Санкт-Петербург, ул. Седова, 80, телефон (812) 449-0770, факс (812) 449-0771, E-mail: info@bilim.com

Николай Малых
nmalykkh@bilim.com

Описание программы tcpdump подготовленное на основе информации из руководства man. Программа tcpdump работает на всех платформах UNIX, а сейчас существует и аналог этой программы для платформ Windows -Windump.

tcpdump

http://www.tcpdump.org

Программа **tcpdump**, включаемая в большинство дистрибутивов UNIX, выводит заголовки пакетов для сетевого интерфейса в соответствии с заданным логическим выражением. Программа также допускает использование с флагом **-w** для записи пакетов 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 прекращен также после обработки определенного числа пакетов.

При завершении работы tcpdump выводит значения счетчиков для:

- 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 счетчиков по сигналу SIGINFO (этот сигнал может быть подан обычно с помощью клавиш control-T) без прерывания работы программы.

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 псевдо-устройству (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 найдете, 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 **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.

[1] **Promiscuous -** “Неразборчивый” режим (режим захвата), 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.

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. Таблица содержит список опций tcpdump и описание каждой их них. Отметим, что некоторые опции поддерживаются не всеми платформами, на которых может использоваться программа tcpdump.

Таблица 1 Опции командной строки tcpdump

Опция	*Описание*
-A	задает вывод каждого пакета (без заголовков канального уровня) в формате ASCII. Этот режим удобен для сбора трафика HTTP.
-c <число пакетов>	задает завершение работы программы после захвата заданного числа пакетов.
-C <размер файла>	задает необходимость проверки размера файла захвата перед записью в него каждого нового пакета. Если размер файла превышает значение параметра **file_size**, этот файл закрывается и создается 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 номер файла. Переменная file_size задает размер файла в миллионах байтов (не в мегабайтах = 1 048 576 байт).
-d	задает вывод дампа скомпилированного кода соответствия пакетов (packet-matching code) в понятном человеку формате и завершение работы программы.
-dd	выводит дамп кода соответствия в виде фрагмента C-программы.
-ddd	выводит дамп кода соответствия в виде строки десятичных значений, перед которой следует строка со значением счетчика.
-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[3]. 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>	задает использование алгоритма и секрета **spi@ipaddr** для расшифровки пакетов IPsec ESP, направленных по адресу **ipaddr** и содержащих and 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 ее с реальными секретными ключами не следует, поскольку введенный в командной строке ключ IPsec доступен другим пользователям системы[4]. Кроме явного указания параметров в командной строке их можно задать в файле опций, который tcpdump будет читать при получении первого пакета ESP.
-f	задает 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>	задает использование фильтров, содержащихся в указанном файле. В этом случае заданные в командной строке фильтры игнорируются.
-i <интерфейс>	задает 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.

[3]Например, Windows или UNIX-системы, в которых не поддерживается команда **ifconfig -a**

[4]Например, его можно увидеть с помощью команды **ps**.

Опция	*Описание*

-l	задает буферизацию строк **stdout**. Эта опция полезна в тех случаях, когда вы хотите 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	задает вывод списка известных типов канального уровня и завершение работы программы.
-m <файл>	загружает модуль определений SMI MIB из указанного файла. Эта опция может использоваться неоднократно для загрузки нескольких модулей MIB.
-n	отключает преобразование адресов и номеров портов в символьные имена.
-N	задает использование только имен хостов, а не полных доменных имен. Например, вместо **lhotze.bilim-systems.net** при использовании этой опции моя рабочая станция будет обозначаться как **lhotze**.
-O	отключает оптимизатор кода проверки соответствия пакетов условиям фильтрации (стр. 4). Используйте эту опцию, если вам покажется, что оптимизатор работает с ошибками.
-p	указывает программе, что интерфейс не нужно переводить в режим захвата[5]. Опцию **-p** нельзя использовать вместе с фильтром **ether host {local-hw-addr} or ether broadcast**.
-q	задает вывод минимального объема информации.
-R	при установке этого флага предполагается, что пакеты ESP/AH используют старый вариант спецификации[6] и tcpdump не будет выводить поля replay prevention (защита от воспроизведения). Поскольку спецификация ESP/AH не включает поля с номером версии, tcpdump не может определить версию протокола ESP/AH по заголовкам пакетов.
-r <файл>	задает 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	задает вывод абсолютных порядковых номеров TCP взамен относительных.
-s <snaplen>	задает захват из каждого пакета **snaplen** байтов вместо отбираемых по умолчанию 68 байтов[7]. Значение 68 подходит для протоколов IP, ICMP, TCP и UDP но может приводить к потере протокольной информации для некоторых пакетов DNS (см. стр. 12) и NFS (см. стр. 12). Потеря части пакетов по причине малого размера кадра захвата (snapshot) указывается в выходных данных полями вида **[|proto]'**, где **proto** – имя протокольного уровня, на котором произошло отсечение части пакета[8]. Отметим, что увеличение кадра захвата приведет к дополнительным временным затратам на обработку пакетов и уменьшению числа буферизуемых пакетов, что может привести к потере части пакетов. Используйте минимальное значение **snaplen**, которое позволит обойтись без потери информации об интересующем вас протоколе. Установка **snaplen = 0** приведет к захвату полных пакетов.
-T <тип>	задает интерпретацию пакетов, выбранных с помощью фильтра, как пакетов указанного параметром типа. В настоящее время поддерживаются типы **aodv**[9], **cnfp**[10], **rpc**[11], **rtp**[12], **rtcp**[13], **snmp**[14], **tftp**[15], **vat**[16] и **wb**[17].
-t	отключает вывод временных меток в каждой строке дампа.
-tt	задает вывод в каждой строке дампа неформатированных временных меток.
-ttt	задает вывод временных интервалов (в микросекундах) между захватом предыдущего и данного пакетов в каждой строке дампа.
-tttt	задает вывод временных меток в принятом по умолчанию формате для каждой строки дампа.
-u	задает вывод манипуляторов (handle) NFS без декодирования.
-U	задает режим “буферизации на уровне пакетов” для файлов, сохраняемых с помощью опции **-w**. В этом режиме каждый пакет записывается в выходной файл как только он будет захвачен (не дожидаясь заполнения выходного буфера). Флаг **-U** не будет поддерживаться, если программа **tcpdump** была скомпилирована со старой опцией **libpcap**, не поддерживающей функцию **pcap_dump_flush()**.

[5]Интерфейс может быть переведен в режим захвата другими программами, поэтому использование флага **-p** отнюдь не гарантирует работу интерфейса в обычном режиме – программа просто не будет переводить этот интерфейс в режим захвата. 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).
[6]RFC1825 - RFC1829
[7]В SunOS NIT минимум составляет 96 байтов.
[8]Например, при захвате пакетов в сети Ethernet с помощью команда **tcpdump -s 12** на выходе будут появляться строки типа **22:31:43.385357 [|ether]**, показывающие, что усекновение пакетов произошло на уровне Ethernet
[9]Протокол Ad-hoc On-demand Distance Vector.
[10]Протокол Cisco NetFlow.
[11]Протокол Remote Procedure Call (удаленный вызов процедур).
[12]Протокол Real-Time Applications (приложения в реальном масштабе времени).
[13]Протокол управления приложениями реального времени (Real-Time Applications control protocol).
[14]Простой протокол сетевого управления (Simple Network Management Protocol).
[15]Тривиальный протокол обмена файлами (Trivial File Transfer Protocol).
[16]Visual Audio Tool.
[17]Распределенные доски White Board.

Опция	*Описание*
-v	задает вывод дополнительной информации при захвате файлов. 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	задает дополнительное увеличение объема выводимой информации (например, полное декодирование пакетов SMB, вывод дополнительных полей откликов NFS и т. п.).
-vvv	задает максимальный объем выводимой информации (например, полностью выводятся опции **telnet SB ... SE**). При использовании вместе с ключом **-X** опции Telnet выводятся также в шестнадцатеричном представлении.
-w <файл>	задает запись необработанных (raw) пакетов. Собранные в файл пакеты можно впоследствии просматривать с использованием флага **-r** или передавать для анализа другим программам (например, Ethereal). Если в качестве имени файла указан символ **-**, запись осуществляется на стандартное устройство вывода (**stdout**).
-x	задает 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. 3). Отметим, что при захвате полных кадров канального уровня дамп может включать также байты заполнения, если пакет сетевого уровня имеет малый размер.
-xx	задает вывод шестнадцатеричного дампа для каждого пакета с включением заголовков канального уровня.
-X	задает вывод дампа в шестнадцатеричном и ASCII-формате без заголовков канального уровня. Эта опция может быть очень удобна при анализе новых протоколов.
-XX	задает вывод дампа в шестнадцатеричном и ASCII-формате с включением заголовков канального уровня.
-y <тип>	задает тип канального уровня, используемого при захвате пакетов. Поддерживаемые значения можно посмотреть с помощью флага **-L**.

Фильтрация при сборе пакетов

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

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

type

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

задает 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). 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 захвата (например, захват с фиктивного интерфейса **any** в Linux-системах) могут использоваться классификаторы **inbound** и **outbound**.

proto

задает протокол, к которому должны относиться пакеты. Этот классификатор может принимать значения **ether**, **fddi**[18], **tr**[19], **wlan**[20], **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[21].

Кроме объектов и квалификаторов 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 квалификаторов. 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.

[18]**fddi** в действительности является псевдонимом **ether**; 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 содежат 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.

[19]**tr** также является в действительности псевдонимом **ether**, поскольку оба типа кадров используют весьма похожую структуру заголовков.

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

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

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 приведенных 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[22]).
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[23]. Указанное параметром имя хоста должно преобразовываться в IP-адрес механизмами преобразования имен, доступными локальному компьютеру (файл /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, указанному 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.
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 сети>	будет отбирать все пакеты IP[24], 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.
dst port <порт>	отберет 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 номер или неоднозначное обозначение порта (только порт, без протокола) 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** – трафик **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 трех 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).
less <размер>	будет собирать пакеты, размер которых не превышает указанного значения.
greater <размер>	будет собирать пакеты, размер которых не меньше указанного значения.
ip proto <протокол>	отбирает все пакеты IP, содержащие заданный идентификатор типа в поле типа 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, перед этими идентификаторами следует помешать символ \ (слэш)[25]. Отметим, что этот примитив не проверяет цепочки протокольных заголовков.
ip6 proto <протокол>	будет отбирать все пакеты IPv6 указанного типа. Отметим, что этот примитив не проверяет цепочки протокольных заголовков.

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

[23]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.

[24]Этот примитив не может использоваться для сетей IPv6.

[25]При работе с командным интерпретатором C-shell следует добавлять еще один слэш (****)

Примитив	*Описание*
ip6 protochain <протокол>	отберет 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.
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 учетом маски подсети для интерфейса, который используется для захвата пакетов. Если маска подсети для интерфейса недоступна[26], фильтр может работать некорректно.
ether multicast	собирает все кадры с групповыми адресами Ethernet. Ключевое слово **ether** использовать необязательно. Логически это правило эквивалентно выражению **ether[0] & 1 != 0**.
ip multicast	отбирает пакеты с групповыми адресами IP.
ip6 multicast	отбирает пакеты с групповыми адресами IPv6.
ether proto <протокол>	отбирает кадры Ethernet с заданным типом протокола. Протокол может быть указан по номеру или имени[27] (**ip**, **ip6**, **arp**, **rarp**, **atalk**, **aarp**, **decnet**, **sca**, **lat**, **mopdl**, **moprc**, **iso**, **stp**, **ipx**, **netbeui**). В случаях использования правила для протоколов FDDI (например, **fddi protocol arp**), Token Ring (например, tr protocol arp) и IEEE 802.11 (например, **wlan protocol arp**) в большинстве случаев идентификация протокола производится на основании заголовка 802.2 Logical Link Control (LLC), который обычно помещается после заголовка FDDI, Token Ring или 802.11. При фильтрации для большинства протоколов FDDI, Token Ring и 802.11 программа **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 для пакетов использования формата SNAP с OUI = 0x000000 не выполняется за исключением перечисленных ниже случаев: **iso** tcpdump проверяет поля **DSAP**[28] и **SSAP**[29] в заголовках LLC; **stp** **netbeui** tcpdump проверяет поле DSAP в заголовке LLC; **atalk** tcpdump проверяет использование в кадре формата SNAP с OUI = 0x080007 и тип (etype) AppleTalk. Для случая Ethernet проверяются поля типа Ethernet для большинства протоколов. Исключениями являются протоколы **iso** **sap** **netbeui** tcpdump проверяет для кадра принадлежность к 802.3 и заголовок LLC (как это описано выше для FDDI, Token Ring и 802.11); **atalk** проверяется типа **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.
decnet src <хост>	собирает все пакеты от указанного хоста DECNET, который может быть задан по адресу или имени DECNET[30].
decnet dst <хост>	отбирает все пакеты, адресованные указанному хосту DECNET.

[26]Интерфейс не имеет маски или сбор пакетов осуществляется со всех интерфесов хоста Linux (фиктивный интерфейс **any**).
[27]Перечисленные здесь имена протоколов могут использоваться также в качестве ключевых слов, поэтому имени протокола должен предшествовать символ \ (например, **arp**).
[28]Destination Service Access Point – точка доступа к сервису для получателя.
[29]Source Service Access Point – точка доступа к сервису для отправителя.
[30]Поддержка имен DECNET обеспечивается только на хостах ULTRIX, настроенных для использования DECNET.

Примитив	*Описание*
decnet host <хост>	собирает все пакеты, содержащие адрес указанного хоста DECNET в поле отправителя или получателя.
ifname <интерфейс>	отбирает все пакеты, полученные от указанного интерфейса[31].
on <интерфейс>	синоним **ifname**.
rnr <номер>	собирает только пакеты, записанные в файл программой pf в соответствии с правилом, имеющим указанных номер. Это правило работает только при просмотре файлов, собранных с помощью **pf**.
rulenum <номер>	синоним для **rnr**.
reason <код>	собирает только пакеты, соответствующие указанному коду причины (PF reason code). Известные коды причин включают **match**, **bad-offset**, **fragment**, **short**, **normalize**, **memory**. Правило работает только при просмотре файлов, собранных с помощью **pf**.
action <действие>	отбирает пакеты, записанные в файл указанной операцией PF (**pass** или **block**). Правило работает только при просмотре файлов, собранных с помощью **pf**.
ip, ip6, arp, rarp, atalk, aarp, decnet, iso, stp, ipx, netbeui	используются в качестве сокращения для: **ether proto p** где p – один из перечисленных протоколов.
lat, moprc, mopdl	сокращения для: **ether proto p** где p – один из перечисленных протоколов. Отметим, что в настоящее время tcpdump не умеет разбирать эти протоколы.
vlan [vlan_id]	отбирает кадры IEEE 802.1Q VLAN. Если указан необязательный идентификатор VLAN, выделяются лишь пакеты, относящиеся в указанной виртуальной ЛВС. Отметим, что первое ключевой слово **vlan** изменяет расчет смещения полей для оставшейся части выражения с учетом размеров полей VLAN в заголовке кадра.
tcp, udp, icmp	используется в качестве сокращения для **ip proto p** или **ip6 proto p** где p – один из перечисленных протоколов.
iso proto <протокол>	собирает пакеты с указанным типом протокола OSI. Протокол может быть указан по номеру или имени (**clnp**, **esis**, **isis**).
clnp, esis, isis	сокращения для выражений: **iso proto p** где p – один из перечисленных протоколов.
l1, l2, iih, lsp, snp, csnp, psnp	сокращения 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.
llc	собирает пакеты ATM с инкапсуляцией LLC для SunATM (Solaris).
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 мета-устройствам (VPI=0, VCI=1).
bcc	собирает пакеты ATM для SunATM (Solaris), относящиеся к широковещательным сигнальным устройствам (VPI=0, VCI=2).

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

Примитив	*Описание*
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 мета-устройствам и содержащие сообщения **Q.2931 Setup**, **Call Proceeding**, **Connect**, **Connect Ack**, **Release**, **Release Done**.

Логические выражения

Выражения типа

```
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 +, -, *, /, **&**, |, <<, >>, оператор длины (offset) и данные из пакетов. Для получения значений полей из пакетов применяется синтаксис:

```
proto [offset : size]
```

Параметр **proto** может содержать идентификатор одного из протоколов (**ether**, **fddi**, **tr**, **wlan**, **ppp**, **slip**, **link**, **ip**, **arp**, **rarp**, **tcp**, **udp**, **icmp**, **ip6**) и задает уровень протокола[32], для которого извлекаются данные. Отметим, что **tcp**, **udp** и другие протоколы верхних уровней относятся только к пакетам IPv4, а не IPv6[33]. Параметр **offset** задает смещение в байтах относительно начала заголовка указанного уровня. Необязательный параметр **size** определяет размер интересующего поля в байтах. Допустимы значения размера 1, 2 и 4, по умолчанию просматривается 1 байт.

Оператор длины, 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
```

соберет только нефрагментированные дейтаграммы 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 поля и значения смещений могут задаваться не только числами, но и именами. В частности, для протокола поддерживается параметр **icmptype** (поле типа ICMP), который может принимать значения **icmp-echoreply**, **icmp-unreach**, **icmp-sourcequench**, **icmp-redirect**, **icmp-echo**, **icmp-routeradvert**, **icmp-routersolicit**, **icmp-timxceed**, **icmp-paramprob**, **icmp-tstamp**, **icmp-tstam-preply**, **icmp-ireq**, **icmp-ireqreply**, **icmp-maskreq**, **icmp-maskreply**. Для флагов TCP можно использовать идентификаторы **tcp-fin**, **tcp-syn**, **tcp-rst**, **tcp-push**, **tcp-ack** и **tcp-urg**.

Примитивы в выражениях можно группировать с использованием

- скобок[34];
- отрицания (**!** или **not**);
- логического пересечения (**&&** или **and**);
- логического объединения (|| или **or**).

Оператор отрицания имеет высший уровень приоритета, операции объединения и пересечения имеют одинаковый приоритет и выполняются слева направо в порядке следования. Отметим, что для операции логического пересечения 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 мета-символы командного интерпретатора, должны передаваться как один аргумент, заключенный в кавычки.

Примеры фильтров

[32]Протоколы **ether**, **fddi**, **wlan**, **tr**, **ppp**, **slip** и **link** указывают на канальный уровень.
[33]Поддержка IPv6 будет реализована в следующих версиях.
[34]В зависимости от используемого командного интерпретатора перед символами открывающих и закрывающих скобок может потребоваться включение esc-символа.

Таблица 3 Примеры фильтров tcpdump

Фильтр	*Выполняемые действия*
`tcpdump host sundown`	Выводит все пакеты, принимаемые и передаваемые хостом **sundown**
`tcpdump host helios and \( hot or ace \)`	Выводит пакеты, передаваемые между хостом **helios** и любым из хостов **hot** или **ace.**
`tcpdump ip host ace and not helios`	Выводит пакеты передаваемые между хостом **ace** и любым хостом, за исключением **helios**.
`tcpdump net ucb-ether`	Выводит все пакеты, передаваемые или принимаемые хостами сети **ucb-ether**.
`tcpdump 'gateway snup and (port ftp or ftp-data)'`	Выводит весь трафик **ftp**, проходящий через шлюз **snup**[35].
`tcpdump ip and not net localnet`	Выводит весь трафик, за исключением исходящего от локальных хостов и адресованного им.
`tcpdump 'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst net localnet'`	Выводит стартовые (SYN) и конечные (FIN) пакеты TCP, исключая соединения локальных хостов.
`tcpdump 'gateway snup and ip[2:2] > 576'`	Выводит переданные через шлюз **snup** пакеты IP, размер которых превышает 576 байтов.
`tcpdump 'ether[0] & 1 = 0 and ip[16] >= 224'`	Выводит широковещательные и групповые пакеты, которые не были переданы с использованием широковещательных и групповых адресов Ethernet.
`tcpdump 'icmp [icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply'`	Выводит все пакеты ICMP, кроме запросов и откликов **echo** (т. е., кроме пакетов **ping**).

Формат вывода

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

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. Нормальные 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 так называемым SNAP-пакетом.

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. Как и для сетей FDDI предполагается, что кадры содержат пакеты LLC. Независимо от наличия опции **-e** выводится информация о заданном отправителем маршруте (source routing), если пакет содержит такую информацию.

При использовании 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[36]. Поле типа пакета выводится первым и может принимать значение **ip**, **utcp** или **ctcp**. Остальные сведения относятся к пакету IP. Для пакетов TCP вслед за типом выводится идентификатор соединения. Если для пакета используется компрессия, сжатый заголовок декодируется перед выводом. Для специальных случаев[37] 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 - urgent pointer), **W** (окно - window), **A** (подтверждение - ack), **S** (порядковый номер - sequence number) и I (идентификатор пакета - packet ID), сопровождаемые величиной изменения (+n или -n) или указателем на новое значение параметра (=n). Далее выводится информация о количестве 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/RARP

[35]Кавычки позволяют избавиться от ошибок при анализе скобок командным интерпретатором.

[36]Компрессия заголовков TCP/IP для каналов SLIP описана в документе RFC 1144, который вы можете загрузить с сайта http://rfc-editor.org/rfc/rfc1144.txt.

[37]RFC 1144 определяет как специальные случаи интерактивный трафик и передачу больших объемов трафика.

Информация, выводимая для пакетов arp/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 нашем примере IP-адрес обозначен как **csam**, а MAC-адрес – как **CSAM**). Если ввести команду с опцией **-n**, результат будет выглядеть как

```
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
```

Пакеты TCP

Для понимания приведенной здесь информации вы должны быть знакомы с протоколом TCP. Если протокол известен вам недостаточно хорошо, обратитесь к документу RFC 793[38].

Формат вывода для протокола TCP в общем случае имеет вид:

```
src > dst: flags data-seqno ack window urg options
```

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 приемном 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, заключенные в угловые скобки.

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

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

```
1) rtsg.1023 > csam.login: S 768512:768512(0) win 4096 <mss 1024>
2) csam.login > rtsg.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>
3) rtsg.1023 > csam.login: . ack 1 win 4096
4) rtsg.1023 > csam.login: P 1:2(1) ack 1 win 4096
5) csam.login > rtsg.1023: . ack 2 win 4096
6) rtsg.1023 > csam.login: P 2:21(19) ack 1 win 4096
7) csam.login > rtsg.1023: P 1:2(1) ack 21 win 4077
8) csam.login > rtsg.1023: P 2:3(1) ack 21 win 4077 urg 1
9) 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[39]. Пакет не содержит в себе подтверждения, доступный размер приемного 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 программа **tcpdump** видит первый пакет в данном соединении, она выводит 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. 3) 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 передается с флагом PUSH. Строка 7 содержит отправленное хостом **csam** подтверждение приема данных от хоста **rtsg** вплоть до байта с номером 21 (но не включая этот байт). Большая часть этих данных сохраняется в приемном буфере, поскольку **csam** показывает уменьшение приемного окна на 19 байтов. В этом пакете **csam** также передает 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. 3), 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[40], 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 полей:

[38]Документ вы можете загрузить с сайта http://rfc-editor.org/rfc/rfc793.txt. На сайте http://www.protocols.ru имеется перевод документа на русский язык.

[39]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.

[40]Размер опции слишком мал или указанный размер опции выходит за пределы указанного размера заголовка.

```
CWR | ECE | URG | ACK | PSH | RST | SYN | FIN
```

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

1) Инициатор соединения передает пакет с установленным флагом SYN.

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

3) Инициатор передает в ответ пакет с флагом ACK.

Давайте создадим фильтр, который будет собирать пакеты, в которых установлен только флаг SYN (этап 1). Отметим, что пакеты этапа 2 (SYN-ACK) нас не будут интересовать, 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 нам 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 = 2
```

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 нам 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.

Пакеты UDP

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

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

Приведенная 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 (на основе номера порта) и для этих протоколов выводится соответствующая информация. В частности, программа tcpdump выводит дополнительные сведения для пакетов DNS[41] и вызовов NFS с помощью Sun RPC[42].

[41]Спецификация протокола DNS (Domain Name System) приведена в RFC 1034 и RFC 1035, которые можно загрузить с сайта http://rfc-editor.org/rfc/.

[42]Спецификацию протокола RPC (Remote Procedure Call – удаленный вызов процедур) содержится в документе RFC 1050, который вы можете загрузить с сайта http://rfc-editor.org/rfc/rfc1050.txt.

Запросы UDP к серверам DNS

Формат вывода для запросов DNS имеет вид

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

Например

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

говорит, что хост **h2opolo** запрашивает у сервера имен **helios** адресную запись (qtype=A) для имени **ucbvax.berkeley.edu**. Идентификатор запроса имеет значение **3**. Знак + показывает наличие флага **recursion desired**[43]. Размер пакета составляет 37 байтов без учета заголовков UDP и IP. Поскольку пакет содержит нормальный запрос, поле **op** опущено. Если бы это поле содержало что-то иное, 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, заключенные в квадратные скобки. Если запрос содержит секции 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 счетчика. Если установлены какие-либо биты отклика (**AA**, **RA** или **rcode**) или в байтах 2 и 3 установлены любые биты **must be zero** (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 имен 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** (address - адрес) и содержит IP-адрес указанного в запросе хоста (128.32.137.3). Общий размер отклика составляет 273 байта без учета 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 **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 (усеченное сообщение) "|". Если секция **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 используйте данную 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 http://www.samba.org/ или одном из его зеркал.

Запросы и отклики NFS

Запросы и отклики Sun NFS[44] имеют вид:

```
src.xid > dst.nfs: len op args
src.nfs > dst.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** передает транзакцию с идентификатором 6709[45] хосту **wrl**. Размер запроса составляет 112 байтов без учета заголовков **UDP** и **IP**. Запрашиваемая операция **readlink**[46] для файла с идентификатором (handle) **fh 21,24/10.731657119** была успешно выполнена и хост **wrl** возвращает результат **ok** с содержимым символьной ссылки.

В третьей строке **sushi** запрашивает у хоста **wrl** поиск файла **xcolors** в каталоге **9,74/4096.6878**.

При использовании опции **-v** вывод становится более информативным[47]:

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

[43] При отсутствии записи на сервере имен следует сделать рекурсивный запрос к другому серверу.
[44] Network File System – сетевая файловая система.
[45] Отметим, что номер, указанный после имени отправителя, задает не порт, а номер транзакции.
[46] Прочесть символьную ссылку.
[47] При использовании опции **-v** будут выводиться также поля заголовков **IP** (**TTL**, **ID**, **length**, **fragmentation**), которые были опущены в приведенном 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** выводится также некоторые атрибуты прочитанного файла (тип **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 идентифицируют явно операции RPC. Программа 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[48] 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** передает пакет **RX** хосту **pike**. Этот пакет адресован файловому серверу (**fs**) и начинает вызов удаленной 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 успешном изменении имени файла.

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

Формат вывода должен быть достаточно 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. Вы можете задать размер кадра захвата с помощью опции **-s** для обеспечения более читаемых результатов (например, **-s 256**).

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

KIP AppleTalk (DDP in UDP)

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

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

В приведенном примере первые две строки содержат имена сетей AppleTalk, а третья строка – имя хоста[49]. Для разделения номера и имени в файле могут использоваться 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[50] сети **icsd-net**, прослушивающим порт NBP

[48] Andrew File System

[49] Хост отличается от сети наличием в номере третьего октета.

[50] Отметим, что широковещательный адрес 255 показывается просто именем или номером сети без указания хоста. По этой причине разумно сохранять имена хостов и сетей в файле **/etc/atalk.names** по-отдельности.

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.

Пакеты NBP выводятся в формате, подобном приведенному 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 учета заголовка ATP. Символ ***** для пакета 7 показывает наличие флага **EOM**.

Хост **jssmag.209** после получения пакетов запрашивает повторную передачу пакетов 3 и 5 и **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.

Фрагментация IP

Фрагментированные дейтаграммы IP выводятся в формате:

```
(frag id:size@offset+)
(frag id:size@offset)
```

Знак + в первой строке показывает наличие дополнительных фрагментов.

Поле **id** показывает идентификатор фрагмента, **size** – его размер в байтах без учета заголовка IP, а **offset** – смещение (в байтах) фрагмента в исходной дейтаграмме.

Информация выводится для каждого фрагмента. Первый фрагмент выводится с заголовком протокола вышележащего уровня и сведениями о фрагментации. Все последующие фрагменты не включают заголовка протокола вышележащего уровня и сведения о фрагменте выводятся сразу после адресов отправителя и получателя. Ниже приведен пример пакетов связанных с передачей файла по протоколу FTP с сайта arizona.edu на хост **rtsg** через сеть CSNET, которая не поддерживает передачу дейтаграмм размером 576 байтов.

```
arizona.ftp-data > rtsg.1170: . 1024:1332(308) ack 1 win 4096 (frag 595a:328@0+)
arizona > rtsg: (frag 595a:204@328)
rtsg.1170 > arizona.ftp-data: . ack 1536 win 2560
```

Отметим, что адреса во второй строке приведены без номеров портов, поскольку фрагменты (за исключением первого) не содержат заголовков TCP, что не позволяет получить сведений о номера портов и порядковых номерах TCP. Порядковые номера в первой строке показывают наличие в пакет 308 байтов пользовательских данных, хотя фактически в дейтаграмме содержится 512 байтов (308 байтов в первом фрагменте и 204 – во втором).

Пакеты с флагом запрета фрагментирования (don't fragment) помечаются символами **DF**.

Временные метки

По умолчанию каждая строка вывода включает временную метку, содержащую время захвата кадра в формате:

```
hh:mm:ss.frac
```

Точность отображения времени зависит от точности системного таймера. Временная метка фиксирует момент, когда кадр становится доступным ядру системы. Время между получением кадра из среды передачи и его доставкой ядру не принимается во внимание.