов IP можно указывать по именам или (**icmp**, **icmp6**, **igmp**, **igrp**, **pim**, **ah**, **esp**, **vrrp**, **udp**, **tcp**) или номерам. Поскольку **tcp**, **udp** и **icmp** используются также в качестве ключевых слов, их следует экранировать символом \. Отметим, что этот примитив не проверяет цепочки протокольных заголовков.
`ip6 proto <протокол>`	Отберёт пакеты IPv6 указанного типа без проверки цепочки протокольных заголовков.
`proto <протокол>`	Отберёт пакеты IPv4 и IPv6 указанного типа без проверки цепочки протокольных заголовков.
`tcp, udp, icmp`	Сокращение для proto <протокол>, где протокол относится к одному из указанных.
`ip6 protochain <протокол>`	Отберёт все пакеты IPv6, содержащие в цепочке протокольных заголовков идентификатор указанного типа протокола. Например, фильтру **ip6 protochain 6** будут соответствовать все пакеты IPv6 с заголовками TCP в цепочке заголовков. Такой пакет может содержать, например, заголовок аутентификации (AH), маршрутный заголовок (routing header), или заголовок опции hop-by-hop между заголовками IPv6 и TCP. Отметим, что порождаемый этим примитивом код BPF достаточно сложен и не может быть оптимизирован средствами tcpdump, поэтому использование данного фильтра может замедлять работу программы.
`ip protochain <протокол>`	Эквивалентно примитиву **ip6 protochain protocol**, но работает с пакетами IPv4.
`ip protochain <протокол>`	Эквивалентно **ip6 protochain protocol**, но работает с пакетами IPv4 и IPv6.
`ether broadcast`	Обеспечивает отбор всех широковещательных кадров Ethernet. Ключевое слово **ether** может быть опущено.
`ip broadcast`	Отбирает все широковещательные пакеты IPv4. Этому правилу будут соответствовать широковещательные адреса, содержащие только нули (all-zeroes) и только единицы (all-ones) с учётом маски подсети для интерфейса, который используется для отбора пакетов. Если маска подсети для интерфейса недоступна[1], фильтр может работать некорректно.
`ether multicast`	Собирает все кадры с групповыми адресами Ethernet. Ключевое слово **ether** необязательно. Логически это правило эквивалентно выражению **ether[0] & 1 != 0**.
`ip multicast`	Отбирает пакеты с групповыми адресами IPv4.
`ip6 multicast`	Отбирает пакеты с групповыми адресами IPv6.

[1]Интерфейс не имеет маски или сбор пакетов осуществляется со всех интерфейсов хоста Linux (интерфейс **any**).

Примитив	*Описание*
`ether proto <протокол>`	Отбирает кадры Ethernet с заданным типом протокола. Протокол может быть указан по номеру или имени[1] (**ip**, **ip6**, **arp**, **rarp**, **atalk**, **aarp**, **decnet**, **sca**, **lat**, **mopdl**, **moprc**, **iso**, **stp**, **ipx**, **netbeui**). При использовании правила для протоколов FDDI (например, **fddi protocol arp**), Token Ring (например, **tr protocol arp**) или IEEE 802.11 (например, **wlan protocol arp**) идентификация протокола выполняется на основании заголовка 802.2 Logical Link Control (LLC), который следует после заголовка FDDI, Token Ring или 802.11. При фильтрации для большинства протоколов FDDI, Token Ring и 802.11 **tcpdump** проверяет только поле идентификатора протокола (protocol ID) в заголовке LLC так называемого SNAP-формата с идентификатором OUI = 0x000000 (Organizational Unit Identifier), указывающим на инкапсуляцию Ethernet. Проверка использования формата SNAP с OUI = 0x000000 не выполняется за исключением перечисленных ниже случаев: **`iso`** tcpdump проверяет поля **DSAP**[2] и **SSAP**[3] в заголовках LLC; **`stp, netbeui`** tcpdump проверяет поле DSAP в заголовке LLC; **`atalk`** проверяется формат SNAP с OUI = 0x080007 и тип (etype) AppleTalk. Для Ethernet проверяются поля типа Ethernet для большинства протоколов. Исключения указаны ниже **`iso, sap, netbeui`** проверяется принадлежность к 802.3 и заголовок LLC (как это описано выше для FDDI, Token Ring и 802.11); **`atalk`** проверяется тип **AppleTalk** в кадре Ethernet и формат заголовка SNAP (как для FDDI, Token Ring и 802.11); **`aarp`** проверяется тип **AppleTalk ARP** в кадре Ethernet или использование формата 802.2 SNAP с OUI = 0x000000; **`ipx`** tcpdump проверяет тип IPX в кадре Ethernet, поле IPX DSAP в заголовке LLC, инкапсуляцию IPX и тип IPX в кадре SNAP.
`ip, ip6, arp, rarp, atalk, aarp, decnet, iso, stp, ipx, netbeui`	Сокращения для ether proto <протокол>, где указан один из протоколов.
`lat, moprc, mopdl`	Сокращения для ether proto <протокол>, где указан один из протоколов. Отметим, что не все приложения, использующие pcap(3PCAP), понимают эти протоколы.
`decnet src <хост>`	Собирает все пакеты от указанного хоста DECNET, который может быть задан по адресу в форме 10.123 или имени DECNET[4].
`decnet dst <хост>`	Отбирает все пакеты, адресованные указанному хосту DECNET.
`decnet host <хост>`	Собирает все пакеты, содержащие адрес указанного хоста DECNET в поле отправителя или получателя.
`llc`	Отбирает пакет с заголовком 802.2 LLC, включая: пакеты Ethernet с полем length вместо поля type, которые не являются неразобранными пакетами Netware IEEE 802.3; пакеты IEEE 802.11; пакеты Token Ring (без проверки LLC); пакеты FDDI (без проверки LLC); пакеты ATM с инкапсуляцией LLC (например, SunATM в Solaris).

[1]Перечисленные здесь имена протоколов могут использоваться также в качестве ключевых слов, поэтому имени протокола должен предшествовать символ **** (например, **\arp**).
[2]Destination Service Access Point - точка доступа к сервису для получателя.
[3]Source Service Access Point - точка доступа к сервису для отправителя.
[4]Поддержка имён DECNET обеспечивается только на хостах ULTRIX, настроенных для использования DECNET.

Примитив	*Описание*
`llc type`	Отбирает пакет с заголовком 802.2 LLC указанного типа: i - информационные PDU (I); s - PDU управления (S, supervisory); u - unnumberd PDU (U); rr - PDU готовности приёмника (RR); rnr - PDU неготовности приёмника (RNR); rej - PDU отторжения (REJ, reject); ui - PDU ненумерованной информации (UI); ua - PDU ненумерованных подтверждений (UA); disc - PDU разрыва соединения (DISC, disconnect); sabme - PDU расширенного сбалансированного режима (SABME); test - тестовые PDU (TEST); xid - PDU информационного обмена (XID); frmr - PDU отторжения кадра (FRMR);
`inbound`	Пакеты, полученные хостом, где выполняется отбор. Поддерживается лишь некоторыми типами интерфейсов, такими как SLIP, фиктивный интерфейс Linux any и др.
`outbound`	Пакеты, переданные хостом, где выполняется отбор. Поддерживается лишь некоторыми типами интерфейсов, такими как SLIP, фиктивный интерфейс Linux any и др.
`ifname <интерфейс>`	Отбирает все пакеты, полученные от указанного интерфейса[2].
`on <интерфейс>`	Синоним **ifname**.
`rnr <номер>`	Собирает только пакеты, записанные в файл программой pf в соответствии с правилом, имеющим указанных номер (доступно только при сборе пакетов с помощью OpenBSD и FreeBSD pf).
`rulenum <номер>`	Синоним для **rnr**.
`reason <код>`	Собирает только пакеты, соответствующие указанному коду причины PF. Известные коды причин включают **match**, **bad-offset**, **fragment**, **short**, **normalize**, **memory** (доступно только при сборе пакетов с помощью OpenBSD и FreeBSD pf).
`rset <имя>`	Собирает пакеты, соответствующие указанному именем правилу привязанного набора (доступно только при сборе пакетов с помощью OpenBSD и FreeBSD pf).
`ruleset <имя>`	Синоним rset.
`srnr <номер>`	Собирает пакеты, соответствующие указанному номером правилу привязанного набора (доступно только при сборе пакетов с помощью OpenBSD и FreeBSD pf).
`subrulenum <номер>`	Синоним srnr.
`action <действие>`	Отбирает пакеты, захваченные указанной операцией pf (**pass** или **block**, а для некоторых версий pf также nat, rdr, binat и scrub). Правило доступно только при сборе пакетов с помощью OpenBSD и FreeBSD pf.
`wlan ra ehost`	Отбирает пакеты с полем IEEE 802.11 RA равным ehost. Поле RA отсутствует в кадрах управления.
`wlan ta ehost`	Отбирает пакеты с полем IEEE 802.11 TA равным ehost. Поле TA отсутствует в кадрах управления, CTS (готовность к передаче) и ACK (подтверждение).
`wlan addr1 ehost`	Отбирает пакеты с первым адресом IEEE 802.11 равным ehost.
`wlan addr2 ehost`	Отбирает пакеты со вторым адресом IEEE 802.11 равным ehost. Второй адрес отсутствует в кадрах CTS (готовность к передаче) и ACK (подтверждение).
`wlan addr3 ehost`	Отбирает пакеты с третьим адресом IEEE 802.11 равным ehost. Третий адрес отсутствует в кадрах управления (control).
`wlan addr4 ehost`	Отбирает пакеты с четвёртым адресом IEEE 802.11 равным ehost. Четвёртый адрес присутствует только в кадрах WDS (Wireless Distribution System).
`type wlan_type`	Отбирает пакеты с типом кадра IEEE 802.11 wlan_type (mgt, ctl, data).
`type wlan_type subtype wlan_subtype`	Отбирает пакеты с типом кадра IEEE 802.11 wlan_type и субтипом wlan_subtype. Для типа mgt, субтип может принимать значения assoc-req, assoc-resp, reassoc-req, reassoc-resp, probe-req, probe-resp, beacon, atim, disassoc, auth, deauth, для типа ctl - ps-poll, rts, cts, ack, cf-end, cf-end-ack, а для data - data, data-cf-ack, data-cf-poll, data-cf-ack-poll, null, cf-ack, cf-poll, cf-ack-poll, qos-data, qos-data-cf-ack, qos-data-cf-poll, qos-data-cf-ack-poll, qos, qos-cf-poll и qos-cf-ack-poll.
`subtype wlan_subtype`	Отбирает пакеты с субтипом кадра IEEE 802.11 wlan_subtype, а кадр имеет тип, для которого указанный субтип определён.
`dir <направление>`	Отбирает кадры IEEE 802.11 указанного направления (nods, tods, fromds, dstods или число).

[2]Это правило применимо только к пакетам, собранным в файл с помощью программы **pf** (OpenBSD).

Примитив	*Описание*

`vlan [vlan_id]`	Отбирает кадры IEEE 802.1Q VLAN. Если указан идентификатор VLAN, выбираются лишь пакеты, относящиеся в указанной виртуальной ЛВС. Первое ключевое слово **vlan** изменяет расчёт смещения полей для оставшейся части выражения с учётом размера поля VLAN в заголовке кадра (4). Выражение vlan [vlan_id] может указываться несколько раз с учётом вложенности VLAN. Каждое такое выражение увеличивает смещение для полей на 4. Например,
	`vlan 100 && vlan 200`
	отберёт кадры VLAN 200, инкапсулированные в кадры VLAN 100, а
	`vlan && vlan 300 && ip`
	отберёт пакеты IPv4, инкапсулированные в кадры VLAN 300, инкапсулированные в любую виртуальную ЛВС.

`mpls [label_num]`	Отбирает пакеты MPLS. При наличии параметра [label_num] выбираются только пакеты с указанной меткой. Первое ключевое слово mpls меняет расчёт смещения полей для оставшейся части выражения в предположении, что пакет относится к протоколу IP, инкапсулированному в MPLS. Выражение mpls [label_num] может указываться несколько раз с учётом вложенности меток. Каждое вхождение увеличивает смещение на 4.
	Например, фильтр
	`mpls 100000 && mpls 1024`
	отберёт пакеты с внешней меткой 100000 и внутренней меткой 1024, а
	`mpls && mpls 1024 && host 192.9.200.1`
	отберёт пакеты с адресом 192.9.200.1, внутренней меткой 1024 и любой внешней меткой.

`pppoed`	Отбирает пакеты PPP-over-Ethernet Discovery (тип 0x8863)
`pppoes [session_id]`	Отбирает пакеты сессий PPP-over-Ethernet (тип 0x8864). При наличии параметра session_id будут отбираться лишь пакеты указанной сессии. Первое ключевое слово pppoes изменяет расчёт смещения полей для оставшейся части выражения в предположении, что пакет относится к сессии PPPoE. Например, фильтр `pppoes 0x27 && ip` отберёт пакеты IPv4, инкапсулированные в сессию PPPoE с идентификатором 0x27.
`geneve [vni]`	Отбирает пакеты Geneve (порт UDP 6081). При наличии параметра vni отбираются лишь соответствующие ему пакеты. Ключевое слово pppoes изменяет расчёт смещения полей для оставшейся части выражения в предположении, что пакет относится к протоколу Geneve. Например, фильтр `geneve 0xb && ip` отберёт пакеты IPv4, инкапсулированные с Geneve VNI 0xb (пакеты, напрямую инкапсулированные в Geneve, а также пакеты IP в кадрах Ethernet).
`iso proto <протокол>`	Собирает пакеты с указанным типом протокола OSI. Протокол может быть указан по номеру или имени (**clnp**, **esis**, **isis**).
`clnp, esis, isis`	Сокращение для iso proto p, где p - один из перечисленных протоколов.
`l1, l2, iih, lsp, snp, csnp, psnp`	Сокращение для типов IS-IS PDU.
`vpi n`	Собирает пакеты ATM с указанным идентификатором виртуального пути для SunATM (Solaris).
`vci n`	Собирает пакеты ATM с указанным идентификатором виртуального канала для SunATM (Solaris).
`lane`	Собирает пакеты эмуляции ЛВС (ATM LANE) для SunATM (Solaris). Первое ключевое слово **lane** в выражении изменяет проверки для остальной части фильтра в предположении, что пакет относится к пакетам эмуляции Ethernet или LANE LE Control. Если ключевое слово **lane** не указано, проверки выполняются в предположении LLC-инкапсуляции.
`oamf4s`	Собирает пакеты ATM для SunATM (Solaris), являющиеся сегментами потока ячеек OAM F4 (VPI=0, VCI=3).
`oamf4e`	Собирает пакеты ATM для SunATM (Solaris), относящиеся к сквозным потокам OAM F4 (VPI=0, VCI=4).
`oamf4`	Собирает пакеты ATM для SunATM (Solaris), являющиеся сегментами сквозного потока ячеек OAM F4 (VPI=0, (VCI=3 или VCI=4)).
`oam`	Собирает пакеты ATM для SunATM (Solaris), являющиеся сегментами сквозного потока ячеек OAM F4 (VPI=0, (VCI=3 или VCI=4)).
`metac`	Собирает пакеты ATM для SunATM (Solaris), относящиеся к сигнальным метаустройствам (VPI=0, VCI=1).
`bcc`	Собирает пакеты ATM для SunATM (Solaris), относящиеся к широковещательным сигнальным устройствам (VPI=0, VCI=2).
`sc`	Собирает пакеты ATM для SunATM (Solaris), относящиеся к сигнальным устройствам (VPI=0, VCI=5).
`ilmic`	Собирает пакеты ATM для SunATM (Solaris), относящиеся к клиентским устройствам ILMI (VPI=0, VCI=16).
`connectmsg`	Собирает пакеты ATM для SunATM (Solaris), относящиеся к сигнальным устройствам и содержащие сообщения **Q.2931 Setup**, **Call Proceeding**, **Connect**, **Connect Ack**, **Release**, **Release Done**.
`metaconnect`	Собирает пакеты ATM для SunATM (Solaris), относящиеся к сигнальным метаустройствам и содержащие сообщения Q.2931 Setup, Call Proceeding, Connect, Connect Ack, Release, Release Done.

Примитив	*Описание*
`expr <операция> expr`	Возвращают логическое значение, соответствующее отношениям между левой и правой часть. В качестве операции могут использоваться **>**, **<**, **>=**, **<=**, **=**, **!=**, а операнды **expr** могут быть арифметическими выражениями, включающими целые константы (запись в стандарте C), бинарные операторы[1] **+**, **-**, *****, **/**, **%**, **&**, **|**, **^**, **<<**, **>>**, оператор размера (len) и данные из пакетов. Во всех операциях сравнения применяются целые числа без знака. Для получения значений полей из пакетов применяется синтаксис: `proto [expr : size]` Параметр **proto** может содержать идентификатор одного из протоколов (**ether**, **fddi**, **tr**, **wlan**, **ppp**, **slip**, **link**, **ip**, **arp**, **rarp**, **tcp**, **udp**, **icmp**, **ip6**, **radio**) и задаёт уровень протокола[2], для которого извлекаются данные. Отметим, что **tcp**, **udp** и другие протоколы верхних уровней относятся только к пакетам IPv4, а не IPv6[3]. Параметр **expr** задаёт смещение в байтах относительно начала заголовка указанного уровня. Необязательный параметр **size** определяет размер интересующего поля в байтах и может принимать значения 1, 2 и 4 (по умолчанию 1 байт). Оператор размера, указываемый ключевым словом **len**, определяет размер пакета в байтах. Например, выражению ether[0] & 1 != 0 будет соответствовать весь multicast-трафик, выражение ip[0] & 0xf != 5 позволяет собрать все пакеты IP, в которых присутствует поле опций, а фильтр ip[6:2] & 0x1fff = 0 соберёт только нефрагментированные дейтаграммы IPv4 и первые фрагменты. При выборе полей из заголовков учитывается структура пакетов соответствующего уровня. Например, tcp[0] всегда будет возвращать первый байт заголовка TCP, игнорируя фрагменты. Некоторые значения полей и смещений могут задаваться не только числами, но и именами. Поддерживаются значения **icmptype**[4], **icmp6type**[5], **icmpcode**, **icmpcode**, **tcpflags**[6].

Примитивы в выражениях можно группировать с использованием

- скобок[7];
- отрицания (**!** или **not**);
- конкатенации (**&&** или **and**);
- выбора варианта (**||** или **or**).

Оператор отрицания имеет высший уровень приоритета, операции выбора и конкатенации имеют одинаковый приоритет и выполняются слева направо в порядке следования. Отметим, что для конкатенации недостаточно просто указать операнды рядом, а требуется явно задать операцию (**&&** или **and**).

Если идентификатор указан без ключевого слова, предполагается ключевое слово, которое до этого использовалось последним. Например, выражение

```
not host vs and ace
```

является простым сокращением от

```
not host vs and host ace
```

Отметим, что эти выражения не эквивалентны фильтру **not (host vs or ace)**.

Аргументы выражений могут передаваться программе tcpdump как один или множество аргументов (используйте более удобную для вас форму). В общем случае выражения, содержащие метасимволы командного интерпретатора, должны передаваться как один аргумент, заключенный в кавычки.

Примеры фильтров

Таблица 3. Примеры фильтров tcpdump.

Фильтр	*Выполняемые действия*
`host sundown`	Все пакеты, принимаемые и передаваемые хостом **sundown**
`host helios and \( hot or ace \)`	Пакеты, передаваемые между хостом **helios** и любым из хостов **hot** или **ace.**
`ip host ace and not helios`	Пакеты передаваемые между хостом **ace** и любым хостом, за исключением **helios**.
`net ucb-ether`	Все пакеты, передаваемые или принимаемые хостами сети **ucb-ether**.
`'gateway snup and (port ftp or ftp-data)'`	Весь трафик **ftp**, проходящий через шлюз **snup**[8].

[1]Операторы % и ^ поддерживаются фильтрами в ядре Linux, начиная с версии 3.7, а в остальных случаях эти операторы используют фильтры в пользовательском пространстве, что существенно повышает издержки при отборе и может вести к потере пакетов.
[2]Протоколы **ether**, **fddi**, **wlan**, **tr**, **ppp**, **slip** и **link** указывают на канальный уровень, **radio** указывает «радио-заголовок», добавляемый к некоторым выборкам 802.11.
[3]Поддержка IPv6 будет реализована в следующих версиях.
[4]Поле типа ICMP, которое может принимать значения **icmp-echoreply**, **icmp-unreach**, **icmp-sourcequench**, **icmp-redirect**, **icmp-echo**, **icmp-routeradvert**, **icmp-routersolicit**, **icmp-timxceed**, **icmp-paramprob**, **icmp-tstamp**, **icmp-tstampreply**, **icmp-ireq**, **icmp-ireqreply**, **icmp-maskreq**, **icmp-maskreply**.
[5]Поле типа ICMPv6, которое может принимать значения **icmp6-echo**, **icmp6-echoreply**, **icmp6-multicastlistenerquery**, **icmp6-multicastlistenerreportv1**, **icmp6-multicastlistenerdone**, **icmp6-routersolicit,** **icmp6-routeradvert**, **icmp6-neighborsolicit**, **icmp6-neighboradvert**, **icmp6-redirect**, **icmp6-routerrenum**, **icmp6-nodeinformationquery**, **icmp6-nodeinformationresponse**, **icmp6-ineighbordiscoverysolicit**, **icmp6-ineighbordiscoveryadvert**, **icmp6-multicastlistenerreportv2**, **icmp6-homeagentdiscoveryrequest**, **icmp6-homeagentdiscoveryreply**, **icmp6-mobileprefixsolicit**, **icmp6-mobileprefixadvert,** **icmp6-certpathsolicit**, **icmp6-certpathadvert**, **icmp6-multicastrouteradvert**, **icmp6-multicastroutersolicit**, **icmp6-multicastrouterterm**
[6]Для флагов TCP можно указать идентификаторы **tcp-fin**, **tcp-syn**, **tcp-rst**, **tcp-push**, **tcp-ack**, **tcp-urg**, **tcp-ece**, **tcp-cwr**.
[7]В зависимости от используемого командного интерпретатора перед символами открывающих и закрывающих скобок может потребоваться включение символа экранирования.
[8]Кавычки позволяют избавиться от ошибок при анализе скобок командным интерпретатором.

Фильтр	*Выполняемые действия*
`ip and not net localnet`	Весь трафик, не относящийся к хостам локальной сети.
`'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst net localnet'`	Стартовые (SYN) и конечные (FIN) пакеты TCP, исключая соединения хостов локальной сети.
`tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)`	Пакеты IPv4 протокола HTTP через порт 80, исключая пакеты SYN, FIN и пакеты ACK без данных.
`'gateway snup and ip[2:2] > 576'`	Переданные через шлюз **snup** пакеты IP, размер которых превышает 576 байтов.
`'ether[0] & 1 = 0 and ip[16] >= 224'`	Широковещательные и групповые пакеты, которые не были переданы с использованием широковещательных и групповых адресов Ethernet.
`'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'`	Все пакеты ICMP, кроме запросов и откликов **echo** (т. е., кроме пакетов **ping**).

Формат вывода

Формат вывода **tcpdump** зависит от протокола. Ниже приведены краткие описания и примеры для большинства используемых форматов вывода.

Временные метки

По умолчанию в начале каждой строки вывода указывается временная метка в формате hh:mm:ss.frac, с поддерживаемым ядром разрешением. Метка указывает время её добавления ядром и может включать задержку между завершением приёма пакета интерфейсом и передачей ядру прерывания для чтения пакета, а также время обработки этого прерывания.

Заголовки канального уровня

При использовании опции **-e** выводится содержимое заголовков канального уровня. Для сетей Ethernet информация из заголовков включает адреса отправителя и получателя, протокол и размер кадра.

Для сетей FDDI опция **-e** обеспечивает вывод поля управления (frame control), адресов отправителя и получателя, а также размера кадра. Значение поля управления определяет интерпретацию остальной части кадра. Обычные кадры (например, содержащие дейтаграммы IP) являются “асинхронными” с уровнем приоритета от 0 до 7 (например async4). Предполагается, что такие кадры содержат пакеты 802.2 LLC. Заголовок LLC выводится в тех случаях, когда пакет не является дейтаграммой ISO или пакетом SNAP.

В сетях Token Ring опция **-e** обеспечивает вывод полей контроля доступа (**access control**) и управления кадром (**frame control**), адресов отправителя и получателя, а также размера кадра. Предполагается, что кадры содержат пакеты LLC. Независимо от наличия опции **-e** выводится информация о заданном отправителем маршруте (source routing), если пакет содержит такую информацию.

В сетях 802.11 опция **-e** выводит значения полей управления (**frame control**), все адреса из заголовка 802.11, а также размер кадра. Предполагается, что кадры содержат пакеты LLC.

Для каналов SLIP информация канального уровня включает индикатор направления (**I** для входящего трафика, **O** - для исходящего), тип пакета и сведения о компрессии[1]. Поле типа пакета выводится первым и может принимать значение **ip**, **utcp** или **ctcp**. Для пакетов IP дополнительной информации о канале не выводится. Для пакетов TCP вслед за типом выводится идентификатор соединения. Если для пакета используется компрессия, сжатый заголовок декодируется перед выводом. Для специальных случаев[2] выводятся значения ***S+n** и ***SA+n** (где n - числовое значение), которые указывают величину изменения порядкового номера для пакета и подтверждения, соответственно. Для остальных пакетов могут выводиться индикаторы изменений - **U** (указатель важности), **W** (окно), **A** (подтверждение), **S** (порядковый номер) и **I** (идентификатор пакета), сопровождаемые величиной изменения (+n или -n) или указателем на новое значение параметра (=n). Далее выводится информация о размере данных в пакете и размер сжатого заголовка.

Например строка вывода

```
O ctcp * A+6 S+49 I+6 3 (6)
```

относится к исходящему сжатому пакету TCP с неявным идентификатором соединения. Порядковый номер подтверждения увеличился на 6, порядковый номер пакета - на 49, идентификатор пакета - на 6. Пакет содержит 3 байта данных и 6-байтовый сжатый заголовок.

Пакеты ARP и RARP

Информация, выводимая для пакетов arp и rarp, включает тип запроса и его аргументы. Выводимой информации вполне достаточно для понимания происходящих процессов. Ниже показан пример вывода для случая, когда хост **rtsg** открывает сессию **rlogin** с хостом **csam**:

```
arp who-has csam tell rtsg
arp reply csam is-at CSAM
```

Первая строка показывает запрос **arp** от хоста **rtsg** для получения адресов (MAC и IP) хоста **csam**. В ответ на это **csam** возвращает свои адреса (в примере IP-адрес обозначен как **csam**, а MAC-адрес - как **CSAM**). Если ввести команду с опцией **-n**, результат будет иметь вид

```
arp who-has 128.3.254.6 tell 128.3.254.68
arp reply 128.3.254.6 is-at 02:07:01:00:01:c4
```

Если же воспользоваться опцией **-e**, можно увидеть, что первый пакет является широковещательным (MAC-адрес отправителя показан как **RSTG**), поле типа содержит значение 0806 (**ETHER_ARP**), а размер пакета составляет 64 байта.

```
RTSG Broadcast 0806  64: arp who-has csam tell rtsg
CSAM RTSG 0806  64: arp reply csam is-at CSAM
```

[1]Компрессия заголовков TCP/IP для каналов SLIP описана в RFC 1144.
[2]RFC 1144 определяет как специальные случаи интерактивный трафик и передачу больших объёмов трафика.

Пакеты IPv4

Если заголовок канального уровня не выводится, для пакетов IPv4 после временной метки указывается заголовок IP. При наличии опции -v содержимое заголовка IPv4 указывается в скобках после заголовка IP или канального уровня. Базовый формат этой информации имеет вид

```
tos tos, ttl ttl, id id, offset offset, flags [flags], proto proto, length length, options (options)
```

Поле tos указывает тип обслуживания, отличные от нуля биты ECN указываются как ECT(1), ECT(0) или CE. Поле ttl указывает «время жизни» (нулевое значение опускается), id - поле идентификации IP, offset - смещение фрагмента, показывающее, является ли пакет частью фрагментированной дейтаграммы. Поле flags содержит флаги MF и DF - при установленном флаге MF выводится +, а при установленном флаге F - DFP. Если флаги не установлены, выводится точка (.). Поле proto содержит идентификатор протокола, length указывает общий размер пакета, а options - опции IP, если они имеются.

Далее для пакетов TCP и UDP указываются IP-адреса отправителя и получателя, а также номера портов TCP или UDP (номер пора отделяется от адреса точкой), а между отправителем и получателем размещается символ >. Для других протоколов выводятся адреса, разделённые символом >. При наличии данных вышележащего протокола, она выводится вслед за указанными полями.

Для фрагментированных дейтаграмм IP первый фрагмент содержит заголовок вышележащего протокола, а остальные фрагменты этот заголовок не включают. Данные фрагментации выводятся лишь при наличии опции -v.

Пакеты TCP

Формат вывода для протокола TCP (RFC 793) в общем случае имеет вид

```
src > dst: Flags [tcpflags], seq data-seqno, ack ackno, win window, urg urgent, options [opts], length len
```

Поля **src** и **dst** содержат IP-адреса и номера портов для отправителя и получателя. Поле **Flags** содержит комбинацию символов **S** (SYN), **F** (FIN), **P** (PUSH), **R** (RST), **W** (ECN CWR) и **E** (ECN-Echo) в соответствии с установленными для пакета флагами или один символ “**.**” (нет флагов). Поле **data-seqno** описывает занятую данным пакетом часть пространства порядковых номеров. Поле **ack** содержит порядковый номер, ожидаемый для следующей порции данных, передаваемой через это соединение в обратном направлении. Поле **window** показывает число байтов в приёмном буфере, доступных для обратного направления в этом соединении. Поле **urg** показывает состояние флага важности (**urgent**) для данных из этого пакета. Поле **options** содержит опции TCP, заключённые в угловые скобки.

Поля **src**, **dst** и **flags** присутствуют во всех случаях, вывод остальных полей зависит от данных в заголовке TCP.

Ниже показан набор пакетов, передаваемых при организации хостом **rtsg** сессии **rlogin** с хостом **csam**.

```
rtsg.1023 > csam.login: S 768512:768512(0) win 4096 <mss 1024>
csam.login > rtsg.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>
rtsg.1023 > csam.login: . ack 1 win 4096
rtsg.1023 > csam.login: P 1:2(1) ack 1 win 4096
csam.login > rtsg.1023: . ack 2 win 4096
rtsg.1023 > csam.login: P 2:21(19) ack 1 win 4096
csam.login > rtsg.1023: P 1:2(1) ack 21 win 4077
csam.login > rtsg.1023: P 2:3(1) ack 21 win 4077 urg 1
csam.login > rtsg.1023: P 3:4(1) ack 21 win 4077 urg 1
```

Первая строка показывает, что порт TCP с номером 1023 хоста **rtsg** отправил пакет в порт **login** хоста **csam**. Символ **S** говорит о наличии в пакете флага **SYN**. Порядковый номер пакета равен 768512 и данных в пакете не содержится[1]. Пакет не содержит в себе подтверждения, доступный размер приёмного окна составляет 4096 байтов, а запрошенное значение MSS составляет 1024 байта.

Сайт **csam** отправляет в ответ подобный полученному пакет, включающий подтверждение для полученного от **rtsg** пакета **SYN**. После этого **rtsg** подтверждает хосту **csam** получение от него пакета **SYN**. Точка (**.**) в поле флагов говорит о том, что пакет не содержит ни одного флага. Данных в пакете не содержится, поэтому в строке вывода отсутствуют значения порядковых номеров. Отметим, что для подтверждения в строке 3 указан порядковый номер 1. Когда **tcpdump** видит первый пакет в данном соединении, выводится порядковый номер из этого пакета. Для последующих пакетов данного соединения выводятся значения разницы между порядковым номером для текущего пакета и начальным порядковым номером. Это значит, что для всех пакетов, кроме первого, порядковые номера указываются относительно начала потока данных для соединения и первый байт данных имеет номер 1. Опция **-S** (стр. 4) отключает относительную нумерацию и обеспечивает вывод порядковых номеров в соответствии со значениями в пакетах.

В строке 6 показан пакет, который **rtsg** отправляет хосту **csam** с 19 байтами данных (байты со 2 по 20). Пакет передаётся с флагом PUSH. Строка 7 содержит отправленное хостом **csam** подтверждение приёма данных от хоста **rtsg** вплоть до байта с номером 21 (не включая этот байт). Большая часть этих данных сохраняется в приёмном буфере, поскольку **csam** показывает уменьшение приёмного окна на 19 байтов. В этом пакете **csam** также передаёт хосту **rtsg** 1 байт данных. В строках 8 и 9 показана передача хостом **csam** двух важных (**urg**) байтов данных, отправленных с использованием флага выталкивания **PUSH**.

Если используется достаточно малый кадр захвата (см. описание опции **-s** на стр. 4), tcpdump может не получить заголовок TCP полностью. В таких случаях интерпретируется полученная часть заголовка, а в строке вывода помещается маркер **[|tcp]**, показывающий невозможность полной интерпретации. Если заголовок содержит некорректную опцию[2], строка вывода будет содержать маркер **[bad opt]** и последующие опции не будут интерпретироваться, поскольку невозможно корректно определить начало следующей опции. Если поле размера заголовка указывает на присутствие опций, но размер пакета IP недостаточно велик для включения всех опций в пакет, **tcpdump** будет помещать в строке вывода маркер **[bad hdr length]**.

Сбор пакетов TCP с заданными комбинациями флагов (SYN-ACK, URG-ACK и т. п.)

Поле флагов заголовка TCP содержит 8 элементов - CWR | ECE | URG | ACK | PSH | RST | SYN | FIN.

[1]Об этом говорит запись **first:last(nbytes)**, в которой указывается порядковый номер первого байта в этом и следующем за ним (т. е. номер последнего байта в данном пакете + 1) пакетах и число байтов, содержащихся в пакете.
[2]Размер опции слишком мал или выходит за пределы указанного размера заголовка.

Предположим, что нужно собрать пакеты, используемые для организации нового соединения TCP. Напомним, что протокол TCP использует 3-этапную процедуру организации новых соединений, как показано ниже.

1) Инициатор соединения передаёт пакет с установленным флагом SYN.

2) Получатель этого пакета передаёт отклик с флагами SYN и ACK.

3) Инициатор передаёт в ответ пакет с флагом ACK.

Создадим фильтр, который будет собирать пакеты, содержащие только флаг SYN (этап 1). Пакеты этапа 2 (SYN-ACK) не будут включаться, поскольку они являются просто откликами на стартовый запрос SYN. Прежде, чем строить выражение для фильтра, вспомним структуру заголовка TCP без опций.

Таблица 4. Структура заголовка TCP.

0										***15***	***31***
Порт отправителя											Порт получателя
Порядковый номер											
Номер подтверждения											
HL	резерв		C	E	U	A	P	R	S	F	Размер окна
Контрольная сумма TCP											Указатель важности

Заголовок TCP состоит из 20 октетов, если не используются необязательные поля опций. Первая строка таблицы 4 соответствует октетам 0 - 3, вторая - октетам 4 - 7 и т. д. Поле битов управления (флагов) TCP содержится в октете 13. Пронумеруем биты флагов справа налево (в соответствии с ростом значимости битов).

Таблица. 5 Биты флагов TCP.

7	***6***	***5***	***4***	***3***	***2***	***1***	***0***
2^7=128	2^6=64	2^5=32	2^4=16	2^3=8	2^2=4	2^1=2	2^0=1
CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

Таким образом, значение октета флагов при наличии в нем только флага SYN будет составлять

```
0*128 + 0*64 + 0*32 + 0*16 + 0*8 + 0*4 + 1*2 + 0*1 = 2
```

и для выделения пакетов с флагом SYN можно воспользоваться выражением

```
tcp[13] == 2
```

Указав интересующий интерфейс, мы можем собрать пакеты с помощью команды

```
tcpdump -i <интерфейс> tcp[13] == 2
```

Эту команду можно перевести на человеческий язык словами: «Собрать с указанного интерфейса пакеты TCP, имеющие в тринадцатом октете заголовка значение 2».

Далее предположим, что нужно собрать пакеты с флагом SYN, независимо от состояния флага ACK и иных флагов. Посмотрим, какое значение будет иметь октет флагов для пакетов SYN-ACK:

```
|C|E|U|A|P|R|S|F|
|0 0 0 1 0 0 1 0|
```

В десятичном формате значение 00010010 будет равно 18

```
0*128 + 0*64 + 0*32 + 1*16 + 0*8 + 0*4 + 1*2 + 0*1 = 18
```

Однако, мы не можем использовать выражение

```
tcp[13] == 18
```

поскольку ему будут соответствовать только пакеты с установленными флагами SYN и ACK, но не будут соответствовать пакеты, имеющие только флаг SYN, который интересует нас в первую очередь.

Для сбора пакетов SYN, независимо от значения флага ACK, следует использовать маску 00000010 (десятичное значение 2). Таким образом, мы можем ввести выражение:

```
tcpdump -i <интерфейс> 'tcp[13] & 2 == 2'
```

которое позволит нам собирать пакеты с установленным флагом SYN независимо от присутствия других флагов. Выражение нужно поместить в кавычки или использовать символ \ для экранирования специального символа &.

Пакеты UDP

Формат вывода пакетов UDP можно проиллюстрировать на примере пакета **rwho**

```
actinide.who > broadcast.who: udp 84
```

Приведённая строка говорит о том, что порт **who** хоста **actinide** передал дейтаграмму **UDP** в порт **who** с использованием широковещательного адреса IP. Пакет содержит 84 байта пользовательских данных.

Для некоторых служб, работающих по протоколу UDP, распознаются протоколы вышележащего уровня (по номеру порта) и для таких протоколов выводится соответствующая информация. В частности, tcpdump выводит дополнительные сведения для пакетов DNS[1] и вызовов NFS с помощью Sun RPC[2].

Запросы UDP к серверам DNS

Формат вывода для запросов DNS имеет вид

```
src > dst: id op? flags qtype qclass name (len)
```

Например

[1]Спецификация протокола DNS (Domain Name System) приведена в RFC 1034 и RFC 1035.
[2]Спецификация протокола RPC (Remote Procedure Call - удалённый вызов процедур) содержится в RFC 1050.

```
h2opolo.1538 > helios.domain: 3+ A? ucbvax.berkeley.edu. (37)
```

говорит, что хост **h2opolo** запрашивает у сервера имён **helios** адресную запись (qtype=A) для имени **ucbvax.berkeley.edu**. Идентификатор запроса имеет значение **3**. Знак **+** показывает наличие флага **recursion desired**[1]. Размер пакета составляет 37 байтов без учёта заголовков UDP и IP. Поскольку пакет содержит обычный запрос, поле **op** опущено. Если бы это поле содержало иную операцию, соответствующий код был бы выведен между **3** и **+**. Поле **qclass** также содержит стандартное значение **C_IN**, которое опущено при выводе. Любое другое значение **qclass** было бы выведено вслед за символом **A**.

При анализе пакета проверяется наличие в нем аномалий и в результате такой проверки строка вывода может содержать дополнительные поля, заключённые в квадратные скобки. Если запрос содержит разделы answer (ответ), authority records (запись о полномочиях) или additional records (дополнительные записи), значения ancount, nscount или arcount выводятся как **[na]**, **[nn]** или **[nau]**, где n показывает значение соответствующего счётчика. Если установлены какие-либо биты отклика (**AA**, **RA** или **rcode**) или в байтах 2 и 3 установлены любые биты **MBZ** (должно быть нулем), выводится поле **[b2&3=x]**, где x - шестнадцатеричное значение байтов 2 и 3 из заголовка.

UDP-отклики от серверов DNS

Для вывода откликов сервера имён используется формат

```
src > dst:  id op rcode flags a/n/au type class data (len)
```

Например,

```
helios.domain > h2opolo.1538: 3 3/3/7 A 128.32.137.3 (273)
helios.domain > h2opolo.1537: 2 NXDomain* 0/1/0 (97)
```

Первая строка показывает, что сервер **helios** отвечает на запрос с **id=3** от хоста **h2opolo** сообщениям с тремя записями **answer**, 3 записями **NS** и 7 дополнительными записями. Первая запись **answer** имеет тип **A** (адрес) и содержит IP-адрес указанного в запросе хоста (128.32.137.3). Общий размер отклика составляет 273 байта без учёта заголовков UDP и IP. Поля **op** (Query) и код отклика (NoError) были опущены при выводе.

Во второй строке **helios** отвечает на запрос 2 с кодом **NXDomain** (несуществующий домен) без записей **answer**, с одной записью **NS** и без записей **authority**. Символ ***** показывает, установленный бит полномочного отклика (**authoritative answer**). Ввиду отсутствия записей answer не выводится никакой информации о типе, классе и данных.

В строке вывода могут также появляться индикаторы флагов RA (рекурсия доступна) "**-**" и TC (усечённое сообщение) "**|**". Если раздел **question** (вопрос) не содержит в точности одну запись, выводится поле **[nq]**.

Отметим, что запросы и отклики DNS могут быть достаточно велики и принятое по умолчанию значение кадра захвата (68 байтов) может не обеспечить достаточное количество данных из пакета. Для просмотра трафика DNS целесообразно увеличить размер кадра захвата вдвое с помощью опции **-s 128**.

Декодирование SMB/CIFS

Программа tcpdump поддерживает функции декодирования пакетов SMB/CIFS/NBT, использующих порты **UDP/137**, **UDP/138** и **TCP/139**. Поддерживаются также некоторые примитивы декодирования данных **IPX** и **NetBEUI SMB**.

По умолчанию декодирование происходит с минимальным выводом информации, а для увеличения информативности служит опция **-v**. Отметим, что при использовании опции **-v** один пакет **SMB** может занимать больше экранной страницы, поэтому указывайте опцию только при необходимости, чтобы не утонуть в море выводимых на экран данных.

При декодировании сеансов **SMB**, содержащих текстовые строки Unicode может потребоваться установка переменной окружения **USE_UNICODE =1**.

Информацию о формате пакетов **SMB** вы сможете найти на сайте www.cifs.org или одном из зеркал samba.org.

Запросы и отклики NFS

Запросы и отклики Sun NFS[2] имеют вид:

```
src.sport > dst.nfs: NFS request xid xid len op args
src.nfs > dst.dport: NFS reply xid xid reply stat len op results
```

Ниже показан пример вывода информации для пакетов NFS

```
sushi.6709 > wrl.nfs:
   112 readlink fh 21,24/10.73165
wrl.nfs > sushi.6709:
   reply ok 40 readlink "../var"
sushi.201b > wrl.nfs:
   144 lookup fh 9,74/4096.6878 "xcolors"
wrl.nfs > sushi.201b:
   reply ok 128 lookup fh 9,74/4134.3150
```

Первая строка показывает, что хост **sushi** передаёт транзакцию с идентификатором 6709[3] хосту **wrl**. Размер запроса составляет 112 байтов без учёта заголовков **UDP** и **IP**. Запрашиваемая операция **readlink**[4] для файла с идентификатором (handle) **fh 21,24/10.731657119** была успешно выполнена и хост **wrl** возвращает результат **ok** с содержимым символьной ссылки.

Затем **sushi** запрашивает у хоста **wrl** поиск файла **xcolors** в каталоге **9,74/4096.6878** и **wrl** возвращает отклик с идентификатором транзакции.

При использовании опции **-v** вывод становится более информативным[5]

```
sushi.1372a > wrl.nfs: 148 read fh 21,11/12.195 8192 bytes @ 24576
```

[1]При отсутствии записи на сервере имён следует сделать рекурсивный запрос к другому серверу.
[2]Network File System - сетевая файловая система.
[3]Номер, указанный после имени отправителя, задаёт не порт, а номер транзакции.
[4]Прочесть символьную ссылку.
[5]При использовании опции **-v** будут выводиться также поля заголовков **IP** (**TTL**, **ID**, **length**, **fragmentation**), которые опущены в приведённом примере.

```
wrl.nfs > sushi.1372a: reply ok 1472 read REG 100664 ids 417/0 sz 29388
```

В первой строке показан запрос **sushi** к хосту **wrl** на чтение 8192 байтов из файла **21,11/12.195**, начиная со смещения 24576. Хост **wrl** возвращает результат **ok**, показанный во второй строке пакет является первым фрагментом отклика, содержащим 1472 байта прочитанных данных. Последующие фрагменты не имеют заголовков **NFS** и **UDP**, поэтому информация об этих пакетов может не появиться на экране, если вы задали в команде тот или иной фильтр. Благодаря использованию опции **-v** выводятся также некоторые атрибуты прочитанного файла (**REG** - обычный файл, восьмеричное представление прав доступа, идентификаторы владельца и группы, а также размер файла).

При использовании опции **-vv** объем выводимой информации может дополнительно возрасти.

Отметим, что запросы NFS могут быть достаточно большими и при использовании опции **-v** выводимая информация может занять несколько экранных страниц. В некоторых случаях будет полезно уменьшить размер кадра захвата с помощью опции **-s** (например, **-s 192)**.

Отклики NFS не указывают явно операции RPC, вместо этого tcpdump сохраняет информацию о последних запросах и при выводе откликов указывает соответствующие идентификаторы транзакций.

Запросы и отклики AFS

Вывод информации для запросов и откликов AFS[1] имеет вид

```
src.sport > dst.dport: rx packet-type
src.sport > dst.dport: rx packet-type service call call-name args
src.sport > dst.dport: rx packet-type service reply call-name args
```

Ниже показан пример вывода информации для пакетов AFS

```
elvis.7001 > pike.afsfs:
     rx data fs call rename old fid 536876964/1/1 ".newsrc.new"
     new fid 536876964/1/1 ".newsrc"
pike.afsfs > elvis.7001: rx data fs reply rename
```

В первой строке хост **elvis** передаёт пакет **RX** хосту **pike**. Этот пакет адресован файловому серверу (**fs**) и начинает вызов удалённой процедуры (RPC). Вызов RPC содержит команду **rename** (переименовать) с идентификатором старого каталога **536876964/1/1** и именем **.newsrc.new**, а также новым идентификатором **536876964/1/1** и именем **.newsrc**. Хост **pike** возвращает отклик RPC с информацией об изменении имени файла.

В общем случаем все пакеты AFS RPC декодируются по меньшей мере как имена процедур RPC. Во многих случаях декодируется также один или несколько передаваемых процедуре аргументов.

Формат вывода должен быть понятен для тех, кто знаком с AFS и RX.

При использовании опции **-v** обеспечивается вывод дополнительной информации (идентификаторы вызовов RX, номера вызовов, порядковые номера, флаги пакетов RX). Опция **-vv** дополнительно увеличивает объем выводимой информации (в частности, сведений о согласовании MTU для пакетов RX ack), а опция **-vvv** обеспечивает также вывод параметров безопасности и идентификаторов сервиса.

Для пакетов **abort** выводятся коды ошибок (за исключением пакетов Ubik, поскольку эти пакеты используются для обозначения пакетов **yes vote** протокола Ubik).

Отметим, что запросы AFS могут быть достаточно велики, поэтому использование флага **-v** иной раз будет приводить к выводу для пакета многостраничной информации. Можно задать размер кадра захвата с помощью опции **-s** для обеспечения более читаемых результатов (например, **-s 256**).

Отклики AFS явно не указывают операции RPC, поэтому tcpdump отслеживает последние запросы и помечает отклики идентификаторами соответствующих запросов.

KIP AppleTalk (DDP по протоколу UDP)

Пакеты AppleTalk DDP, инкапсулированные в дейтаграммы UDP, извлекаются из дейтаграмм и отображаются как пакеты DDP (заголовки UDP отбрасываются). Для преобразования имён сетей и хостов AppleTalk служит файл **/etc/atalk.names**, строки которого имеют форму **адрес (номер) - имя**

```
1.254     ether
16.1      icsd-net
1.254.110 ace
```

В приведённом примере первые две строки содержат имена сетей AppleTalk, а третья - имя хоста[2]. Для разделения номера и имени в файле могут служить пробелы или символы табуляции. Файл **/etc/atalk.names** может содержать пустые строки и строки комментариев, начинающиеся с символа **#**.

Адреса AppleTalk выводятся в формате net.host.port, например,

```
144.1.209.2 > icsd-net.112.220
office.2 > icsd-net.112.220
jssmag.149.235 > icsd-net.2
```

Если файл **/etc/atalk.names** не содержит записи для той или иной сети или хоста, соответствующее поле выводится в цифровом формате. В первой строке показан пакет NBP (DDP порт 2), отправленный узлом **209** сети **144.1** в порт **220** узла **112** сети **icsd-net**. Вторая строка отличается от первой только тем, что указано также символьное имя отправителя (**office**). В третьей строке показан пакет, отправленный из порта **235** хостом **149** сети **jssmag** всем хостам[3] сети **icsd-net**, прослушивающим порт NBP

Пакеты протоколов NBP (name binding protocol) и ATP (AppleTalk transaction protocol) выводятся с интерпретацией их содержимого. Для остальных протоколов просто выводится дамп имени протокола или его номера, если имя неизвестно, и размер пакета.

[1]Andrew File System.
[2]Хост отличается от сети наличием в номере третьего октета.
[3]Отметим, что широковещательный адрес 255 задаётся просто именем или номером сети без указания хоста. По этой причине разумно сохранять имена хостов и сетей в файле **/etc/atalk.names** отдельно.

Пакеты NBP выводятся в формате, подобном приведённому ниже:

```
icsd-net.112.220 > jssmag.2: nbp-lkup 190: "=:LaserWriter@*"
jssmag.209.2 > icsd-net.112.220: nbp-reply 190: "RM1140:LaserWriter@*" 250
techpit.2 > icsd-net.112.220: nbp-reply 190: "techpit:LaserWriter@*" 186
```

Первая строка показывает запрос на преобразование имени для принтеров **LaserWriter**, переданный хостом **112** сети **icsd** по широковещательному адресу сети **jssmag**. Идентификатор запроса **nbp** имеет значение 190. Вторая строка содержит отклик на этот запрос от хоста **jssmag.209**, сообщающего о наличии ресурса **LaserWriter** с именем **RM1140**, зарегистрированного на порту **250**. В третьей строке показан другой отклик на тот же запрос, говорящий, что хост **techpit** имеет ресурс **LaserWriter** с именем **techpit**, зарегистрированный на порту 186.

Пример формата вывода пакетов ATP показан ниже:

```
jssmag.209.165 > helios.132: atp-req  12266<0-7> 0xae030001
helios.132 > jssmag.209.165: atp-resp 12266:0 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:1 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:2 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:4 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:6 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp*12266:7 (512) 0xae040000
jssmag.209.165 > helios.132: atp-req  12266<3,5> 0xae030001
helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
jssmag.209.165 > helios.132: atp-rel  12266<0-7> 0xae030001
jssmag.209.133 > helios.132: atp-req* 12267<0-7> 0xae030002
```

Хост **jssmag.209** инициирует транзакцию **12266** с хостом **helios**, запрашивая до 8 пакетов (**<0-7>**). Шестнадцатеричное число в конце строки содержит значение поля **userdata** из запроса.

Хост **helios** отвечает на полученный запрос 8 пакетами по 512 байтов. Число после номера транзакции указывает порядковый номер пакета для данной транзакции, а число в скобках - размер данных в пакете без учёта заголовка ATP. Символ ***** для пакета 7 показывает наличие флага **EOM**.

Хост **jssmag.209** после получения пакетов запрашивает повторную передачу пакетов 3 и 5, а **helios** повторяет эти пакеты, после чего **jssmag.209** завершает транзакцию. В последней строке показан новый запрос хоста **jssmag.209**. Символ ***** показывает, что флаг **XO** (exactly once) для пакета не установлен.

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru