3-Е ИЗДАНИЕ

АНАЛИЗ ПАКЕТОВ

Практическое руководство по использованию Wireshark и tcpdump для решения реальных проблем в локальных сетях

Крис Сандерс

no starch press

АНАЛИЗ ПАКЕТОВ

PRACTICAL PACKET ANALYSIS

3RD EDITION

Using Wireshark to Solve Real-World Network Problems

by Chris Sanders

no starch press

San Francisco

АНАЛИЗ ПАКЕТОВ

3-Е ИЗДАНИЕ

Практическое руководство по использованию Wireshark и tcpdump для решения реальных проблем в локальных сетях

Крис Сандерс

Диалектика

Москва • Санкт-Петербург
2019

ББК 32.973.26-018.2.75
С18
УДК 681.3.07

ООО "Диалектика"
Зав. редакцией *С.Н. Тригуб*
Перевод с английского и редакция *И.В. Берштейна*

По общим вопросам обращайтесь в издательство "Диалектика" по адресу:
info@dialektika.com, http://www.dialektika.com

Сандерс, Крис.

С18 Анализ пакетов: практическое руководство по использованию Wireshark и tcpdump для решения реальных проблем в локальных сетях, 3-е изд. : Пер. с англ. – СПб. : ООО "Диалектика", 2019 – 448 с. : ил. – Парал. тит. англ.

ISBN 978-5-6040723-0-1 (рус.)

ББК 32.973.26-018.2.75

Научно-популярное издание

Крис Сандерс

Анализ пакетов

практическое руководство по использованию Wireshark и tcpdump для решения реальных проблем в локальных сетях 3-е издание

Подписано в печать 28.11.2018. Формат 70x100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 36,1. Уч.-изд. л. 28,1.
Тираж 400 экз. Заказ № 16503.

Отпечатано в АО "Первая Образцовая типография"
Филиал "Чеховский Печатный Двор"
142300, Московская область, г. Чехов, ул. Полиграфистов, д. 1
Сайт: www.chpd.ru, E-mail: sales@chpd.ru, тел. 8 (499) 270-73-59

ООО "Диалектика", 195027, Санкт-Петербург, Магнитогорская ул., д. 30, лит. А, пом. 848

ISBN 978-5-6040723-0-1 (рус.)
ISBN 978-1-59327-802-1 (англ.)

ОГЛАВЛЕНИЕ

СОДЕРЖАНИЕ

ОТЗЫВЫ О КНИГЕ

"Книга содержит море информации. Написана толково и удобна для чтения. Откровенно говоря, мне было очень интересно читать об анализе пакетов".

из интернет-издания *TechRepublic*.

"Эта книга рекомендуется начинающим сетевым аналитикам, разработчикам программного обеспечения, вновь испеченным инженерам-системотехникам, специалистам в области систем информационной безопасности и всем, кому приходится устранять неполадки в сетях и их защите, закатав рукава".

Гюнтер Оллманн (Gunter Ollmann), бывший главный технический директор компании IOActive.

"Когда мне придется исследовать медленно работающую сеть, я непременно обращусь к этой книге. И это, вероятно, наилучшая похвала для любой технической литературы".

Майкл У. Лукас (Michael W. Lucas), автор книг *Absolute FreeBSD* и *Network Flow Analysis*.

"Необходимая книга для тех, кто отвечает за администрирование сетей на любом уровне".

из *Linux Pro Magazine*.

"Замечательное, удобное и грамотно составленное руководство".

из `ArsGeek.com`.

"Самая лучшая отправная точка для тех, кому требуется тщательно изучить основы анализа пакетов".

из интернет-издания `StateofSecurity.com`.

"Слово *практика* весьма информативно и уместно в названии этой книги. В ней читатели смогут узнать немало об анализе пакетов и сразу же перейти к реальным примерам работы с Wireshark".

из интернет-издания `LinuxSecurity.com`.

"Чтобы выяснить, сообщаются ли друг с другом какие-нибудь неизвестные сетевые узлы или ваш компьютер с чужими, вам потребуется анализатор сетевых пакетов. И для этой цели лучше всего подойдет инструментальное средство Wireshark, а эта книга как нельзя лучше поможет изучить его".

из *Free Software Magazine*.

"Книга идеально подходит для читателей от начинающего до промежуточного уровня".

из *Daemon News*.

"О, благодать, спасен тобой
Я из пучины бед;
Был мертв и чудом стал живой,
Был слеп и вижу свет".
"О, благодать", христианский гимн Джона Ньютона.

БЛАГОДАРНОСТИ

Мне бы хотелось выразить искреннюю признательность всем, кто поддерживал меня в работе над этой книгой.

Благодарю свою жену Эллен за безграничную любовь и способность мириться с тем, что я ночи напролет проводил за клавиатурой, когда она пыталась заснуть в своей постели.

Благодарю свою мать, которая даже после смерти служит мне примером доброты, который продолжает вдохновлять меня. Благодарю также своего отца, научившего меня, что значит тяжкий труд, без которого мне не удалось бы ничего добиться.

Благодарю Джейсона Смита (Jason Smith), который мне как брат. Мне трудно выразить свою признательность ему за его постоянное внимание ко мне.

Что же касается моих бывших и нынешних сотрудников, то мне очень повезло работать в окружении людей, благодаря которым я стал лучше и сообразительнее. Мне трудно перечислить всех, но я искренне признателен Дастину, Алеку, Мартину, Патрику, Крису, Майку и Грейди за ежедневную поддержку и усвоение того, что значит лидерство служителя.

Благодарю Тайлера Регули (Tyler Reguly) за техническое редактирование книги и исправление моих нелепых ошибок. Благодарю также Дэвида Вона (David Vaughan) – за особую опеку, Джеффа Каррелла (Jeff Carrell) – за оказанную помощью в редактировании содержимого посвященному сетевому протоколу IPv6, Бреда Дункана (Brad Duncan) – за предоставление файла перехвата, использованного мной в главе, посвященной безопасности, а также

группу специалистов из компании QA Café — за предоставленную лицензию на инструментальное средство Cloudshark, с помощью которого мне удалось организовать перехват пакетов для демонстрации в этой книге.

И, конечно, мне хотелось бы выразить свою признательность Джеральду Комбсу (Gerald Combs) и команде разработчиков Wireshark. Ведь Wireshark стала столь замечательной платформой благодаря преданности Джеральда и многих других разработчиков. Если бы не их усилия, положение дел в области информационных технологий и безопасности сетей оказалось бы намного хуже.

И, наконец, благодарю Билла, Серену, Яна, Аманду, Элисон и остальных сотрудников издательства No Starch Press за проявленное усердие в редактировании и выпуске всех трех изданий этой книги.

ВВЕДЕНИЕ

Настоящее, третьей издание этой книги было написано и отредактировано в течение полутора лет, т.е. от конца 2015 года и до начала 2017 года, приблизительно через шесть лет после выхода в свет второго издания и десять лет после первого издания. В настоящее издание вошло немало нового, включая совершенно новые файлы и сценарии перехвата, а также отдельную новую главу, посвященную анализу пакетов из командной строки с помощью утилит TShark и `tcpdump`. Если вам понравились два предыдущих издания, то понравится и третье. Оно написано в том же духе с пояснениями простым и доходчивым языком. А если вы сомневались в выборе двух предыдущих изданий, поскольку они не содержали самые последние сведения об организации сетей и обновлениях Wireshark, в таком случае вам придется прочитать настоящее издание, которое было дополнено описанием новых сетевых протоколов и обновленной информацией по версии Wireshark 2.*x*.

Зачем нужна эта книга

У читателя может возникнуть вполне резонный вопрос: чем эта книга отличается от других книг на тему анализа пакетов? Ответ заключается в названии книги *Практика анализа пакетов*. Следует откровенно признать, что ничто не может сравниться с практическим опытом, и точнее всего этот опыт удается передать в книге на практических примерах и реальных сценариях.

В первой половине книги даются знания, необходимые для уяснения особенностей анализа пакетов и работы с Wireshark. А вторая ее половина полностью посвящена разбору случаев из повседневной практики управления сетями.

Усвоив методики анализа пакетов, описанные в этой книге, читатель извлечет для себя немалую пользу, кем бы он ни работал: сетевым техником, сетевым администратором, руководителем информационной службы, специалистом по настольным системам или даже аналитиком по вопросам безопасности сетей.

Основные понятия и принятый подход

Вообще-то, я неторопливый и когда поясняю какое-нибудь понятие, то стараюсь делать это непринужденно. Это же относится и к языку, которым изложен материал книги. Чтобы не увязнуть в техническом жаргоне, я попытался изложить описываемые в ней понятия как можно более простым и обыденным языком. Все термины и понятия определяются в ней без каких-либо несущественных подробностей. Ведь я родом из замечательного штата Кентукки, а следовательно, стараюсь быть немногословным. (Но вы, читатель, простите мне излишнюю многословность, которой я иногда грешу в книге.)

Первые несколько глав существенны для понимания остальной части книги, поэтому непременно усвойте описываемые в них понятия. А вторая половина книги всецело посвящена разбору практических примеров. Возможно, они и не вполне отражают те ситуации, которые возникают в вашей практической деятельности, тем не менее вы сможете применить рассматриваемые в них понятия в возникающих у вас ситуациях.

Ниже приведено краткое описание содержания этой книги.

Глава 1, "Анализ пакетов и основы организации сетей". В этой главе поясняется, что такое анализ пакетов, как он действует и как проводится. Здесь также даются основы передачи данных по сети и анализа пакетов.

Глава 2, "Подключение к сети". В этой главе рассматриваются различные методики подключения анализатора пакетов к сети.

Глава 3, "Введение в Wireshark". В этой главе даются основные сведения о приложении Wireshark. В частности, где его взять, как им пользоваться, что оно позволяет делать, чем оно примечательно и прочие полезные сведения о нем. В настоящее издание включено обсуждение вопросов специальной настройки Wireshark с помощью профилей конфигурации.

Глава 4, "Обработка перехваченных пакетов". Установив и настроив Wireshark, вы, вероятно, захотите узнать, как взаимодействовать с перехваченными пакетами. В этой главе даются основы обработки перехваченных

пакетов, включая новые разделы с более подробным описанием особенностей отслеживания потоков пакетов и преобразования имен.

Глава 5, "Дополнительные возможности Wireshark". Научившись ползать, пора учиться летать. В этой главе подробно рассматриваются дополнительные возможности Wireshark, а также демонстрируется внутренний механизм действия некоторых менее очевидных операций. В эту главу включены новые разделы с подробным описанием особенностей отслеживания потоков пакетов и преобразования имен.

Глава 6, "Анализ пакетов из командной строки". Wireshark — замечательное средство, но иногда требуется оставить удобства графического пользовательского интерфейса, чтобы взаимодействовать с пакетами в режиме командной строки. В этой новой главе показывается, как пользоваться TShark и `tcpdump` — двумя самыми лучшими инструментальными средствами командной строки для анализа пакетов.

Глава 7, "Протоколы сетевого уровня". В этой главе демонстрируется, как обычная передача данных, выполняемая на сетевом уровне, выглядит на уровне пакетов, исследуемых в сетевых протоколах ARP, IPv4, IPv6 и ICMP. Чтобы устранять неполадки в этих сетевых протоколах на практике, следует сначала понять, каким образом они действуют.

Глава 8, "Протоколы транспортного уровня". В этой главе рассматриваются два наиболее распространенных сетевых протокола транспортного уровня: TCP и UDP. Оба эти протокола служат для передачи большинства рассматриваемых здесь пакетов, и поэтому очень важно уяснить, как они выглядят на уровне пакетов и чем они отличаются.

Глава 9, "Распространенные протоколы верхнего уровня". В этой главе продолжается рассмотрение сетевых протоколов и демонстрируется, как выглядят четыре наиболее распространенных протокола верхнего уровня: HTTP, DNS, DHCP и SMTP.

Глава 10, "Основные реальные сценарии". В этой главе представлен анализ некоторых примеров типичного сетевого трафика в первом ряде реальных сценариев. Каждый сценарий представлен в удобном для исследования формате, позволяющем выявить затруднение, возникшее в сети, проанализировать его и найти решение. В этих основных сценариях задействовано совсем немного компьютеров и проводится ограниченный анализ, но и этого должно быть достаточно, чтобы ввести вас в курс дела.

Глава 11, "Меры борьбы с медленной сетью". Наиболее характерные осложнения, с которыми приходится иметь дело сетевым техникам, обычно связаны с низкой скоростью работы сети. Эта глава посвящена решению подобного рода осложнений.

Глава 12, "Анализ пакетов на безопасность". Безопасность сетей является самым насущным предметом в области информационных технологий. В этой главе представлены некоторые сценарии, связанные с решением вопросов, имеющих отношение к безопасности, по методикам анализа пакетов.

Глава 13, "Анализ пакетов в беспроводных сетях". В этой главе даются основы анализа пакетов в беспроводных сетях. В ней обсуждается отличия анализа пакетов в беспроводных сетях от проводных, включая некоторые примеры трафика в беспроводных сетях.

Приложение А, "Дополнительная информация". В этом приложении представлены дополнительные инструментальные средства и ресурсы Интернета, которые могут оказаться полезными для дальнейшего применения методик анализа пакетов, усвоенных в этой книге.

Приложение Б, "Интерпретация пакетов". Если требуется подробнее изучить особенности интерпретации отдельных пакетов, то в этом приложении дается краткий обзор хранения пакетной информации в двоичном виде и ее преобразования в шестнадцатеричное представление. В этом приложении показывается также, как разбирать вручную пакеты, представленные в шестнадцатеричном виде, с помощью схем пакетов. Это удобно в том случае, если приходится уделять немало времени анализу специальных протоколов или пользоваться инструментальными средствами анализа пакетов в режиме командной строки.

Как пользоваться этой книгой

- **Как учебным пособием.** В таком случае читайте ее последовательно главу за главой, уделяя особое внимание реальным примерам в последующих главах, чтобы постепенно усвоить порядок анализа пакетов.
- **Как справочным пособием.** Некоторыми средствами Wireshark вам придется пользоваться нечасто, и поэтому вы можете забыть, каким образом они действуют. Поэтому иметь эту книгу под рукой очень удобно, чтобы освежить в памяти порядок применения отдельных средств Wireshark. Когда вы проводите анализ пакетов на своей работе, у вас может возникнуть потребность обратиться за справкой к диаграммам, схемам и методикам, представленным в этой книге.

О примерах файлов перехвата

Все файлы перехвата, употребляемые в примерах из этой книги, доступны на веб-странице издательства No Starch Press, посвященной этой книге, по адресу `https://www.nostarch.com/packetanalysis3/`. Чтобы извлечь

наибольшую пользу из книги, загрузите эти файлы и пользуйтесь ими, прорабатывая приведенные в ней практические примеры.

Фонд поддержки технологий в сельской местности

Во введении к этой книге нельзя не упомянуть о самом главном, что дало ее издание. Вскоре после выхода в свет первого издания я основал общественную организацию под названием "Фонд поддержки технологий в сельской местности" (Rural Technology Fund – RTF).

Современные информационные технологии обычно доступны учащимся в сельской местности – даже с отличной успеваемостью – в меньшей степени, чем в крупных городах. Основание "Фонда поддержки технологий в сельской местности" в 2008 году стало верхом всех моих мечтаний. Тем самым я пытался восполнить цифровой пробел между жителями сельской и городской местности. С этой целью в "Фонде поддержки технологий в сельской местности" действуют специальные учебные программы, мероприятия по привлечению жителей сельской местности, сбору пожертвований на ресурсы по обучению цифровым технологиям, проводятся учебные курсы и пропагандируются современные технологии в сельских и бедных районах.

В 2016 году "Фонд поддержки технологий в сельской местности" передал ресурсы по обучению цифровым технологиям в руки более 10 тыс. учащихся в сельских и бедных районах США. И мне приятно объявить, что весь свой авторский гонорар я направил на эти цели в Фонд. Если вы желаете узнать больше об этой общественной организации или принять посильное участие в ее деятельности, посетите ее веб-сайт по адресу `http://www.ruraltechfund.org/` или найдите ее в Twitter по адресу `@RuralTechFund`.

Как связаться с автором книги

Мне всегда любопытно знать, что читатели думают о моей книге. Если вы желаете связаться со мной по какой-нибудь причине, направляйте все свои вопросы, комментарии, замечания и предложения прямо мне по адресу `chris@chrissander.org`. Кроме того, я регулярно веду свой блог по адресу `http://www.chrissanders.org/`, и меня можно найти в Twitter по адресу `@chrissanders88`.

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам

интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо, либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг.

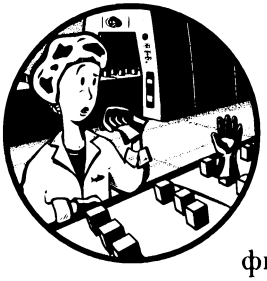
Наши электронные адреса:

E-mail: info@williamspublishing.com

WWW: http://www.williamspublishing.com

1

АНАЛИЗ ПАКЕТОВ И ОСНОВЫ ОРГАНИЗАЦИИ СЕТЕЙ



Каждый день в вычислительной сети может произойти миллион самых разных событий: от простого заражения шпионской программой до сложной ошибки конфигурирования маршрутизатора. И самое лучшее, на что можно надеяться, — быть готовым во всеоружии знаний и инструментальных средств отреагировать на осложнения подобного рода.

Чтобы действительно разобраться в затруднениях, возникающих в сети, необходимо перейти на уровень пакетов. Все затруднения в сети начинаются именно на этом уровне, где могут обнаружиться скверные реализации даже самых опрятно выглядящих приложений, а заслуживающие, на первый взгляд, полного доверия сетевые протоколы оказаться зловредными. Здесь от нас ничего не скроется, поскольку на этом уровне нет ни вводящей в заблуждение структуры меню, ни привлекательной графики, ни неблагонадежных работников и вообще никакой секретной информации, кроме зашифрованной. И чем больше нам удается сделать на уровне пакетов, тем лучше мы можем контролировать свою сеть и разрешать возникающие в ней затруднения. Это и есть область действия анализа пакетов.

Рассмотрению именно этой области и посвящена данная книга. Из реальных сценариев вам предстоит узнать, как бороться с медленной передачей данных по сети, выявлять узкие места в приложениях и даже отслеживать

хакеров. Прочитав эту книгу, вы сможете реализовать методики анализа пакетов, которые помогут вам разрешать даже самые сложные затруднения, возникающие в вашей сети.

В этой главе представлены самые основы с акцентом на передачу данных по сети. Материал этой главы поможет вам получить в свое распоряжение инструментальные средства, которые потребуются для дальнейшего изучения различных сценариев из реальной эксплуатации сетей.

Анализ пакетов и их анализаторы

Анализ пакетов, иногда еще называемый *анализом протоколов*, описывает процесс перехвата и интерпретации действующих данных по мере их продвижения по сети, чтобы лучше понять, что же в ней происходит. Как правило, анализ пакетов проводится *анализатором пакетов* — инструментальным средством, применяемым для перехвата первичных данных, передаваемых по проводам сети.

Анализ пакетов может оказать помощь в следующем.

- Уяснить характеристики сети.
- Выяснить, кто находится в сети.
- Определить, кто или что "съедает" доступную пропускную способность сети.
- Выявить моменты, когда использование сети достигает своего пика.
- Выявить зловредную деятельность в сети.
- Обнаружить небезопасные и громоздкие приложения.

Для анализа пакетов имеются различные программы: как бесплатные, так и коммерческие. Каждая такая программа предназначена для определенных целей. К числу самых распространенных программ анализа пакетов относятся `tcpdump`, OmniPeek и Wireshark (именно последней и уделяется основное внимание в этой книге). Программы OmniPeek и Wireshark снабжены графическим пользовательским интерфейсом, тогда как `tcpdump` является утилитой командной строки.

Оценка анализатора пакетов

Выбирая анализатор пакетов, необходимо принять во внимание целый ряд факторов, включая следующие.

- **Поддержка сетевых протоколов.** Все анализаторы пакетов способны интерпретировать различные протоколы, а большинство из них — наиболее распространенные сетевые протоколы (например, IPv4 и ICMP),

транспортные протоколы (например, TCP и UDP) и даже протоколы уровня приложений (например, DNS и HTTP). Хотя они могут и не поддерживать нетрадиционные и более сложные протоколы (например, IPv6, SMBv2 и SIP). Поэтому, выбирая анализатор пакетов, убедитесь, что в нем поддерживаются применяемые вами протоколы.

- **Удобство использования.** Обращайте особое внимание на интерфейс анализатора пакетов, простоту его установки и общую последовательность операций. Выбранная вами программа должна соответствовать уровню вашей квалификации. Так, если у вас имеется весьма скромный опыт анализа пакетов, вам вряд ли стоит выбирать такие сложные анализаторы пакетов, действующие в режиме командной строки, как утилита `tcpdump`. А если вы имеете немалый опыт анализа пакетов, то вам подойдет и более развитая, хотя и сложная программа. По мере приобретения необходимого опыта вы можете даже найти полезным сочетать в отдельных случаях несколько программ анализа пакетов.
- **Стоимость.** Самое замечательное, что многие анализаторы пакетов бесплатны и практически ничем не уступают их коммерческим аналогам. А самое главное отличие бесплатных анализаторов пакетов от коммерческих заключается в их механизмах отчетности. В состав коммерческих продуктов, как правило, входит специальный модуль формирования отчетов, тогда как в бесплатных приложениях такие средства отсутствуют, и поэтому они обеспечивают лишь ограниченную отчетность.
- **Поддержка программ.** Даже если вы овладели основами работы с программой анализа пакетов, вам иногда потребуется дополнительная поддержка для решения новых задач по мере их появления. Оценивая имеющуюся поддержку программ, обращайте внимание на документацию для разработчиков, публичные форумы и списки рассылки пользователей. И несмотря на возможную нехватку формализованной коммерческой поддержки бесплатных программ анализа пакетов вроде Wireshark, сообщество пользователей и участников их разработки нередко ведет активные дискуссионные клубы, вики-страницы и блоги, чтобы помочь извлечь наибольшую пользу из выбранного вами анализатора пакетов.
- **Доступ к исходному коду.** Некоторые анализаторы пакетов относятся к программному обеспечению с открытым исходным кодом. Это дает возможность просматривать исходный код программы, а иногда и вносить в него необходимые коррективы. Если у вас особый или сложный случай для применения анализатора пакетов, то такая возможность окажется для вас весьма привлекательной. Исходный код большинства коммерческих анализаторов пакетов недоступен.

- **Поддержка операционной системы.** К сожалению, не во всех анализаторах пакетов поддерживается каждая операционная система. Если вы работаете консультантом, у вас может возникнуть потребность перехватывать и анализировать пакеты в самых разных операционных системах. Следовательно, вам потребуется инструментальное средство, работающее в большинстве операционных систем. Следует также иметь в виду, что пакеты иногда придется перехватывать на одной машине, а просматривать на другой. Отличия в операционных системах могут вынудить вас пользоваться разными приложениями на отдельных машинах.

Принцип действия анализаторов пакетов

В процессе анализа пакетов задействованы как программные, так и аппаратные средства. Этот процесс делится на следующие стадии.

1. **Сбор данных.** Прежде всего анализатор пакетов собирает первичные двоичные данные из сети. Как правило, это делается переключением избранного сетевого интерфейса в *смешанный режим (promiscuous mode)*. В этом режиме сетевая плата может принимать весь трафик в сегменте сети, а не только адресуемый ей трафик.
2. **Преобразование.** Далее перехваченные двоичные данные преобразуются в удобочитаемую форму. На это способно большинство развитых анализаторов пакетов, работающих в режиме командной строки. На этой стадии сетевые данные могут автоматически интерпретироваться только на самом элементарном уровне, оставляя бо́льшую часть анализа конечному пользователю вручную.
3. **Анализ.** Наконец, анализатор пакетов проводит анализ перехваченных и преобразованных данных. В частности, он проверяет протокол перехваченных в сети данных, исходя из извлекаемой информации, и далее начинает анализ характерных особенностей этого протокола.

Установление связи между компьютерами

Чтобы полностью уяснить анализ пакетов, необходимо точно знать, каким образом устанавливается связь между компьютерами. В этом разделе мы рассмотрим основные положения о сетевых протоколах, модель OSI (Open Systems Interconnections – взаимодействие открытых систем), сетевые фреймы данных и аппаратную поддержку всего этого.

Сетевые протоколы

Современные сети состоят из разных систем, работающих на различных платформах. Для сообщения между этими системами в качестве общепонятного языка служит ряд сетевых *протоколов*. К числу самых распространенных относятся TCP (Transmission Control Protocol – протокол управления передачей), IP (Internet Protocol – межсетевой протокол), ARP (Address Resolution Protocol – протокол преобразования адресов), а также DHCP (Dynamic Host Configuration Protocol – протокол динамического конфигурирования хоста). Все эти протоколы логически сгруппированы для совместной работы в так называемый *стек протоколов*.

Проколы удобно сравнить с правилами, регулирующими употребление естественного языка. В каждом языке имеются определенные правила, например, порядок спряжения глаголов, приветствия людей и даже надлежащей благодарности кого-нибудь. Протоколы действуют сходным образом, давая возможность определить порядок маршрутизации пакетов, установления сетевого соединения и подтверждения приема данных.

Протокол может быть как очень простым, так и крайне сложным в зависимости от его функций. И хотя различные протоколы могут существенно отличаться, многие сетевые протоколы призваны решать следующие вопросы.

- **Установление соединения.** Кто инициирует установление соединения: клиент или сервер? Какой информацией следует обменяться, прежде чем устанавливать соединение?
- **Согласование характеристик соединения.** Шифруется ли передача данных по сетевому протоколу? Каким образом происходит обмен ключами шифрования между связывающимися хостами (т.е. сетевыми узлами)?
- **Форматирование данных.** Каким образом организованы данные, содержащиеся в пакете? В каком порядке данные обрабатываются принимающим их устройством?
- **Обнаружение и исправление ошибок.** Что происходит в том случае, если пакет слишком долго достигает места своего назначения? Каким образом клиент возобновляет свою работу, если он не в состоянии установить соединение с сервером в кратчайший период времени?
- **Разрыв соединения.** Каким образом один хост дает знать другому, что связь окончена? Какую завершающую информацию следует передать, чтобы корректно разорвать соединение?

Семиуровневая модель OSI

Сетевые протоколы подразделяются по своим функциям на основании принятой в данной отрасли стандартной модели OSI. Эта иерархическая модель состоит из семи уровней и очень удобна для понимания особенностей связи и передачи данных по сети. На рис. 1.1, *справа*, показаны уровни модели OSI, а соответствующая терминология для обозначения данных на каждом из этих уровней — *слева*. На самом верху данной модели находится уровень приложений, представляющий программы, предназначенные для доступа к сетевым ресурсам, а в самом низу — физический уровень, где, по существу, данные перемещаются по сети. Протоколы на каждом уровне действуют совместно, обеспечивая надлежащую обработку данных на уровнях, расположенных непосредственно выше и ниже.

Рис. 1.1. Иерархическое представление семи уровней модели OSI

Каждый уровень модели OSI выполняет конкретную функцию, как поясняется ниже.

- **Уровень приложений (седьмой).** На этом самом верхнем уровне модели OSI предоставляются средства для доступа пользователей к сетевым ресурсам. Как правило, это единственный уровень, доступный конечным пользователям, поскольку на нем предоставляется интерфейс, на основании которого они осуществляют всю свою деятельность в сети.

ПРИМЕЧАНИЕ *Первоначально модель OSI была обнародована в 1983 году Международной организацией по стандартизации (ISO) в виде документа под названием ISO 7498. Модель OSI является всего лишь рекомендуемым в данной отрасли стандартом. Это означает, разработчики сетевых протоколов не должны строго придерживаться данной модели. На самом деле модель OSI не является единственной для организации сетей. Например, некоторые разработчики отдают предпочтение модели DoD, т.е. модели Министерства обороны США, иначе называемой моделью TCP/IP.*

- **Уровень представления данных (шестой).** На этом уровне получаемые данные преобразуются в формат, удобный для их чтения на уровне приложений. Порядок кодирования и декодирования данных на этом уровне зависит от протокола, применяемого на уровне приложений для передачи и приема данных. На уровне приложений может также использоваться несколько форм шифрования и дешифрования данных для их защиты.
- **Сеансовый уровень (пятый).** На этом уровне происходит *диалог*, или *сеанс связи*, между двумя компьютерами. Сеансовый уровень отвечает также за установление дуплексного (т.е. двунаправленного) или полудуплексного (т.е. однонаправленного) соединения, а также для корректного (т.е. не резкого и внезапного) разрыва связи между двумя хостами (т.е. сетевыми узлами).
- **Транспортный уровень (четвертый).** Основное назначение транспортного уровня — предоставить надежные транспортные услуги нижележащим уровням. Благодаря управлению потоком данных, их сегментации и десегментации, исправлению ошибок на транспортном уровне обеспечивается безошибочная доставка данных из одной точки сети в другую. Обеспечить надежную доставку данных крайне сложно, поэтому в модели OSI для этой цели выделен отдельный уровень. На транспортном уровне используются протоколы как с установлением соединения, так и без него. Именно на этом уровне и действуют определенные брандмауэры и промежуточные, так называемые прокси-серверы.
- **Сетевой уровень (третий).** Один из самых сложных уровней модели OSI, обеспечивающий маршрутизацию данных между физическими сетями и правильную адресацию сетевых узлов (например, по IP-адресу). На этом уровне происходит также разбиение потоков данных на более мелкие части, а иногда и обнаружение ошибок. Именно на этом уровне и действуют маршрутизаторы.
- **Канальный уровень (второй).** На этом уровне предоставляются средства для переноса данных по физической сети. Основное назначение данного уровня — предоставить схему адресации для обозначения

физических устройств (например, MAC-адреса). Именно на этом уровне и действуют такие физические устройства, как мосты и коммутаторы.

- **Физический уровень (первый).** Это самый нижний уровень модели OSI, где находится среда, по которой переносятся сетевые данные. На этом уровне определяется физические и электрические характеристики всего сетевого оборудования, включая уровни напряжений в сети, концентраторы, сетевые адаптеры, повторители и кабельную разводку. На физическом уровне устанавливаются и разрываются сетевые соединения, предоставляются средства для совместного использования общих сетевых ресурсов и преобразования сигналов из цифровой в аналоговую форму, и наоборот.

ПРИМЕЧАНИЕ *Для удобства запоминания уровней модели OSI на английском языке служит мнемоническая фраза "Please Do Not Throw Sausage Pizza Away" (Пожалуйста, не выбрасывайте пиццу с колбасой), где прописные буквы отдельных слов обозначают названия каждого уровня данной модели, начиная с первого*[1].

В табл. 1.1 перечислены некоторые из наиболее распространенных протоколов, употребляемых на каждом уровне модели OSI.

Таблица 1.1. Типичные протоколы, используемые на каждом уровне модели OSI

Уровень	Протоколы
Приложений	HTTP, SMTP, FTP, Telnet
Представления	ASCII, MPEG, JPEG, MIDI
Сеансовый	NetBIOS, SAP, SDP, NWLink
Транспортный	TCP, UDP, SPX
Сетевой	IP, IPX
Канальный	Ethernet, Token Ring, FDDI, AppleTalk
Физический	Проводной или беспроводный

Несмотря на то что модель OSI является всего лишь рекомендуемым стандартом, ее следует знать наизусть, поскольку она предоставляет удобный словарь терминов для осмысления и описания затруднений, возникающих в сетях. По мере чтения книги вы обнаружите, что вопросы маршрутизации относятся к третьему уровню данной модели, а вопросы программирования – к седьмому.

[1] В качестве варианта на русском языке предлагается мнемоническая фраза "Федя, Как Стать Тебе Самым Первым Парнем", естественно, на деревне. – *Примеч. ред.*

ПРИМЕЧАНИЕ *Однажды коллега рассказал мне о жалобе одного пользователя на невозможность доступа к сетевому ресурсу. Это затруднение возникло из-за того, что пользователь ввел неверный пароль. Мой коллега отнес данное затруднение к восьмому, неофициальному пользовательскому уровню модели OSI. И такое обозначение часто употребляется в среде тех, кто работает на уровне пакетов.*

Прохождение данных по модели OSI

Исходно передача данных по сети начинается на уровне приложений передающей системы. Данные проходят сверху вниз по всем уровням модели OSI до тех пор, пока не достигнут физического уровня, где находится точка, откуда данные отправляются из передающей системы в принимающую. А принимающая система получает данные на своем физическом уровне, откуда данные проходят снизу вверх по всем уровням модели OSI, достигая в конечном итоге уровня приложений.

Каждый уровень модели OSI может взаимодействовать только с уровнями, расположенными непосредственно выше и ниже его. Например, на уровне 2 данные можно передавать и принимать только с уровней 1 и 3.

Ни одна из услуг, предоставляемых различными протоколами на любом заданном уровне модели OSI, не должна быть избыточной. Так, если на одном уровне протокол предоставляет конкретную услугу, то ни один другой протокол на любом другом уровне не должен предоставлять такую же самую услугу. У протоколов на разных уровнях могут быть средства, служащие сходным целям, но их функционирование будет хотя бы немного, но все равно отличаться.

Протоколы на соответствующих уровнях передающего и приемного устройств дополняют друг друга. Так, если протокол на седьмом уровне передающего устройства отвечает за форматирование передаваемых данных, то соответствующий протокол на седьмом уровне приемного устройства должен отвечать за чтение полученных отформатированных данных.

На рис. 1.2 показано графическое представление модели OSI на двух связанных вместе устройствах. Как видите, передача данных происходит сверху вниз на одном устройстве, а прием — в обратном порядке на другом устройстве.

Инкапсуляция данных

Протоколы, действующие на разных уровнях модели OSI, обмениваются данными посредством *инкапсуляции данных*. Каждый уровень данной модели отвечает за добавление в начало и конец передаваемых данных соответствующего заголовка и концевика в виде дополнительных битов информации, предназначенных для взаимодействия между уровнями. Когда, например, данные получаются на транспортном уровне из сеансового уровня, на транспортном

уровне добавляется свой заголовок, содержащий требуемую информацию, в эти данные, перед передачей их дальше на сетевой уровень.

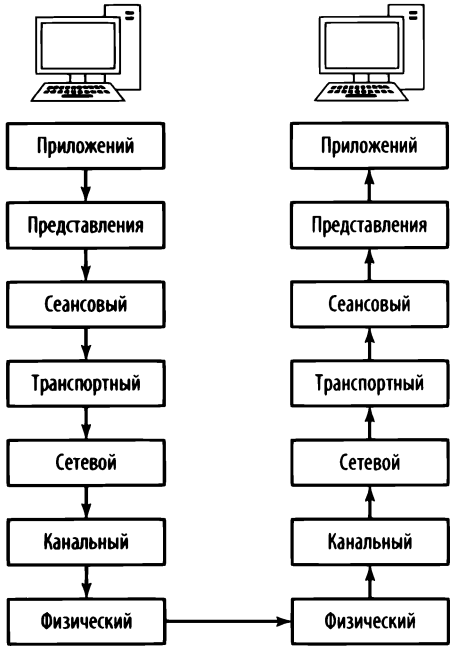


Рис. 1.2. Протоколы, действующие на одном и том же уровне как в передающей, так и в принимающей системе

В процессе инкапсуляции создается блок протокольных данных (PDU – protocol data unit), включающий в себя передаваемые данные и всю добавленную заголовочную и концевую информацию. По мере продвижения данных сверху вниз по модели OSI и добавления в нее заголовочной и концевой информации на разных уровнях блок PDU изменяется, постепенно разрастаясь. В окончательном виде блок PDU формируется, когда он достигает физического уровня, где он посылается приемному устройству по месту назначения. А на приемном устройстве заголовки и концевики извлекаются соответствующими протоколами из блока PDU в обратном их вводу порядке по мере прохождения данных снизу вверх по уровням модели OSI. И как только блок PDU достигнет верхнего уровня модели OSI, останутся лишь данные из исходного уровня приложений.

Чтобы продемонстрировать принцип действия инкапсуляции данных, рассмотрим упрощенный практический пример создания, передачи и приема пакета по отношению к модели OSI. Однако специалисты по анализу пакетов редко упоминают о сеансовом или уровне представлений модели OSI. Именно поэтому оба эти уровня отсутствуют в рассматриваемом здесь примере, как, впрочем, и в остальных примерах из данной книги.

ПРИМЕЧАНИЕ *Для описания данных, упакованных на каждом уровне, в модели OSI употребляются специальные термины. Так, на физическом уровне данные содержатся в виде отдельных битов, на канальном – в виде фреймов, на сетевом – в виде пакетов, на транспортном – в виде сегментов, а на трех верхних уровнях – просто в виде данных. Но на практике такая терминология не находит особого применения, и поэтому далее в книге мы будем пользоваться только термином пакет для обозначения полного или частичного блока PDU, включая заголовочную и концевую информацию из нескольких или всех уровней модели OSI.*

В данном случае предпринимается попытка просмотреть веб-страницу по адресу `http://www.google.com/`. Прежде всего необходимо сформировать пакет запроса, передаваемый из исходного клиентского компьютера на целевой серверный компьютер. В данном случае предполагается, что сеанс связи по сетевому протоколу TCP/IP уже установлен. Процесс инкапсуляции данных в рассматриваемом здесь примере наглядно показан на рис. 1.3.

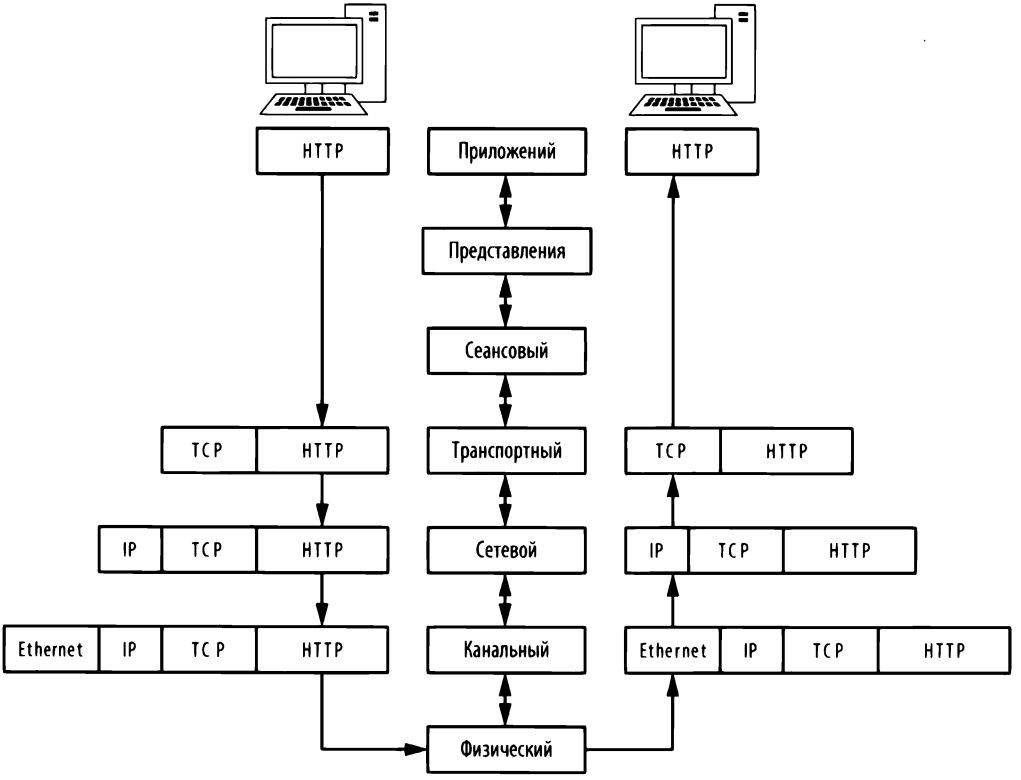


Рис. 1.3. Графическое представление инкапсуляции данных при их обмене между клиентом и сервером

Итак, начнем с клиентского компьютера на уровне приложений. Здесь речь идет о просмотре страницы веб-сайта, и поэтому применяется сетевой протокол HTTP. По этому протоколу выдается команда на загрузку файла `index.html` со страницы по указанному выше адресу.

ПРИМЕЧАНИЕ *На практике браузер запросит сначала у веб-сайта корневой каталог документов, обозначаемый косой чертой (/). Как только сервер получит этот запрос, он переадресует браузер к тому файлу, который сконфигурирован для обслуживания запросов на выдачу корневого каталога документов. Обычно это файл вроде* **`index.html`** *или* **`index.php`**. *Подробнее об этом речь пойдет в главе 9, "Распространенные протоколы верхнего уровня", при обсуждении сетевого протокола HTTP.*

Как только из протокола уровня приложений будет отправлена команда, останется лишь доставить пакет по месту его назначения. Данные из пакета передаются сверху вниз на транспортный уровень. В протоколе HTTP уровня приложений применяется протокол TCP, так как первый располагается в стеке протоколов *над* последним, а по существу, "сидит" на нем. Поэтому TCP служит в качестве протокола транспортного уровня, обеспечивая надежную доставку пакета. На транспортном уровне формируется TCP-заголовок, который добавляется в блок PDU, как показано на рис. 1.3. В этот TCP-заголовок входят порядковые номера и прочие данные, добавляемые к пакету и обеспечивающие надлежащую его доставку.

ПРИМЕЧАНИЕ *Специалисты по сетям нередко говорят, что один сетевой протокол "сидит" или "ездит" на другом протоколе из-за нисходящего характера архитектуры модели OSI. Так, протокол HTTP уровня приложений предоставляет определенные услуги, полагаясь на протокол TCP транспортного уровня для надежного оказания своих услуг. А услуги обоих этих протоколов опираются на протокол IP сетевого уровня для доставки данных по адресу. Таким образом, протокол HTTP "сидит" на протоколе TCP, а тот – на протоколе IP.*

Выполнив свое задание, протокол TCP передает пакет протоколу IP третьего уровня, отвечающему за логическую адресацию пакета. С этой целью в протоколе IP формируется информация о логической адресации, которая добавляется в блок PDU, после чего пакет передается протоколу Ethernet на канальном уровне, где физические адреса Ethernet сохраняются в Ethernet-заголовке. Таким образом, полностью собранный пакет переносится на физический уровень, где он передается в двоичном виде единиц и нулей по сети.

Полностью готовый пакет переносится по кабельной системе сети, достигая в конечном итоге веб-сервера компании Google. Этот веб-сервер начинает чтение полученного пакета снизу вверх, т.е. начиная с информации транспортного уровня, где содержатся сведения о физической адресации по протоколу Ethernet. Эта информация используется в сетевом адаптере с целью выяснить, предназначается ли полученный пакет для данного конкретного веб-сервера. После обработки этой информации из пакета удаляется информация второго уровня и обрабатывается информация третьего уровня.

Информация об адресации по протоколу IP третьего уровня читается с целью убедиться, что пакет адресуется правильно и не разбит на части. Эта информация также удаляется из пакета, чтобы обеспечить обработку информации следующего уровня.

Далее читается информация из протокола TCP четвертого уровня с целью убедиться, что пакет поступил в правильной последовательности. После этого информация четвертого уровня удаляется из пакета, где остаются лишь данные уровня приложений, которые могут быть переданы серверному приложению, развернутому на указанном веб-сервере. В ответ на этот пакет от клиента сервер должен передать сначала пакет подтверждения по протоколу TCP, чтобы уведомить клиента о благополучном получении его запроса, а затем запрашиваемый файл `index.html`.

Все пакеты создаются и обрабатываются описанным в данном примере способом независимо от применяемых протоколов. Но в то же время далеко не каждый пакет в сети формируется, начиная с протокола уровня приложений. В сети нередко можно обнаружить пакеты, содержащие информацию только из протоколов второго, третьего или четвертого уровня.

Сетевое оборудование

А теперь рассмотрим сетевое оборудование, где выполняется вся черновая работа по обмену данными в сети. В этом разделе основное внимание уделяется наиболее характерным компонентам сетевого оборудования, к которым относятся концентраторы, коммутаторы и маршрутизаторы.

Концентраторы

Обычный концентратор представляет собой прямоугольный корпус с целым рядом портов типа RJ-45 подобно модели NETGEAR, приведенной на рис. 1.4. Концентраторы разнятся от совсем небольших устройств на 4 порта до крупных устройств на 48 портов в стоечном исполнении для корпоративной среды.

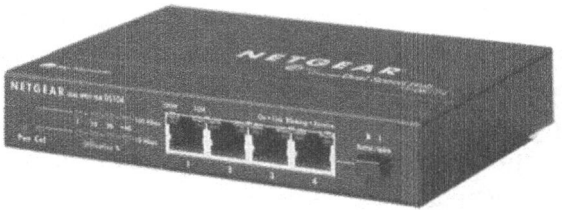


Рис. 1.4. Типичный концентратор на четыре порта для сети Ethernet

Концентраторы могут формировать немало излишнего сетевого трафика и способны работать только в *полудуплексном режиме*, т.е. они не в состоянии одновременно передавать и принимать данные. Поэтому концентраторы, как правило, не применяются в большинстве современных сетей, а также в сетях с высоким трафиком, где вместо них используются коммутаторы, рассматриваемые далее. Тем не менее принцип действия концентраторов следует знать, поскольку им принадлежит важная роль в методике анализа пакетов, называемой "перехватом пакетов через концентратор" и обсуждаемой в главе 2, "Подключение к сети".

Концентратор — это всего лишь *повторитель*, работающий на физическом уровне модели OSI. Он принимает пакеты, посылаемые из одного порта, и передает их во все остальные порты, т.е. повторяет их, а обязанность приемного устройства — принять или отвергнуть каждый пакет. Так, если из компьютера, подключенного к порту 1 концентратора на 4 порта, требуется передать данные на компьютер, подключенный к порту 2, концентратор направит эти пакеты в порты 2, 3 и 4. Клиенты, подключенные в портам 3 и 4, проверят поле адреса получателя по стандарту Media Access Control (MAC — управление доступом к среде передачи данных), или просто MAC-адреса в Ethernet-заголовке пакета и обнаружат, что данный пакет предназначен не для них, и поэтому пропустят (т.е. отвергнут) его. На рис. 1.5 приведен пример, где компьютер А передает данные компьютеру Б. Когда компьютер А посылает эти данные, их получают все компьютеры, подключенные к концентратору. Но лишь компьютер Б фактически принимает отправленные данные, тогда как остальные компьютеры отвергают их.

В качестве аналогии допустим, что сообщение на тему "Внимание всем сотрудникам отдела маркетинга" рассылается по электронной почте всем сотрудникам в компании, а не только тем, кто работает в отделе маркетинга. Увидев данное сообщение, сотрудники этого отдела откроют его, а остальные сотрудники компании проигнорируют его, поскольку оно их не касается. Этот пример наглядно показывает, что такой подход к передаче данных по сети приводит к появлению излишнего трафика и напрасной трате времени. Но именно таким образом и действует концентратор. Наилучшим вариантом

замены концентраторов в производственных и сетях с высоким трафиком являются *коммутаторы — дуплексные* устройства, способные синхронно передавать и принимать данные.

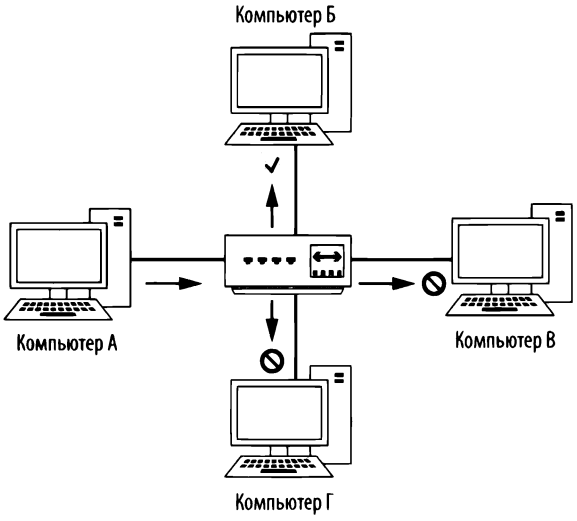


Рис. 1.5. Порядок прохождения трафика, когда компьютер А передает данные компьютеру Б через концентратор

Коммутаторы

Как и концентратор, коммутатор предназначен для повторения пакетов. Но в отличие от концентратора, коммутатор не рассылает данные в каждый порт, а посылает их только тому компьютеру, для которого они предназначены. Коммутаторы очень похожи на концентраторы, как показано на рис. 1.6.

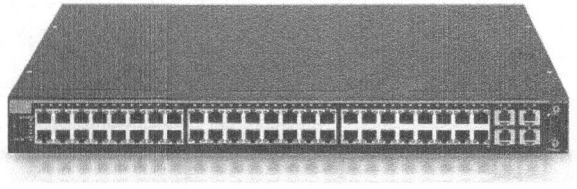


Рис. 1.6. Коммутатор на 48 портов в стоечном исполнении для сети Ethernet

Некоторые крупные коммутаторы (например, компании Cisco) работают под управлением специализированного программного обеспечения или веб-интерфейсов, разработанных изготовителем оборудования. Такие коммутаторы обычно называются *управляемыми* и предоставляют ряд полезных возможностей для управления сетью, включая активизацию и дезактивизацию

портов, просмотр статистики портов, внесение корректив в конфигурацию и удаленную перезагрузку.

Коммутаторы предоставляют дополнительные функциональные возможности для обработки передаваемых пакетов. Для установления непосредственной связи с конкретными устройствами коммутаторы должны однозначно распознавать устройства по их MAC-адресам. Это означает, что они должны действовать на канальном уровне модели OSI.

Коммутаторы хранят адреса второго уровня каждого подключенного к ним устройства в *таблице ассоциативной памяти* (CAM –Content Addressable Memory), которая выполняет роль регулировщика дорожного движения. Когда передается пакет, коммутатор читает информацию из заголовка второго уровня в этом пакете и, используя таблицу ассоциативной памяти как справочник, определяет, в какие порты следует направить данный пакет. Коммутаторы посылают пакеты только в конкретные порты, тем самым значительно сокращая сетевой трафик.

На рис. 1.7 наглядно показан порядок прохождения сетевого трафика через коммутатор. В частности, компьютер А посылает данные только компьютеру Б как назначенному их получателю. Одновременно в сети может вестись множество диалогов, но данные передаются непосредственно между коммутатором и назначенным получателем, а не между коммутатором и всеми подключенными к нему компьютерами.

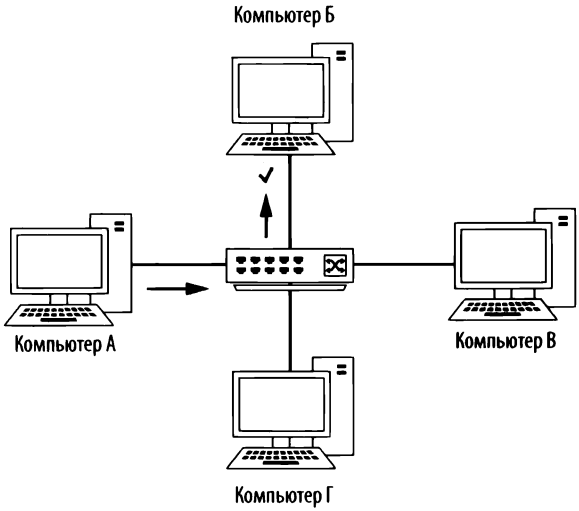


Рис. 1.7. Порядок прохождения трафика, когда компьютер А передает данные компьютеру Б через коммутатор

Маршрутизаторы

Маршрутизатор – это усовершенствованное сетевое устройство с намного более высоким уровнем функционирования, чем коммутатор или концентратор. Маршрутизатор может принимать самые разные виды и формы, но, как правило, на его передней панели установлено несколько светодиодных индикаторов, а на задней панели – ряд сетевых портов в зависимости от масштабов сети. Пример небольшого маршрутизатора приведен на рис. 1.8.

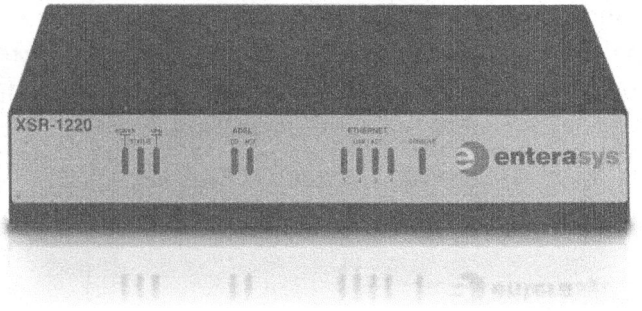


Рис. 1.8. Низкоуровневый маршрутизатор компании Enterasys, пригодный для применения в сетях малых и средних масштабов

Маршрутизаторы действуют на третьем уровне модели OSI, где они отвечают за пересылку пакетов между двумя или несколькими сетями. Процесс, в ходе которого маршрутизаторы направляют сетевой трафик между сетями, называется *маршрутизацией*. Имеется несколько видов протоколов маршрутизации, определяющих порядок, в котором разнотипные пакеты направляются в другие сети. Обычно маршрутизаторы пользуются адресами третьего уровня (например, IP-адресами) для однозначного распознавания устройств в сети.

Чтобы наглядно продемонстрировать понятие маршрутизации, обратимся к аналогии соседства городских улиц, представив дома с их адресами в качестве компьютеров, а каждую улицу – в качестве сетевого сегмента (рис. 1.9). Выйдя из своего дома, можно легко посетить своих соседей в других домах на той же самой улице, пройдя прямо по тротуару от парадной двери одного дома к другой. Аналогично коммутатор обеспечивает связь между всеми компьютерами в одном сегменте сети.

Но общение с соседями, проживающими на другой улице, подобно связи с компьютером в другом сегменте сети. Глядя на рис. 1.9, допустим, что вы проживаете на улице Виноградной, 502, и вам требуется добраться до дома по адресу Кизиловая аллея, 206. Чтобы сделать это, вы должны свернуть сначала на Дубравную улицу, а затем – на Кизиловую аллею. Это можно сравнить с пересечением разных сегментов сети. Так, если устройству, находящемуся

по адресу `192.168.0.3`, требуется связаться с устройством, расположенным по адресу `192.168.0.54`, оно должно сделать это сначала через свой маршрутизатор, чтобы достичь сети по адресу `10.100.1.x`, а затем через маршрутизатор целевого сегмента сети, прежде чем достичь этого сегмента.

Размеры и количество маршрутизаторов в сети, как правило, зависят от ее масштабов и функций. На границе личных, домашних и небольших учрежденческих сетей может быть установлен лишь один небольшой маршрутизатор. А в крупных корпоративных сетях может потребоваться несколько маршрутизаторов, установленных в разных отделах организации и подключенных к одному центральному маршрутизатору или коммутатору третьего уровня (т.е. к усовершенствованному типу коммутатора со встроенными возможностями функционирования в качестве маршрутизатора).

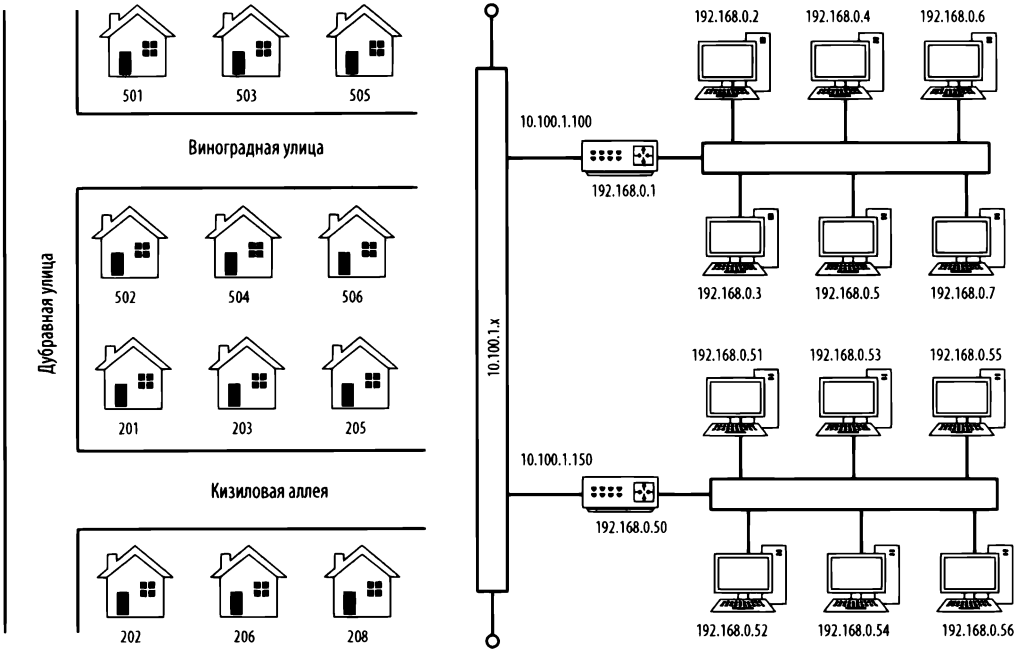


Рис. 1.9. Сравнение маршрутизируемой сети с соседством городских улиц

Рассматривая приведенные далее схемы сетей, вы в конечном итоге поймете, каким образом данные проходят через различные точки сети. Так, на рис. 1.10 приведена весьма распространенная форма компоновки маршрутизируемой сети. В данном случае две отдельные сети соединены через один маршрутизатор. Если компьютеру в сети А потребуется связаться с компьютером в сети Б, то передаваемые из него данные должны непременно пройти через маршрутизатор.

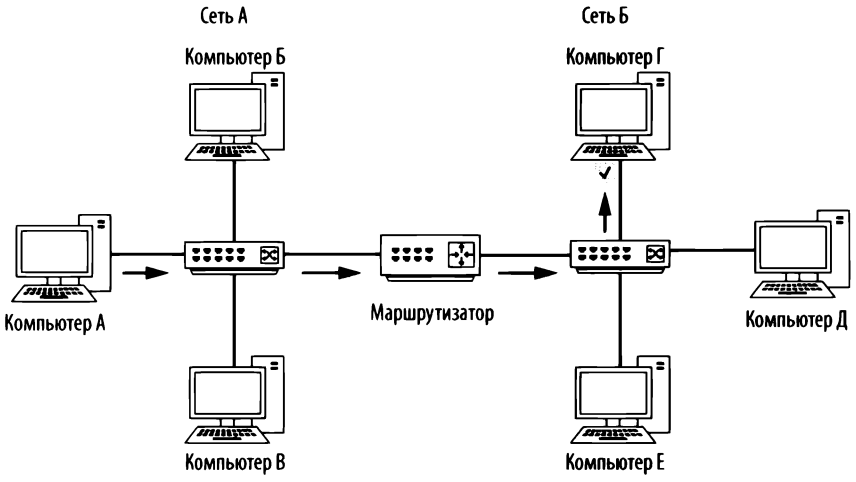


Рис. 1.10. Порядок прохождения сетевого трафика, когда компьютер А в одной сети передает данные компьютеру Г в другой сети через маршрутизатор

Классификация сетевого трафика

Сетевой трафик можно классифицировать по трем типам: широковещательный, много- и одноадресатный. Каждая из этих классификаций сетевого трафика имеет свои особенности, определяющие порядок обработки пакетов в сетевом оборудовании.

Широковещательный трафик

Широковещательным (broadcast) называется такой пакет, который посылается во все порты в сетевом сегменте независимо от того, где установлен данный порт: на концентраторе или коммутаторе. Имеются формы широковещательного трафика второго и третьего уровня. На втором уровне MAC-адрес **`ff:ff:ff:ff:ff:ff`** является зарезервированным широковещательным адресом, и любой трафик, направляемый по этому адресу, распространяется по всему сетевому сегменту. Широковещательный адрес имеется и на третьем уровне, но он меняется в зависимости от используемого диапазона сетевых адресов.

Самый большой из всех IP-адресов, допустимых в диапазоне адресов IP-сети, зарезервирован для применения в качестве широковещательного адреса. Так, если у компьютера имеется адрес **`192.168.0.20`** и маска подсети **`255.255.255.0`**, адрес **`192.168.0.255`** оказывается широковещательным (подробнее об IP-адресации речь пойдет в главе 7, "Протоколы сетевого уровня").

Предел, до которого могут распространяться широковещательные пакеты, называется *широковещательным доменом*, который представляет собой сетевой сегмент, где любой компьютер может непосредственно передавать данные другому компьютеру без помощи маршрутизатора. В крупных сетях со многими концентраторами или коммутаторами, соединяемыми через разные средства связи и среды передачи данных, широковещательные пакеты, передаваемые от одного коммутатора, достигают всех портов на всех остальных коммутаторах в сети, поскольку пакеты повторяются, проходя от одного коммутатора к другому. На рис. 1.11 приведен пример двух широковещательных доменов в небольшой сети. Каждый широковещательный домен простирается до тех пор, пока он не достигнет маршрутизатора, и поэтому широковещательные пакеты циркулируют только в этом конкретном домене.

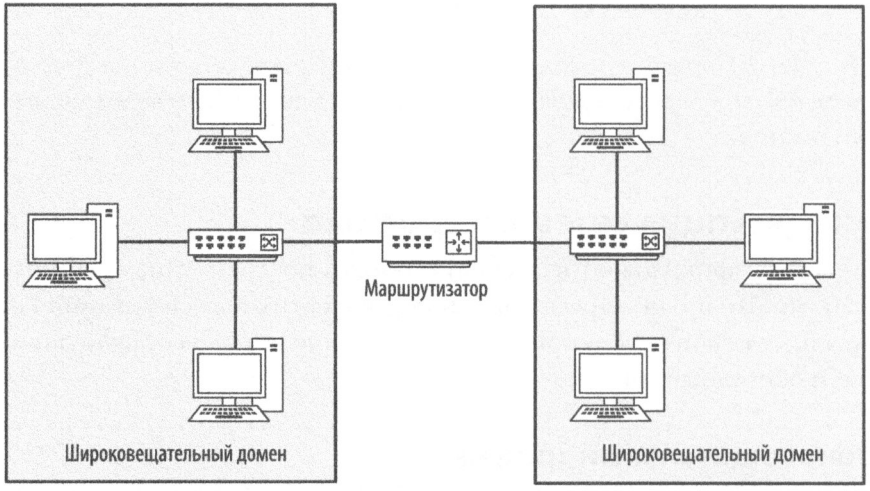


Рис. 1.11. Широковещательный домен охватывает весь текущий сегмент маршрутизируемой сети до маршрутизатора

Упоминавшаяся ранее аналогия с соседством городских улиц позволяет также наглядно показать принцип действия широковещательных доменов. В частности, широковещательный домен можно рассматривать как соседнюю улицу, где все соседи сидят на своем крыльце. Если вы станете на свое крыльцо и крикнете, то жильцы на вашей улице смогут вас услышать. Но если вам захочется пообщаться с кем-нибудь на другой улице, то придется найти какой-нибудь способ сделать это непосредственно, а не широко вещать (криком) со своего крыльца.

Многоадресатный трафик

Многоадресатной[2] *(multicast)* называется одновременная передача пакета из одного источника во многие места назначения. Цель многоадресатной передачи — использовать как можно меньше пропускной способности сети. Оптимизация многоадресатного трафика состоит в том, чтобы поменьше тиражировать поток данных по пути его следования к месту назначения. Конкретная обработка многоадресатного трафика зависит от его реализации в отдельных сетевых протоколах.

Многоадресатный трафик реализуется в основном через схему адресации, добавляющую получателей пакетов к многоадресатной группе. Именно так и действует многоадресатная передача по межсетевому протоколу IP. Такая схема адресации гарантирует от передачи пакетов тем компьютерам, для которых они не предназначены. Если вы обнаружите IP-адрес в пределах от `224.0.0.0` до `239.255.255.255`, то по нему, вероятнее всего, обрабатывается многоадресатный трафик, поскольку именно в этих пределах зарезервированы адреса для подобных целей.

Одноадресатный трафик

Одноадресатный (unicast) пакет передается непосредственно от одного компьютера к другому. Конкретное функционирование одноадресатной передачи зависит от применяемого сетевого протокола. В качестве примера можно привести устройство, которому требуется связаться с веб-сервером. Это одноточечное соединение, а следовательно, процесс передачи данных, должен быть начат клиентским устройством, передающим пакет только веб-серверу.

Заключительные соображения

В этой главе были представлены основы организации сетей, которые требуется знать для анализа пакетов. В частности, необходимо ясно понимать, что именно происходит на данном уровне передачи данных по сети, прежде чем приступить к диагностике сети. В главе 2, "Подключение к сети", будет рассмотрен ряд методик перехвата пакетов, которые требуется проанализировать.

---

[2] Поскольку пакет, посланный по одному адресу, достигает многих получателей, или адресатов. — *Примеч. ред.*

2

ПОДКЛЮЧЕНИЕ К СЕТИ

Главное решение для эффективного анализа пакетов лежит в физическом расположении анализатора пакетов для надлежащего перехвата данных. Специалисты по анализу пакетов нередко называют местоположение анализатора пакетов *прослушиванием сети* или *подключением к сети.*

К сожалению, для анализа пакетов недостаточно подключить свой переносной компьютер к сетевому порту и перехватить трафик. На самом деле иногда разместить анализатор пакетов в сети намного сложнее, чем фактически проанализировать пакеты. Расположить анализатор пакетов нелегко потому, что устройства могут быть подключены к сети с помощью самых разных видов сетевого оборудования. Типичная ситуация для анализа пакетов приведена на рис. 2.1. Каждое из устройств в современной сети (коммутаторы и маршрутизаторы) по-разному обрабатывает сетевой трафик, и поэтому приходится принимать во внимание физическую организацию анализируемой сети.

В этой главе преследуется цель помочь вам лучше понять порядок расположения анализатора пакетов в сетях с самой разной топологией. Но прежде рассмотрим, каким образом можно увидеть прохождение всех пакетов по сети, к которой мы подключаемся для анализа пакетов.

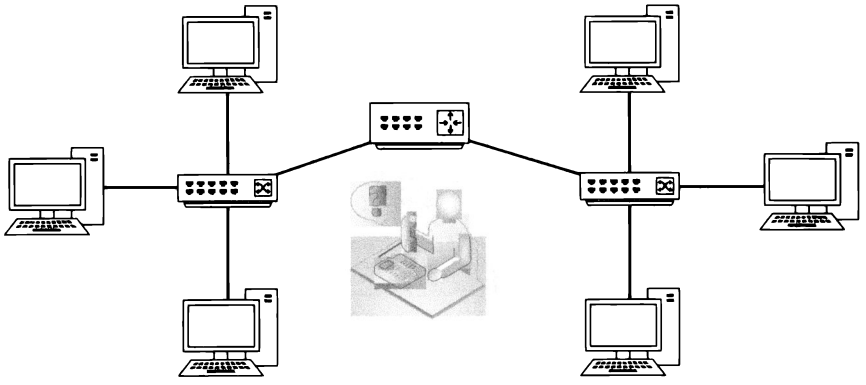


Рис. 2.1. Разместить анализатор пакетов в сети не так-то просто, если в ней имеется немало соединений, поскольку такая ее организация затрудняет получение нужных данных

Прослушивание сети в смешанном режиме

Прежде чем анализировать пакеты в сети, необходимо обзавестись сетевой интерфейсной платой (NIC), иначе называемой сетевым адаптером, с поддержкой драйвера смешанного режима (promiscuous mode) для прослушивания сети. В *смешанном режиме* сетевой адаптер может просматривать все пакеты, проходящие по сети.

Как пояснялось в главе 1, "Анализ пакетов и основы организации сетей", из широковещательного сетевого трафика устройства обычно получают пакеты, которые фактически не предназначены для них. Например, сетевой протокол ARP (Address Resolution Protocol – протокол преобразования адресов) служит крайне важным средством в любой исследуемой сети для определения MAC-адресов, соответствующих конкретному IP-адресу. Чтобы обнаружить подходящий MAC-адрес, устройство посылает широковещательный пакет по сетевому протоколу ARP каждому устройству, находящемуся в его широковещательном домене, в надежде, что нужное устройство отреагирует на данный пакет.

Широковещательный домен (т.е. сетевой сегмент, где любой компьютер может передавать данные непосредственно любому другому компьютеру без помощи маршрутизатора) может состоять из нескольких устройств. Но лишь нужное приемное устройство в этом домене должно быть заинтересовано в получении широковещательного пакета, передаваемого по сетевому протоколу ARP. И было бы совершенно неэффективно, если бы каждое устройство в сети обрабатывало широковещательный пакет, передаваемый по сетевому протоколу ARP. Напротив, если пакет не предназначен для устройства, а сле-

довательно, не нужен ему, то сетевой адаптер данного устройства отбросит пакет вместо того, чтобы передать его на обработку центральному процессору (ЦП).

Отбрасывание пакетов, не предназначенных для принимающего хоста, повышает эффективность обработки данных в сети, но для анализа пакетов этот режим работы сетевой платы совсем не подходит. Специалистам по анализу пакетов, как правило, требуется перехватывать *каждый* пакет, посылаемый по сети, чтобы не пропустить какой-нибудь важный фрагмент информации.

Используя смешанный режим работы сетевого адаптера, можно гарантировать, что будет перехвачен весь сетевой трафик. Когда сетевой адаптер работает в смешанном режиме, он передает каждый обнаруживаемый им пакет процессору хоста независимо от его адреса его получателя. И как только пакет поступит на обработку в ЦП, приложение, анализирующее пакеты, сможет взять его на анализ.

Смешанный режим поддерживается в большинстве современных сетевых адаптеров, и в состав Wireshark входит драйвер libpcap/WinPcap, позволяющий переключить сетевой адаптер непосредственно в смешанный режим из графического интерфейса инструментального средства Wireshark. (Подробнее о драйвере libpcap/WinPcap речь пойдет в главе 3, "Введение в Wireshark".)

ПРИМЕЧАНИЕ *В большинстве операционных систем, включая Windows, не допускается применять сетевой адаптер в смешанном режиме, если только у вас нет расширенных пользовательских полномочий. Если же у вас нет законных оснований получить такие полномочия в своей системе, вам, скорее всего, не удастся провести такого рода анализ пакетов в данной конкретной сети.*

Анализ пакетов через концентраторы

Анализ пакетов в сети, где установлены концентраторы, — мечта любого исследователя пакетов. Как пояснялось в главе 1, "Анализ пакетов и основы организации сетей", сетевой трафик, пропускаемый через концентратор, поступает на каждый порт, подключенный к этому концентратору. Следовательно, чтобы проанализировать трафик, проходящий через компьютер, подключенный к концентратору, достаточно подключить анализатор пакетов к пустому порту концентратора. Это даст возможность просматривать весь обмен данными как с этим компьютером, так и с любыми другими устройствами, подключенными к тому же самому концентратору. Как показано на рис. 2.2, пределы видимости безграничны, когда анализатор пакетов подключен к сети, созданной на основе концентраторов.

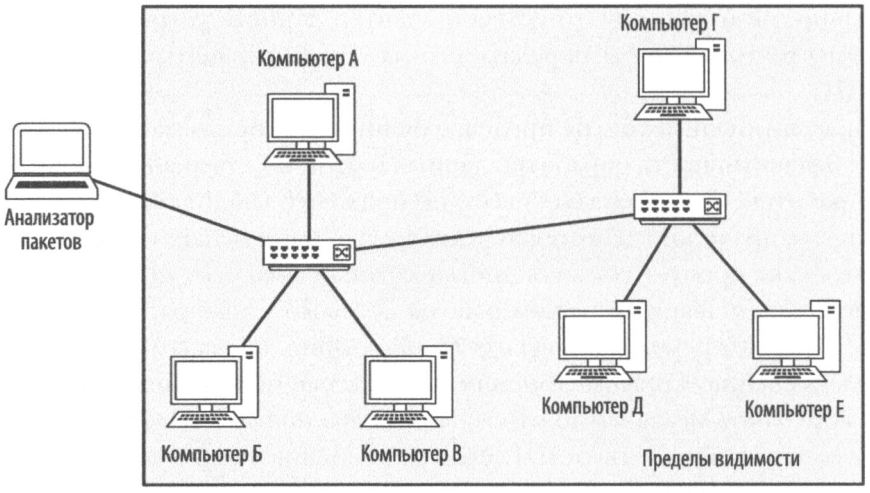


Рис. 2.2. Анализ пакетов в радиально-узловой сети, созданной на основе концентраторов, обеспечивает безграничные пределы видимости

ПРИМЕЧАНИЕ *Как следует из различных блок-схем, приведенных в данной книге, термин пределы видимости охватывает устройства в сети, трафик которых можно просматривать с помощью анализатора пакетов.*

Но теперь, к сожалению, сети очень редко строятся на основе концентраторов, поскольку их администрирование весьма затруднено. В таких сетях лишь одно устройство может одновременно передавать данные через концентратор, и поэтому все подключенные к нему устройства должны конкурировать за право передать данные со всеми остальными устройствами сети. Если данные будут одновременно передавать два или больше устройств, то возникает так называемое явление *коллизии пакетов*, как показано на рис. 2.3. В итоге может произойти потеря пакетов, и для выхода из сложившейся ситуации передающим устройствам нужно будет повторить их передачу, что приводит к падению пропускной способности сети. По мере увеличения уровня трафика и количества коллизий устройствам, возможно, придется повторять передачу пакетов по три или четыре раза, а это в значительной степени снизит производительность сети. Таким образом, нетрудно понять, почему в большинстве современных сетей любых масштабов применяются коммутаторы. Но, несмотря на то что концентраторы редко применяются в современных сетях, иногда они все же встречаются в тех сетях, где поддерживается устаревшее оборудование или специализированные устройства, например, в сетях автоматизированных систем управления производственными процессами (ICS).

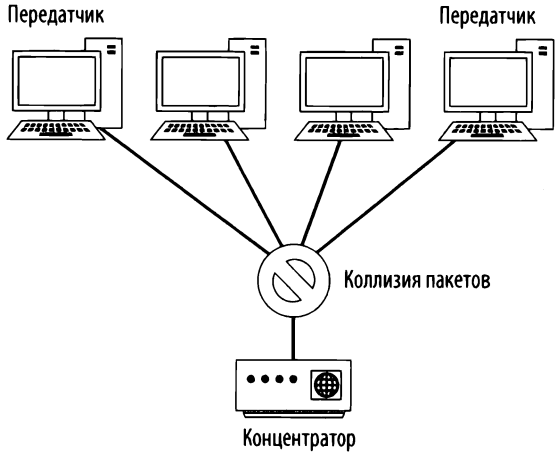


Рис. 2.3. В радиально-узловой сети, созданной на основе концентраторов, может возникнуть коллизия пакетов, когда два или больше устройств будут передавать данные одновременно

Чтобы выяснить наличие концентратора в сети, проще всего заглянуть в серверное помещение или сетевой монтажный шкаф, где большинство концентраторов специально обозначены метками. Если и это не поможет, загляните в самый темный угол серверного помещения, где обычно находится покрытое толстым слоем пыли сетевое оборудование.

Анализ пакетов в коммутируемой среде

В качестве соединительных устройств в современных сетях чаще всего применяются коммутаторы. Они обеспечивают эффективный порядок переноса данных через широковещательный, много- и одноадресатный трафик. Коммутаторы допускают дуплексную передачу данных, а это означает, что машины могут передавать и принимать данные одновременно.

К сожалению, коммутаторы усложняют задачу исследователям пакетов. Подключив свой анализатор пакетов к порту коммутатора, вы сможете просмотреть лишь часть широковещательного трафика, а также трафик, передаваемый и принимаемый тем устройством, на котором установлен анализатор пакетов (рис. 2.4). Чтобы перехватить трафик из целевого устройства в коммутируемой сети, вам придется предпринять дополнительные шаги. Перехватить такой трафик можно следующими четырьмя способами: зеркальное отображение портов (port mirroring), перехват пакетов через концентратор, применение сетевого ответвителя и заражение ARP-кеша.

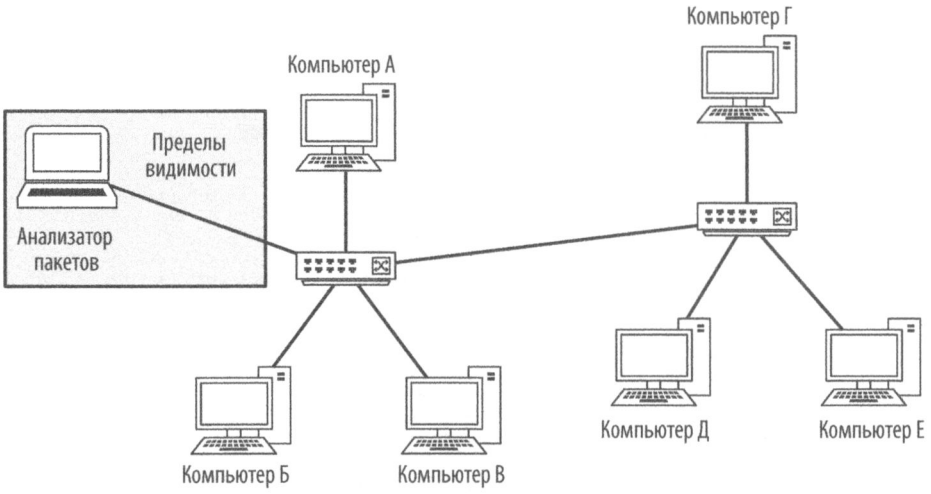


Рис. 2.4. Пределы видимости в коммутируемой сети ограничиваются портом, к которому подключен анализатор пакетов

Зеркальное отображение портов

Зеркальное отображение портов (port mirroring), иначе называемое *расширением портов (port spanning)*, считается едва ли не самым простым способом перехвата сетевого трафика из целевого устройства в коммутируемой сети. Для этого необходимо иметь доступ к интерфейсу командной строки или веб-управления того коммутатора, к которому подключен целевой компьютер. Кроме того, коммутатор должен поддерживать зеркальное отображение портов и иметь пустой порт, к которому можно подключить анализатор пакетов.

Чтобы активизировать зеркальное отображение портов, следует выдать команду, вынуждающую коммутатор копировать весь трафик из одного порта в другой. Например, чтобы перехватить весь трафик, который передает и принимает устройство, подключенное к порту 3 коммутатора, достаточно подключить анализатор пакетов к его порту 4 и зеркально отобразить порт 3 на порт 4. Процесс зеркального отображения портов наглядно показан на рис. 2.5.

Порядок организации зеркального отображения портов зависит от конкретной модели коммутатора. В большинстве промышленных коммутаторов придется пройти регистрацию через интерфейс командной строки и настроить зеркальное отображение портов по специальной команде. Перечень таких команд приведен в табл. 2.1.

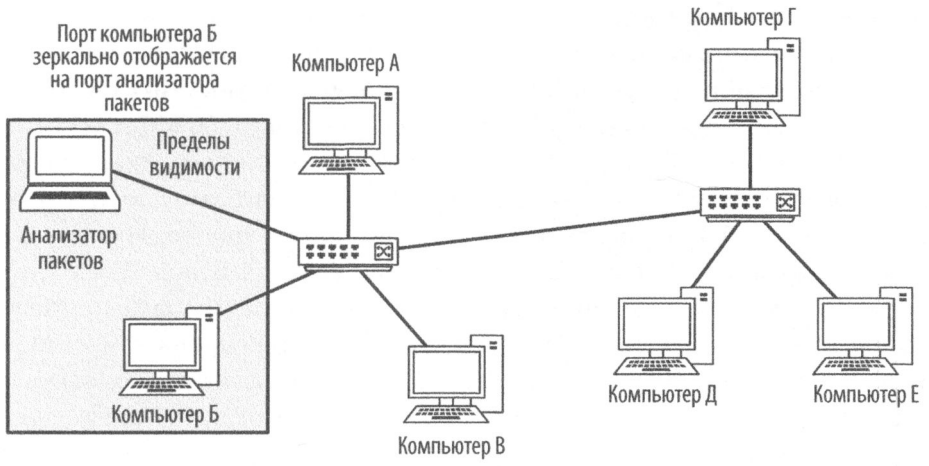


Рис. 2.5. Зеркальное отображение портов позволяет расширить пределы видимости в коммутируемой сети

Таблица 2.1. Команды, применяемые для активизации зеркального отображения

Производитель коммутатора	Команда
Cisco	`set span <порт источника> <порт назначения>`
Enterasys	`set port mirroring create <порт источника> <порт назначения>`
Nortel	`port-mirroring mode mirror-port <порт источника>` `monitor-port <порт назначения>`

ПРИМЕЧАНИЕ *В некоторых промышленных коммутаторах предоставляется веб-ориентированный графический интерфейс, где зеркальное отображение портов не всегда поддерживается, хотя такие коммутаторы нетипичны и не стандартизированы. Но если коммутатор предоставляет эффективный способ настройки зеркального отображения портов через веб-интерфейс, то такой возможностью следует, безусловно, воспользоваться. Кроме того, в коммутаторах для небольших учрежденческих и домашних сетей (SOHO) все чаще начинают предоставляться возможности для зеркального отображения портов, которое, как правило, настраивается через веб-интерфейс.*

Зеркально отображая порты, следует принимать во внимание их пропускную способность. Некоторые производители коммутаторов позволяют зеркально отображать несколько портов на один порт, и такая функция может оказаться полезной при анализе передачи данных между двумя или несколькими устройствами на одном коммутаторе. Рассмотрим, однако, что может при

этом произойти, произведя несложные математические расчеты. Если, например, имеется коммутатор на 24 порта, где 23 порта зеркально отображаются на один порт при дуплексной передаче данных со скоростью 100 Мбит/с, то данные должны поступать в этот порт со скоростью 4600 Мбит/с. Но ведь это выходит далеко за пределы физических возможностей одного порта, а следовательно, может привести к потере пакетов или замедлению работы сети, если сетевой трафик достигнет определенного уровня. Иногда подобная ситуация называется *переподпиской (oversubscription)*. В подобных случаях коммутаторы, как известно, пропускают лишние пакеты или даже нарушается работа внутренних электрических схем, что полностью препятствует передаче данных. Поэтому непременно убедитесь, что, перехватывая пакеты рассматриваемым здесь способом, вы не вызовите подобные осложнения в сети.

Зеркальное отображение портов может оказаться привлекательным, недорогим решением для корпоративных сетей и в тех случаях, когда требуется постоянно контролировать отдельные сегменты сети, например, в ходе текущего контроля сетевой безопасности. Но такая методика, как правило, оказывается недостаточно надежной для подобного применения. Зеркальное отображение портов может давать противоречивые результаты особенно при высоких уровнях трафика в сети, приводя к потере данных, которую трудно проследить. В подобных случаях рекомендуется пользоваться сетевым ответвителем, как поясняется далее в соответствующем разделе.

Перехват пакетов через концентратор

Еще одним способом перехвата трафика, проходящего через целевое устройство в коммутируемой сети, является *перехват пакетов через концентратор*. По этой методике целевое устройство и анализатор пакетов размещаются в одном сегменте коммутируемой сети и подключаются непосредственно к концентратору. Многие считают перехват пакетов через концентратор обманом, но на самом деле это вполне обоснованное решение, когда нельзя выполнить зеркальное отображение портов, но в то же время имеется физический доступ к тому коммутатору, к которому подключено целевое устройство.

Чтобы организовать перехват пакетов через концентратор, достаточно иметь в своем распоряжении сам концентратор и несколько сетевых кабелей. Если у вас имеется такое оборудование, подключите его следующим образом.

1. Найдите коммутатор, к которому подключено целевое устройство, и отключите последнее от сети.
2. Подключите сетевой кабель целевого устройства к своему концентратору.

3. Подключите еще один сетевой кабель, соединяющий ваш анализатор пакетов с концентратором.
4. Подключите сетевой кабель, идущий от концентратора, к сетевому коммутатору, чтобы присоединить концентратор к сети.

Итак, вы разместили целевое устройство и анализатор пакетов в одном широковещательном домене. В итоге весь сетевой трафик от данного устройства будет передаваться в широковещательном режиме. В результате анализатор пакетов может перехватывать из него все пакеты, как показано на рис. 2.6.

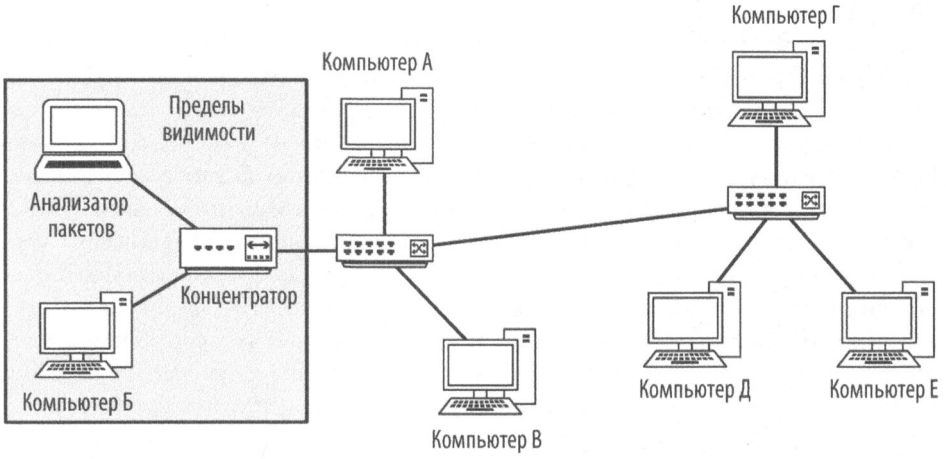


Рис. 2.6. При перехвате пакетов через концентратор целевое устройство и анализатор пакетов изолированы

В большинстве случаев перехват пакетов через концентратор сводит дуплексную (т.е. двунаправленную) передачу данных на целевом устройстве к полудуплексной (т.е. однонаправленной). И хотя эта методика не является самой лучшей для перехвата пакетов, она остается единственно возможной в тех случаях, когда в коммутаторе не поддерживается зеркальное отображение портов. Следует, однако, иметь в виду, что концентратор потребуется также подключить к розетке электросети, которую бывает нелегко найти.

ПРИМЕЧАНИЕ *Не забудьте своевременно предупредить пользователя целевого устройства, что вы собираетесь отключить его от сети, особенно если это большой начальник!*

ВЫБОР "ПРАВИЛЬНЫХ" КОНЦЕНТРАТОРОВ

Чтобы перехватывать пакеты через концентратор, воспользуйтесь настоящим концентратором, а не коммутатором, ложно отнесенным к категории концентраторов. Некоторые производители сетевого оборудования имеют скверную привычку рекламировать и продавать свои устройства как "концентраторы", хотя они на самом деле выполняют функции низкоуровневых коммутаторов. Если вы не пользуетесь проверенным, испытанным концентратором, то сможете просматривать только собственный трафик, а не тот, что проходит через целевое устройство.

Выбирая устройство, которое якобы считается концентратором, убедитесь, что оно отвечает своему назначению. Чтобы выяснить, является ли выбираемое устройство настоящим концентратором, лучше всего подключить к нему пару компьютеров и проверить, можно ли на одном из них анализировать сетевой трафик между вторым и другими устройствами в сети, например, компьютерами или принтерами. Если это можно сделать, значит, вы выбрали именно то, что нужно!

Концентраторы стали теперь настолько редки, что уже не производятся массово. Приобрести имеющийся в продаже настоящий концентратор практически невозможно, и поэтому вам придется проявить творческую инициативу, чтобы найти нужный товар. Отличным местом для поиска концентраторов служит аукцион бывших в употреблении предметов, проводимый в районе местной школы. Государственные школы обязаны попытаться выставить на аукцион бывшие в употреблении предметы, прежде чем избавиться от них, а ведь у них нередко залеживается старое оборудование. Мне не раз приходилось видеть людей, уходивших с таких аукционов с несколькими концентраторами, приобретенными по цене меньше стоимости тарелки фасоли или кукурузного хлеба. Еще одним удобным местом для приобретения концентраторов может служить электронный аукцион eBay, но будьте на чеку, чтобы не приобрести коммутатор, ложно отнесенный к категории концентраторов.

Применение сетевого ответвителя

Всем известно выражение "Зачем мне курица, если я могу съесть бифштекс?" А если вы с юга США, то вам знакомо такое выражение: "Зачем мне ливерный хлеб, если я могу съесть бутерброд с поджаренной болонской колбасой?" Это же относится и к выбору между концентратором и сетевым ответвителем для перехвата и анализа пакетов.

Сетевой *ответвитель* – это оборудование, которое можно установить между двумя точками кабельной разводки сети, чтобы перехватывать пакеты, проходящие между этими точками. И в данном случае, как и при перехвате пакетов через концентратор, в сети устанавливается оборудование, которое позволяет перехватывать нужные для анализа пакеты. Отличие рассматриваемого здесь способа заключается в том, что вместо концентратора в данном

случае применяется оборудование, специально предназначенное для анализа сетевого трафика.

Имеются два основных типа сетевых ответвителей: *агрегированный* и *неагрегированный*. Оба типа сетевых ответвителей устанавливаются между двумя устройствами для анализа обмена данными между ними. Главное отличие неагрегированного сетевого ответвителя от агрегированного заключается в том, что у него имеются четыре порта, как показано на рис. 2.7, и ему требуются отдельные интерфейсы для контроля текущего сетевого трафика в обоих направлениях. А у агрегированного сетевого ответвителя имеются лишь три порта, но он позволяет проводить контроль текущего сетевого трафика в обоих направлениях с помощью единственного интерфейса. Кроме того, сетевые ответвители, как правило, требуется подключать к розетке электросети, хотя некоторые из них работают и от батарей электропитания, допуская краткосрочное проведение анализа пакетов.

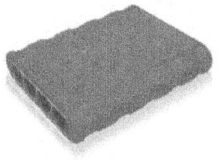


Рис. 2.7. Неагрегированный сетевой ответвитель типа Barracuda

Агрегированные сетевые ответвители

Проще всего пользоваться агрегированным сетевым ответвителем, у которого имеется лишь один физический контрольный порт для анализа двунаправленного сетевого трафика. Чтобы перехватывать с помощью агрегированного сетевого ответвителя весь входящий и исходящий сетевой трафик одного компьютера, подключенного к коммутатору, выполните следующие действия.

1. Отключите компьютер от коммутатора.
2. Подключите один конец первого сетевого кабеля к компьютеру, а другой — к входному порту сетевого ответвителя.
3. Подключите один конец второго сетевого кабеля к выходному порту сетевого ответвителя, а другой — к коммутатору.
4. Подключите один конец третьего сетевого кабеля к контрольному порту сетевого ответвителя, а другой — к компьютеру, выполняющему роль анализатора пакетов.

Агрегированный сетевой ответвитель должен быть подключен к сети так, как показано на рис. 2.8. В итоге анализатор пакетов должен перехватывать весь входящий и исходящий трафик компьютера, подключенного к данному ответвителю.

Рис. 2.8. Применение агрегированного сетевого ответвителя для перехвата сетевого трафика

Неагрегированные сетевые ответвители

Неагрегированный сетевой ответвитель немного сложнее, чем агрегированный. Тем не менее он допускает чуть больше удобств в перехвате сетевого трафика. Вместо единственного контрольного порта, применяемого для прослушивания двунаправленного обмена данными, неагрегированный сетевой ответвитель предоставляет два контрольных порта. Один контрольный порт служит для анализа сетевого трафика в одном направлении (от компьютера, подключенного к сетевому ответвителю), а другой — для анализа сетевого трафика в другом направлении (к компьютеру, подключенному к сетевому ответвителю).

Чтобы перехватывать весь входящий и исходящий сетевой трафик компьютера, подключенного к коммутатору, выполните следующие действия.

1. Отключите компьютер от коммутатора.
2. Подключите один конец первого сетевого кабеля к компьютеру, а другой — к входному порту сетевого ответвителя.
3. Подключите один конец второго сетевого кабеля к выходному порту сетевого ответвителя, а другой — к коммутатору.
4. Подключите один конец третьего сетевого кабеля к контрольному порту A сетевого ответвителя, а другой его конец — к одному сетевому адаптеру на компьютере, выполняющем роль анализатора пакетов.

5. Подключите один конец четвертого сетевого кабеля к контрольному порту B сетевого ответвителя, а другой – к другому сетевому адаптеру на компьютере, выполняющем роль анализатора пакетов.

В итоге неагрегированный сетевой ответвитель должен быть подключен к сети так, как показано на рис. 2.9.

Рис. 2.9. Применение неагрегированного сетевого ответвителя для перехвата сетевого трафика

Приведенные выше примеры могут навести на мысль, что с помощью сетевого ответвителя можно контролировать только одно устройство. Но на самом деле контролировать можно многие устройства, проявив творческий подход к размещению сетевого ответвителя. Так, если требуется полностью контролировать обмен данными между целым сегментом сети и Интернетом, сетевой ответвитель можно установить между коммутатором, к которому подключены все остальные устройства, и маршрутизатором восходящего потока данных в сети. Такое расположение в узком месте сети позволяет собирать требующийся сетевой трафик. Подобная стратегия обычно применяется при текущем контроле сетевой безопасности.

Выбор сетевого ответвителя

Какой же тип сетевого ответвителя лучше? Как правило, предпочтение следует отдавать агрегированным сетевым ответвителям, поскольку для них требуется меньше кабельной разводки и не нужно устанавливать два сетевых адаптера на компьютере анализатора пакетов. Но если требуется перехватывать большие объемы сетевого трафик или контролировать трафик, проходящий только в одном направлении, то лучше выбрать неагрегированный сетевой ответвитель.

Приобрести можно сетевые ответвители всех размеров: от простых ответвителей сети Ethernet за 150 долл. до оптоволоконных ответвителей корпоративного уровня, цена которых обозначается шестизначными цифрами. Мне лично приходилось пользоваться сетевыми ответвителями корпоративного уровня от компаний Ixia (бывшей Net Optics), Dualcomm и Fluke Networks, и я был ими очень доволен, хотя имеется много других замечательных сетевых ответвителей. Если вы предполагаете применять сетевой ответвитель на корпоративном уровне, убедитесь в достаточной его отказоустойчивости. Это означает, что если сетевой ответвитель неверно функционирует или выходит из строя, то он должен все равно пропускать пакеты, не нарушая связность сети на ее ответвляемом участке.

Заражение ARP-кеша

К числу наиболее предпочитаемых мною методик перехвата сетевого трафика относится заражение ARP-кеша. Более подробно сетевой протокол ARP будет рассматриваться в главе 7, "Протоколы сетевого уровня", а здесь приводится лишь краткое его описание, необходимое для понимания особенностей данной методики.

ARP-процесс

Как упоминалось в главе 1, "Анализ пакетов и основы организации сетей", в модели OSI адресация пакетов может осуществляться на двух уровнях — втором и третьем. Адреса второго уровня или MAC-адреса применяются вместе с выбранной вами системой адресации третьего уровня. В данной книге система адресации третьего уровня обозначается как *система IP-адресации* в соответствии с принятой в данной отрасли стандартной терминологией.

Все устройства в сети связываются вместе по IP-адресам на третьем уровне модели OSI. А поскольку коммутаторы действуют на втором уровне данной модели, то они осведомлены только о MAC-адресах второго уровня. Следовательно, для обмена пакетами между собой устройства должны включать в них информацию о MAC-адресах. Если же MAC-адрес неизвестен, он должен быть получен из известного IP-адреса третьего уровня, чтобы можно было пересылать сетевой трафик соответствующему устройству. И такой процесс преобразования адресов выполняется с помощью сетевого протокола ARP на втором уровне модели OSI.

Для компьютеров, подключенных к сетям Ethernet, ARP-процесс начинается в тот момент, когда одному компьютеру требуется связаться с другим. Сначала передающий компьютер проверяет свой ARP-кеш, чтобы выяснить, имеется ли в нем уже MAC-адрес, связанный с IP-адресом компьютера-получателя. Если такой адрес отсутствует, передающий компьютер посылает ARP-запрос

по широковещательному адресу `ff:ff:ff:ff:ff:ff` канального уровня, в котором указывает IP-адрес получателя, как пояснялось в главе 1, "Анализ пакетов и основы организации сетей". Формируемый в итоге широковещательный пакет принимается всеми компьютерами в данном конкретном сегменте сети Ethernet. По существу, этот пакет содержит такой запрос: "Какой MAC-адрес имеет компьютер с указанным IP-адресом?"

Те устройства, которые не соответствуют указанному в запросе IP-адресу получателя, просто игнорируют данный ARP-запрос. А компьютер, IP-адрес которого совпал с указанным в ARP-запросе, формирует ответный ARP-пакет, в котором указывает свой MAC-адрес. Таким образом, исходный передающий компьютер получает информацию об адресации канального уровня, которая требуется ему для связывания с удаленным компьютером. И эту информацию он сохраняет в своем ARP-кеше для быстрого ее извлечения.

Принцип действия заражения ARP-кеша

Заражение ARP-кеша, иногда еще называемое *ARP-подменой*, является усовершенствованной формой подключения к коммутируемой сети с целью ее прослушивания. Принцип его действия состоит в том, чтобы посылать ARP-сообщения коммутатору или маршрутизатору сети Ethernet с поддельными MAC-адресами (второго уровня) для перехвата сетевого трафика другого компьютера, как наглядно показано на рис. 2.10.

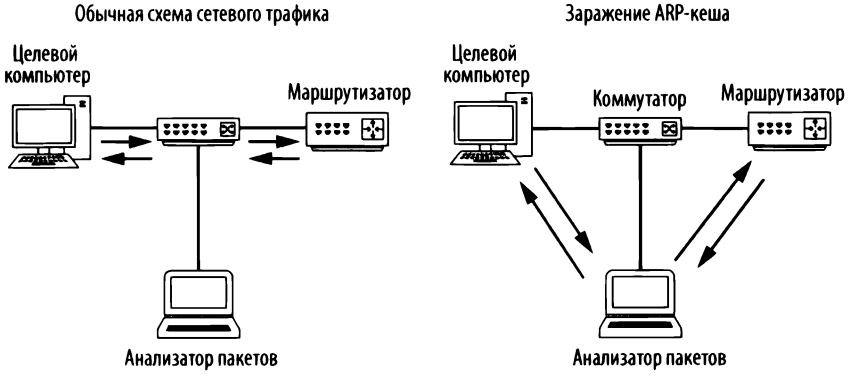


Рис. 2.10. Заражение ARP-кеша позволяет перехватывать сетевой трафик целевого компьютера

Такая методика обычно применяется атакующими злоумышленниками для посылки ложно адресуемых пакетов клиентским системам, чтобы перехватить определенный сетевой трафик или вызвать атаку типа отказа в обслуживании (DoS) на целевой компьютер. Но данную методику можно применять и вполне законно для перехвата пакетов целевого компьютера в коммутируемой сети.

Применение Cain & Abel

Прежде чем пытаться применить заражение ARP-кеша, следует приобрести необходимые инструментальные средства и собрать некоторую информацию. В целях демонстрации данной методики воспользуемся распространенным инструментальным средством защиты Cain & Abel от компании oxid.it (`http://www.oxid.it/`), которое поддерживает операционные системы Windows. Загрузите и установите это инструментальное средство, следуя инструкциям на веб-сайте по указанному выше адресу.

ПРИМЕЧАНИЕ *При попытке загрузить инструментальное средство Cain & Abel, вероятнее всего, антивирусная программа или браузер отметит его как зловредное программное обеспечение или "хакерское инструментальное средство". Оно действительно находит самое разное применение, включая и злонамеренное. Но для рассматриваемых здесь целей оно не представляет никакой угрозы вашей системе.*

Прежде чем воспользоваться Cain & Abel, вам придется собрать определенную информацию, включая IP-адрес вашей анализирующей пакеты системы, удаленной системы, откуда вы хотели бы перехватывать сетевой трафик, а также маршрутизатора, от которого удаленная система получает нисходящий поток данных.

Открыв Cain & Abel в первый раз, вы заметите ряд вкладок у верхнего края главного окна. Ведь заражение ARP-кеша — это лишь одна из функциональных возможностей Cain & Abel. Для рассматриваемых здесь целей выберите вкладку **Sniffer** (Анализатор пакетов). Щелкнув на этой вкладке, вы должны увидеть пустую таблицу (рис. 2.11).

Чтобы заполнить эту таблицу, вам придется активизировать встроенный в Cain & Abel анализатор пакетов и просканировать хосты в своей сети. С этой целью выполните следующие действия.

1. Щелкните на второй слева пиктограмме с изображением сетевого адаптера на панели инструментов.
2. Вам будет предложено выбрать интерфейс для анализа пакетов. Выберите интерфейс, подключенный к той сети, где вы будете выполнять заражение ARP-кеша. Если это ваша первая попытка воспользоваться средствами Cain & Abel, выберите этот интерфейс и щелкните на кнопке **OK**. А если вы выбирали интерфейс в Cain & Abel прежде, то результат вашего выбора был сохранен, и вам нужно лишь щелкнуть на пиктограмме с изображением сетевого адаптера еще раз, чтобы выбрать подходящий интерфейс. (Убедитесь, что кнопка с этой пиктограммой находится в нажатом состоянии, чтобы активизировать встроенный в Cain & Abel анализатор пакетов.)

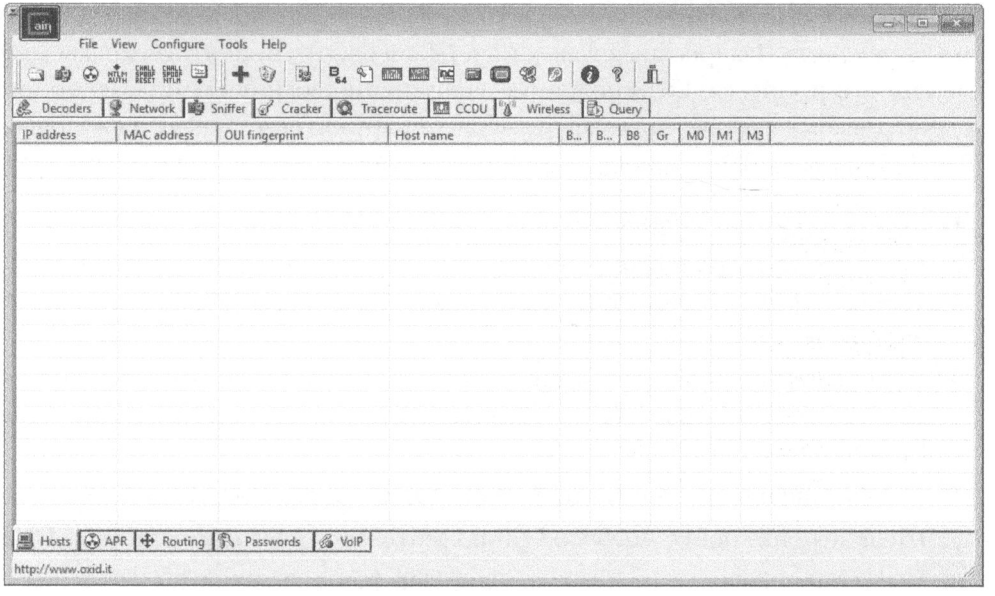


Рис. 2.11. Вкладка Sniffer *в главном окне Cain & Abel*

3. Чтобы составить список хостов, имеющихся в вашей сети, щелкните на кнопке со знаком "плюс" (+). Откроется диалоговое окно MAC Address Scanner (Сканер MAC-адресов), как показано на рис. 2.12. В этом окне должен быть выбран переключатель All hosts in my subnet (Все хосты в подсети), либо вы можете указать диапазон адресов. Щелкните на кнопке OK, чтобы продолжить дальше.

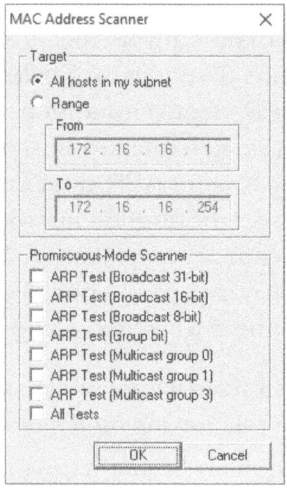


Рис. 2.12. Сканирование MAC-адресов средствами Cain & Abel для обнаружения узлов сети

Некоторые пользователи Windows 10 жалуются, что Cain & Abel не в состоянии определить IP-адрес их сетевых интерфейсов, что препятствует завершению данного процесса. Если подобное затруднение возникнет и у вас, то при настройке своих сетевых интерфейсов вы обнаружите, что их IP-адрес равен `0.0.0.0`. Чтобы выйти из этого затруднения, выполните следующие действия.

1. Если инструментальное средство Cain & Abel открыто, закройте его.
2. Введите **`ncpa.cpl`** в строке поиска на рабочем столе операционной системы, чтобы открыть диалоговое окно Network Connections (Сетевые подключения).
3. Щелкните сначала правой кнопкой мыши на сетевом интерфейсе, из которого вы собираетесь выполнять анализ пакетов, а затем на кнопке Properties (Свойства).
4. Дважды щелкните на опции Internet Protocol Version 4 (TCP/IPv4).
5. Щелкните на кнопке Advanced (Дополнительно) и выберите вкладку DNS.
6. Установите флажок рядом с меткой Use this connection's DNS suffix in DNS registration (Использовать DNS-суффикс для подключения при регистрации в DNS), чтобы активизировать соответствующий режим.
7. Щелкните на кнопке OK, чтобы выйти из открытых диалоговых окон, а затем перезапустите Cain & Abel.

В итоге таблица должна заполниться списком всех хостов, подключенных к вашей сети, наряду с их MAC-адресами, IP-адресами и сведениями о производителях. Именно с этим списком вам придется работать при настройке заражения ARP-кеша.

У нижнего края рабочего окна Cain & Abel должен появиться ряд вкладок для перехода в другие окна под заголовком Sniffer. Итак, составив список хостов, перейдите на вкладку APR, чтобы приступить к работе по методике заражения ARP-кеша в соответствующем диалоговом окне.

В открывшемся диалоговом окне APR появятся две пустые таблицы. По окончании приведенных ниже действий по настройке в верхней таблице появятся устройства, задействованные в заражении ARP-кеша, а в нижней таблице – весь обмен данными между зараженными вами машинами.

Чтобы настроить заражение ARP-кеша, выполните следующие действия.

1. Щелкните сначала на пустом участке в верхней части экрана, а затем на кнопке со знаком "плюс" (+) стандартной панели инструментов Cain & Abel.
2. В открывшемся окне появятся две панели выбора. Слева вы увидите список всех хостов, имеющихся в вашей сети. Если вы щелкнете на IP-адресе целевого компьютера, то в панели справа появится список

всех хостов в вашей сети, кроме хоста, имеющего IP-адрес целевого компьютера.

3. Щелкните сначала в правой панели на IP-адресе маршрутизатора, непосредственно направляющего поток данных, исходящий из целевого компьютера (рис. 2.13), а затем на кнопке OK. В итоге IP-адреса обоих устройств должны появиться в верхней таблице в главном окне прикладной программы.
4. Чтобы завершить процесс, щелкните на черно-желтом знаке радиации, что на стандартной панели инструментов. Это приведет к активизации средств Cain & Abel для заражения ARP-кеша и позволит вашей анализирующей пакеты системе стать посредником во всех обменах данными между целевой системой и маршрутизатором восходящего потока.

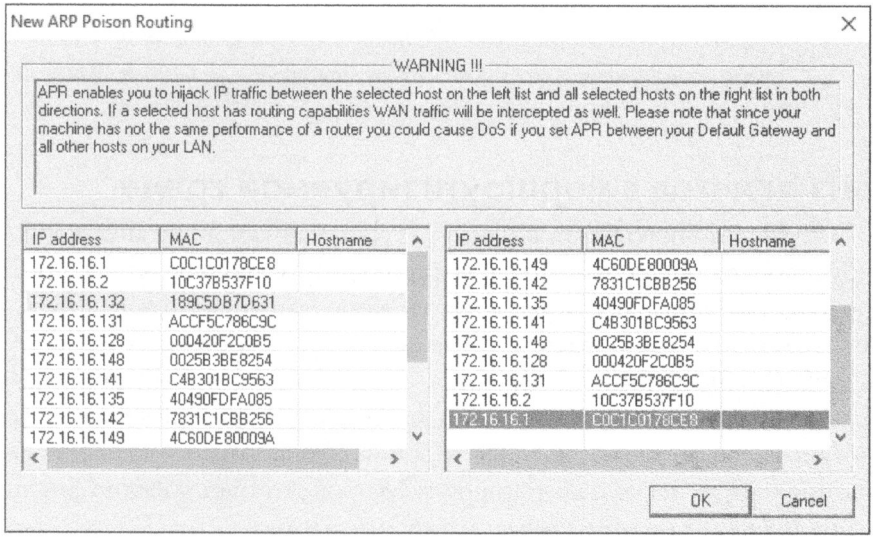


Рис. 2.13. Выбор устройств, для которых требуется активизировать заражение ARP-кеша

Теперь вы сможете запустить свой анализатор пакетов и приступить к процессу их анализа. Завершив перехват сетевого трафика, щелкните еще раз на черно-желтом знаке радиации, чтобы прекратить заражение ARP-кеша.

Предупреждение о заражении ARP-кеша

В качестве заключительного замечания по поводу заражения ARP-кеша хочется сказать, что вы должны учитывать назначение тех систем, для которых реализуется данный процесс. Данную методику не следует, в частности, применять для устройств с очень высокой степенью использования в сети.

Примером тому служит файловый сервер, имеющий канал связи с сетью на скорости 1 Гбит/с, особенно когда система анализа пакетов подключена на скорости только 100 Мбит/с.

Когда сетевой трафик перенаправляется по методике, представленной в данном примере, весь трафик, передаваемый и получаемый целевой системой, должен сначала пройти через систему анализа пакетов, а следовательно, она станет узким местом в процессе обмена данными. Такая перемаршрутизация может иметь эффект вроде отказа в обслуживании на анализируемой машине, что в конечном счете приведет к снижению производительности сети и получению ложных данных анализа. Перегрузка по трафику может также воспрепятствовать нормальному обмену данными по сетевому протоколу SSL.

ПРИМЕЧАНИЕ *Чтобы исключить прохождение всего сетевого трафика через вашу систему анализа пакетов, воспользуйтесь таким средством, как асимметричная маршрутизация. Подробнее с этой методикой можно ознакомиться в разделе APR руководства пользователя Cain & Abel (*`http://www.oxid.it/ca_um/topics/apr.htm`*).*

Анализ пакетов в маршрутизируемой среде

Все методики подключения к коммутируемым сетям доступны и для маршрутизируемых сетей. Для работы в маршрутизируемых средах следует лишь обращать особое внимание на местоположение анализатора пакетов, когда устраняются неполадки, охватывающие несколько сегментов сети.

Как вам должно быть уже известно, широковещательный домен целевого устройства простирается вплоть до маршрутизатора, где сетевой трафик передается следующему маршрутизатору восходящего потока. Если данные должны проходить несколько маршрутизаторов, то очень важно анализировать сетевой трафик со всех сторон маршрутизатора.

В качестве примера рассмотрим затруднение, которое может возникнуть в сети с несколькими сегментами, соединенными через ряд маршрутизаторов. В такой сети каждый сегмент связывается с восходящим сегментом для сохранения и извлечения данных. Затруднение, которое мы пытаемся разрешить, состоит в том, что нисходящая подсеть (т.е. сеть Г на рис. 2.14) не может связаться ни с одним из устройств в сети А.

Если проанализировать трафик устройства в сети Г, испытывающего трудности связывания с устройствами в других сетях, можно явно обнаружить данные, передаваемые в другой сегмент сети, но не обнаружить данные, поступающие обратно. Если же изменить положение анализатора пакетов и приступить к анализу сетевого трафика в следующем восходящем сегменте сети (т.е. в сети Б), то можно обнаружить, что сетевой трафик пропускается или

неверно направляется маршрутизатором сети Б. В конечном итоге это приведет к проблеме настройки маршрутизатора, разрешив которую можно разрешить и более крупную дилемму. И хотя рассматриваемый здесь случай выглядит несколько общим, из него можно сделать следующий важный вывод: когда приходится иметь дело с несколькими маршрутизаторами и сетевыми сегментами, анализатор пакетов придется устанавливать в разных местах сети, чтобы получить полную картину происходящего и точно определить возникшее в ней затруднение.

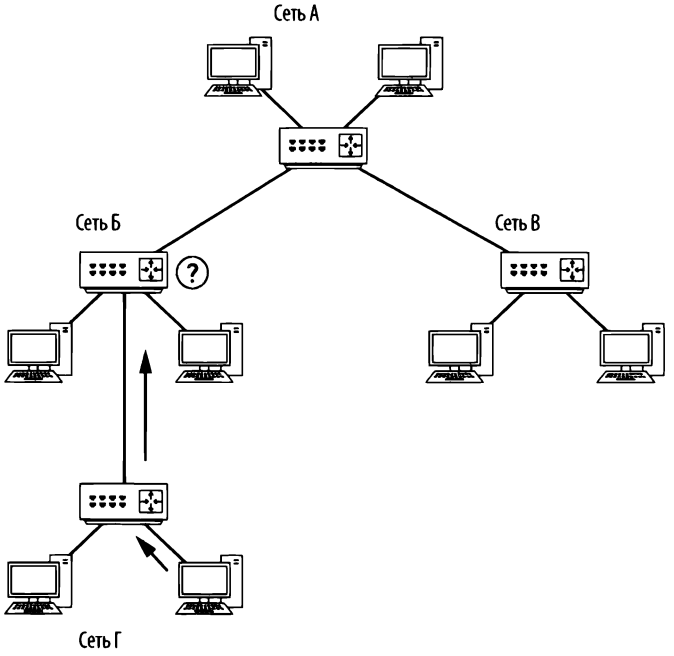


Рис. 2.14. Компьютер в сети Г не может связаться с компьютерами в сети А

КАРТЫ СЕТИ

Обсуждая расположение анализатора пакетов в сети, мы исследовали несколько так называемых карт сети. *Карта*, или *схема*, сети показывает все технические ресурсы в сети и порядок их соединения.

Чтобы определить место для расположения анализатора пакетов в сети, лучше всего представить ее наглядно. Если у вас имеется карта сети, держите ее под рукой, поскольку она является весьма ценным ресурсом для процесса диагностики и анализа. Для этого вам, возможно даже, придется составить подробную карту своей сети. Помните, что иногда половину дела в устранении неисправностей обеспечивает сбор надлежащих данных.

Размещение анализатора пакетов на практике

Мы рассмотрели четыре способа перехвата сетевого трафика в коммутируемой среде. К ним можно добавить еще один, для чего достаточно рассмотреть установку приложения для анализа пакетов на одном устройстве, из которого требуется перехватывать сетевой трафик (такая методика называется *прямой установкой*). Из этих пяти способов не так-то просто выбрать наиболее подходящий. Поэтому в табл. 2.2 сведены основные положения по каждому способу и соответствующей методике анализа пакетов.

Исследователям пакетов необходимо действовать как можно более скрытно. В идеальном случае требующиеся данные следует собирать, не оставляя следов. Подобно тому, как следователи-криминалисты не должны портить место преступления, так и аналитикам пакетов не следует портить перехватываемый сетевой трафик.

Таблица 2.2. Основные положения по анализу пакетов в коммутируемой среде

Методика	Основные положения
Зеркальное отображение портов	• Не оставляет следы и не формирует дополнительные пакеты • Можно настроить, не отключая клиента от сети, что удобно при зеркальном отображении портов маршрутизатора или сервера • Задействует ресурсы коммутатора для целей обработки и может быть неприемлемой при высоком трафике
Перехват пакетов через концентратор	• Подходит в тех случаях, когда хост можно безболезненно отключить от сети • Не дает желаемого эффекта, когда требуется перехватить сетевой трафик из нескольких хостов, поскольку в этом случае вполне возможны коллизии и потеря пакетов • Может привести к потере пакетов в современных хостах, работающих на скорости 100–1000 Мбит/с, поскольку большинство настоящих концентраторов работают на скорости лишь 10 Мбит/с
Применение сетевого ответвителя	• Идеально подходит в том случае, если хост можно безболезненно отключить от сети • Единственная возможность, когда требуется проанализировать сетевой трафик по оптоволоконному соединению • Наиболее предпочтительное решение для перехвата пакетов и непрерывного контроля сети в корпоративной среде, поскольку сетевые ответвители весьма надежны и допускают наращивание до масштабов каналов связи с высокой пропускной способностью • Сетевые ответвители предназначены для решения насущных задач и вполне соответствуют уровням скоростей в современных сетях, что выгодно отличает данную методику от перехвата пакетов через концентратор • Может обойтись недешево, особенно в крупных масштабах, а следовательно, применять данную методику невыгодно

Окончание табл. 2.2

Методика	Основные положения
Заражение ARP-кеша	• Считается весьма нестабильной методикой, поскольку требует внедрения пакетов в сеть с целью перенаправить сетевой трафик через анализатор пакетов • Если зеркальное отображение портов невозможно, то может оказаться эффективной методикой для быстрого перехвата сетевого трафика из целевого устройства, не отключая его от сети • Требует особой аккуратности, чтобы не оказывать влияния на функционирование сети
Прямая установка	• Обычно не рекомендуется, ведь если возникнут трудности в работе хоста, они могут привести к потере пакетов или такому манипулированию ими, что они не будут представлены точно • От сетевого адаптера хоста не требуется поддержки смешанного режима работы • Лучше всего подходит для тестовых сред, исследования или сравнения с исходными характеристиками производительности, анализа файлов перехвата, создаваемых в других местах сети

Перейдя к практическим сценариям в последующих главах, мы обсудим наилучшие способы перехвата требующихся данных в каждом конкретном случае. А пока обратите внимание на рис. 2.15, на котором приведена блок-схема, призванная помочь вам выбрать наилучшую методику для перехвата сетевого трафика в конкретной ситуации. На этой блок-схеме во внимание приняты самые разные факторы, начиная с места для перехвата пакетов: дома или на работе, но она служит лишь в качестве общей справки и не охватывает все возможные сценарии подключения к сети с целью прослушивания и анализа пакетов.

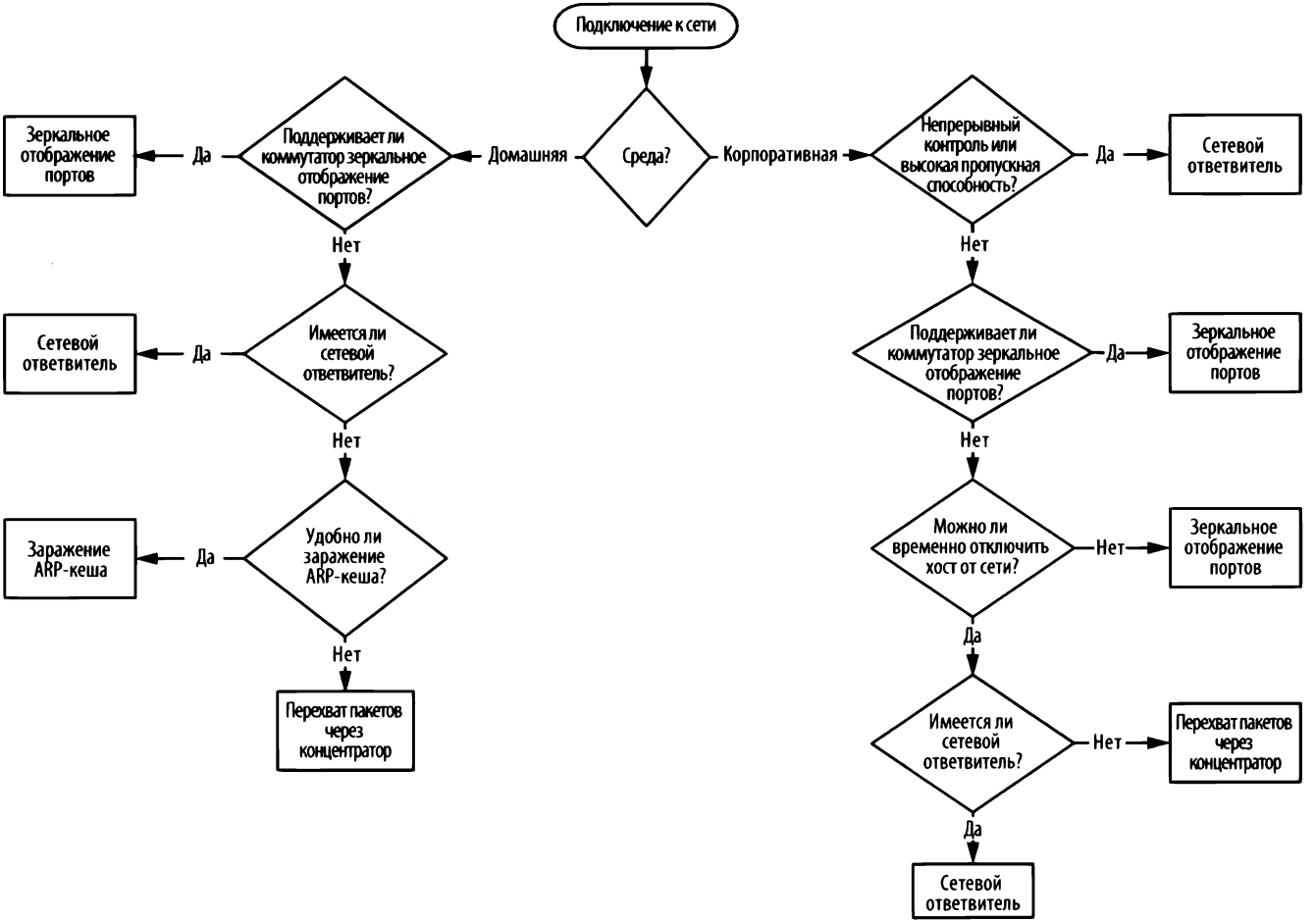


Рис. 2.15. Блок-схема, оказывающая помощь в выборе наилучшей методики для подключения к сети с целью прослушивания и анализа пакетов

3

ВВЕДЕНИЕ В WIRESHARK

Как упоминалось в главе 1, "Анализ пакетов и основы организации сетей", для анализа сетевых пакетов имеется несколько приложений, но в этой книге основное внимание уделяется приложению Wireshark. А в этой главе представлено введение в Wireshark.

Краткая история создания Wireshark

У приложения Wireshark очень богатая история. Джеральд Комбс (Gerald Combs), окончивший курс вычислительной техники в университете штата Миссури, расположенном в Канзас-Сити, разработал это приложение по необходимости. Первая версия его приложения была выпущена под названием Ethereal в 1998 году по универсальной общедоступной лицензии GNU (GPL).

Через восемь лет после выпуска версии Ethereal Комбс оставил свою работу в поисках других карьерных возможностей. К сожалению, у его тогдашнего работодателя остались все права на торговую марку Ethereal, и поэтому Комбсу не удалось добиться согласия на получение контроля над фирменным названием Ethereal. Поэтому Комбс и остальные члены команды разработчиков переименовали свой проект на *Wireshark* в середине 2006 года.

Ныне популярность приложения Wireshark существенно возросла, а в его разработке приняло участие более 500 специалистов. В то же время приложение, существующее под названием Ethereal, больше не разрабатывается.

Преимущества Wireshark

Приложение Wireshark дает ряд преимуществ, благодаря которым оно оказывается весьма привлекательным для повседневного применения. Оно рассчитано на разные категории аналитиков пакетов: от начинающих до опытных, предоставляя заманчивые возможности как для тех, так и для других. Итак, исследуем возможности Wireshark по критериям, определенным в главе 1, "Анализ пакетов и основы организации сетей", для выбора инструментальных средств анализа пакетов.

- **Поддержка сетевых протоколов.** Wireshark отличается поддержкой целого ряда сетевых протоколов – на момент написания этой книги их насчитывалось около 1000. К числу поддерживаемых сетевых протоколов относятся как общеупотребительные протоколы вроде IP и DHCP, так и более развитые специализированные протоколы вроде DNP3 и BitTorrent. А поскольку приложение Wireshark разработано по модели открытого исходного кода, то при каждом его обновлении вводится поддержка нового сетевого протокола.

ПРИМЕЧАНИЕ *В том маловероятном случае, если требующийся вам сетевой протокол не поддерживается в Wireshark, можете запрограммировать его поддержку сами. После этого передайте свой исходный код разработчикам Wireshark на рассмотрение, чтобы включить его в данное приложение. Узнать о том, что требуется для участия со своим исходным кодом в проекте Wireshark, можно по адресу* `https://www.wireshark.org/develop.html`.

- **Удобство для пользователей.** Интерфейс Wireshark – самый простой для усвоения среди всех приложений для анализа пакетов. Он создан на основе графического интерфейса с ясно составленными контекстными меню и простой компоновкой. Кроме того, в нем предоставляется ряд средств, предназначенных для повышения удобства его практического применения, в том числе выделение сетевых протоколов разным цветом и подробное графическое представление исходных данных. В отличие от некоторых более сложных приложений, работающих в режиме командной строки (например, утилиты `tcpdump`), интерфейс приложения Wireshark вполне доступен тем, кто делает только первые шаги в области анализа пакетов.
- **Стоимость.** Приложение Wireshark совершенно бесплатно, поскольку оно выпущено по универсальной общедоступной лицензии GNU (GPL). Его можно свободно загрузить и применять в любых целях: как в личных, так и в коммерческих.

ПРИМЕЧАНИЕ *Несмотря на то что приложение Wireshark может быть получено бесплатно, некоторые по ошибке внесли за него плату. Если вы ищете анализаторы пакетов на электронном аукционе eBay, вас может удивить, сколько людей готовы продать "профессиональную корпоративную лицензию" на Wireshark всего лишь за 39,95 долларов. Если вы действительно решите приобрести такую лицензию, свяжитесь со мной, и мы сможем обсудить выгодные варианты приобретения недвижимости с видом на океан, которые я мог бы предложить вам у себя в Кентукки!*

- **Поддержка программы.** Уровень поддержки программного обеспечения может решить его судьбу. Свободно распространяемое программное обеспечение вроде Wireshark может и не иметь никакой формальной поддержки. Поэтому сообщество разработчиков программного обеспечения с открытым исходным кодом нередко опирается на свою базу пользователей для оказания помощи. К счастью для нас, сообщество разработчиков Wireshark относится к числу самых активных. В частности, ссылки на веб-сайте, посвященном приложению Wireshark, направляют непосредственно к нескольким формам поддержки, включая оперативно доступную документацию, вики-страницы, часто задаваемые вопросы и место для подписки на список рассылки, который контролируется большинством главных разработчиков Wireshark. Имеется также платная поддержка Wireshark со стороны компании Riverbed Technology.
- **Доступ к исходному коду.** Wireshark относится к категории программного обеспечения с открытым исходным кодом, который доступен в любой момент. Это может оказаться удобным для поиска и устранения неисправностей в данном приложении, понимании принципа действия дешифраторов сетевых протоколов или внесения своего вклада в разработку Wireshark.
- **Поддержка операционных систем.** В приложении Wireshark поддерживаются все основные современные операционные системы, включая Windows, Linux и Mac OS X. Полный перечень поддерживаемых операционных систем можно посмотреть на начальной странице веб-сайта, посвященного приложению Wireshark.

Установка Wireshark

Процесс установки Wireshark удивительно прост. Но прежде чем установить Wireshark, убедитесь, что ваша система отвечает следующим требованиям.

- Любой современный 32- или 64-разрядный ЦП типа x86.
- 400 Мбайт доступной оперативной памяти, хотя для крупных файлов перехвата потребуется больший объем памяти.
- Не меньше 300 Мбайт на жестком диске плюс место для файлов перехвата.
- Сетевой адаптер, поддерживающий комбинированный (promiscuous mode) режим работы.
- Драйвер перехвата WinPcap/libpcap.

Драйвер перехвата WinPcap является реализованным в Windows вариантом pcap — интерфейса API для перехвата пакетов. Проще говоря, этот драйвер взаимодействует с операционной системой для перехвата исходных данных пакета, применения фильтров и переключения сетевого адаптера в комбинированный режим, и обратно.

Несмотря на то что драйвер WinPcap можно загрузить отдельно (по адресу `http://www.winpcap.org/`), его, как правило, лучше устанавливать из дистрибутивного пакета Wireshark, поскольку версия драйвера WinPcap, входящая в этот пакет, уже проверена на работоспособность вместе с Wireshark.

Установка в системах Windows

Текущая версия Wireshark проверена на работоспособность в разных версиях ОС Windows, которые поддерживаются до сих пор их производителем. На момент написания этой книги к поддерживаемым относились версии Windows Vista, Windows 7, Windows 8, Windows 10, а также Windows Server 2003, 2008 и 2012. И хотя Wireshark зачастую будет работать и в других версиях Windows (вроде Windows XP), официально эти версии не поддерживаются.

Для установки Wireshark в ОС Windows прежде всего необходимо получить последнюю версию сборки установочного пакета с официального веб-сайта Wireshark по адресу `http://www.wireshark.org/`. С этой целью перейдите в раздел **Download** (Загрузки) этого веб-сайта и выберите нужный вариант установочного пакета в зависимости от применяемой версии Windows. А после загрузки установочного пакета выполните следующие действия.

1. Дважды щелкните на файле с расширением **.exe**, чтобы начать установку, а затем щелкните на кнопке Next (Далее) во вступительном окне.
2. Прочитайте лицензионное соглашение и, если согласны с ним, щелкните на кнопке I Agree (Соглашаюсь).
3. Выберите компоненты Wireshark, которые требуется установить, как показано на рис. 3.1. Для своих целей можете принять выбранные по умолчанию компоненты и щелкнуть на кнопке Next.

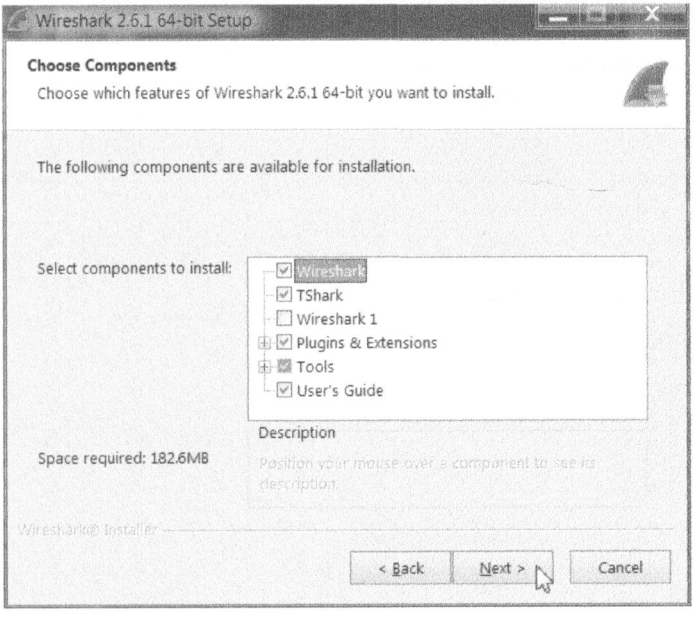


Рис. 3.1. Выберите компоненты Wireshark, которые требуется установить

4. Щелкните на кнопке Next в открывшемся окне Select Additional Tasks (Выбор дополнительных задач).
5. Выберите место для установки Wireshark и щелкните на кнопке Next.
6. Когда появится диалоговое окно, в котором запрашивается установка драйвера WinPcap, установите сначала флажок Install WinPcap (Установить драйвер WinPcap), как показано на рис. 3.2, а затем щелкните на кнопке Install. В итоге должен начаться процесс установки.
7. Далее вам представится возможность установить USBPcap — утилиту для сбора данных из USB-устройств. Установите соответствующий флажок, если желаете установить эту утилиту, а затем щелкните на кнопке Next.
8. Где-то посредине процесса установки Wireshark должна начаться установка драйвера WinPcap. И как только это произойдет, щелкните на кнопке Next во вступительном окне, прочитайте лицензионное соглашение и, если согласны с ним, щелкните на кнопке I Agree.
9. В итоге выбранный драйвер WinPcap и утилита USBPcap должны быть установлены на вашем компьютере. По окончании данной установки щелкните на кнопке Finish (Готово).
10. На этом установка Wireshark должна быть завершена. И как только это произойдет, щелкните на кнопке Next.
11. Щелкните на кнопке Finish в открывшемся окне подтверждения установки Wireshark.

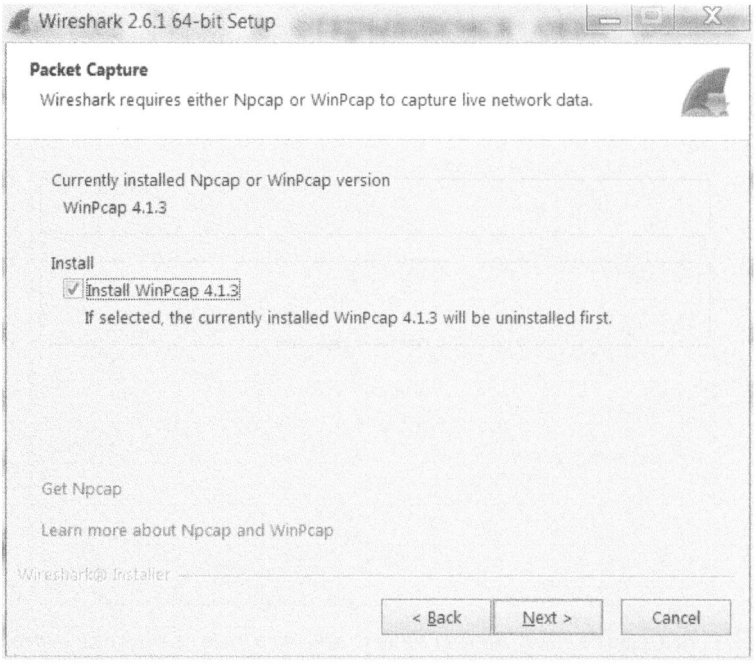


Рис. 3.2. Выбор варианта установки драйвера WinPcap

Установка в системах Linux

Приложение Wireshark работает на большинстве современных платформ, построенных на основе ОС Unix. Его можно установить с помощью одного из диспетчеров дистрибутивных пакетов или же загрузив и установив дистрибутивный пакет, подходящий для вашей операционной системы. Было бы нереально охватить все процедуры установки в каждой версии ОС Linux, поэтому рассмотрим лишь некоторые из них.

Как правило, для установки системного программного обеспечения в ОС Unix требуется права доступа суперпользователя (`root`). Но для установки локальных версий программного обеспечения, скомпилированных из исходного кода, права суперпользователя не требуется.

Системы на основе RPM-пакетов

Если вы пользуетесь версией Red Hat Linux или основанным на ней дистрибутивом вроде CentOS, то в этой ОС, скорее всего, имеется устанавливаемое по умолчанию инструментальное средство Yum для управления пакетами. В таком случае для быстрой установки приложения Wireshark вам достаточно извлечь его из хранилища дистрибутивного программного обеспечения. С этой целью откройте окно терминала и введите в нем следующую команду:

```
$ sudo yum install wireshark
```

Если потребуются какие-нибудь дополнительные зависимости, вам будет предложено установить и их. Если же все пройдет удачно, вы сможете запустить приложение Wireshark на выполнение из командой строки и получить к нему доступ через графический интерфейс.

Системы на основе DEB-пакетов

В состав большинства дистрибутивов на основе DEB-пакетов, например Debian или Ubuntu, входит инструментальное средство APT для управления пакетами. Оно позволяет установить Wireshark из хранилища программного обеспечения ОС. Чтобы установить Wireshark с помощью этого инструментального средства, откройте окно терминала и введите в нем следующую команду:

```
$ sudo apt-get install wireshark wireshark-qt
```

И в этом случае вам будет предложено установить любые зависимости, чтобы завершить процесс установки.

Компиляция исходного кода Wireshark

В силу изменений в архитектуре операционной системы и функциональных средствах Wireshark инструкции по компиляции исходного кода Wireshark могут со временем меняться. И это одна из причин, по которым рекомендуется пользоваться диспетчером пакетов операционной системы для установки данного приложения. Но если в вашем дистрибутиве Linux не применяется программное обеспечение для автоматического управления пакетами или вам требуется специальная установка Wireshark, то у вас есть возможность установить данное приложение вручную, скомпилировав его исходный код. С этой целью выполните следующие действия.

1. Загрузите исходный пакет с соответствующей страницы веб-сайта, посвященного Wireshark по указанному ранее адресу.
2. Извлеките архив, введя следующую команду и подставив в ней соответствующее имя файла загруженного пакета:

```
$ tar -jxvf <укажите здесь имя файла пакета>.tar.bz2
```

3. Прежде чем выбирать конфигурацию и устанавливать Wireshark, возможно, придется установить ряд дополнительных пакетов в зависимости от выбранной вами версии Linux. Например, в Ubuntu 14.04 требуется установить ряд дополнительных пакетов для нормальной работы

Wireshark. И это можно сделать по приведенной ниже команде. Но для этого потребуются права доступа суперпользователя, а иначе придется сначала ввести команду **sudo**.

```
$ sudo apt-get install pkg-config bison flex qt5-default libgtk-3-dev
  libpcap-dev qttools5-dev-tools
```

4. Установив требуемые дополнительные пакеты, перейдите к тому каталогу, в который были извлечены исходные файлы Wireshark.
5. Настройте исходный код на правильную его сборку в вашем дистрибутиве Linux по команде **./configure**. Если вы желаете изменить стандартные параметры установки, укажите их на данной стадии установки. Если же какие-нибудь зависимости (т.е. дополнительные пакеты) отсутствуют, то компиляция исходного кода, скорее всего, завершится с ошибкой. Поэтому установите и настройте отсутствующие зависимости, прежде чем продолжить дальше. И если настройка пройдет удачно, вы увидите сообщение, извещающее об этом, как показано на рис. 3.3.

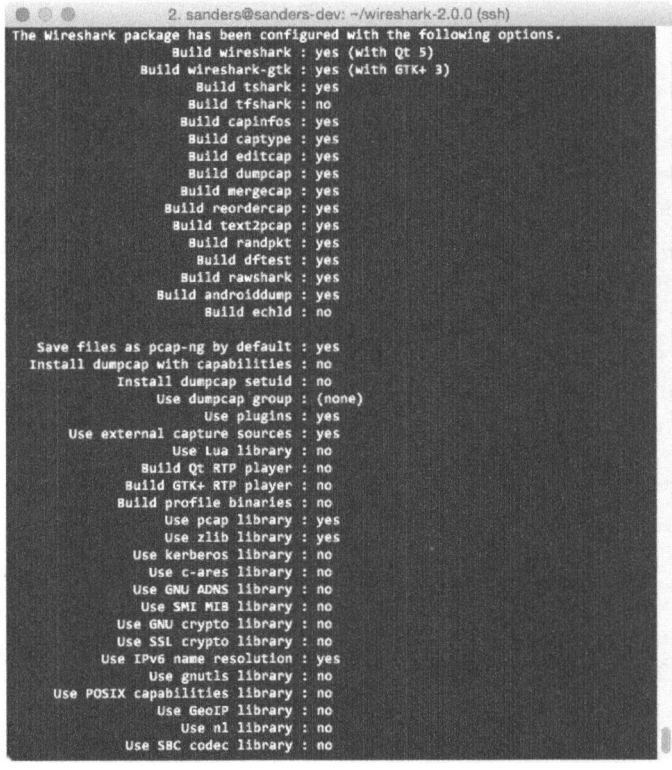


*Рис. 3.3. При удачном завершении команды **./configure** появится приведенное здесь сообщение с выбранными настройками*

6. Введите команду **make**, чтобы скомпилировать исходный код и собрать двоичный исполняемый файл.
7. Запустите завершающую стадию установки по команде **sudo make install**.
8. Выполните команду **sudo /sbin/ldconfig**, чтобы завершить процесс установки.

ПРИМЕЧАНИЕ *Если при выполнении описанных выше действий возникнут какие-нибудь ошибки, возможно, придется установить еще один дополнительный пакет.*

Установка в системах Mac OS X

Чтобы установить Wireshark в одной из систем Mac OS X, выполните следующие действия.

1. Загрузите установочный пакет для Mac OS X с соответствующей страницы веб-сайта, посвященного Wireshark по указанному ранее адресу.
2. Запустите на выполнение утилиту в виде мастера установки и далее следуйте указанным в нем инструкциям. Приняв условия лицензии конечного пользователя, выберите место для установки Wireshark.
3. Завершите процесс установки в соответствующем мастере.

Основы работы в Wireshark

Установив успешно приложение Wireshark в своей системе, можете приступить к его изучению. Открыв полностью функционирующий анализатор пакетов, вы не увидите в нем практически ничего интересного! А дело в том, что для анализа пакетов Wireshark требуются какие-нибудь данные.

Первый перехват пакетов

Чтобы ввести данные пакетов в Wireshark, вам придется сделать свой первый перехват пакетов. И здесь у вас может возникнуть следующий вопрос: "Как же мне перехватывать пакеты, если в моей сети все в порядке?"

Во-первых, в сети *всегда* что-нибудь да не так. Если не верите, пошлите сообщение по электронной почте всем пользователям своей сети, известив их, что все работает идеально. Во-вторых, для анализа пакетов совсем не обязательно, чтобы в сети было что-нибудь не так. На самом деле аналитики пакетов большую часть своего рабочего времени тратят на анализ исправного, а не проблемного сетевого трафика. Ведь нужно же иметь какое-то исходное основание для сравнения, чтобы эффективно исправить сетевой трафик. Так,

если вы надеетесь разрешить затруднение, связанное с сетевым протоколом DHCP, проанализировав его трафик, то должны знать, каким образом выглядит поток рабочего трафика по протоколу DHCP.

В более широком смысле, чтобы найти аномалии в повседневной работе сети, необходимо знать, каким образом выглядит эта повседневная работа. Если ваша сеть работает бесперебойно, то ваши наблюдения за ней послужат исходным основанием для представления сетевого трафика в его нормальном состоянии.

Итак, перехватите немного пакетов, выполнив следующие действия:

1. Откройте Wireshark.
2. Выберите команду Capture⇨Options (Перехват⇨Параметры) из главного меню. В итоге должно появиться диалоговое окно, где перечислены различные сетевые интерфейсы, которые могут быть использованы для перехвата пакетов, а также самые основные сведения о каждом из них (рис. 3.4). Обратите внимание на столбец Traffic, в котором приведен линейный график, наглядно показывающий объем сетевого трафика, проходящего в настоящий момент через данный интерфейс. Пиковые точки на этом графике фактически указывают на перехват пакетов. Если они отсутствуют, линейный график должен быть плоским. Кроме того, каждый интерфейс можно расширить, щелкнув на стрелке слева от него, чтобы увидеть связанную с ним информацию об адресации, например, MAC-адрес или IP-адрес.

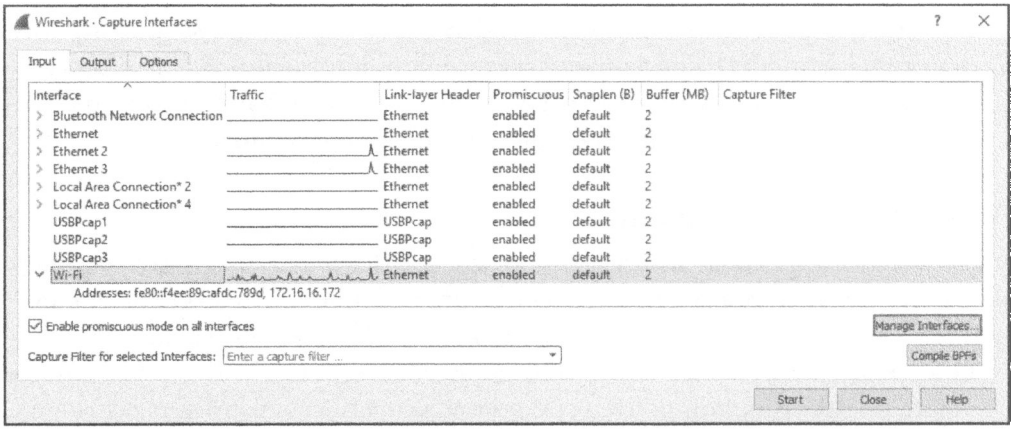


Рис. 3.4. Выбор сетевого интерфейса для перехвата пакетов

3. Щелкните сначала на том сетевом интерфейсе, которым вы хотели бы воспользоваться, а затем на кнопке Start (Пуск). Текущее окно должно заполниться перехватываемыми данными.

4. Подождите около минуты, и как только будете готовы остановить перехват и просмотреть полученные данные, щелкните на кнопке Stop (Остановка), выбираемой из раскрывающегося меню Capture.

Как только вы выполните описанные выше действия, завершив процесс перехвата, главное окно Wireshark должно заполниться полученными в итоге данными. В действительности объем этих данных может вас ошеломить, но они быстро обретут для вас определенный смысл, стоит вам научиться разбирать содержимое главного окна Wireshark по частям.

Главное окно Wireshark

Большую часть времени вам придется работать в главном окне Wireshark. Именно здесь перехватываемые пакеты отображаются и преобразуются в более удобный для анализа формат. Итак, рассмотрим содержимое главного окна Wireshark, используя сделанный только что перехват, как показано на рис. 3.5.

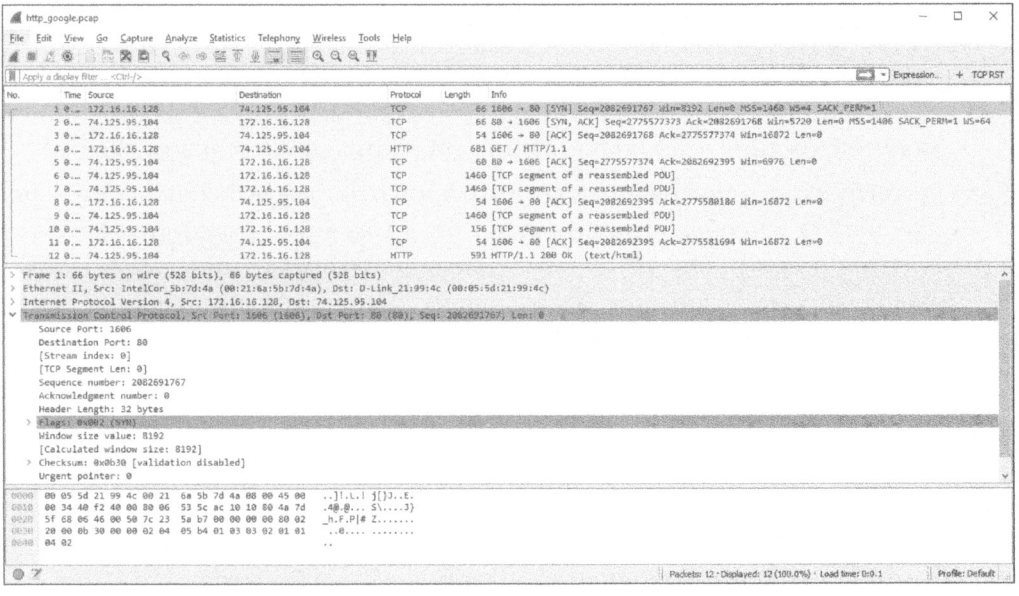


Рис. 3.5. Вид главного окна Wireshark, состоящего из трех панелей

Главное окно Wireshark состоит из панелей Packet List (Список пакетов), Packet Details (Подробные сведения о пакете) и Packet Bytes (Байты из пакетов), которые располагаются сверху вниз и зависят друг от друга. Чтобы просмотреть подробные сведения об отдельном пакете в панели Packet Details, необходимо сначала выбрать этот пакет в панели Packet List. Если же выбрать часть пакета в панели Packet Details, то в панели Packet Bytes появятся отдельные байты, соответствующие данной части пакета.

ПРИМЕЧАНИЕ *На рис. 3.5 обратите внимание на то, что в панели* Packet List *перечислены разные сетевые протоколы. Здесь отсутствует визуальное разделение протоколов на разные уровни, кроме их выделения разным цветом. А все пакеты показаны в том порядке, в каком они получены по сети.*

Ниже приведено краткое описание каждой панели.

- Packet List. Это верхняя панель, в которой отображается таблица, содержащая все пакеты из текущего файла перехвата. Она состоит из столбцов, содержащих номер пакета, относительное время перехвата пакета, адреса источника и получателя пакета, тип сетевого протокола пакета, а также некоторые общие сведения, находящиеся в пакете.

ПРИМЕЧАНИЕ *Здесь и далее под сетевым трафиком подразумеваются все пакеты, отображаемые в панели* Packet List. *А если речь идет только о трафике DNS, то имеются в виду пакеты по протоколу DNS (служба доменных имен), отображаемые в той же самой панели.*

- Packet Details. Это средняя панель, где в иерархическом виде отображаются сведения об одном пакете. Она может быть свернута или развернута для отображения всей информации, собранной об отдельном пакете.
- Packet Bytes. Это нижняя панель, в которой отображаются исходные данные пакета в необработанном виде, т.е. в том виде, в каком пакет переносится по сети. Эти исходные данные не содержат ничего, что упрощало бы их отслеживание. Методики их интерпретации подробнее рассматриваются в приложении Б, "Интерпретация пакетов", к этой книге.

Глобальные параметры настройки Wireshark

Имеется несколько глобальных параметров Wireshark, которые можно настроить под свои нужды. Чтобы получить доступ к глобальным параметрам настройки Wireshark, выберите команду Edit⇨Preferences (Правка⇨Параметры) из главного меню. В итоге откроется диалоговое окно Preferences с несколькими специально настраиваемыми параметрами, как показано на рис. 3.6.

Глобальные параметры настройки Wireshark разбиты на шесть основных разделов и дополнительный раздел Advanced. Ниже приведено краткое описание этих разделов.

- **Appearance (Представление).** В этом разделе находятся глобальные параметры, которые определяют порядок представления данных в Wireshark. Большую часть этих параметров можно изменить, исходя из своих личных предпочтений, включая необходимость сохранять

положение окон, компоновку трех основных панелей, расположение полосы прокрутки и столбцов в панели Packet List, шрифты, которыми выделяются перехваченные данные, а также цвета фона и символов шрифта.

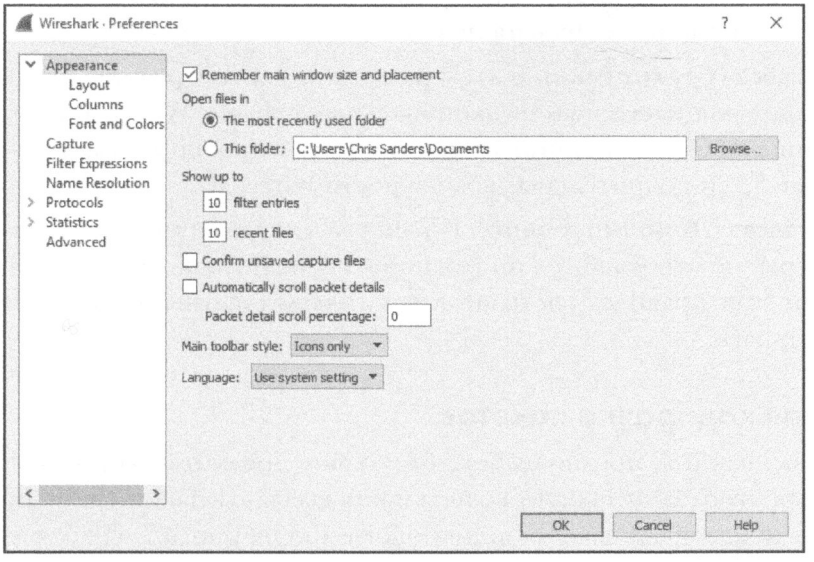


Рис. 3.6. В диалоговом окне Preferences можно специально настроить глобальные параметры Wireshark

- **Capture (Перехват).** В этом разделе находятся глобальные параметры, которые определяют порядок перехвата пакетов, включая стандартный интерфейс для перехвата, необходимость перехода в комбинированный режим по умолчанию и обновления панели Packet List в реальном времени.
- **Filter Expressions (Фильтрующие выражения).** В дальнейшем мы обсудим, каким образом в Wireshark можно фильтровать сетевой трафик по отдельным критериям. А в этом разделе находятся глобальные параметры, которые позволяют создавать фильтры сетевого трафика и манипулировать ими.
- **Name Resolution (Преобразование имен).** С помощью глобальных параметров из этого раздела можно активизировать функциональные средства Wireshark, позволяющие преобразовывать адреса в их более удобные для различения имена, в том числе адреса канального, сетевого и транспортного уровня, а также указывать максимальное количество параллельных запросов на преобразование имен.
- **Protocols (Протоколы).** В этом разделе находятся глобальные параметры, имеющие отношение к перехвату и отображению различных

пакетов, которые Wireshark в состоянии декодировать. Настраиваемые глобальные параметры имеются не для всех сетевых протоколов, но параметры некоторых из них все же можно изменить. Впрочем, эти параметры лучше оставить установленными по умолчанию, если только нет особых причин для их изменения.

- **Statistics (Статистика).** В этом разделе находится ряд глобальных параметров для настройки функциональных средств Wireshark, предназначенных для ведения статистики и более подробно рассматриваемых в главе 5, "Дополнительные возможности Wireshark".
- **Advanced (Дополнительно).** В этом разделе находятся глобальные параметры, не относящиеся ни к одной из перечисленных выше категорий. Они, как правило, настраиваются только опытными пользователями Wireshark.

Цветовая кодировка пакетов

Если вы, как и я, предпочитаете блестящие предметы и приятные цвета, то вас, вероятно, заинтересует возможность выделять пакеты разными цветами в панели Packet List, как, например, показано на рис. 3.7. И хотя это всего лишь черно-белый рисунок в печатном издании, тем не менее, разные оттенки серого на нем дают общее представление о цветовой кодировке пакетов. На первый взгляд может показаться, что пакеты выделяются цветами произвольно, но на самом деле это не так.

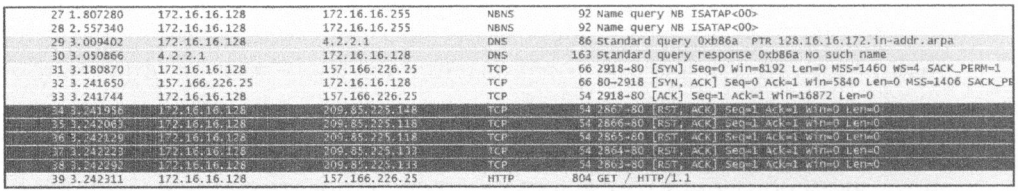


Рис. 3.7. Цветовая кодировка пакетов в Wireshark позволяет быстро распознавать сетевые протоколы

Каждый пакет отображается отдельным цветом по весьма веской причине: цвет может отражать сетевой протокол и значения в отдельных полях пакета. Например, весь трафик по сетевому протоколу UDP выделяется по умолчанию голубым цветом, а весь трафик по сетевому протоколу HTTP – салатовым. Такая цветовая кодировка позволяет быстро распознавать различные сетевые протоколы, не обращаясь к полю протокола в каждом пакете, отображаемом в панели Packet List. Со временем вы сами убедитесь, насколько это экономит время, затрачиваемое на просмотр крупных файлов перехвата.

Цвета, назначенные для каждого сетевого протокола, нетрудно изменить в диалоговом окне Coloring Rules (Правила выделения цветом), как показано на рис. 3.8. Чтобы открыть это окно, выберите команду View⇨Coloring Rules (Вид⇨ Цветовые правила) из главного меню.

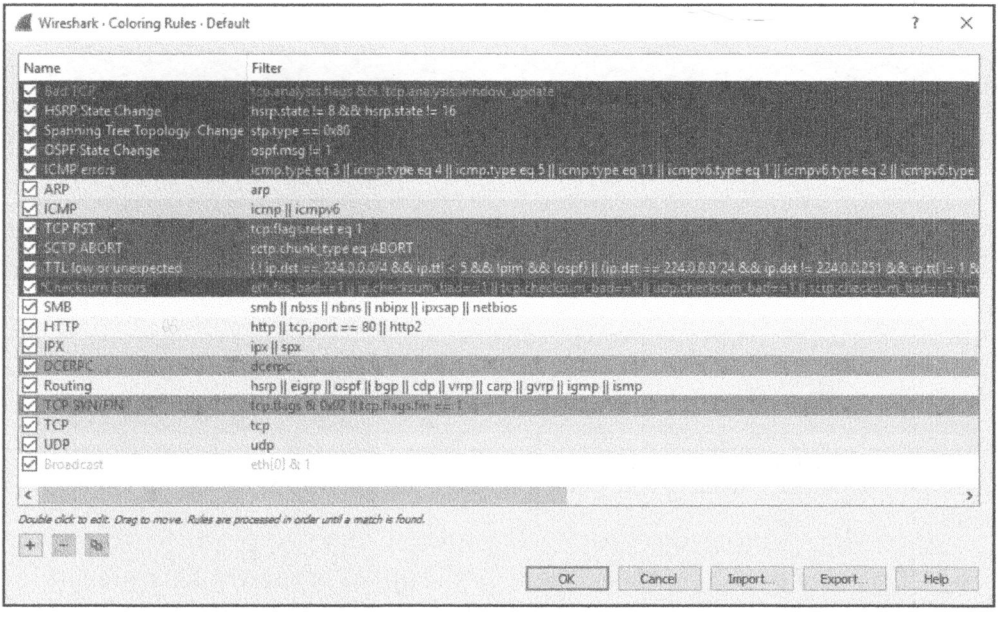


Рис. 3.8. В диалоговом окне Coloring Rules можно просматривать и изменять порядок выделения пакетов разными цветами

Правила выделения цветом основываются на фильтрах, применяемых в Wireshark и подробнее рассматриваемых в главе 4, "Обработка перехваченных пакетов". С помощью этих фильтров можно определить собственные правила выделения цветом или изменить уже существующие. Например, чтобы изменить с салатового на бледно-лиловый цвет фона, которым выделяется сетевой трафик по протоколу HTTP, выполните следующие действия.

1. Откройте Wireshark, а затем диалоговое окно Coloring Rules (по команде View⇨Coloring Rules).
2. Найдите правило выделения цветом сетевого протокола HTTP в списке подобных правил и выберите его, щелкнув на нем кнопкой мыши.
3. В нижней части экрана появятся кнопки выбора цветов символов и фона, как показано на рис. 3.9.
4. Щелкните на кнопке Background (Фон).
5. Выберите требующийся цвет на палитре цветов и щелкните на кнопке OK.

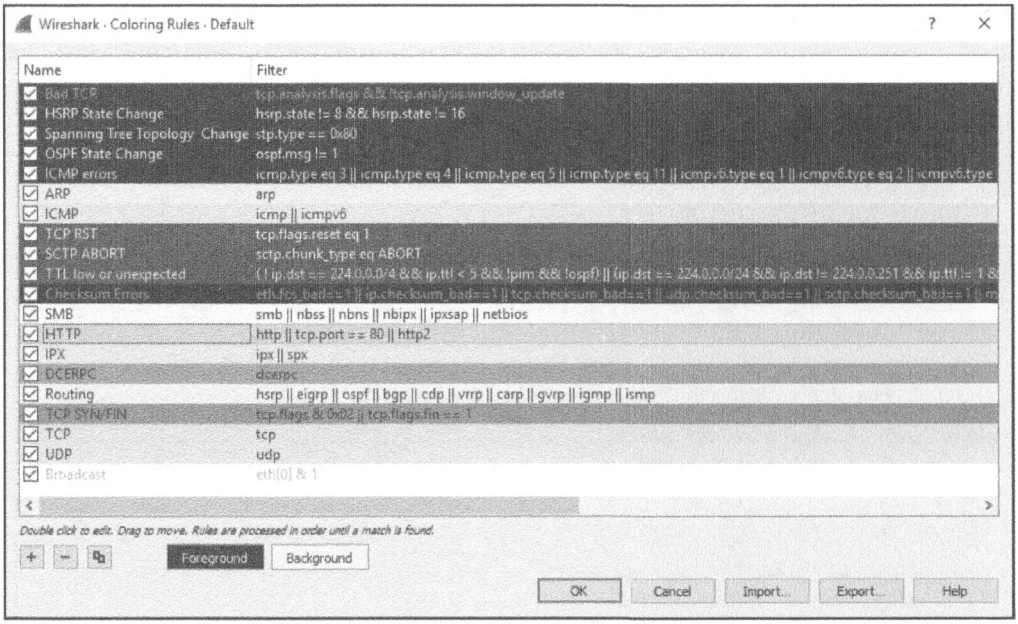


Рис. 3.9. Редактируя фильтр выделения цветом, можно изменить цвета символов и фона

6. Щелкните на кнопке OK еще раз, чтобы принять внесенные изменения и вернуться в главное окно. В итоге пользовательский интерфейс должен перезагрузиться, чтобы отразить обновленную цветовую схему.

В ходе работы со своей сетью в Wireshark вы постепенно начнете замечать, что с одними сетевыми протоколами вам приходится иметь дело чаще, чем с другими. Именно здесь и приходит на помощь цветовая кодировка пакетов, упрощающая их анализ. Так, если вы считаете, что в вашей сети имеется непослушный DHCP-сервер, самовольно назначающий IP-адреса, можете изменить правила выделения цветом сетевого протокола DHCP таким образом, чтобы отображать соответствующие пакеты светло-желтым или любым другим легко различаемым цветом. Это даст вам возможность намного быстрее различать весь сетевой трафик по протоколу DHCP, а следовательно, повысить эффективность анализа пакетов.

ПРИМЕЧАНИЕ *Не так давно мне пришлось обсуждать принятые в Wireshark правила выделения цветом в ходе презентации для группы местных учащихся. Один из учащихся признался, что ему удалось изменить установленные по умолчанию правила выделения цветом, поскольку он не может различать некоторые цвета. Такое изменение правил выделения пакетов разным цветом лишь расширяет специальные возможности пользователей Wireshark.*

Файлы конфигурации

Полезно знать, где именно Wireshark сохраняет настройки своей конфигурации, на тот случай, если потребуется внести в них непосредственные коррективы. Чтобы найти местоположение файлов конфигурации Wireshark, достаточно выбрать сначала команду Help⇨About Wireshark (Сцравка⇨О программе Wireshark), а затем вкладку Folders (Папки). Диалоговое окно с этой развернутой вкладкой приведено на рис. 3.10.

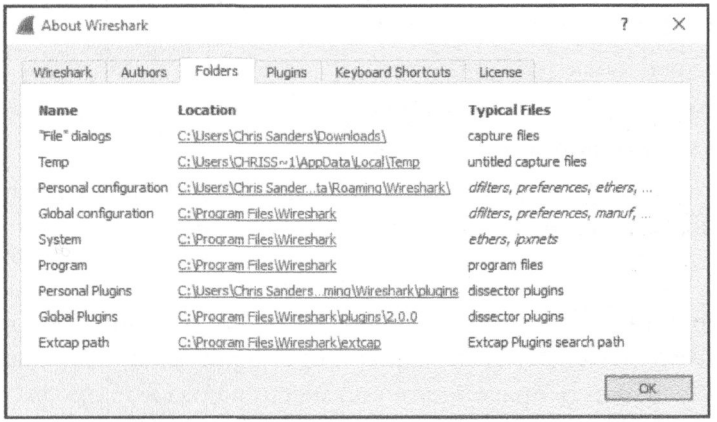


Рис. 3.10. Местоположение файлов конфигурации Wireshark

Что касается специальной настройки Wireshark, то двумя самыми важными местами для ее проведения служат каталоги личной и глобальной конфигурации. Так, в каталоге глобальной конфигурации содержатся все устанавливаемые и сохраняемые по умолчанию настройки и профили Wireshark, а в каталоге личной конфигурации — специальные настройки и профили, характерные для вашей учетной записи. Любые новые файлы, которые вы создаете, будут сохраняться в подкаталоге, расположенном в каталоге личной конфигурации по указанным вами именам. Различать каталоги личной и глобальной конфигурации важно потому, что любые изменения в файлах глобальной конфигурации окажут влияние на каждого пользователя Wireshark в системе.

Профили конфигурации

После ознакомления с глобальными параметрами настройки Wireshark у вас может порой возникнуть потребность воспользоваться сначала одним рядом этих параметров, а затем быстро перейти к другим глобальным параметрам, чтобы учесть изменившуюся ситуацию. Но вместо того чтобы перенастраивать глобальные параметры каждый раз, когда в этом возникает потребность, можно воспользоваться внедренными в Wireshark профилями

конфигурации, дающими пользователям возможность создавать и сохранять определенный ряд глобальных параметров.

В профиле конфигурации хранится следующее.

- Глобальные параметры настройки
- Фильтры перехвата
- Фильтры отображения
- Правила выделения цветом
- Запрещенные сетевые протоколы
- Принудительные расшифровки
- Недавние установки, в том числе размеры панелей, настройки представления меню и ширина столбцов
- Характерные для протоколов таблицы, содержащие, например, перечень пользователей сетевого протокола SNMP и специальные HTTP-заголовки

Чтобы просмотреть список профилей конфигурации, выберите команду Edit⇨Configuration Profiles (Правка⇨Профили конфигурации) из главного меню. Можно также щелкнуть правой кнопкой мыши на разделе профилей в правом нижнем углу окна и выбрать команду Manage Profiles (Управлять профилями) из контекстного меню. В открывшемся окне Configuration Profiles вы обнаружите ряд стандартных профилей Wireshark, включая следующие: **`Default`** (Стандартный), **`Bluetooth`** (Беспроводная персональная сеть Bluetooth) и **`Classic`** (Классический). Здесь же находится специально созданный автором книги профиль **`Latency Investigation`** (Исследование сетевой задержки), который выделяется простым текстом, тогда как остальные профили — курсивом, как показано на рис. 3.11.

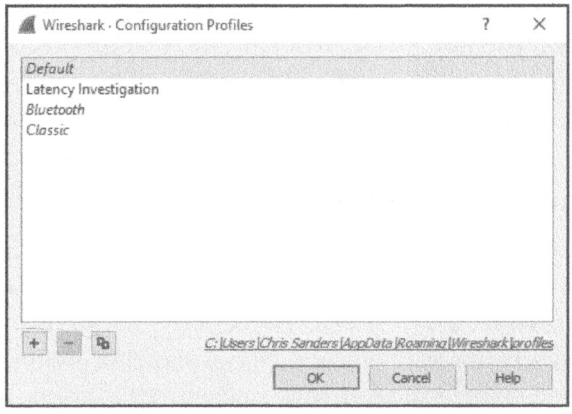


Рис. 3.11. Просмотр профилей конфигурации

В окне Configuration Profiles можно создавать, копировать, удалять и применять профили конфигурации. Процесс создания нового профиля довольно прост. Для этого достаточно выполнить следующие действия.

1. Настройте конфигурацию Wireshark с установками, которые требуется сохранить в отдельном профиле.
2. Перейдите в окно Configuration Profiles, выбрав команду Edit⇨Configuration Profiles из главного меню.
3. Щелкните на кнопке со знаком "плюс" (+) и присвойте описательное имя новому профилю конфигурации.
4. Щелкните на кнопке OK.

Когда у вас возникнет потребность сменить профиль конфигурации, перейдите в окно Configuration Profiles, щелкните сначала на имени нужного профиля, а затем на кнопке OK. Эту операцию можно ускорить, щелкнув правой кнопкой мыши в правом нижнем углу окна Wireshark на заголовке **Profile** и выбрав нужный профиль, как показано на рис. 3.12.

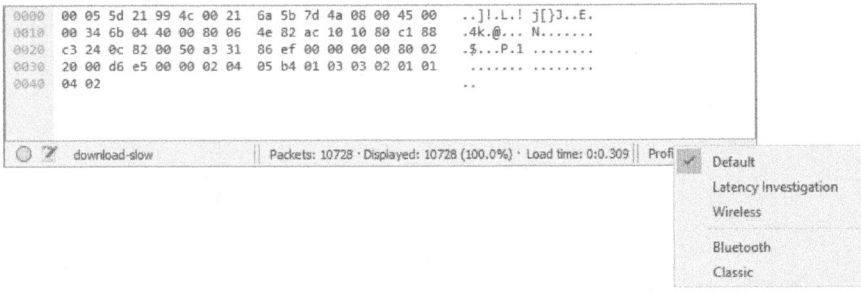


Рис. 3.12. Быстрая смена профиля конфигурации через заголовок ***Profile***

К числу самых полезных свойств профилей конфигурации относится возможность сохранять их в отдельном каталоге вместе с целым рядом файлов конфигурации. Это означает, что можно создавать резервные копии профилей, а затем обмениваться ими с другими пользователями. На вкладке Folders, приведенной на рис. 3.10, указаны пути к каталогам с файлами личной и глобальной конфигурации. Чтобы обменяться профилем с пользователем на другом компьютере, достаточно скопировать папку, совпадающую с именем обмениваемого профиля, и вставить ее в тот же самый каталог для соответствующего пользователя на другом компьютере.

По ходу чтения этой книги у вас может возникнуть потребность создать несколько профилей высокого уровня для общей диагностики сети, выявления источника сетевой задержки и исследования вопросов безопасности. Не бойтесь свободно пользоваться профилями. Они действительно экономят время,

когда требуется быстро сменить ряд глобальных параметров настройки. Мне известны пользователи Wireshark, которые употребляли десятки профилей для успешного разрешения самых разных ситуаций в сети.

Итак, установив и настроив Wireshark, вы готовы приступить к анализу пакетов. В главе 4, "Обработка перехваченных пакетов", поясняется, как обрабатывать перехваченные пакеты.

4

ОБРАБОТКА ПЕРЕХВАЧЕННЫХ ПАКЕТОВ

После знакомства с Wireshark вы готовы приступить к перехвату и анализу пакетов. Из этой главы вы узнаете, как обращаться с файлами перехвата, пакетами и форматами отображения времени. В ней также рассматриваются дополнительные возможности для перехвата пакетов и особенности применения фильтров.

Обработка файлов перехвата

Со временем вы обнаружите, что немалая доля анализа пакетов приходится на стадию, следующую после их перехвата. Как правило, несколько перехватов производится в разные моменты времени. При этом они сохраняются, а затем анализируются сообща. Таким образом, Wireshark позволяет сохранять файлы перехвата для последующего их анализа. Кроме того, несколько файлов перехвата можно объединить вместе.

Сохранение и экспорт файлов перехвата

Чтобы сохранить результат перехвата пакетов, выберите команду File⇨Save As (Файл⇨Сохранить как) из главного меню. В итоге появится диалоговое окно Save file as (Сохранение файла), приведенное на рис. 4.1, в котором вам будет предложено указать место для сохранения полученного перехвата

пакетов, а также выбрать для него подходящий формат файла. Если вы не укажете формат файла, Wireshark автоматически выберет задаваемый по умолчанию формат файла с расширением **.pcapng**.

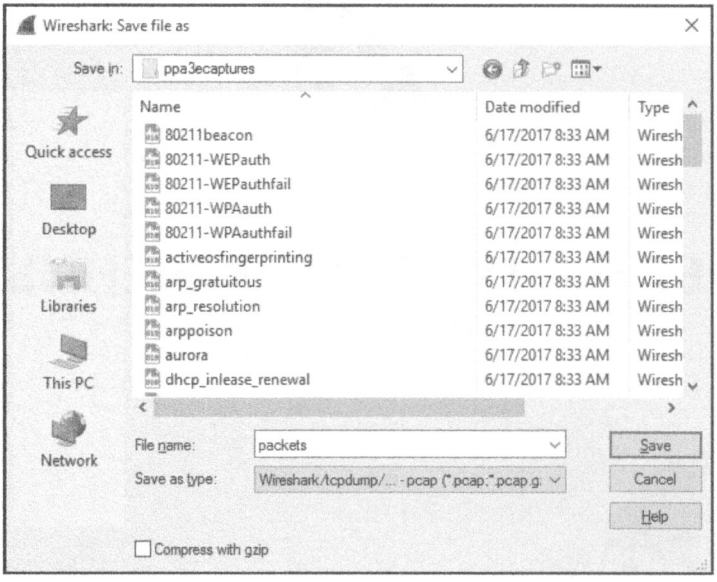


Рис. 4.1. В диалоговом окне Save file as можно сохранять полученные перехваты пакетов

Зачастую требуется сохранить лишь часть перехваченных пакетов. С этой целью выберите команду File⇨Export Specified Packets (Файл⇨Экспортировать указанные пакеты) из главного меню. В итоге появится диалоговое окно, приведенное на рис. 4.2. Это удобный случай сократить размеры чрезмерно раздутых файлов перехвата. В частности, можно выбрать сохранение пакетов лишь в конкретном диапазоне номеров, отмеченных пакетов или же тех пакетов, которые появляются на экране после применения фильтра отображения (отмеченные пакеты и фильтры рассматриваются далее в этой главе).

Объединение файлов перехвата

Для некоторых видов анализа требуется возможность объединять вместе несколько файлов перехвата. И это обычная норма практики, когда приходится сравнивать два потока данных или объединять потоки, относящиеся к одному сетевому трафику, но перехваченные отдельно.

Чтобы объединить файлы перехвата, откройте один из них и выберите команду File⇨Merge (Файл⇨Объединить) из главного меню. В итоге откроется диалоговое окно Merge with capture file (Объединение с файлом перехвата),

приведенное на рис. 4.3. Выберите сначала новый файл, с которым требуется объединить уже открытый файл, а затем способ их объединения. В частности, выбранный файл можно присоединить в начале или в конце открытого в настоящий момент файла или же объединить файлы в хронологическим порядке, исходя из отметок времени их создания или изменения.

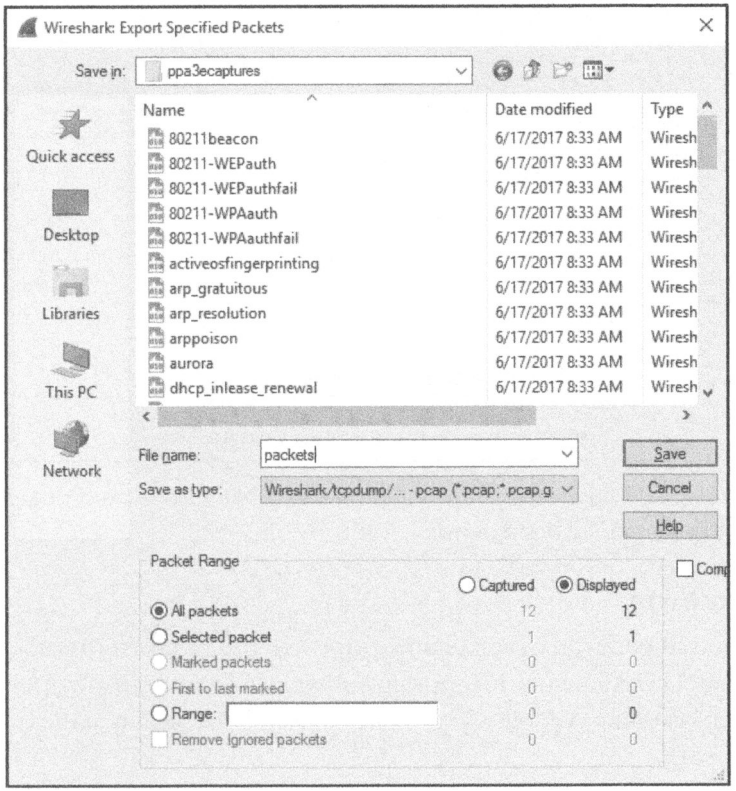


Рис. 4.2. В диалоговом окне Export Specified Packets можно более точно указать порядок сохранения перехваченных пакетов

Обработка пакетов

Дело в конечном итоге дойдет до того, что вам придется обрабатывать очень много пакетов. По мере увеличения количества пакетов до тысяч и даже миллионов у вас возникнет потребность эффективно перемещаться по этим пакетам. И для этой цели в Wireshark предусмотрен поиск и отметка пакетов, отвечающих определенным критериям. Чтобы упростить обращение к пакетам для получения быстрой справки, их можно распечатать.

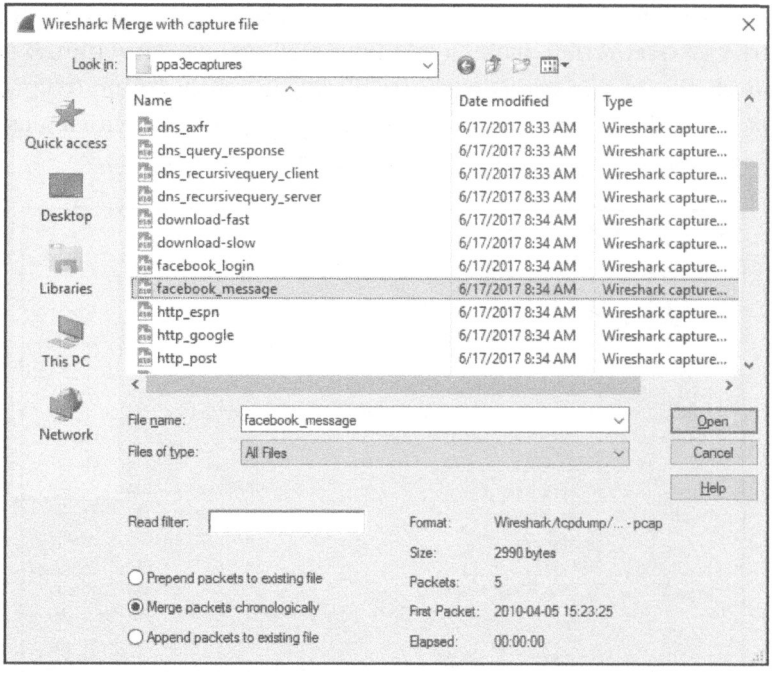


Рис. 4.3. В диалоговом окне Merge with capture file можно объединить два файла перехвата

Поиск пакетов

Чтобы найти пакеты, отвечающие определенным критериям, откройте панель Find Packet (Поиск пакета), приведенную в сжатом виде на рис. 4.4, нажав комбинацию клавиш <Ctrl+F>. Эта панель появится между панелью Filter и окном Packet List.

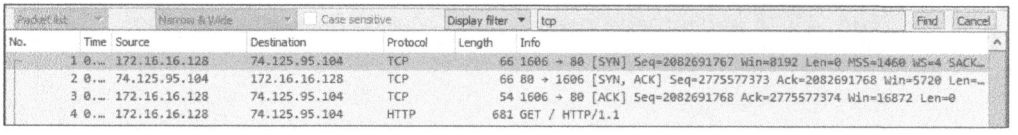


Рис. 4.4. Поиск пакетов в Wireshark по заданным критериям. В данном случае показаны пакеты, совпадающие с выражением **tcp**, *заданным в фильтре отображения*

На этой панели предоставляются три варианта поиска пакетов, как поясняется ниже.

- **Display filter (Фильтр отображения).** Этот вариант позволяет создать фильтр, чтобы найти только те пакеты, которые отвечают заданному в нем выражению. Именно этот вариант поиска пакетов использован для получения результатов, приведенных на рис. 4.4.

- **Hex value (Шестнадцатеричное значение).** Этот вариант предназначен для поиска пакетов по указанному шестнадцатеричному значению.
- **String (Символьная строка).** Этот вариант предназначен для поиска пакетов по указанной символьной строке. В частности, можно указать имя объекта для поиска пакетов или учитывать регистр букв в строке поиска.

Виды поиска пакетов сведены в табл. 4.1.

Таблица 4.1. Виды поиска пакетов

Вид поиска	Примеры
Фильтр отображения	`not ip`
	`ip.addr==192.168.0.1`
	`Arp`
Шестнадцатеричное значение	`00ff`
	`ffff`
	`00ABB1f0`
Символьная строка	`Workstation1`
	`UserB`
	`domain`

Выбрав вид поиска пакетов, введите критерий поиска в текстовом поле и щелкните на кнопке Find, чтобы найти первый пакет, отвечающий заданному критерию. Чтобы найти следующий пакет, щелкните на кнопке Find еще раз или нажмите комбинацию клавиш <Ctrl+N>, а для поиска предыдущего пакета, совпавшего с заданным критерием, – комбинацию клавиш <Ctrl+B>.

Отметка пакетов

Итак, найдя пакеты, отвечающие заданному критерию, можете отметить те из них, которые интересуют вас в первую очередь. Отметка пакетов, в частности, позволяет сохранить только эти пакеты. Кроме того, отмеченные пакеты можно быстро отыскать по их белому тексту на черном фоне, как показано на рис. 4.5.

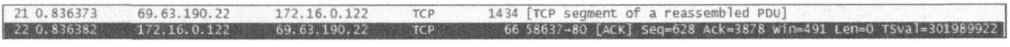


Рис. 4.5. Отмеченный пакет выделяется на экране. В данном примере второй пакет оказывается отмеченным и выглядит более темным на экране

Чтобы отметить пакет, щелкните на нем правой кнопкой мыши в панели Packet List и выберите команду Mark Packet (Отметить пакет) из контекстного

меню или нажмите комбинацию клавиш <Ctrl+M>, находясь в панели Packet List. Отметить можно столько перехваченных пакетов, сколько потребуется. А для перехода между отмеченными пакетами вперед или назад нажмите комбинацию клавиш <Shift+Ctrl+N> или <Shift+Ctrl+B> соответственно.

Вывод пакетов на печать

Несмотря на то что бо́льшая часть анализа проводится на экране компьютерного монитора, перехваченные данные иногда требуется распечатать. Время от времени я распечатываю перехваченные пакеты и приклеиваю липкой лентой полученные распечатки к своему рабочему столу, чтобы быстро обращаться за справкой к их содержимому по ходу анализа. Пакеты удобно также распечатать в формате PDF, особенно при подготовке отчетов.

Чтобы распечатать перехваченные пакеты, откройте диалоговое окно Print, выбрав команду File⇨Print (Файл⇨Печать) из главного меню (рис. 4.6).

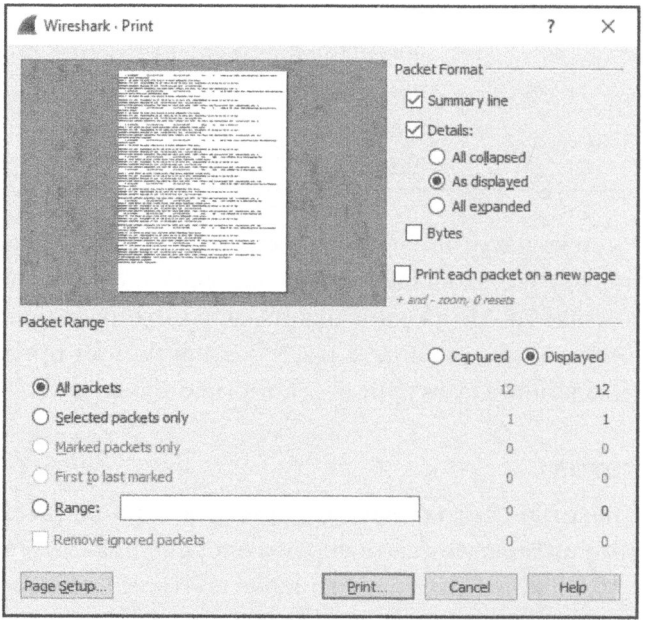


Рис. 4.6. В диалоговом окне Print можно распечатать указанные пакеты

Как и в диалоговом окне Export Specified Packets, в данном окне можно указать печать пакетов только в определенных пределах или же тех пакетов, которые отображаются после применения фильтра. Кроме того, можно указать требуемую степень детализации для печати каждого пакета. После того как выберете все параметры настройки печати, щелкните на кнопке Print.

Задание форматов отображения времени и привязок к нему

Время очень существенно, особенно для анализа пакетов. Все, что происходит в сети, зависит от времени, и поэтому в файлах перехвата приходится часто изучать тенденции и сетевые задержки. В приложении Wireshark предоставляется несколько настраиваемых параметров, имеющих отношение ко времени. В этом разделе будут рассмотрены форматы отображения времени и привязки к нему.

Форматы отображения времени

Каждый пакет, перехватываемый в Wireshark, снабжается отметкой времени, присваиваемой ему на уровне операционной системы. Приложение Wireshark способно отображать абсолютную отметку времени, обозначающую конкретный момент, когда пакет был перехвачен, время относительно последнего перехваченного пакета, а также начало и конец перехвата.

Параметры, имеющие отношение к отображению времени, находятся под заголовком View в главном меню. Пункт Time Display Format (Формат отображения времени) под этим заголовком главного меню позволяет настроить формат представления времени, а также точность его отображения, как показано на рис. 4.7.

Параметры настройки формата для представления времени позволяют выбрать разные варианты отображения времени. К их числу относится отображение даты и времени суток в стандартном формате или же в формате всеобщего скоординированного времени (UTC), количества секунд, прошедших с момента последнего перехвата, и прочее.

А параметры настройки точности позволяют задать точность отображения времени автоматически, т.е. в формате, взятом из файла перехвата, или вручную, например, в секундах, миллисекундах, микросекундах и т.д. Эти параметры будут настраиваться в примерах, приведенных далее в книге, поэтому вы должны ознакомиться с ними уже теперь.

ПРИМЕЧАНИЕ *Сравнивая данные из пакетов, поступающих из разных устройств, убедитесь, что эти устройства синхронизированы с одним и тем же источником времени, особенно если вы выполняете ретроспективный анализ или диагностику сети. Для синхронизации устройств в сети можно воспользоваться протоколом NTP (Network Time Protocol – протокол сетевого времени). Анализ пакетов, поступающих из устройств в разных часовых поясах, следует проводить в формате UTC, а не местного времени, чтобы избежать недоразумений при сообщении полученных результатов анализа.*

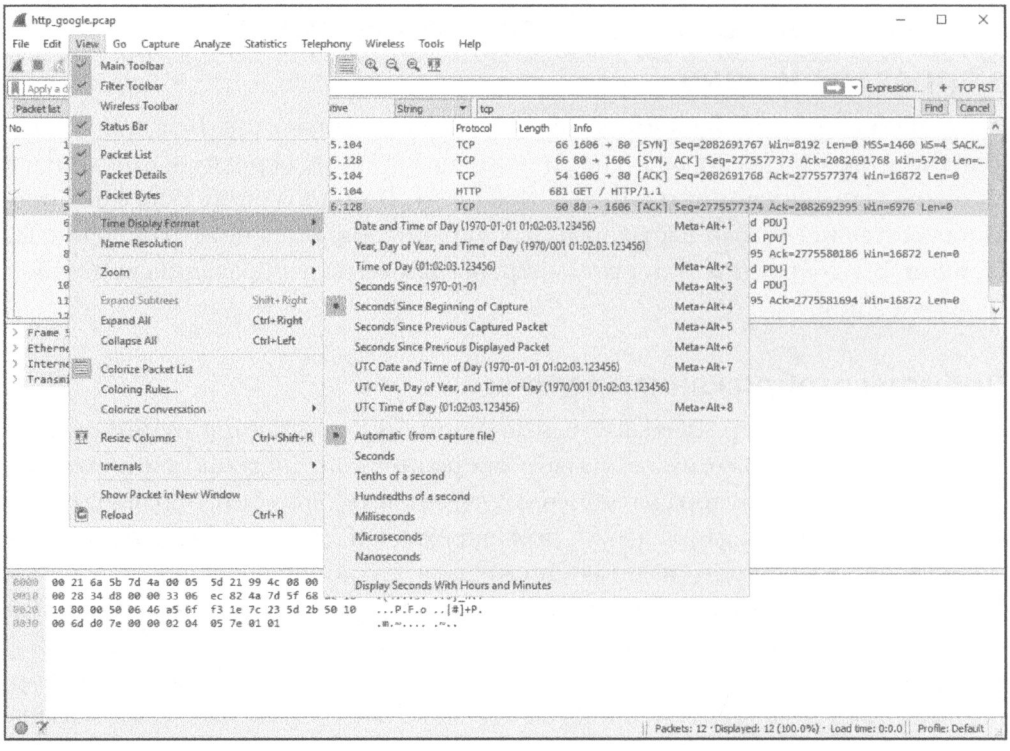


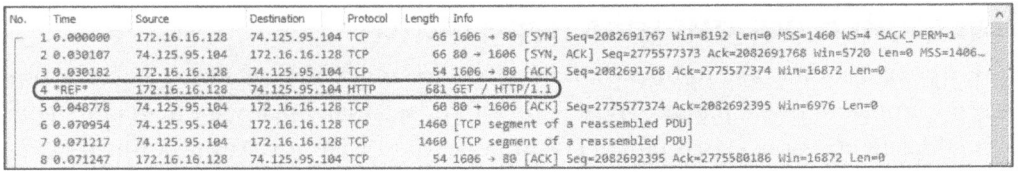
Рис. 4.7. По команде View⇨Time Display Format из главного меню можно выбрать один из нескольких форматов отображения времени

Временная привязка к пакетам

Временная привязка к пакетам позволяет настроить определенный пакет таким образом, чтобы последующие расчеты времени производились относительно данного пакета. Такая возможность особенно удобна при исследовании ряда последовательных событий, наступающих в какой-нибудь другой момент, а не в момент начала формирования файла перехвата.

Чтобы установить временную привязку к пакету, щелкните правой кнопкой мыши на избранном в качестве опорного пакете в панели Packet List и выберите команду Set/Unset Time Reference (Установить/Сбросить временную привязку) из контекстного меню. А для того чтобы сбросить временную привязку, повторите эту же операцию еще раз. Кроме того, устанавливать и сбрасывать временную привязку к пакету можно, выбрав опорный пакет в подокне Packet List и нажав комбинацию клавиш <Ctrl+T>.

Если временная привязка к пакету активизирована, в столбце **Time** (Время), отображаемом в панели Packet List, появится метка ***REF*** (рис. 4.8).



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.030107	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406…
3	0.030182	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	*REF*	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.048778	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.070954	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
7	0.071217	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
8	0.071247	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082692395 Ack=2775580186 Win=16872 Len=0

Рис. 4.8. Пакет №4 с активизированной временной привязкой к нему

Устанавливать временную привязку к пакету удобно лишь в том случае, если формат отображения времени перехвата настроен на показ времени относительно начала перехвата. А в любом другом случае установка временной привязки к пакету ничего полезного не даст и на самом деле приведет к отображению моментов времени, в которых будет очень трудно разобраться.

Временной сдвиг

Иногда встречаются пакеты из разных источников, которые не синхронизированы с одним и тем же источником. Это особенно характерно для исследования файлов перехвата, взятых из двух мест, содержащих один и тот же поток данных. Большинство сетевых администраторов стремятся к тому, чтобы каждое устройство в их сети было синхронизировано. Но нередко между определенными типами устройств возникает временной сдвиг. В приложении Wireshark предоставляется возможность сдвигать отметку времени в пакетах, чтобы устранить данное затруднение в ходе анализа.

Чтобы сдвинуть отметку времени в одном или нескольких пакетах, выберите команду Edit⇨Time Shift (Правка⇨Временной сдвиг) из главного меню или нажмите комбинацию клавиш <Ctrl+Shift+T>. В открывшемся диалоговом окне Time Shift можно указать пределы для временного сдвига во всем файле перехвата в целом или же установить время для отдельных пакетов. В примере, приведенном на рис. 4.9, выбран сдвиг отметки времени в каждом пакете из файла перехвата на две минуты и пять секунд.

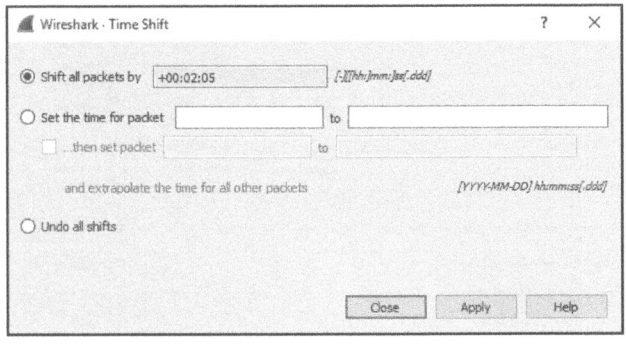


Рис. 4.9. Диалоговое окно Time Shift

Настройка параметров перехвата

В предыдущей главе процесс элементарного перехвата пакетов рассматривался с применением параметров настройки, устанавливаемых в диалоговом окне Capture Interfaces (Интерфейсы для перехвата). Но в Wireshark предоставляется немало других возможностей для настройки параметров перехвата, которые мы еще не рассматривали. Для доступа к этим параметрам выберите команду Capture⇨Options из главного меню.

В диалоговом окне Capture Interfaces имеется немало настраиваемых параметров, позволяющих сделать процесс перехвата пакетов более удобным. Оно разделено на три вкладки: Input (Ввод), Output (Вывод) и Options (Параметры). Рассмотрим их по очереди.

Вкладка Input

Основное назначение вкладки Input (рис. 4.10) — отображать все сетевые интерфейсы, доступные для перехвата пакетов, а также некоторые основные сведения о каждом из этих интерфейсов. К этим сведениям относится удобное имя сетевого интерфейса, предоставляемое на уровне операционной системы, график сетевого трафика, наглядно показывающий пропускную способность данного интерфейса, а также дополнительные параметры настройки конфигурации, в том числе состояние смешанного режима работы сетевого интерфейса и размер буфера. На правом краю рассматриваемой здесь вкладки, не показанном на рис. 4.10, находится также столбец для отображения фильтра, применяемого при перехвате. Подробнее об этом речь пойдет в разделе“Фильтры перехвата”.

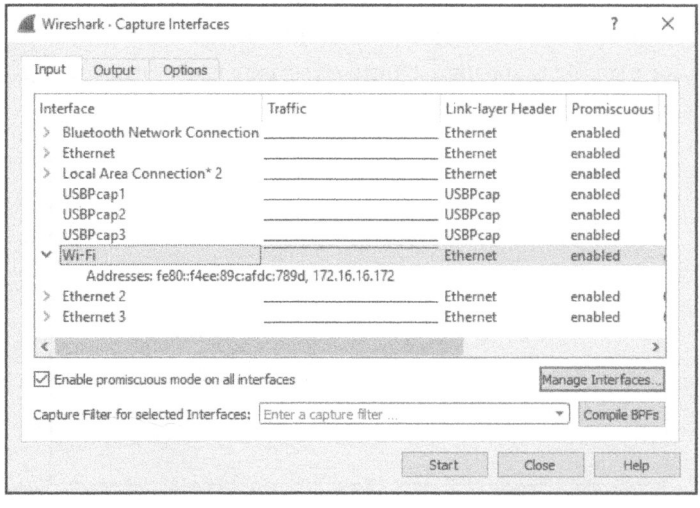


Рис. 4.10. Вкладка Input диалогового окна Capture Interfaces

На вкладке Input можно щелкнуть на большинстве параметров, чтобы сразу же отредактировать их. Так, если требуется отменить смешанный режим работы сетевого интерфейса, достаточно щелкнуть на его поле и сменить активное состояние этого режима на неактивное, выбрав соответствующий пункт из раскрывающегося меню.

Вкладка Output

На вкладке Output (рис. 4.11) можно организовать автоматическое сохранение перехваченных пакетов в файле вместо того, чтобы сначала перехватывать их, а затем сохранять в файле. Ведь манипулировать сохраненными пакетами намного легче. В частности, перехваченные пакеты можно сохранить в одном файле, в нескольких файлах и даже в кольцевом буфере, как поясняется далее, чтобы манипулировать целым рядом созданных файлов. Чтобы активизировать такой режим сохранения перехваченных пакетов, введите полный путь и имя файла в текстовом поле File. А для того чтобы выбрать каталог и предоставить имя файла, щелкните на кнопке Browse... (Просмотр).

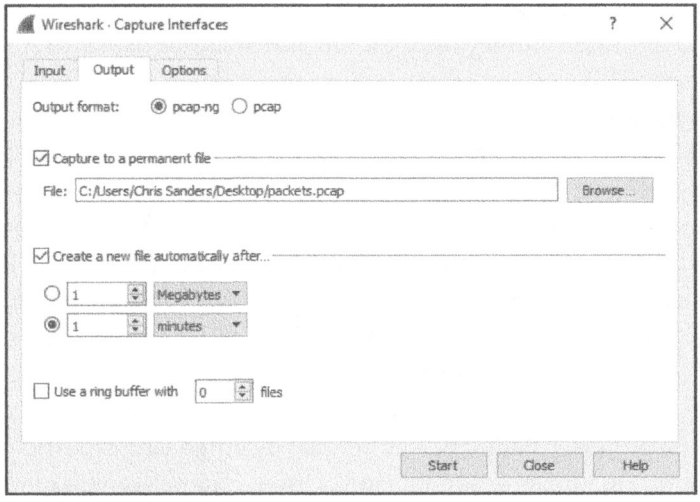
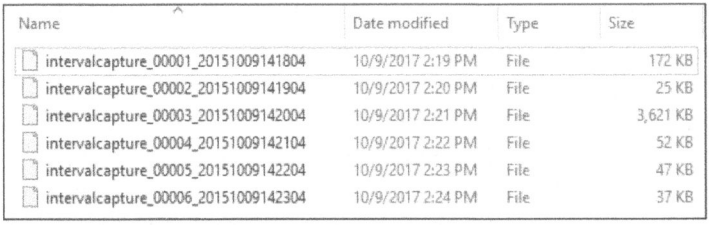


Рис. 4.11. Вкладка Output диалогового окна Capture Interfaces

Если приходится перехватывать большой объем сетевого трафика или выполнять перехват в долгосрочной перспективе, то особенно удобными для этой цели окажутся *наборы файлов* — совокупности нескольких файлов, выделяемые в отдельные группы по особому условию. Чтобы задать набор файлов, установите флажок Create a new file automatically after... (Создать автоматически новый файл после...).

Для сохранения перехваченных пакетов в наборах файлов по заданному размеру или промежутку времени в Wireshark применяются специальные

триггеры. Чтобы активизировать один из таких триггеров, выберите кнопку-переключатель, расположенную ниже упомянутого выше флажка, а затем укажите требуемое пороговое значение и единицу его измерения для срабатывания триггера. Например, можно установить триггер для запуска процесса создания нового файла после каждого перехвата сетевого трафика объемом 1 Мбайт, как показано на рис. 4.12, или через каждую минуту при перехвате сетевого трафика.



Name	Date modified	Type	Size
intervalcapture_00001_20151009141804	10/9/2017 2:19 PM	File	172 KB
intervalcapture_00002_20151009141904	10/9/2017 2:20 PM	File	25 KB
intervalcapture_00003_20151009142004	10/9/2017 2:21 PM	File	3,621 KB
intervalcapture_00004_20151009142104	10/9/2017 2:22 PM	File	52 KB
intervalcapture_00005_20151009142204	10/9/2017 2:23 PM	File	47 KB
intervalcapture_00006_20151009142304	10/9/2017 2:24 PM	File	37 KB

Рис. 4.12. Набор файлов, создаваемый в Wireshark через каждую минуту при перехвате сетевого трафика

Флажок Use a ring buffer with... (Использовать кольцевой буфер с...) позволяет указать определенное количество файлов, которые должны быть накоплены в наборе средствами Wireshark, прежде чем перезаписывать их. И хотя термин *кольцевой буфер* имеет разный смысл, в данном случае он, по существу, обозначает набор файлов, где первый файл может быть перезаписан, как только в набор будет введен последний записанный файл, чтобы сохранить вновь поступившие данные. Иными словами, этот флажок устанавливает режим записи файлов по принципу "первым пришел, первым обслужен" (FIFO). Установив этот флажок, можно указать максимальное количество файлов для хранения в кольцевом буфере с циклическим сдвигом. Допустим, что для хранения перехваченного сетевого трафика выбран набор из шести файлов, причем новый файл создается через каждый час. Таким образом, формируется кольцевой буфер размером 6 файлов. Как только будет создан шестой файл, в кольцевом буфере произойдет циклический сдвиг с перезаписью первого файла вместо создания седьмого файла для хранения очередного массива перехваченных данных. Этим гарантируется, что на жестком диске будет храниться не больше шести файлов данных в течение 6 часов с возможностью записи новых данных.

И, наконец, на вкладке Output можно также указать конкретный формат файла данных. Так, можно выбрать традиционный формат **.pcap**, если предполагается взаимодействие с сохраненными пакетами в инструментальном средстве, не способном обрабатывать файлы с расширением **.pcapng**.

Вкладка Options

На вкладке Options содержится целый ряд других вариантов выбора режима сохранения перехваченных пакетов, включая параметры настройки отображения, преобразования имен и прерывания перехвата, как показано на рис. 4.13.

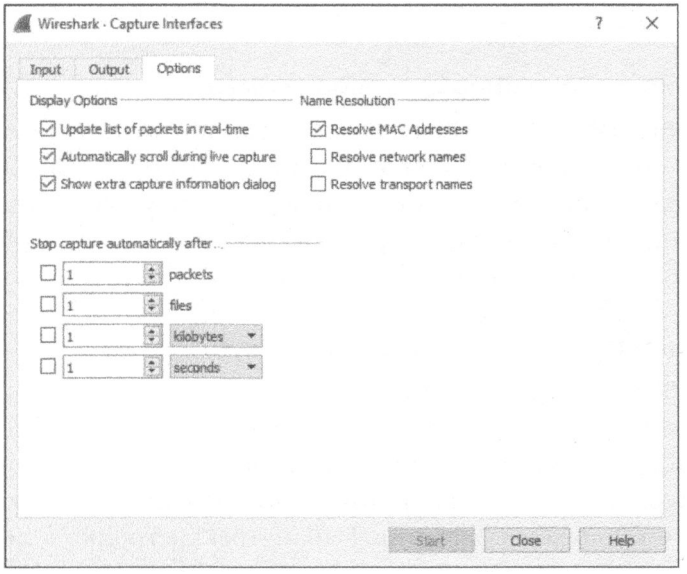


Рис. 4.13. Вкладка Options диалогового окна Capture Interfaces

Параметры настройки отображения

В разделе Display Options (Параметры отображения) находятся элементы, управляющие порядком отображения пакетов в процессе их перехвата. Назначение флажка Update list of packets in real-time (Обновление списка в реальном времени) не требует особых пояснений. Его можно устанавливать вместе с флажком Automatically scroll during live capture (Автоматическая прокрутка в ходе перехвата). Когда оба эти флажка установлены, на экране отображаются все перехваченные пакеты, причем самые последние из них отображаются немедленно.

ПРЕДУПРЕЖДЕНИЕ *Если установлены оба флажка,* Update list of packets in real-time *и* Automatically scroll during live capture, *то для поддержания соответствующего режима отображения потребуется немало ресурсов ЦП – даже если перехватывается скромный объем данных. Поэтому оба эти флажка лучше сбросить, если только нет особой необходимости просматривать перехваченные пакеты в реальном времени.*

Флажок Show extra capture information dialog (Показывать диалоговое окно с дополнительными сведениями) позволяет активизировать или подавить отображение небольшого окна, в котором представлено в процентах количество перехваченных пакетов, отсортированных по их сетевому протоколу. Я лично предпочитаю отображать это информационное окно, поскольку обычно запрещаю активную прокрутку окна отображения пакетов во время их перехвата.

Параметры настройки преобразования имен

В разделе Name Resolution (Преобразование имен) можно установить режим автоматического преобразования MAC-адресов (второго уровня), а также сетевых и транспортных имен (третьего и четвертого уровня соответственно). Более подробно общие вопросы преобразования имен будут рассмотрены в главе 5, "Дополнительные возможности Wireshark".

Параметры настройки прекращения перехвата

В разделе Stop capture automatically after... (Автоматически прекратить перехват после...) можно задать определенные условия, при которых перехват будет прекращен. Как и при создании набора файлов, прекратить перехват можно по заданному размеру файла или промежутку времени. Но это можно сделать и по указанному количеству пакетов. Параметры настройки в данном разделе удобно сочетать с параметрами настройки, доступными на вкладке Output.

Применение фильтров

С помощью фильтров можно указать пакеты, которые требуются для анализа. Проще говоря, фильтр – это выражение, в котором задаются критерии для включения или исключения пакетов из анализа. Так, если имеются пакеты, которые не требуется просматривать, можно создать фильтр, чтобы избавиться от них. А если требуется просматривать только вполне определенные пакеты, можно создать фильтр, чтобы отображать только их.

В приложении Wireshark предоставляются следующие типы фильтров.

- **Фильтры перехвата.** Применяются в том случае, если требуется перехватывать только те пакеты, которые указаны для включения или исключения в заданном выражении.
- **Фильтры отображения.** Применяются к существующему ряду пакетов с целью скрыть ненужные пакеты или показать нужные, исходя из заданного выражения.

Рассмотрим сначала фильтры перехвата.

Фильтры перехвата

Фильтры данного типа применяются в процессе перехвата пакетов с целью изначально ограничить количество пакетов, предоставляемых для анализа. Одной из главных причин для применения фильтров перехвата служит производительность. Если заранее известно, что анализировать определенную форму сетевого трафика не нужно, ее достаточно отсеять с помощью фильтра перехвата, сэкономив процессорное время, которое обычно затрачивается для перехвата соответствующих пакетов.

Возможность создавать специальные фильтры перехвата окажется удобной в том случае, если приходится иметь дело с большими массивами данных. Их анализ можно ускорить, если отсеять только те пакеты, которые потребуются для решения текущей задачи анализа.

Допустим, решается задача диагностики службы, работающей через порт **262**, но на анализируемом сервере действует целый ряд служб через разные порты. Перехватить и проанализировать сетевой трафик только через один порт – дело совсем не простое. И здесь может пригодиться фильтр перехвата. Для этого достаточно воспользоваться диалоговым окном Capture Interfaces следующим образом.

1. Выберите команду Capture⇨Options из главного меню, чтобы открыть диалоговое окно Capture Interfaces.
2. Выберите сетевой интерфейс, где требуется перехватывать пакеты, а затем перейдите к крайнему справа столбцу Capture Filter (Фильтр перехвата).
3. Чтобы применить фильтр перехвата, щелкните на этом столбце и введите фильтрующее выражение. В данном случае требуется отфильтровать только сетевой трафик, входящий и исходящий через порт **262**, и поэтому введите **`port 262`**, как показано на рис. 4.14. (Более подробно фильтрующие выражения рассматриваются в следующем разделе.) Цвет ячейки в данном столбце должен измениться на зеленый, указывая на то, что введено достоверное выражение. Если же оно недостоверно, ячейка будет выделена красным цветом.
4. Задав фильтр, щелкните на кнопке Start, чтобы приступить к перехвату.

В итоге вы должны видеть только тот сетевой трафик, который проходит через порт **262**. Следовательно, это даст вам возможность более эффективно анализировать данные этого конкретного трафика.

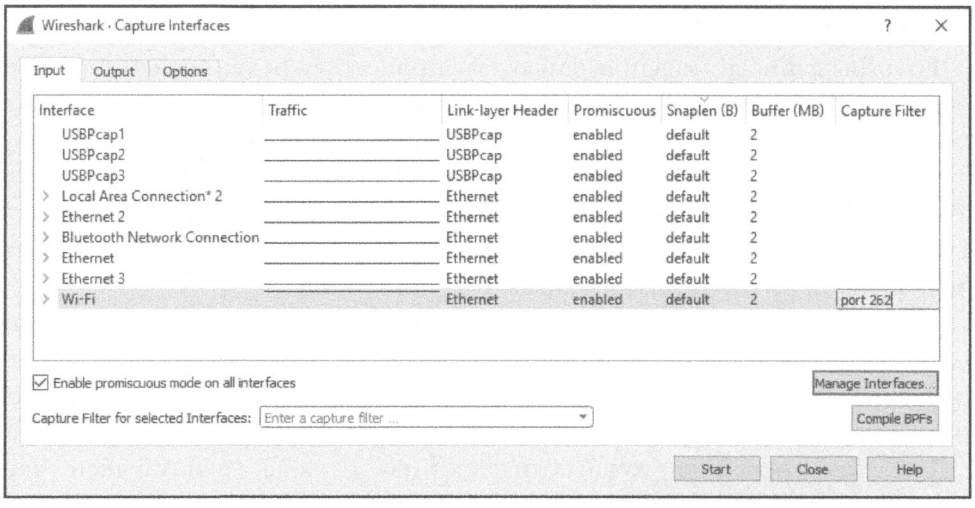


Рис. 4.14. Создание фильтра перехвата в диалоговом окне Capture Interfaces

Синтаксис BPF для фильтров перехвата

Фильтры перехвата, применяемые в драйверах libpcap/WinPcap, описываются с помощью синтаксиса BPF (Berkeley Packet Filter – фильтр пакетов Беркли). Этот синтаксис характерен для целого ряда приложений анализа пакетов, главным образом потому, что они опираются на библиотеки libpcap/WinPcap, в которых для описания фильтров используется синтаксис BPF. Для углубленного анализа сетей на уровне пакетов знание синтаксиса BPF имеет решающее значение.

Фильтр, созданный с помощью синтаксиса BPF, называется *выражением*, причем каждое выражение состоит из одного или нескольких *примитивов*. В свою очередь, примитивы состоят из одного или нескольких *квалификаторов*, перечисленных в табл. 4.2, а также имени или номера идентификатора, как показано на рис. 4.15.

Таблица 4.2. Квалификаторы BPF

Квалификатор	Описание	Примеры
`type`	Обозначает имя или номер идентификатора, на который делается ссылка	`host`, `net`, `port`
`dir`	Обозначает направление передачи при обмене данными с сетевым узлом, доступным по имени или номеру идентификатора	`src`, `dst`
`proto`	Ограничивает соответствие конкретному протоколу	`ether`, `ip`, `tcp`, `udp`, `http`, `ftp`

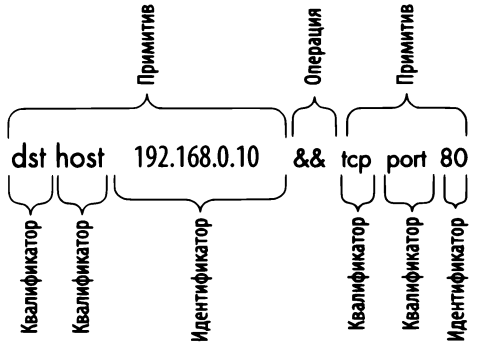


Рис. 4.15. Пример структуры фильтра перехвата

Принимая во внимание составляющие выражения, квалификатор `dst host` и идентификатор **`192.168.0.10`** совместно образуют примитив. Такой примитив сам является выражением, определяющим перехват сетевого трафика только по IP-адресу получателя **`192.168.0.10`**.

Для объединения примитивов при создании более сложных выражений можно использовать логические операции. И для этой цели имеются следующие логические операции.

- Логическая операция конкатенации по И (**`&&`**).
- Логическая операция дизъюнкции по ИЛИ (**`||`**).
- Логическая операция отрицания по НЕ (**`!`**).

Например, следующее выражение служит для перехвата только сетевого трафика, исходящего из источника, расположенного по IP-адресу **`192.168.0.10`** и проходящего через порт отправителя или получателя **`80`**:

```
src host 192.168.0.10 && port 80
```

Фильтры имен хостов и адресации

Большинство фильтров создается для анализа конкретного сетевого устройства или группы подобных устройств. В зависимости от обстоятельств фильтрация может основываться на MAC-адресе устройства, адресах IPv4 или IPv6, а также на имени хоста в службе DNS.

Допустим, что требуется проанализировать сетевой трафик конкретного хоста, взаимодействующего с сервером в сети. Чтобы перехватывать весь сетевой трафик, связанный с адресом данного хоста по протоколу IPv4, на сервере можно создать следующий фильтр, используя квалификатор `host`:

```
host 172.16.16.149
```

Если же анализируется сеть, действующая по протоколу IPv6, то фильтр следует создать на основе адреса IPv6, используя квалификатор `host`:

```
host 2001:db8:85a3::8a2e:370:7334
```

Создать фильтр можно и на основе имени хоста сетевого устройства, используя квалификатор `host` следующим образом:

```
host testserver2
```

А если требуется принять во внимание возможное изменение IP-адреса хоста, то фильтр можно создать на основе его MAC-адреса, добавив квалификатор протокола `ether`, как показано ниже.

```
ether host 00-1a-a0-52-e2-a0
```

Квалификаторы направления передачи данных нередко применяются вместе с фильтрами (например, из приведенных выше примеров) для перехвата сетевого трафика в зависимости от того, является ли этот трафик входящим или исходящим из хоста. Например, чтобы перехватить только сетевой трафик, исходящий из конкретного хоста, необходимо дополнить фильтр квалификатором `src` следующим образом:

```
src host 172.16.16.149
```

А для того чтобы перехватить только те данные, которые предназначены для хоста по адресу **`172.16.16.149`**, следует дополнить приведенный выше фильтр квалификатором `dst`:

```
dst host 172.16.16.149
```

Если в фильтре не употребляется квалификатор типа (`host`, `net` или `port`) с примитивом, то подразумевается квалификатор `host`. Поэтому приведенное ниже выражение, где отсутствует такой квалификатор, равнозначно выражению из предыдущего примера.

```
dst 172.16.16.149
```

Фильтры портов

Помимо фильтрации хостов, создаваемый фильтр можно основывать на портах, применяемых в каждом пакете. Фильтрацию портов можно выполнять для отсеивания служб и приложений, в которых применяются порты известных служб. Так, в следующем примере демонстрируется простой фильтр,

предназначенный для перехвата сетевого трафика, входящего или исходящего только из порта **8080**:

```
port 8080
```

Для перехвата всего сетевого трафика, кроме проходящего через порт **8080**, подойдет следующий фильтр:

```
!port 8080
```

Фильтры портов можно объединять с фильтрами направления передачи данных. Например, чтобы перехватить только сетевой трафик, направляемый веб-серверу для приема через стандартный порт **80** сетевого протокола HTTP, в фильтре можно употребить квалификатор `dst` следующим образом:

```
dst port 8080
```

Фильтры протоколов

Фильтры данного типа позволяют отсеивать пакеты по определенным сетевым протоколам. Они служат для сопоставления с сетевыми протоколами, которые не относятся к уровню приложений и не могут быть выявлены только по определенном порту. Так, если требуется просмотреть сетевой трафик только по протоколу ICMP, можно воспользоваться следующим фильтром:

```
icmp
```

А для просмотра всего сетевого трафика, кроме протокола IPv6, подойдет такой фильтр:

```
!ip6
```

Фильтры полей сетевых протоколов

К числу самых сильных сторон синтаксиса BPF относится возможность исследовать каждый байт в заголовке протокола, чтобы создавать специальные фильтры на основании полученных сведений о конкретном протоколе. Специальные фильтры, рассматриваемые в этом подразделе, позволяют извлекать из пакета определенное количество байтов, начиная с конкретного места.

Допустим, требуется отсеять сетевой трафик по полю типа в заголовке ICMP. Поле типа находится в самом начале пакета, т.е. имеет нулевое смещение. Чтобы обозначить местоположение исследуемого поля в пакете, достаточно указать в квадратных скобках смещение в байтах после квалификатора

протокола (в данном примере — `icmp[0]`). В итоге будет возвращено однобайтовое значение, которое можно с чем-нибудь сравнить. Например, чтобы получить только пакеты, передаваемые по сетевому протоколу ICMP и представляющие сообщения типа 3 о недосягаемости получателя, в фильтрующем выражении следует употребить операцию равенства, как показано ниже.

```
icmp[0] == 3
```

А для того чтобы исследовать только пакеты, передаваемые по сетевому протоколу ICMP и представляющие эхо-запрос (типа 8) или эхо-ответ (типа 0), в фильтрующем выражении можно употребить два примитива и логическую операцию ИЛИ следующим образом:

```
icmp[0] == 8 OR icmp[0] == 0
```

Несмотря на то что специальные фильтры вполне справляются со своей задачей, они основываются только на однобайтовой информации из заголовка пакета. Но в фильтрующем выражении можно также указать длину возвращаемых данных в байтах после числовой величины смещения в квадратных скобках, отделив ее двоеточием.

Допустим, требуется создать фильтр для перехвата всех пакетов, передаваемых по сетевому протоколу ICMP для недосягаемого получателя, хоста и определяемых по типу 3 и коду 1. Однобайтовые поля с этими данными расположены один за другим в заголовке пакета, начиная с нулевого смещения. С этой целью можно создать фильтр, где проверяются два начальных байта с нулевым смещением в заголовке пакета, которые сравниваются с шестнадцатеричным значением **0301** (т.е. тип 3 и код 1) следующим образом:

```
icmp[0:2] == 0x0301
```

Зачастую требуется перехватывать только пакеты, передаваемые по сетевому протоколу TCP с установленным флагом RST. Более подробно о протоколе TCP речь пойдет в главе 8, "Протоколы транспортного уровня", а пока что достаточно сказать, что флаги располагаются в поле пакета TCP со смещением **13**. Это поле интересно тем, что его длина равна одному байту, где каждый бит обозначает отдельный флаг. Как поясняется в приложении Б, "Интерпретация пакетов", каждый бит с установленным в нем флагом представлен в таком байте числом по основанию 2. Так, первый бит представлен числом **1**, второй бит — числом **2**, третий — числом **4** и т.д. В пакете TCP можно одновременно установить несколько флагов. Следовательно, для эффективной их фильтрации одного значения `tcp[13]` недостаточно, поскольку установленному биту RST могут соответствовать несколько значений.

Вместо этого необходимо указать местоположение проверяемого флага в байте, дополнив фильтрующее выражение знаком амперсанда (**&**) и числом, представляющим бит, в котором хранится этот флаг. В частности, флаг RST хранится в бите, представленном в данном байте числом **4**, которое обозначает, что флаг RST установлен. Таким образом, фильтр для рассматриваемых здесь целей будет выглядеть следующим образом:

```
tcp[13] & 4 == 4
```

А для того чтобы просмотреть все пакеты с установленным флагом PSH, который хранится в бите, представленном числом **8** в поле пакета TCP со смещением **13**, достаточно воспользоваться следующим фильтром:

```
tcp[13] & 8 == 8
```

Примеры выражений для фильтров перехвата

Зачастую удачный или неудачный исход анализа пакетов зависит от умения создавать фильтры, пригодные для конкретной ситуации. В табл. 4.3 перечислены некоторые примеры выражений для фильтров перехвата, которые часто применяются при анализе пакетов.

Таблица 4.3. Наиболее употребительные фильтры перехвата

Фильтр	Описание
`tcp[13] & 32 == 32`	Пакеты TCP с установленным флагом URG
`tcp[13] & 16 == 16`	Пакеты TCP с установленным флагом ACK
`tcp[13] & 8 == 8`	Пакеты TCP с установленным флагом PSH
`tcp[13] & 4 == 4`	Пакеты TCP с установленным флагом RST
`tcp[13] & 2 == 2`	Пакеты TCP с установленным флагом SYN
`tcp[13] & 1 == 1`	Пакеты TCP с установленным флагом FIN
`tcp[13] == 18`	Пакеты TCP с установленными флагами SYN и ACK
`ether host` *`00:00:00:00:00:00`*	Входящий и исходящий сетевой трафик по указанному MAC-адресу
`!ether host` *`00:00:00:00:00:00`*	Входящий и исходящий сетевой трафик, кроме указанного MAC-адреса
`broadcast`	Только широковещательный трафик
`icmp`	Трафик только по сетевому протоколу ICMP
`icmp[0:2] == 0x0301`	Трафик по сетевому протоколу ICMP для недосягаемого получателя и хоста
`ip`	Трафик только по сетевому протоколу IPv4
`ip6`	Трафик только по сетевому протоколу Ipv6
`udp`	Трафик только по сетевому протоколу UDP

Фильтры отображения

К этому типу относятся фильтры, применение которых к файлу перехвата означает, что в Wireshark должны быть отображены только те пакеты, которые соответствуют данному фильтру. Фильтр отображения можно создать в текстовом поле Filter, расположенном над панелью Packet List.

Фильтры отображения применяются чаще, чем фильтры перехвата, поскольку они позволяют отсеивать ненужные данные не удаляя при этом их физически из файла перехвата. Так, если потребуется отменить первоначальное условие перехвата, для этого достаточно просто очистить фильтрующее выражение. Кроме того, фильтры отображения оказываются намного более эффективными благодаря поддержке со стороны обширной библиотеки дешифратора пакетов, доступной в Wireshark.

Например, в некоторых случаях фильтр отображения может применяться для отсеивания неуместного широковещательного трафика из файла перехвата. При этом на панели Packet List отсеиваются те широковещательные пакеты ARP, которые не имеют никакого отношения к текущей анализируемой проблеме в сети. Но поскольку эти широковещательные пакеты ARP могут оказаться полезными в дальнейшем, то лучше отсеять их временно, а не удалять полностью.

Чтобы отсеять все пакеты ARP в окне перехвата, установите курсор в текстовом поле Filter, расположенном над панелью Packet List, и введите **!arp**. В итоге все пакеты ARP будут удалены из списка (рис. 4.16). Чтобы удалить введенный фильтр отображения, щелкните на кнопке X, а для того чтобы сохранить этот фильтр для последующего применения, щелкните на кнопке со знаком "плюс" (+).

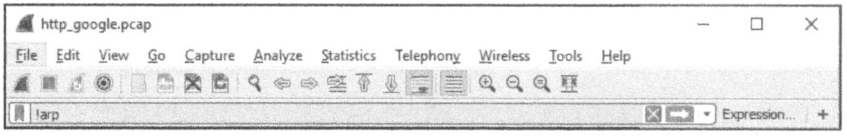


Рис. 4.16. Создание фильтра отображения в текстовом поле Filter, расположенном над подокном Packet List

Фильтры отображения можно применять двумя способами. Один из них состоит в непосредственном употреблении подходящего синтаксиса, как демонстрировалось в предыдущем примере, а другой — в применении диалогового окна Display Filter Expression (Выражение для фильтра отображения) для создания фильтра в диалоговом режиме. Это более простой способ для тех, кто только начинает пользоваться фильтрами. Рассмотрим оба способа применения фильтров отображения, начав с более простого.

Диалоговое окно Display Filter Expression

Возможности диалогового окна Display Filter Expression, приведенного на рис. 4.17, существенно упрощают начинающим пользователям Wireshark задачу по созданию фильтров перехвата и отображения. Для доступа к этому окну выберите команду Filter⇨Expression из главного меню.

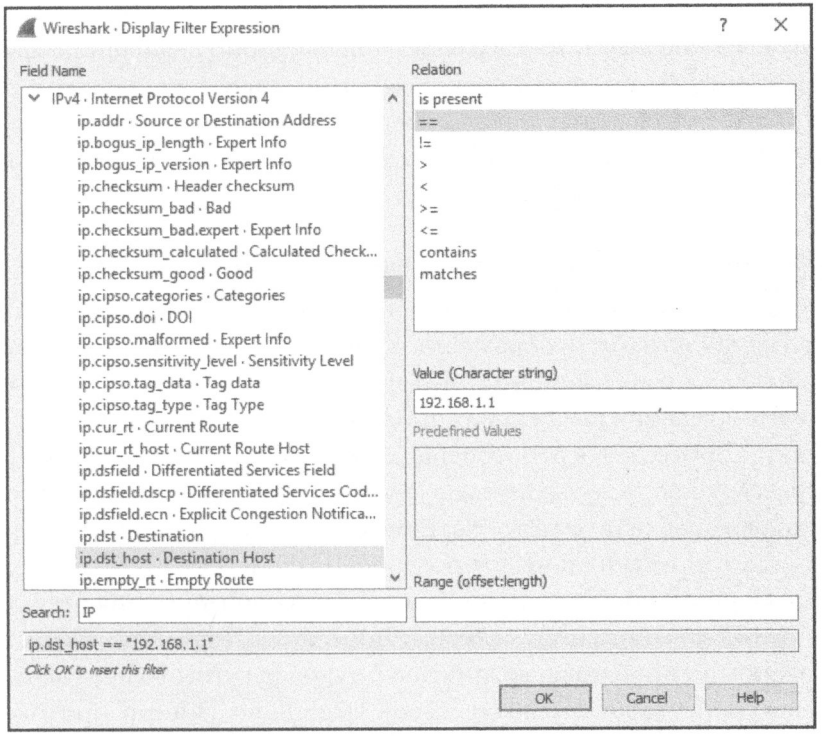


Рис. 4.17. В диалоговом окне Display Filter Expression заметно упрощается задача создания фильтров в Wireshark

Слева в данном окне перечислены все доступные поля сетевого протокола, где указаны все возможные критерии фильтрации. Чтобы создать фильтр, выполните следующие действия.

1. Щелкните на стрелке рядом с названием сетевого протокола, чтобы просмотреть связанные с ним поля критериев. Найдя нужный критерий в качестве основания для создания фильтра, щелкните на нем кнопкой мыши, чтобы выбрать его.
2. Выберите порядок сравнения выбранного поля со значением из критерия. Такое сравнение обозначается с помощью операции больше, меньше, равно и т.д.

3. Составьте фильтрующее выражение, указав значение из критерия, с которым должно сравниваться выбранное поле. Это значение можно указать вручную или выбрать из списка значений, предопределенных в Wireshark.
4. Полученный в итоге фильтр появится в нижней части экрана. По завершении щелкните на кнопке OK, чтобы ввести его на панели фильтров.

Диалоговое окно Display Filter Expression очень удобно для начинающих пользователей Wireshark, но, приобретя некоторый опыт составления фильтров, вы обнаружите, что вводить фильтрующие выражения вручную намного эффективнее. Структура синтаксиса для создания фильтров отображения очень проста, хотя и довольно эффективна.

Структура синтаксиса для создания фильтров отображения

Чем чаще вы станете пользоваться Wireshark, тем больше вам придется пользоваться фильтрами отображения непосредственно в главном окне данного приложения ради экономии времени. Правда, синтаксис для создания фильтров отображения несложен и придерживается стандартной структуры. Зачастую эта структура сосредоточена на сетевых протоколах и следует формату *протокол.средство.подчиненное_средство*. В этом нетрудно убедиться, глядя на диалоговое окно Display Filter Expression. А теперь рассмотрим ряд примеров, демонстрирующих порядок построения фильтров отображения.

Чаще всего фильтры перехвата или отображения применяются для просмотра пакетов по отдельному сетевому протоколу. Допустим, требуется провести диагностику проблемы, возникшей на уровне сетевого протокола TCP, и с этой целью перехватить только сетевой трафик по данному протоколу. Для этого вполне подойдет простой фильтр `tcp`.

А теперь рассмотрим эту задачу под другим углом зрения. Допустим, что в ходе диагностики проблемы на уровне сетевого протокола TCP довольно интенсивно использовалась утилита Ping, что привело к формированию большого объема сетевого трафика по протоколу ICMP. Чтобы отсеять этот трафик из файла перехвата, достаточно применить фильтр `expression !icmp`.

Операции сравнения позволяют сравнивать отдельные значения. Например, в ходе диагностики сетей, действующих по протоколу TCP/IP, нередко требуется просматривать все пакеты со ссылками на конкретный IP-адрес. С помощью операции сравнения на равенство (==) можно создать следующий фильтр, отображающий все пакеты с IP-адресом **`192.168.0.1`**:

```
ip.addr==192.168.0.1
```

А теперь допустим, что требуется просмотреть только те пакеты, длина которых не превышает 128 байт. С этой целью можно использовать операцию сравнения "меньше или равно" (**<=**) в следующем фильтре:

```
frame.len<=128
```

В табл. 4.4 перечислены все операции сравнения, используемые в фильтрующих выражениях, составляемых в Wireshark.

Таблица 4.4. Операции сравнения, используемые в фильтрах Wireshark

Операция	Описание
==	Равно
!=	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно

Логические операции позволяют объединять несколько фильтрующих выражений в одном операторе. И благодаря этому заметно повышается эффективность применяемых фильтров.

Допустим, требуется отобразить пакеты только по двум IP-адресам. Чтобы составить единое выражение для фильтрации пакетов, содержащих оба IP-адреса, можно воспользоваться логической операцией `or` следующим образом:

```
ip.addr==192.168.0.1 or ip.addr==192.168.0.2
```

В табл. 4.5 перечислены все логические операции, употребляемые в фильтрующих выражениях, составляемых в Wireshark.

Таблица 4.5. Логические операции, употребляемые в фильтрах Wireshark

Операция	Описание
`and`	Оба задаваемых условия должны быть истинны
`or`	Истинным должно быть хотя бы одно задаваемое условие
`xor`	Истинным должно быть одно и только одно задаваемое условие
`not`	Ни одно из задаваемых условий не истинно

Примеры выражений для фильтров отображения

Несмотря на то что принципы, по которым составляются фильтрующие выражения, довольно просты, в процессе создания новых фильтров для решения различных задач приходится употреблять ряд особых ключевых слов

и операций. В табл. 4.6 приведены некоторые примеры выражений для наиболее употребительных фильтров отображения, а полный их перечень можно найти в справочном руководстве по фильтрам отображения, доступном по адресу `http://www.wireshark.org/docs/dfref/`.

Таблица 4.6. Наиболее употребительные фильтры отображения

Фильтр	Описание
`!tcp.port==3389`	Отсеять сетевой трафик по протоколу RDP
`tcp.flags.syn==1`	Отобразить пакеты TCP с установленным флагом SYN
`tcp.flags.reset==1`	Отобразить пакеты TCP с установленным флагом RST
`!arp`	Удалить сетевой трафик по протоколу ARP
`http`	Отобразить весь сетевой трафик по протоколу HTTP
`tcp.port==23 || tcp.port==21`	Отобразить сетевой трафик по протоколу Telnet или FTP
`smtp || pop || imap`	Отобразить сетевой трафик электронной почты (по протоколу SMTP, POP или IMAP)

Сохранение фильтров

Начав однажды создавать немалое количество фильтров перехвата и отображения, можно в какой-то момент обнаружить, что некоторые из них применяются довольно часто. Правда, их не придется вводить всякий раз, когда они потребуются, поскольку в Wireshark предоставляется возможность сохранять фильтры для дальнейшего применения. Так, чтобы сохранить специальный фильтр перехвата, выполните следующие действия.

1. Выберите команду Capture⇨Capture Filters (Перехват⇨Фильтры перехвата), чтобы открыть диалоговое окно Capture Filter.
2. Создайте новый фильтр, щелкнув на кнопке со знаком "плюс" (+) в левой нижней части диалогового окна.
3. Введите имя нового фильтра в поле Filter Name (Имя фильтра).
4. Введите фильтрующее выражение в поле Filter String (Строка фильтра).
5. Щелкните на кнопке OK, чтобы сохранить фильтрующее выражение.

А для того чтобы сохранить специальный фильтр отображения, выполните приведенные ниже действия.

1. Введите фильтр на панели Filter, расположенной над панелью Packet List в главном окне Wireshark, а затем щелкните на кнопке с зеленой лентой на левом краю данной панели.
2. Выберите команду Save this Filter (Сохранить этот фильтр) из всплывающего меню. В отдельном диалоговом окне будет представлен список

сохраненных фильтров отображения (рис. 4.18). Здесь можно присвоить своему фильтру имя, прежде чем сохранять его, щелкнув на кнопке OK.

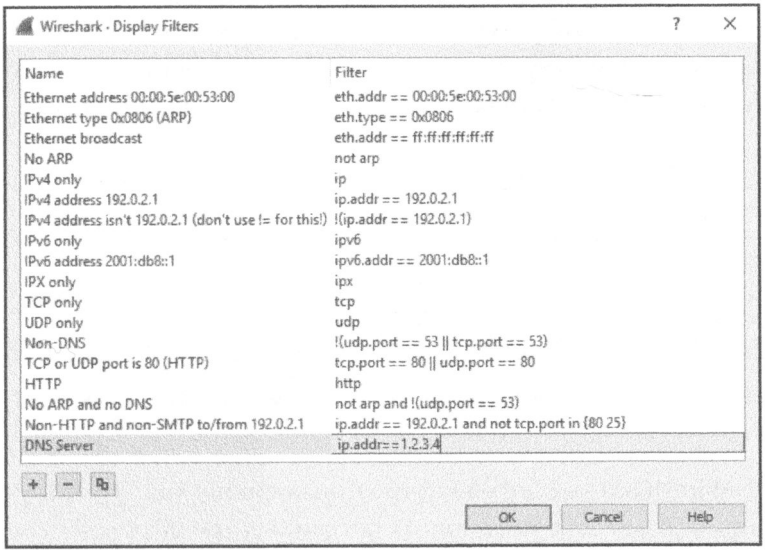


Рис. 4.18. Фильтры отображения можно сохранять непосредственно из главного окна Wireshark

Помещение фильтров отображения на панель инструментов

Если у вас имеются фильтры, которыми вы часто пользуетесь, для удобства взаимодействия с ними проще всего поместить их метки на упомянутой выше панели Filter. Для этого выполните следующие действия.

1. Введите свой фильтр на панели Filter, расположенной над панелью Packet List в главном окне программы Wireshark, а затем щелкните на кнопке со знаком "плюс" (+) справа на данной панели.
2. Ниже панели Filter появится новая панель, где вы можете указать имя своего фильтра в поле Label (Метка; рис. 4.19). Указанная вами метка послужит для представления фильтра на панели инструментов. Введя метку в данном поле, щелкните на кнопке OK, чтобы создать метку для быстрого доступа к вашему фильтру на панели Filter.

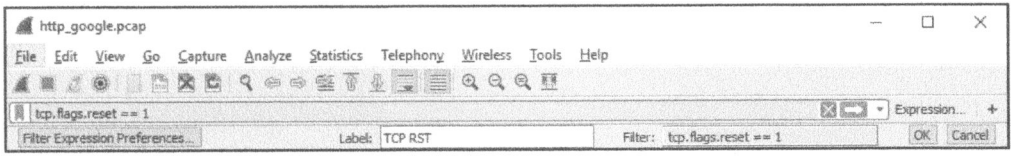


Рис. 4.19. Помещение метки быстрого доступа к фильтру на панель Filter

Как показано на рис. 4.20, в данном случае удалось создать метку быстрого доступа к фильтру для оперативного отображения любых пакетов TCP с установленным флагом RST. Фильтры, помещаемые на панель инструментов, сохраняются в файле конфигурации, упоминавшемся в главе 3, "Введение в Wireshark". Тем самым существенно расширяются ваши возможности выявлять в самых разных ситуациях затруднения, возникающие в сети на уровне пакетов.

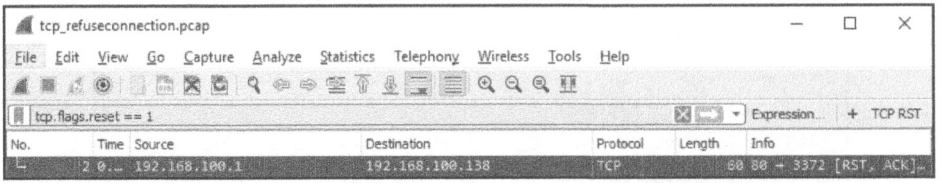


Рис. 4.20. Фильтрация пакетов с помощью меток быстрого доступа на панели инструментов

В состав Wireshark входит ряд встроенных фильтров, которые служат отличными примерами для создания собственных фильтров. Воспользуйтесь ими вместе со справочными страницами Wireshark при создании собственных фильтров. Мы будем часто пользоваться фильтрами в примерах, демонстрируемых далее в этой книге.

5

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ WIRESHARK

Итак, овладев основами приложения Wireshark, можете перейти к изучению его дополнительных возможностей для анализа пакетов и графического представления получаемых результатов. В этой главе будут рассмотрены некоторые из подобных возможностей и средств, включая окна Endpoints и Conversations, более сложные детали процесса преобразования имен, исследование протоколов, интерпретацию потоков, графическое представление ввода-вывода и многое другое.

Эти особые возможности Wireshark как графического средства анализа пакетов окажутся полезными на самых разных стадиях данного процесса. Поэтому постарайтесь хотя бы опробовать все рассматриваемые здесь дополнительные возможности Wireshark, прежде чем двигаться дальше, поскольку мы еще не раз вернемся к ним при рассмотрении примеров практического анализа пакетов в остальной части этой книги.

Конечные точки и сетевые диалоги

Для того чтобы данные передавались по сети, они должны перемещаться потоком хотя бы между двумя устройствами. Каждое устройство, передающее или принимающее данные в сети, представлено в приложении Wireshark так называемой *конечной точкой*. А передача данных между конечными точками

называется *диалогом*. Конечные точки и диалоги описываются в Wireshark на основании свойств передачи данных и, в частности, с помощью таких понятий, как адреса, применяемые в различных сетевых протоколах.

Конечные точки обозначаются несколькими адресами, которые присваиваются им на разных уровнях модели OSI. Например, на канальном уровне конечная точка получает MAC-адрес, который является однозначным адресом, встроенным в сетевое устройство, хотя он может быть видоизменен и тогда потенциально станет ненужным. А на сетевом уровне конечная точка получает IP-адрес, который может быть изменен в любой момент. О том, как используются оба эти типа адресов, речь пойдет в последующих главах.

На рис. 5.1 приведены два примера применения адресов для обозначения конечных точек в сетевых диалогах. В частности, диалог А состоит из двух конечных точек, обменивающихся данными на канальном уровне (по MAC-адресам). У конечной точки А имеется MAC-адрес **`00:ff:ac:ce:0b:de`**, тогда как у конечной точки Б – MAC-адрес **`00:ff:ac:e0:dc:0f`**. А диалог Б определяется двумя устройствами, обменивающимися данными на сетевом уровне (по IP-адресам). У конечной точки А имеется IP-адрес **`192.168.1.25`**, тогда как у конечной точки Б – IP-адрес **`192.168.1.30`**. Выясним, каким образом в Wireshark можно предоставить сведения о передаче данных по сети на основании конечных точек или диалогов.

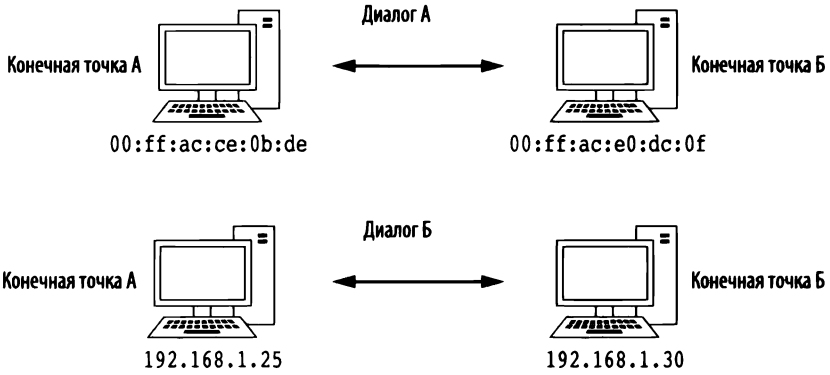


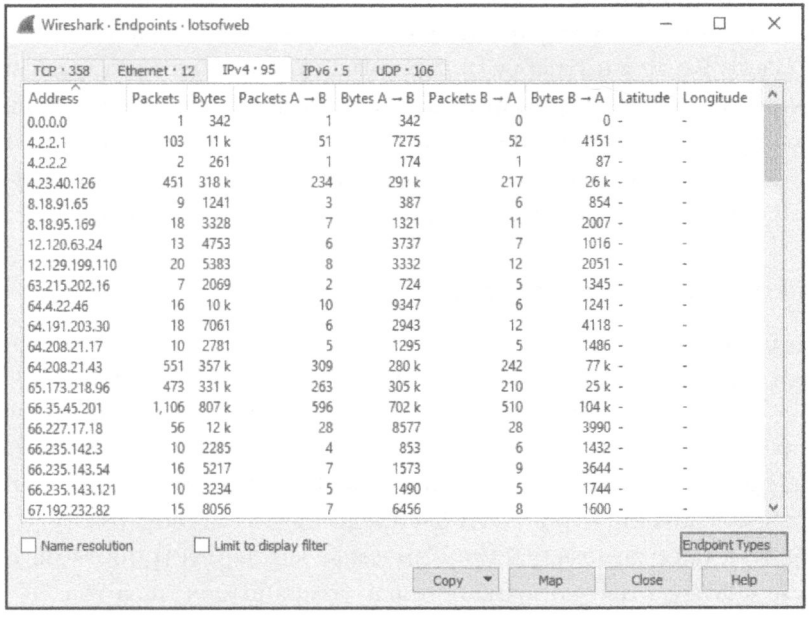
Рис. 5.1. Конечные точки и диалоги в сети

Просмотр статистики в конечных точках

Файл перехвата **`lotsofweb.pcapng`**

Анализируя сетевой трафик, можно выявить проблему в сети вплоть до конкретной конечной точки. В качестве примера откройте сначала файл перехвата `lotsofweb.pcapng`, а затем окно Endpoints, выбрав команду Statistics⇨Endpoints (Статистика⇨Конечные точки) из главного меню. В этом окне отображается ряд полез-

ных статистических данных по каждой конечной точке, включая адрес, количество переданных и принятых пакетов и байтов, как показано на рис. 5.2.



Address	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Latitude	Longitude
0.0.0.0	1	342	1	342	0	0	-	-
4.2.2.1	103	11 k	51	7275	52	4151	-	-
4.2.2.2	2	261	1	174	1	87	-	-
4.23.40.126	451	318 k	234	291 k	217	26 k	-	-
8.18.91.65	9	1241	3	387	6	854	-	-
8.18.95.169	18	3328	7	1321	11	2007	-	-
12.120.63.24	13	4753	6	3737	7	1016	-	-
12.129.199.110	20	5383	8	3332	12	2051	-	-
63.215.202.16	7	2069	2	724	5	1345	-	-
64.4.22.46	16	10 k	10	9347	6	1241	-	-
64.191.203.30	18	7061	6	2943	12	4118	-	-
64.208.21.17	10	2781	5	1295	5	1486	-	-
64.208.21.43	551	357 k	309	280 k	242	77 k	-	-
65.173.218.96	473	331 k	263	305 k	210	25 k	-	-
66.35.45.201	1,106	807 k	596	702 k	510	104 k	-	-
66.227.17.18	56	12 k	28	8577	28	3990	-	-
66.235.142.3	10	2285	4	853	6	1432	-	-
66.235.143.54	16	5217	7	1573	9	3644	-	-
66.235.143.121	10	3234	5	1490	5	1744	-	-
67.192.232.82	15	8056	7	6456	8	1600	-	-

Рис. 5.2. В окне Endpoints можно просмотреть статистические данные по каждой конечной точке в файле перехвата

На вкладках (TCP, Ethernet, IPv4, IPv6 и UDP), расположенных вверху данного окна, отображается количество конечных точек, организованных по отдельному сетевому протоколу. Чтобы отобразить конечные точки только по конкретному протоколу, щелкните на соответствующей вкладке. Чтобы добавить дополнительные вкладки для отсеивания конечных точек по отдельным протоколам, щелкните на кнопке Endpoint Types (Типы конечных точек) в правом нижнем углу экрана и выберите добавляемый сетевой протокол. Если же потребуется преобразование имен для просмотра адресов конечных точек (см. далее раздел "Преобразование имен"), установите флажок Name resolution слева внизу экрана. А если приходится анализировать крупный перехват и требуется отфильтровать отображаемые конечные точки, то можно применить фильтр отображения в главном окне Wireshark и установить флажок Limit to display filter (Наложить ограничение с помощью фильтра отображения) в окне Endpoints. Если этот флажок установлен, в данном окне будут отображаться только те конечные точки, которые соответствуют условию, заданному в фильтре отображения.

В окне Endpoints предоставляется еще одна удобная возможность отсеивать отдельные пакеты для отображения на панели Packet List. Это быстрый способ

углубленного анализа пакетов в отдельной конечной точке. С этой целью щелкните правой кнопкой мыши на конечной точке, чтобы выбрать имеющиеся варианты фильтрации. В открывшемся диалоговом окне вам представится возможность показать или скрыть пакеты, связанные с выбранной конечной точкой. Здесь же можно выбрать вариант Colorize (Выделить цветом), чтобы экспортировать адрес конечной точки непосредственно в правило выделения цветом (о правилах выделения цветом см. в главе 3, "Введение в Wireshark"). Таким образом, можно быстро выделить пакеты, связанные с отдельными конечными точками, чтобы оперативно находить их во время анализа.

Просмотр сетевых диалогов

Файл перехвата **`lotsofweb.pcapng`**

Если файл перехвата `lotsofweb.pcapng` все еще открыт, откройте окно Conversations, выбрав команду Statistics⇨Conversations (Статистика⇨Диалоги) из главного меню, чтобы отобразить все диалоги из данного файла перехвата, как показано на рис. 5.3. Окно Conversations аналогично окну Endpoints, но в нем отображаются по два адреса в каждой строке, чтобы представить сетевой диалог, а также пакеты и байты, передаваемые и принимаемые каждым устройством. В столбце **`Address A`** указывается конечная точка отправителя пакетов, а в столбце **`Address B`** – конечная точка получателя.

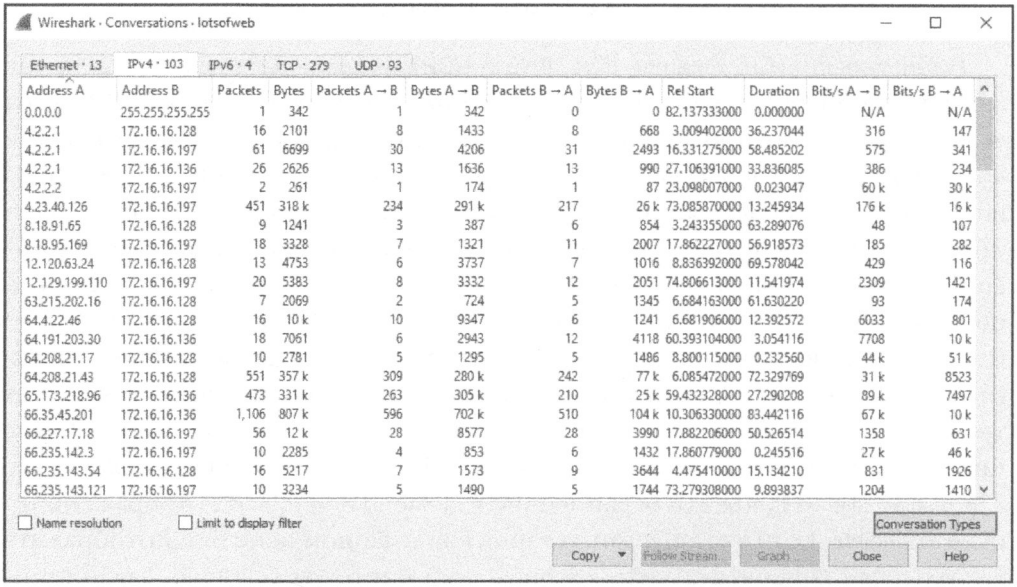


Рис. 5.3. В окне Conversations можно выделить каждый сетевой диалог из файла перехвата

Окно Conversations организовано по отдельным протоколам. Чтобы просмотреть сетевые диалоги по отдельному протоколу, щелкните на соответствующей вкладке у верхнего края данного окна или введите другие типы сетевых протоколов, щелкнув на кнопке Conversation Types справа внизу. Как и в окне Endpoints, здесь можно воспользоваться преобразованием имен, наложить ограничение на показываемые сетевые диалоги с помощью фильтра отображения, а также щелкнуть правой кнопкой мыши на отдельном диалоге, чтобы создать фильтры на основании конкретных диалогов. Фильтры, основанные на сетевых диалогах, удобны для углубленного анализа последовательностей обмена данными по сети, которые представляют особый интерес.

Выявление наиболее активных сетевых узлов с помощью конечных точек и диалогов

Файл перехвата **`lotsofweb.pcapng`**

Окнами Endpoints и Conversations удобно пользоваться при диагностике сети и особенно при попытке найти источник значительного объема сетевого трафика. Обратимся снова за примером к файлу перехвата `lotsofweb.pcapng`. Как подразумевает имя этого файла, он содержит сетевой трафик, формируемый по протоколу HTTP многими клиентами, просматривающими веб-сайты в Интернете. На рис. 5.4 приведен список конечных точек из этого файла, отсортированных по количеству байтов.

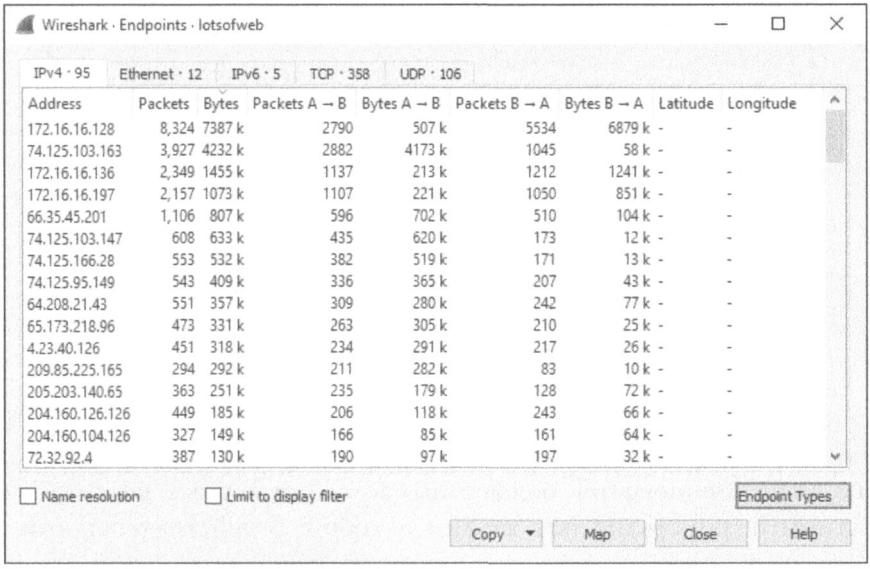


Рис. 5.4. В окне Endpoints показывается, какие сетевые узлы обмениваются данными больше всего

Обратите внимание на конечную точку, находящуюся по адресу **172.16.16.128** и служащую причиной самого интенсивного (в байтах) сетевого трафика. Адрес этой конечной точки является внутренним сетевым адресом, как поясняется в главе 7, "Протоколы сетевого уровня", а устройство, вызвавшее самый интенсивный обмен данными в перехваченном трафике, называется *наиболее активным сетевым узлом (top talker)*.

Вторая по интенсивности сетевого трафика конечная точка находится по адресу **74.125.103.163**, который является внешним (межсетевым) адресом. Если встретится внешний адрес, чтобы выяснить его сетевой узел, можно обратиться к базе данных WHOIS в поисках зарегистрированного владельца. В данном случае из реестра American Registry for Internet Numbers (ARIN – Американский регистратор интернет-адресов; `https://whois.arin.net/ui/`) выявится, что рассматриваемым здесь IP-адресом владеет компания Google (рис. 5.5).

Network	
Net Range	74.125.0.0 - 74.125.255.255
CIDR	74.125.0.0/16
Name	GOOGLE
Handle	NET-74-125-0-0-1
Parent	NET74 (NET-74-0-0-0-0)
Net Type	Direct Allocation
Origin AS	
Organization	Google Inc. (GOGL)
Registration Date	2007-03-13
Last Updated	2012-02-24
Comments	
RESTful Link	https://whois.arin.net/rest/net/NET-74-125-0-0-1
See Also	Related organization's POC records.
See Also	Related delegations.

*Рис. 5.5. Результаты поиска в реестре службы WHOIS по IP-адресу **74.125.103.163** указывают на то, что он принадлежит компании Google*

Принимая во внимание эту информацию, можно предположить одно из двух: конечные точки, находящиеся по адресам **172.16.16.128** и **74.125.103.163**, обмениваются большими массивами данных по сети с многими другими устройствами или же между собой. В действительности конечные точки, как правило, обмениваются данными между собой, что весьма характерно для самых активных пар конечных точек в сети. Чтобы убедиться в этом, откройте окно Conversations, выберите вкладку IPv4 и отсортируйте

ВЫЯВЛЕНИЕ ВЛАДЕЛЬЦЕВ IP-АДРЕСОВ СРЕДСТВАМИ WHOIS

Присваиванием IP-адресов занимаются разные органы в зависимости от их географического местоположения. В частности, ARIN отвечает за присваивание IP-адресов в Соединенных Штатах и соседних регионах, AfriNIC — на территории Африки, RIPE — в Европе, тогда как APNIC — Азии и Тихоокеанском регионе. Как правило, выявление владельца конкретного IP-адреса средствами WHOIS осуществляется на веб-сайте регистратора, отвечающего за этот IP-адрес. Разумеется, глядя на IP-адрес, трудно выяснить, какой региональный регистратор отвечает за него. Эту нелегкую задачу берут за себя веб-сайты вроде Robtex (`http://robtex.com/`), делая запрос в соответствующий реестр и предоставляя полученные результаты. Но даже если вы обратитесь к неверному регистратору, вас все равно перенаправят к верному регистратору.

список по количеству байтов. В итоге вы должны увидеть, что рассматриваемые здесь две конечные точки образуют диалог с наибольшим количеством переданных байтов. Сам характер передачи данных предполагает крупную загрузку, поскольку количество байтов, передаваемых из конечной точки с внешним адресом A (**`74.125.103.163`**), оказывается намного больше, чем количество байтов, передаваемых из конечной точки с внутренним адресом B (**`172.16.16.128`**), как показано на рис. 5.6.

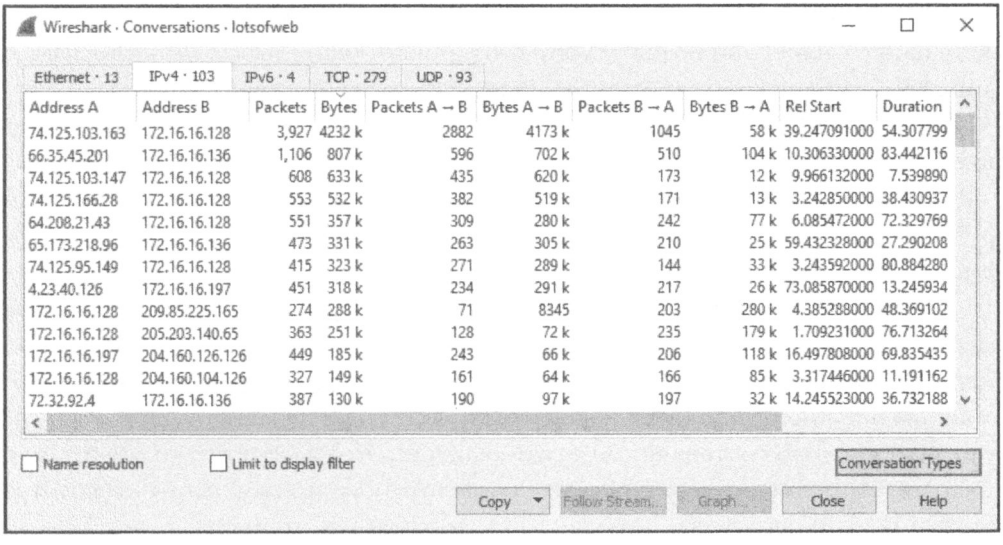


Рис. 5.6. Информация в окне Conversations подтверждает, что два наиболее активных сетевых узла обмениваются данными друг с другом

Этот сетевой диалог можно исследовать отдельно, применив следующий фильтр отображения:

```
ip.addr == 74.125.103.163 && ip.addr == 172.16.16.128
```

Прокрутив список пакетов, можно обнаружить несколько DNS-запросов к домену youtube.com в столбце **Info** на панели Packet List. Это вполне согласуется с представленными ранее выводами о том, что владельцем IP-адреса **74.125.103.163** является компания Google, поскольку ей принадлежит компания YouTube. О том, как пользоваться окнами Endpoints и Conversations в конкретных сценариях анализа пакетов, речь пойдет в остальных главах этой книги.

Статистические данные по иерархии сетевых протоколов

Файл перехвата **lotsofweb.pcapng**

Когда приходится иметь дело с незнакомыми файлами перехвата, иногда требуется выяснить распределение сетевого трафика по отдельным протоколам, т.е. ту долю в процентах, которая приходится на протокол TCP, IP, DHCP и т.д. в перехваченном трафике. Вместо того чтобы подсчитывать пакеты и подытоживать результаты вручную, эти сведения можно получить автоматически в Wireshark, открыв окно Protocol Hierarchy Statistics (Статистические данные по иерархии протоколов).

Так, если файл перехвата lotsofweb.pcapng все еще открыт, а любые применявшиеся ранее фильтры удалены, откройте окно Protocol Hierarchy Statistics (рис. 5.7), выбрав команду Statistics⇨Protocol Hierarchy из главного меню.

В окне Protocol Hierarchy Statistics предоставляется моментальный снимок того вида деятельности, который происходит в сети. Как следует из рис. 5.7, доля 100% приходится на сетевой трафик по протоколу Ethernet, 99,7% – на трафик по протоколу IPv4, 98% – на трафик по протоколу TCP и 13,5% – на трафик по протоколу HTTP, с помощью которого просматриваются веб-страницы в Интернете. Эти сведения служат отличным подспорьем для оценивания загрузки сети, особенно если имеется ясное представление о том, как обычно выглядит трафик в данной сети. Так, если заранее известно, что 10% сетевого трафика обычно приходится на протокол ARP, то из недавнего перехвата следует, что эта доля составляет 50%, это означает, что в сети что-то может быть неладно. Иногда само наличие пакетов некоторого протокола в сетевом трафике может представлять интерес. Так, если в сети отсутствуют устройства, настроенные на применение протокола STP (Spanning Tree Protocol – протокол связующего дерева), то его появление в иерархии сетевых протоколов может означать, что соответствующие устройства настроены неверно.

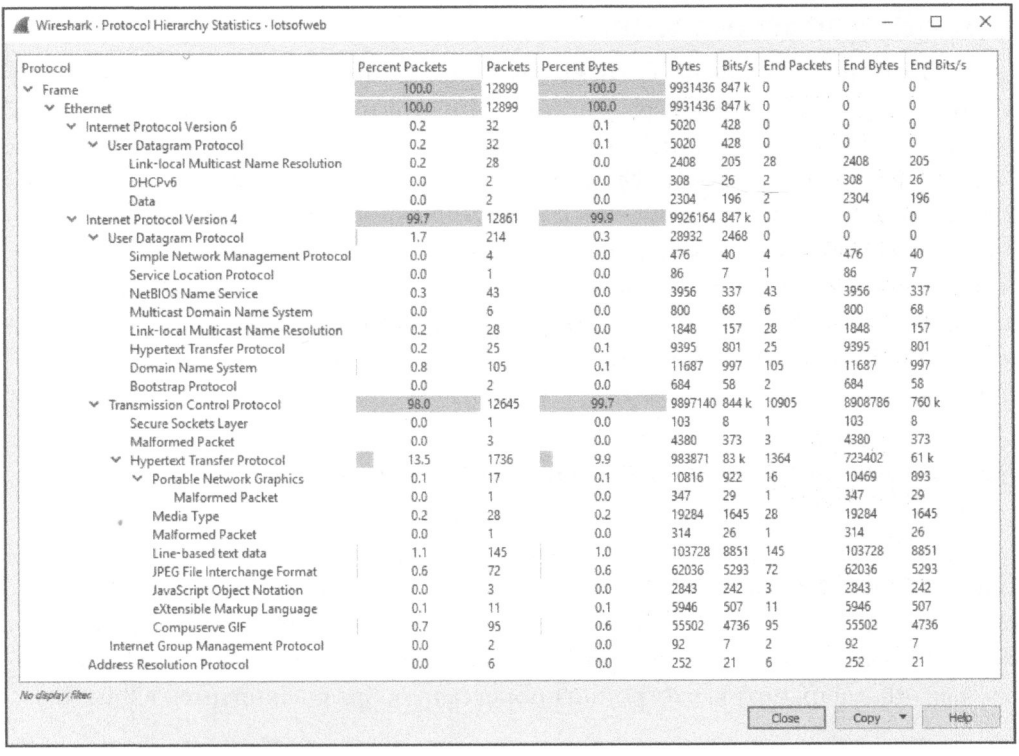


Рис. 5.7. В окне Protocol Hierarchy Statistics показывается распределение сетевого трафика по отдельным протоколам

Со временем вы обнаружите, что окном Protocol Hierarchy Statistics можно пользоваться для профилирования пользователей и устройств в сети, просто глядя на распределение применяемых протоколов. Например, большой объем сетевого трафика по протоколу HTTP может свидетельствовать об интенсивном просмотре веб-страниц в Интернете. Кроме того, вы сможете выявлять отдельные устройства в сети, просто глядя на трафик из сетевого сегмента, принадлежащего конкретному подразделению организации. Например, в отделе информационных технологий могут применяться дополнительные административные протоколы вроде ICMP или SNMP, в отделе обслуживания заказчиков – наблюдаться большой объем почтового трафика по протоколу SMTP, а вредный молодой специалист может незаметно наводнить сеть трафиком, тихо наслаждаясь в своем уголке многопользовательской ролевой игрой *World of Warcraft*!

Преобразование имен

Данные передаются по сети между конечными точками с помощью различных буквенно-цифровых систем адресации, которые зачастую оказываются слишком длинными и сложными для запоминания, например, MAC-адрес **`00:16:ce:6e:8b:24`**, адрес IPv4 **`192.168.47.122`** или адрес IPv6 **`2001:db8:a0b:12f0::1`**. Чтобы облегчить запоминание этих адресов им назначают удобные и запоминающиеся имена, а процесс нахождения адреса по имени называется *преобразованием имен* (*name resolution*). Например, запомнить адрес `google.com` намного легче, чем адрес **`216.58.217.238`**. Благодаря связыванию удобочитаемых имен с неудобочитаемыми адресами облегчается их запоминание и распознавание.

Активизация процесса преобразования имен

При отображении данных из пакетов в Wireshark можно пользоваться преобразованием имен, чтобы упростить их анализ. Чтобы воспользоваться в Wireshark преобразованием имен, выберите команду Edit⇨Preferences⇨Name Resolution (Правка⇨Глобальные параметры⇨Преобразование имен) из главного меню. В итоге откроется окно (рис. 5.8), в котором доступны основные, описываемые ниже параметры настройки процесса преобразования имен в Wireshark.

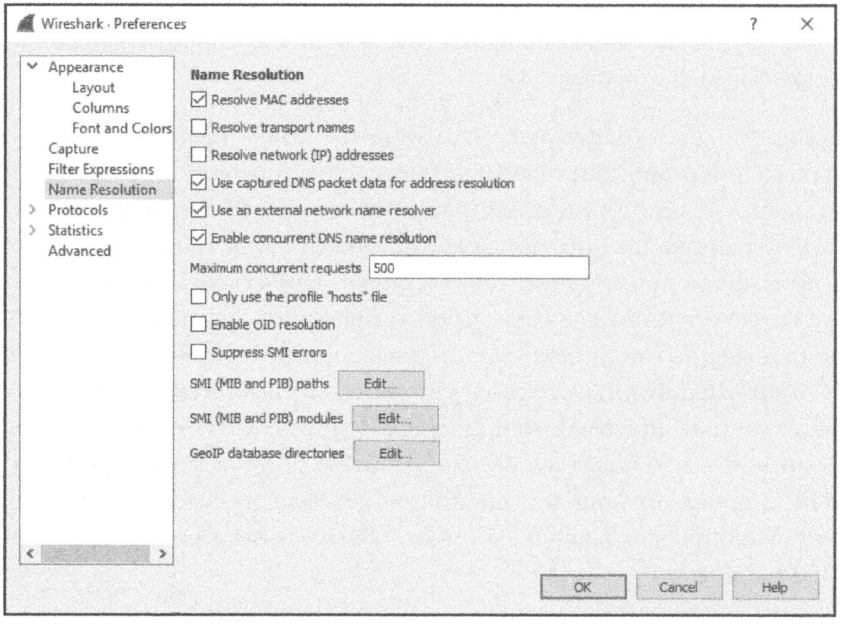


Рис. 5.8. Активизация процесса преобразования имен в окне Preferences. Среди первых трех флажков, относящихся к типам преобразования имен, здесь выбран только флажок Resolve MAC addresses

- **Resolve MAC addresses (Преобразовать MAC-адреса).** В этом режиме применяется протокол ARP, чтобы попытаться преобразовать MAC-адреса второго уровня (например, такие как `00:09:5b:01:02:03`) в адреса третьего уровня (например, `10.100.12.1`). Если подобные попытки потерпят неудачу, Wireshark выберет из своего каталога файл `ethers`, чтобы предпринять новую попытку преобразования адресов. А в качестве последнего средства Wireshark прибегнет к преобразованию первых трех байтов MAC-адреса в имя устройства, обозначаемое производителем по стандарту IEEE (например, `Netgear_01:02:03`).
- **Resolve transport names (Преобразовать транспортные имена).** В этом режиме предпринимается попытка преобразовать номер порта в связанное с ним имя протокола – например, отобразить порт **80** как **http**. Это удобно, когда в ходе анализа встречается неизвестный порт и непонятно, с какой службой он обычно связан.
- **Resolve network (IP) addresses (Преобразовать сетевые (IP) адреса).** В этом режиме предпринимается попытка преобразовать адрес третьего уровня (например, **192.168.1.50**) в удобочитаемое имя DNS вроде `MarketingPC1.domain.com`. Это удобно для выяснения назначения или владельца системы, при условии, что у нее имеется описательное имя.

В разделе Name Resolution окна Preferences, приведенного на рис. 5.8, имеются и другие глобальные, описываемые ниже параметры настройки преобразования имен.

- **Use captured DNS packet data for address resolution (Использовать данные из перехваченных пакетов DNS для преобразования адресов).** В этом режиме производится синтаксический анализ данных из перехваченных пакетов DNS для преобразования IP-адресов в имена DNS.
- **Use an external network name resolver (Использовать внешний преобразователь сетевых имен).** В этом режиме допускается формировать запросы к DNS-серверу той машиной, где выполняется анализ пакетов, для преобразования IP-адресов в имена DNS. Это удобно в том случае, если требуется воспользоваться службой DNS для преобразования имен, но анализируемый перехваченный трафик не содержит соответствующие пакеты DNS.
- **Maximum concurrent requests (Максимум параллельных запросов).** В этом режиме устанавливается ограничение на количество параллельных DNS-запросов, одновременно ожидающих обработки. Этот режим задается в том случае, если в перехваченном трафике формируется слишком много DNS-запросов, что может отрицательно сказаться на пропускной способности сети или производительности DNS-сервера.

- **Only use the profile "hosts" file (Использовать только профильные файлы сетевых узлов hosts).** В этом режиме работа системы DNS ограничивается только файлом **hosts**, связанным с активным профилем Wireshark. О том, как пользоваться этим файлом, речь пойдет в следующем разделе.

Изменения, вносимые в окне Preferences, сохраняются после закрытия и повторного открытия Wireshark. Чтобы оперативно вносить изменения в процесс преобразования имен, не сохраняя их, настраивайте параметры преобразования имен, выбирая команду View⇨Name Resolution из главного меню. У вас есть возможность активизировать или отменять преобразование имен для физических, транспортных и сетевых адресов.

Чтобы сделать файлы перехвата более удобочитаемыми, сэкономив немало времени в определенных случаях, воспользуйтесь различными инструментальными средствами преобразования имен. Например, преобразование имен DNS помогает сразу же выяснить имя компьютера, который вы пытаетесь идентифицировать как отправителя отдельного пакета.

Потенциальные недостатки преобразования имен

Принимая во внимание преимущества преобразования имен, его применение считается вполне очевидным, но у него имеется и ряд потенциальных недостатков. Прежде всего, преобразование сетевых имен может потерпеть неудачу, если отсутствует подходящий DNS-сервер для предоставления имен, связанных с IP-адресом. Сведения о преобразовании имен не сохраняются в файле перехвата, и поэтому процесс их преобразования должен происходить всякий раз, когда открывается этот файл. Если вы перехватываете пакеты в одной сети, а затем открываете файл перехвата пакетов в другой сети, то ваша система не сможет получить доступ к DNS-серверам из исходной сети, а следовательно, преобразование имен потерпит неудачу.

Кроме того, преобразование имен требует дополнительных издержек на обработку. Если приходится иметь дело с очень крупным файлом перехвата, то, возможно, придется отказаться от преобразования имен ради экономии системных ресурсов. Если же попробовать открыть крупный файл перехвата и при этом система не сможет справиться с нагрузкой или приложение Wireshark вообще завершится аварийно, в таком случае может помочь отмена преобразования имен.

Еще одно затруднение возникает, когда при преобразовании сетевых имен задействуется служба DNS. В этом случае могут быть сформированы лишние пакеты, способные засорить файл перехвата сетевым трафиком, посылаемым DNS-серверам для преобразования адресов. Дело усложняется еще и тем, что если анализируемый файл перехвата содержит зловредные IP-адреса, то

попытки преобразовать их в имена могут привести к формированию запросов к инфраструктуре, которая контролируется атакующим злоумышленником и может предупредить его о том, что о его злых намерениях сделать данную систему своей мишенью известно. Чтобы снизить риск засорения файла перехвата пакетов или непреднамеренного связывания с атакующим злоумышленником, сбросьте флажок Use an external network name resolver в разделе Name Resolution окна Preferences.

Применение специального файла hosts

Отслеживать трафик из многих сетевых узлов в крупных файлах перехвата может оказаться довольно утомительным занятием, особенно в том случае, когда преобразование имен недоступно. В таком случае может, в частности, помочь разметка систем вручную на основании их IP-адресов с помощью специального файла `hosts`, создаваемого для сетевых узлов (или хостов) в Wireshark. Это текстовый файл, содержащий список IP-адресов и соответствующих имен. Файл `hosts` может быть использован для назначения в Wireshark соответствующих имен адресам для краткой ссылки. Эти имена будут отображены на панели Packet List.

Чтобы воспользоваться файлом `hosts`, выполните следующие действия.

1. Выберите команду Edit⇨Preferences⇨Name Resolution из главного меню и установите флажок Only use the profile "hosts" file.
2. Создайте новый файл в с помощью программы Notepad в системе Windows или аналогичном текстовом редакторе. Этот файл должен содержать по одной записи в каждой строке с IP-адресом и тем именем, в которое он преобразуется (рис. 5.9). Имя, выбираемое справа в каждой строке, будет отображаться в списке пакетов всякий раз, когда в Wireshark встречается IP-адрес, указываемый слева.

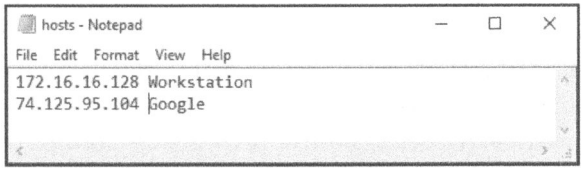


*Рис. 5.9. Создание файла **hosts** в Wireshark*

3. Сохраните созданный файл как обычный текстовый файл под именем **hosts** в соответствующем каталоге, как показано ниже. Обратите внимание, что у файла **hosts** нет расширения!
   - Windows:
   - `<USERPROFILE>\Application Data\Wireshark\hosts`.

- Mac OS X:
- /Users/<*имя_пользователя*>/.wireshark/hosts.
- Linux: /home/<username>/.wireshark/hosts.

Откройте далее файл перехвата, и тогда любые IP-адреса из файла hosts должны быть обозначены соответствующими именами, как показано рис. 5.10. Вместо IP-адресов в столбцах **Source** (Отправитель) и **Destination** (Получатель) на панели Packet List появятся более описательные имена.

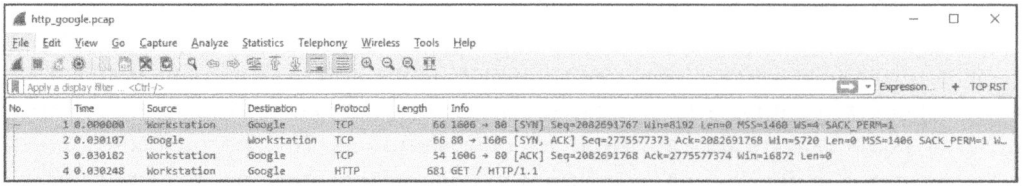


Рис. 5.10. Преобразование имен из специального файла ***hosts*** *в Wireshark*

Такое применение специальных файлов hosts может значительно улучшить вашу способность распознавать определенные сетевые узлы (или хосты) во время анализа пакетов. Работая в команде сетевых аналитиков, рассмотрите возможность обмениваться со своими коллегами файлом hosts, содержащим известные ресурсы. Это поможет вашей команде быстро распознавать системы со статическими адресами, в том числе серверы и маршрутизаторы.

ПРИМЕЧАНИЕ *Если ваш файл* **hosts** *окажется неработоспособным, убедитесь, не дополнили ли вы случайно его имя расширением файла* **.txt**. *Этот файл должен просто носить имя* **hosts** *без всякого расширения!*

Инициируемое вручную преобразование имен

В приложении Wireshark имеется также возможность по требованию активизировать на время преобразование имен. Для этого достаточно щелкнуть правой кнопкой мыши на пакете в панели Packet List и выбрать команду Edit Resolved Name (Изменить преобразуемое имя) из контекстного меню. В открывшемся окне можно указать имя, соответствующее конкретному адресу, как метку. Такое преобразование будет утрачено, как только файл перехвата будет закрыт, хотя это краткий способ пометить адрес, не внося никаких постоянных изменений, которые придется отменять впоследствии. Я часто пользуюсь данным способом, поскольку это немного проще, чем править файл hosts вручную при анализе каждого файла с перехваченными пакетами.

Дешифрирование сетевых протоколов

К числу самых сильных сторон Wireshark относится поддержка анализа тысяч сетевых протоколов. И такая возможность объясняется тем, что Wireshark является приложением с открытым исходным кодом, а следовательно, служит основанием для создания *дешифраторов протоколов*. Они дают возможность распознавать и декодировать различные поля сетевого протокола в Wireshark, чтобы отобразить сетевой протокол в пользовательском интерфейсе. Для интерпретации каждого пакета в Wireshark совместно используется несколько дешифраторов. Например, дешифратор сетевого протокола ICMP позволяет Wireshark выяснить, что IP-пакет содержит данные из протокола ICMP, извлечь тип и код протокола ICMP и отформатировать его поля для отображения в столбце **Info** панели Packet List.

Дешифратор можно рассматривать как интерпретатор исходных данных в приложении Wireshark. Чтобы сетевой протокол нашел поддержку в Wireshark, для него должен быть отдельный дешифратор в данном приложении, а иначе вам придется написать свой дешифратор сетевых протоколов.

Смена дешифратора

Файл перехвата **wrongdissector.pcapng**

Дешифраторы применяются в Wireshark для того, чтобы обнаружить отдельные протоколы и выяснить, как отобразить сетевую информацию. К сожалению, Wireshark далеко не всегда делает правильный выбор дешифратора для применения к пакетам. И это особенно справедливо, когда в сетевом протоколе применяется нестандартная конфигурация, в том числе нестандартный порт, который зачастую настраивается сетевыми администраторами из соображений безопасности или сотрудниками организации, пытающимися обойти средства управления доступом.

Если дешифраторы неверно применяются в Wireshark, их выбор можно переопределить. Например, откройте файл трассировки `wrongdissector.pcapng`, содержащий немало сведений об обмене данными между двумя компьютерами по сетевому протоколу SSL (Secure Socket Layer – уровень защищенных сокетов), применяемый для шифрованного обмена данными между хостами. При обычных условиях просмотр сетевого трафика по протоколу SSL в Wireshark не даст особенно полезных сведений в силу того, что они зашифрованы. Но здесь определенно что-то не так. Если вы просмотрите содержимое нескольких таких пакетов, щелкнув на них и исследовав содержимое панели Packet Bytes, то обнаружите сетевой трафик, представленный простым текстом. Так, если проанализировать пакет 4, то в нем можно обнаружить

упоминание о приложении FileZilla для работы с FTP-сервером. А в ряде следующих пакетов ясно показан запрос и ответ как имени пользователя, так и пароля.

Если бы это был сетевой трафик по протоколу SSL, вы не смогли бы прочитать любые данные, содержащиеся в пакетах, и вряд ли увидели бы все имена пользователей и пароли, передаваемые в виде открытого текста (рис. 5.11). Принимая во внимание представленные здесь сведения, можно с уверенностью предположить, что это скорее трафик по протоколу FTP, а не SSL. Этот трафик, скорее всего, будет интерпретирован в Wireshark как относящийся к протоколу SSL, поскольку в нем употребляется порт **443**, как следует из сведений, приведенных в столбце **Info**. А ведь это стандартный порт, используемый в сетевом протоколе HTTPS (т.е. надстройке HTTP над SSL).

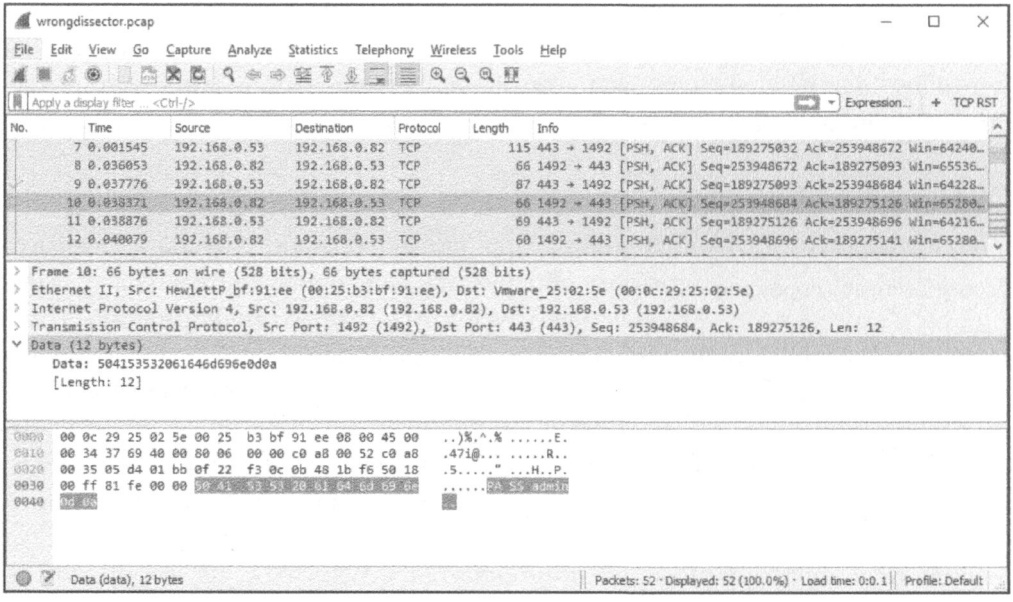


Рис. 5.11. Если имена пользователей и пароли представлены в виде открытого текста, это похоже скорее на протокол FTP, а не SSL!

Чтобы устранить подобное затруднение, можно применить в Wireshark *принудительную расшифровку* с целью воспользоваться дешифратором протокола FTP к анализируемым пакетам, выполнив следующие действия.

1. Щелкните правой кнопкой мыши на избранном пакете SSL (например, на пакете 30) в столбце **Protocol** и выберите из контекстного меню команду Decode As (Расшифровать как), чтобы открыть новое диалоговое окно.

2. Дайте Wireshark команду расшифровывать весь трафик TCP, проходящий через порт **443**, как FTP, выбрав вариант TCP port из списка в столбце **Field** (Поле), введя номер порта **443** в столбце **Value** (Значение) и выбрав вариант FTP из списка в столбце **Current** (Текущий протокол), как показано на рис. 5.12.

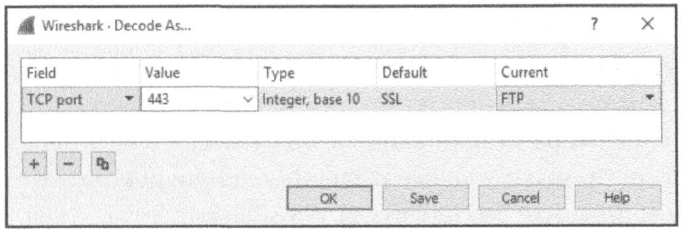


Рис. 5.12. В диалоговом окне Decode As... можно задать условия для принудительной расшифровки анализируемых пакетов

3. Щелкните на кнопке OK, чтобы увидеть изменения, немедленно внесенные в файл перехвата.

Данные будут декодированы как трафик по протоколу FTP, что даст возможность проанализировать их на панели Packet List, не особенно вдаваясь в отдельные байты (рис. 5.13).

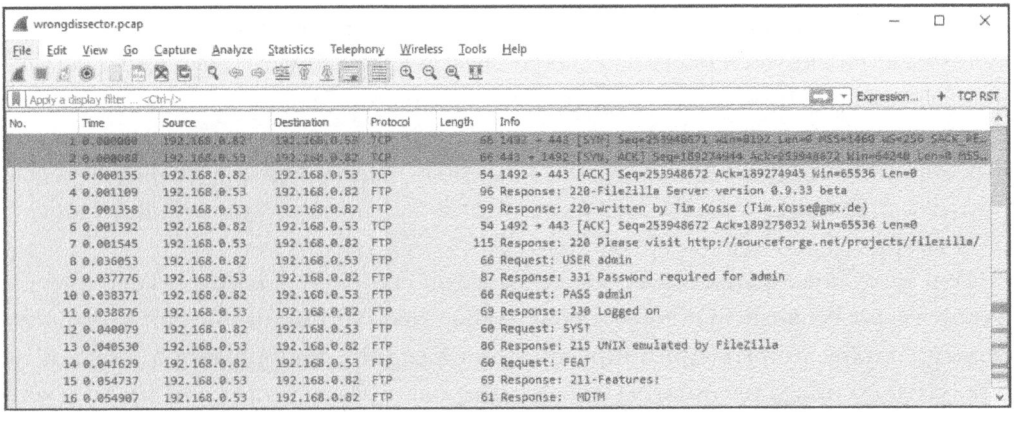


Рис. 5.13. Просмотр правильно расшифрованного трафика FTP

Возможностью принудительной расшифровки можно пользоваться многократно в одном и том же файле перехвата. Принудительные расшифровки, установленные в диалоговом окне Decode As..., будут автоматически отслеживаться в Wireshark. В этом окне можно просматривать и изменять все созданные до сих пор принудительные расшифровки.

По умолчанию принудительные расшифровки не сохраняются при закрытии файла перехвата. Но это положение можно исправить, щелкнув на кнопке Save в диалоговом окне Decode As.... В итоге правила расшифровки протоколов будут сохранены в профиле текущего пользователя Wireshark. Они будут применяться из данного профиля при открытии любого файла перехвата. Сохраненные правила расшифровки можно удалить, щелкнув на кнопке со знаком "минус" (-) в данном окне.

Сохраненные правила расшифровки можно легко забыть, что может привести к большому недоразумению для тех, кто к этому не готов, поэтому обращайтесь с правилами расшифровки благоразумно. Чтобы уберечься от подобной оплошности, принудительные расшифровки не рекомендуется сохранять в своем главном профиле пользователя Wireshark.

Просмотр исходного кода дешифраторов

Прелесть работы в приложении с открытым исходным кодом состоит в том, что при возникновении каких-нибудь недоразумений можно всегда просмотреть исходный код и выяснить их причину. И это особенно удобно при попытке выяснить причины, по которым конкретный протокол интерпретируется неверно. Ведь для этого достаточно проанализировать исходный код соответствующего дешифратора.

Просмотреть и проанализировать исходный код дешифраторов сетевых протоколов можно непосредственно на веб-сайте, посвященном приложению Wireshark, щелкнув сначала на ссылке Develop (Разработка), а затем на ссылке Browse the Code (Просмотр кода). По этой ссылке произойдет переход к хранилищу исходного кода Wireshark, где можно просмотреть код выпуска последних версий Wireshark. В частности, исходные файлы дешифраторов сетевых протоколов находятся в каталоге `epan/dissectors`, где исходный файл каждого дешифратора обозначен именем `packet-<`*`имя_протокола`*`>.c`.

Эти исходные файлы могут оказаться довольно сложными, но все они соответствуют общему шаблону и снабжены довольно подробными комментариями. Чтобы понять принцип действия каждого дешифратора, совсем не обязательно обладать опытом программирования на языке C. Если же требуется досконально разобраться в том, что отображается в Wireshark, рекомендуется начать просмотр и анализ с дешифраторов самых простых сетевых протоколов.

Отслеживание потоков

Файл перехвата **`http_google.pcapng`**

К числу наиболее удовлетворяющих потребностям анализа пакетов относится возможность повторно соби-

рать в Wireshark данные из многих пакетов в едином удобочитаемом формате, нередко называемом *выпиской из пакетов*. Благодаря этому отпадает необходимость просматривать данные, посылаемые от клиента к серверу, мелкими фрагментами при переходе от одного пакета к другому. При *отслеживании потока* данные сортируются с целью упростить их просмотр.

Ниже перечислены типы потоков, которые можно отслеживать.

- **Поток TCP.** В этот поток собираются данные из тех протоколов, где применяется протокол TCP, например, из сетевых протоколов HTTP и FTP.
- **Поток UDP.** В этот поток собираются данные из тех протоколов, где применяется протокол UDP, например, из сетевого протокола DNS.
- **Поток SSL.** В этот поток собираются данные из тех протоколов, где они шифруются. Для дешифровки сетевого трафика необходимо предоставить соответствующие ключи.
- **Поток HTTP.** В этот поток собираются данные из протокола HTTP. Это удобно для отслеживания данных HTTP через поток TCP без полной расшифровки полезной информации из протокола HTTP.

В качестве примера рассмотрим простую транзакцию по протоколу HTTP в файле перехвата `http_google.pcapng`. С этой целью щелкните сначала на любом из пакетов TCP или HTTP в этом файле, затем щелкните правой кнопкой мыши на выбранном пакете и выберите команду Follow⇨TCP Stream (Отслеживать⇨Поток TCP) из контекстного меню. В итоге будет получен единый поток TCP и открыта выписка из диалога в отдельном окне, как показано на рис. 5.14.

Текст, отображаемый в окне Follow TCP Stream, выделен двумя цветами: красным (более светлым оттенком серого на рис. 5.14) — текст, обозначающий сетевой трафик, проходящий от отправителя к получателю, а синим (более темным оттенком серого на рис. 5.14) — текст, обозначающий сетевой трафик, проходящий в противоположном направлении: от получателя к отправителю. Цвет связан с той стороной, которая инициировала обмен данными. В данном примере установление сетевого соединения с веб-сервером инициировал клиент, и поэтому его трафик выделен красным цветом.

Обмен данными в потоке TCP начинается с исходного запроса по методу GET корневого каталога (/) на веб-сервере и продолжается ответом сервера в форме `HTTP/1.1 200 OK` об успешной обработке запроса. По тому же самому образцу происходит обмен данными и в других потоках перехваченных пакетов по мере того, как клиент запрашивает отдельные файлы, а сервер присылает их в ответ. В данном примере можно увидеть, что пользователь просматривает начальную страницу веб-сайта Google. Но вместо того чтобы просматривать

пакеты по очереди, можно без труда прокрутить выписку из пакетов. По существу, вы видите то же самое, что и конечный пользователь, но только изнутри.

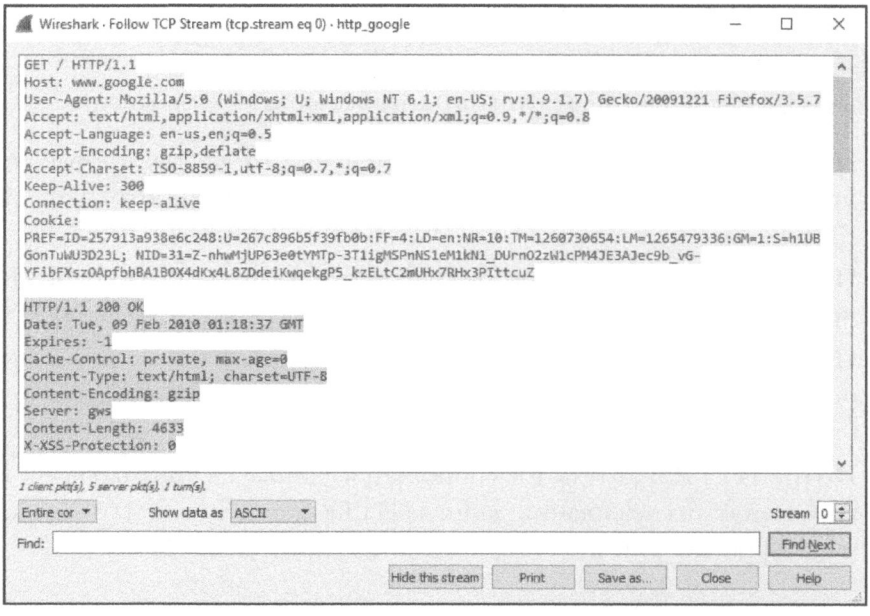


Рис. 5.14. В окне Follow TCP Stream повторно собранные передаваемые данные представлены в удобочитаемом формате

Помимо просмотра исходных данных, в этом окне можно производить поиск в тексте, сохранять его в файле, выводить на печать или выбирать представление данных в коде ASCII, EBCDIC, шестнадцатеричном виде или в форме массива на языке C. Все эти возможности, упрощающие анализ крупных массивов данных, находятся в нижней части данного окна.

Отслеживание потоков SSL

Чтобы отследить потоки TCP и UDP, достаточно выполнить всего пару щелчков, но для просмотра потоков SSL в удобочитаемом формате придется выполнить ряд дополнительных действий. Сетевой трафик по протоколу SSL зашифрован, поэтому необходимо предоставить секретный ключ, связанный с сервером, отвечающим за шифрованный трафик. Методика получения такого ключа зависит от конкретной серверной технологии, и поэтому ее рассмотрение выходит за рамки этой книги. Но как только вы получите секретный ключ, загрузите его в Wireshark, выполнив следующие действия.

1. Перейдите к глобальным параметрам настройки Wireshark, выбрав команду Edit⇨Preferences из главного меню.

2. Разверните раздел Protocols в открывшемся окне и щелкните на заголовке протокола **SSL**, как показано на рис. 5.15. Щелкните на кнопке Edit рядом с меткой RSA keys list (Список ключей RSA).

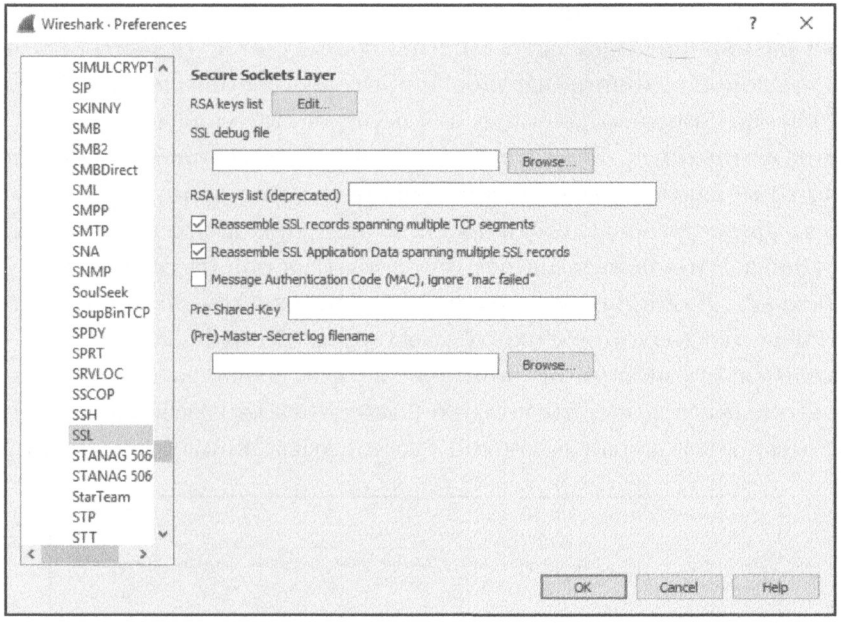


Рис. 5.15. Ввод сведений о дешифровке по протоколу SSL

3. Щелкните на кнопке со знаком "плюс" (+).
4. Предоставьте требующиеся сведения. К их числу относится IP-адрес сервера, отвечающего за шифрование, номер порта, сетевой протокол, местонахождение файла ключей и пароль к этому файлу, если таковой используется.
5. Перезапустите Wireshark.

В итоге у вас должна появиться возможность перехватывать зашифрованный сетевой трафик, проходящий между клиентом и сервером. Щелкните правой кнопкой мыши на пакете HTTPS и выберите команду Follow⇨SSL Stream (Отслеживать⇨Поток SSL), чтобы увидеть выписку из расшифрованного текста пакетов.

Возможность просматривать выписки из пакетов относится к числу наиболее употребительных при анализе пакетов в Wireshark, и вам придется ею часто пользоваться, чтобы быстро выяснить, какие сетевые протоколы при этом применяются. В последующих главах книги будет рассмотрен ряд других сценариев, основанных на просмотре выписок из пакетов.

Длина пакетов

Файл перехвата **download-slow.pcapng**

Размер одного пакета или группы пакетов может сообщить немало интересного о сложившейся ситуации в сети. При обычных обстоятельствах максимальный размер фрейма в сети Ethernet составляет 1518 байт. Если вычесть из этой числовой величины заголовки протоколов Ethernet, IP и TCP, то останется 1460 байт, предназначенных для передачи заголовка протокола седьмого уровня или данных. Если известны минимальные требования к передаче пакетов, то можно начать с анализа распределения длин пакетов в перехваченном трафике, чтобы сделать обоснованное предположение о составе данного трафика. Это очень помогает при попытках понять состав крупных файлов перехвата. Для просмотра распределения пакетов по длине в Wireshark предоставляется диалоговое окно Packet Lengths (Длины пакетов).

Обратимся к конкретному примеру из файла перехвата `download-slow.pcapng`. Открыв его, выберите команду Statistics⇨Packet Lengths из главного меню. В итоге откроется диалоговое окно Packet Lengths, приведенное на рис. 5.16.

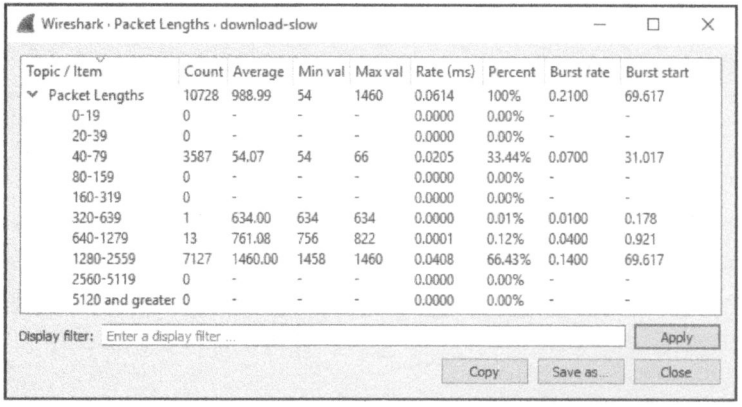


Рис. 5.16. Диалоговое окно Packet Lengths помогает сделать обоснованное предположение о сетевом трафике в файле перехвата

Обратите особое внимание на строку со статическими данными о пакетах длиной от 1280 до 2559 байт. Крупные пакеты вроде этих обычно свидетельствуют о передаче данных, тогда как более мелкие пакеты — о последовательностях управления протоколами. В данном случае наблюдается немалая доля крупных пакетов (66,43%). Даже не просматривая пакеты в файле перехвата, можно сделать вполне обоснованное предположение, что перехваченный трафик содержит одну или несколько передач данных, которые могут принимать форму загрузки по протоколу HTTP, выгрузки по протоколу FTP или любой другой операции в сети, где данные передаются между хостами.

Длина большинства оставшихся пакетов (33,44%) находится в пределах от 40 до 79 байт. К этой категории обычно относятся управляющие пакеты TCP, не несущие полезную информацию. Рассмотрим типичный размер заголовков протоколов. Так, заголовок протокола Ethernet занимает 14 байт (плюс 4 байта на циклический избыточный код CRC), заголовок протокола IP – как минимум 20 байт, а пакет TCP без данных или параметров – те же 20 байт. Это означает, что длина стандартных управляющих пакетов TCP (например, пакетов SYN, ACK, RST и FIN) составит около 54 байт и укладывается в рассматриваемые здесь пределы. Разумеется, эта длина увеличится, если добавить параметры протокола IP или TCP. Более подробно сетевые протоколы IP и TCP рассматриваются в главах 7, "Протоколы сетевого уровня", и 8, "Протоколы транспортного уровня", соответственно.

Анализ длин пакетов позволяет составить общее представление о крупном перехваченном трафике. Если в нем имеется много крупных пакетов, то можно с уверенностью предположить, что в сети передается большой объем данных. Если же длина большинства пакетов невелика, это означает, что передается немного данных, и можно предположить, что перехваченный трафик состоит из команд управления сетевыми протоколами. Но это не правила, которые следует считать непреложными, а всего лишь предположения, помогающие приступить к более углубленному анализу.

Составление графиков

Графики служат основой анализа пакетов и одним из лучших способов получения итогового представления о массиве данных. В состав Wireshark входит несколько средств для составления графиков, помогающих лучше понять перехваченные данные. И в первую очередь – это возможности графического представления ввода-вывода.

Просмотр графиков ввода-вывода

Файлы перехвата **download-fast.pcapng**, **download-slow.pcapng**, **http_espn.pcapng**

В окне IO Graph (График ввода-вывода) предоставляется возможность построить график передачи данных в сети. Такие графики позволяют быстро выявлять всплески и провалы в работе канала связи, обнаруживать задержки в производительности отдельных протоколов и сравнивать параллельные потоки данных.

В качестве примера построения графика ввода-вывода при загрузке файла на компьютер из Интернета обратимся к файлу перехвата `download-fast.pcapng`. Откройте этот файл, щелкните на любом пакете TCP, чтобы выбрать его, и выберите команду Statistics⇨IO Graph из главного меню.

В открывшемся окне IO Graph появится графическое представление потока данных во времени. Как следует из примера, приведенного на рис. 5.17, на данном графике загрузки файла передается около 500 пакетов в секунду. И этот показатель остается постоянным почти до конца графика, где он резко снижается.

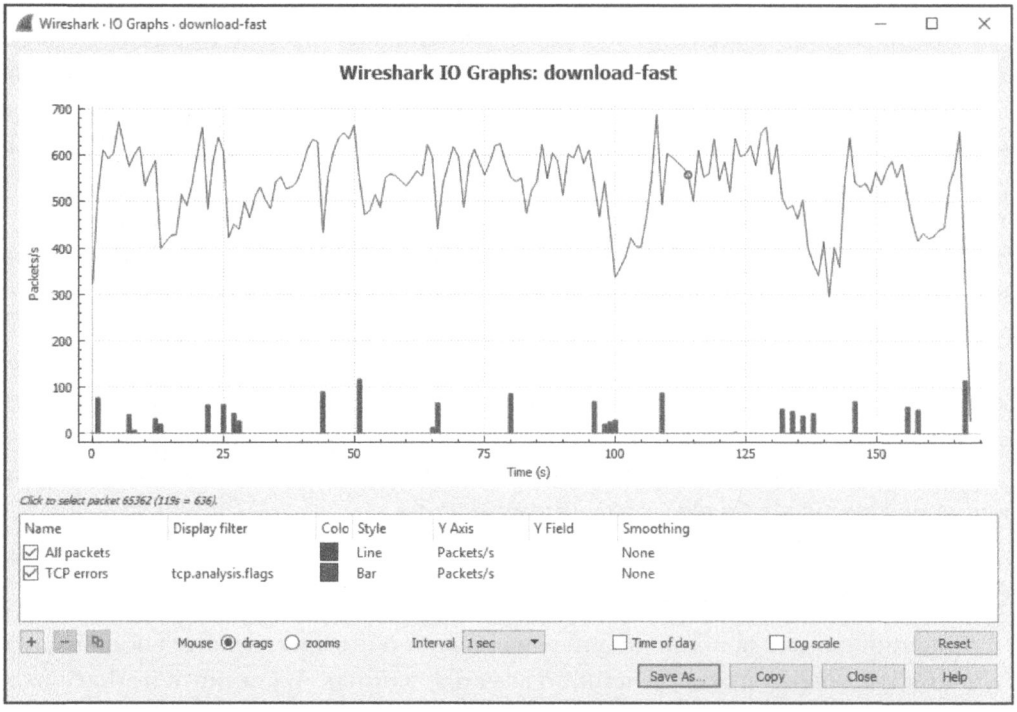


Рис. 5.17. Этот график ввода-вывода демонстрирует практически постоянную скорость передачи пакетов при быстрой загрузке файла

А теперь рассмотрим для сравнения пример более медленной загрузки файла. Оставив открытым текущий файл перехвата, откройте файл перехвата `download-slow.pcapng` в другом экземпляре Wireshark. Составив график ввода-вывода для данного примера загрузки файла описанным выше способом, вы увидите совсем другую картину, как показано на рис. 5.18.

Рассматриваемая здесь загрузка происходит со скоростью от 15 до 100 пакетов в секунду, и такая скорость далека от постоянной, а иногда даже падает до нуля пакетов в секунду. Подобное непостоянство можно лучше увидеть, если расположить рядом графики ввода-вывода обоих файлов, как показано на рис. 5.19. Сравнивая оба графика, обратите особое внимание на значения, откладываемые по осям X и Y, чтобы сравнивать сопоставимые величины. Масштаб в обоих случаях автоматически корректируется в зависимости от количества пакетов и/или объема переданных данных, что составляет главное

отличие сравниваемых графиков. Так, медленная загрузка файла демонстрируется в масштабе от 0 до 100 пакетов в секунду, тогда как быстрая загрузка – от 0 до 700 пакетов в секунду.

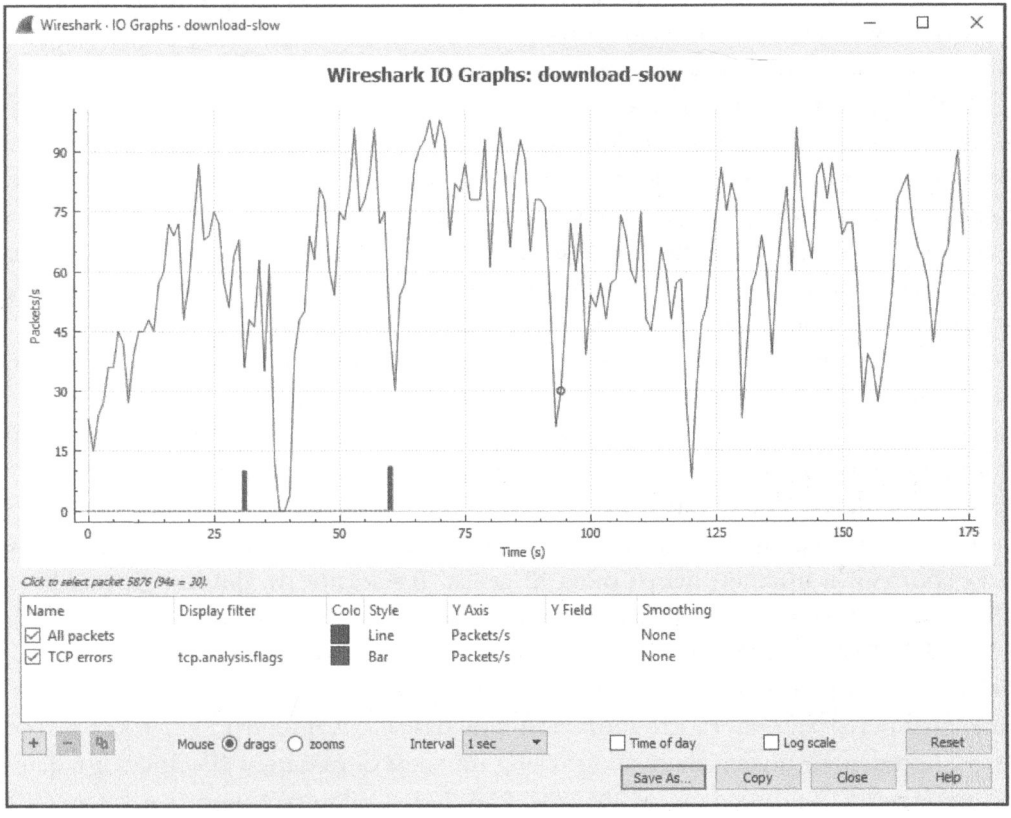


Рис. 5.18. Этот график ввода-вывода демонстрирует непостоянную скорость передачи пакетов при медленной загрузке файла

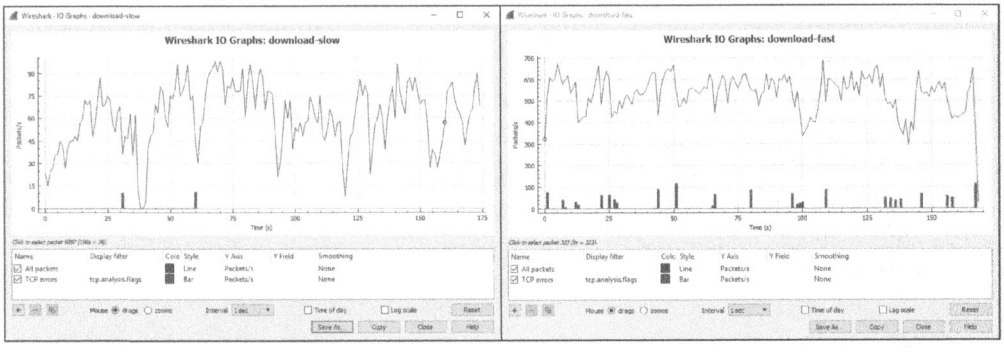


Рис. 5.19. Просмотр двух расположенных рядом графиков ввода-вывода может помочь в выявлении отклонений

Параметры, настраиваемые в нижней части окна IO Graph, позволяют применять ряд особых фильтров, составляемых с помощью того же самого синтаксиса, что и для фильтров отображения, а также выбирать цвета для отображения данных из этих фильтров. Например, можно создать фильтры конкретных IP-адресов и назначить для них особые цвета для просмотра отклонений в пропускной способности каждого устройства.

Опробуйте такую возможность, открыв файл перехвата `http_espn.pcapng`, который был получен при посещении начальной страницы американского кабельного спортивного телеканала ESPN из анализируемого устройства. В окне Conversations вы обнаружите, что наиболее активным оказывается сетевой узел с внешним IP-адресом **`205.234.218.129`**. Из этого можно сделать вывод, что данный сетевой узел, вероятнее всего, служит основным поставщиком содержимого, которое получается при посещении веб-страницы по адресу `espn.com`. Но в диалоге принимают участие и сетевые узлы по другим IP-адресам, вероятнее всего, потому, что дополнительное содержимое загружается из внешних его поставщиков и рекламодателей. Отличия в прямой и сторонней доставке содержимого можно продемонстрировать с помощью графика ввода-вывода, приведенного на рис. 5.20.

Оба фильтра, применяемых на этом графике, представлены отдельными строками в нижней части окна IO Graph. В частности, фильтр `Top Talker` (Наиболее активный сетевой узел) показывает ввод-вывод только по IP-адресу **`205.234.218.129`** основного в данном случае поставщика содержимого. Объем этого ввода-вывода отображается на графике черным цветом, заполняющим верхнюю часть столбиковой диаграммы. А фильтр `Everything Else` (Все остальные) показывает ввод-вывод по всем остальным IP-адресам в файле перехвата, кроме адреса **`205.234.218.129`**. Следовательно, он включает в себя всех сторонних поставщиков содержимого. Объем этого ввода-вывода отображается на графике красным цветом (светлым оттенком серого на рис. 5.20), заполняющим нижнюю часть столбиковой диаграммы. Обратите внимание на то, что единицы измерения по оси Y данного графика были изменены на байты в секунду. С учетом этих изменений очень легко увидеть отличия в объеме трафика основного и сторонних поставщиков содержимого, а также выяснить, сколько содержимого поступает из стороннего источника. Вам, вероятно, будет любопытно повторить это упражнение на часто посещаемых вами веб-сайтах и взять эту полезную стратегию на вооружение для сравнения объемов ввода-вывода в разных хостах.

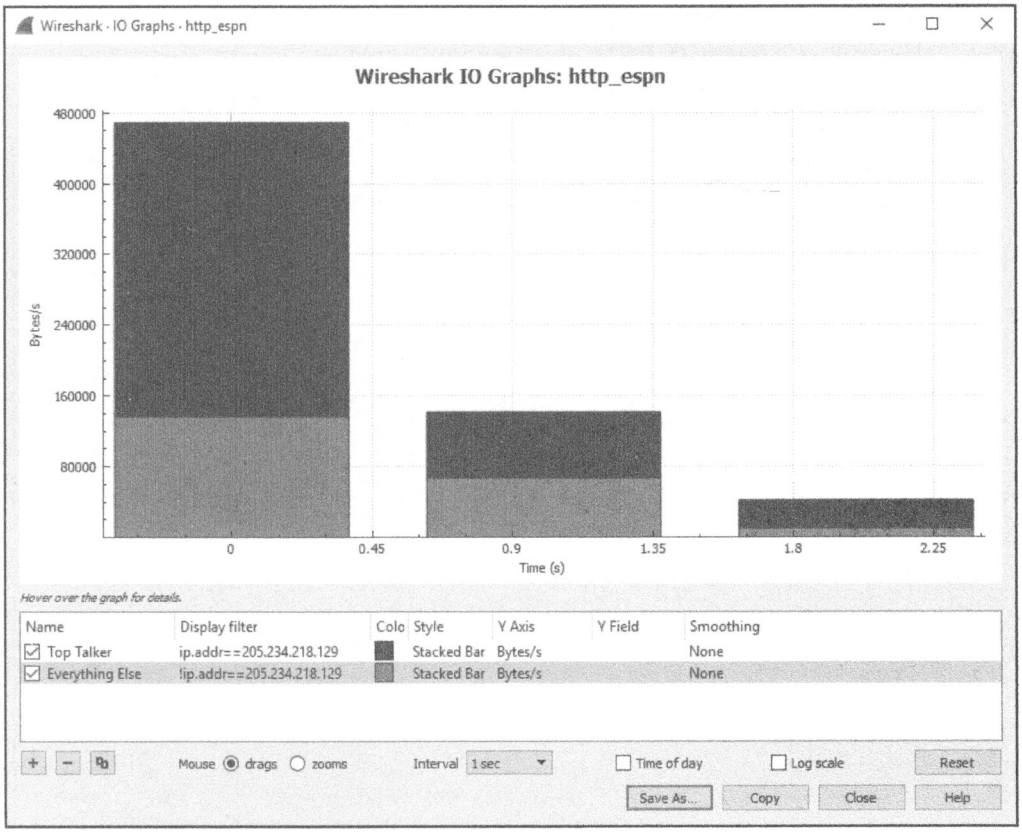


Рис. 5.20. График, демонстрирующий отличия ввода-вывода в двух отдельных устройствах

Составление графика времени круговой передачи пакетов

Файл перехвата **`download-fast.pcapng`**

В приложении Wireshark имеется также возможность составлять и просматривать график времени круговой (round-trip) передачи пакетов из заданного файла перехвата. *Время круговой передачи* (Round-Trip Time, RTT) — это время, которое требуется на получение подтверждения доставки пакета адресату. По существу, это время, которое требуется, чтобы пакет достиг получателя, а отправленное обратно подтверждение его получения — отправителя этого пакета. Анализ времени круговой передачи пакета зачастую выполняется с целью выявить места замедления или узкие места в передаче данных, а также любые задержки, возникающие в этой связи.

Чтобы опробовать такую возможность на конкретном примере, откройте файл перехвата `download-fast.pcapng`. Чтобы просмотреть график времени круговой передачи пакетов из этого файла, выберите сначала любой пакет

TCP, а затем команду Statistics⇨TCP Stream Graphs⇨Round Trip Time Graph (Статистика⇨Графики потоков TCP⇨График времени круговой передачи) из главного меню. В итоге появится график, приведенный на рис. 5.21.

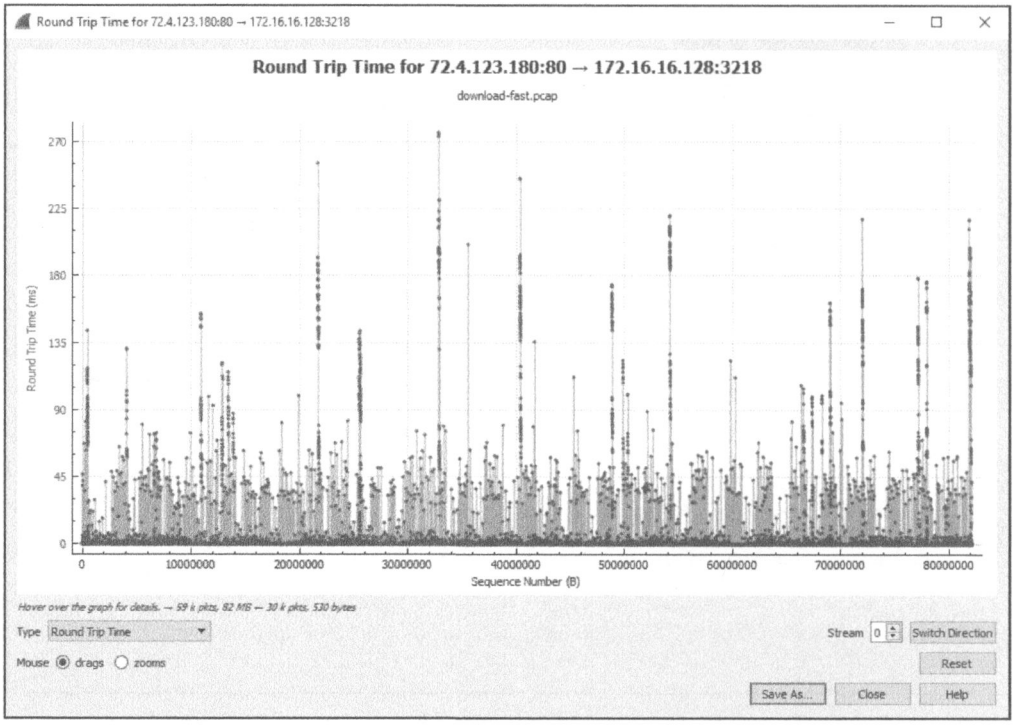


Рис. 5.21. График времени круговой передачи пакетов при быстрой загрузке файла демонстрирует практическое постоянство этой временной характеристики, за исключением нескольких случайных отклонений

Каждая точка на этом графике представляет время на передачу и подтверждение приема пакета. По умолчанию эти величины отсортированы по порядковым номерам пакетов. Чтобы перейти непосредственно к пакету на панели Packet List, достаточно щелкнуть на соответствующей точке графика.

ПРИМЕЧАНИЕ *График времени круговой передачи пакетов носит однонаправленный характер, поэтому очень важно выбрать для анализа нужное направление сетевого трафика. Если у вас этот график выглядит иначе, чем на рис. 5.21, попробуйте дважды щелкнуть на кнопке* Switch Direction *(Сменить направление).*

На этом графике, построенном для быстрой загрузки файла, величины времени круговой передачи пакетов в основном оказываются меньше 0,05 с,

и лишь в некоторых точках они находятся в пределах от 0,10 до 0,25 с. И несмотря на немалое количество больших величин, они в основном обозначают вполне допустимое для загрузки файла время на передачу и подтверждение приема. Анализируя график времени круговой передачи пакетов с точки зрения пропускной способности сети, следует выявлять большие величины времени задержки, обозначаемые множеством точек при больших значениях, откладываемых по оси Y.

Составление графиков потоков

Файл перехвата **dns_recursivequery_server.pcapng**

Возможность составлять графики потоков оказывается полезной для наглядного представления сетевых соединений и передачи потоков данных во времени. Подобные сведения облегчают понимание характера обмена данными между устройствами в сети. График потоков состоит из столбцов, обозначающих соединение между хостами, наглядно представляя сетевой трафик для удобства его интерпретации.

Чтобы создать график потоков, откройте файл перехвата `dns_recursivequery_server.pcapng` и выберите команду Statistics⇨Flow Graph (Статистика⇨График потоков) из главного меню. Полученный в итоге график приведен на рис. 5.22.

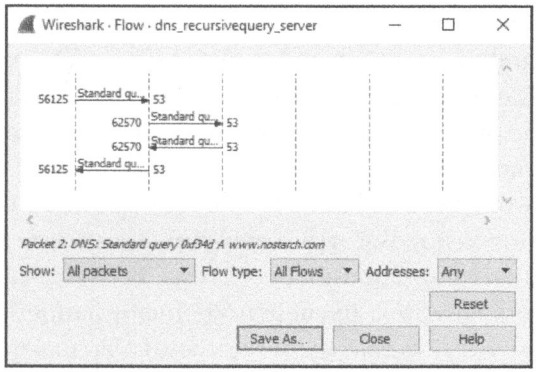


Рис. 5.22. График потоков TCP позволяет намного лучше представить наглядно сетевое соединение

На этом графике потоков наглядно представлен рекурсивный DNS-запрос, получаемый одним хостом и пересылаемый другому хосту (подробнее о протоколе DNS речь пойдет в главе 9, "Распространенные протоколы верхнего уровня"). Каждая вертикальная линия на этом графике обозначает отдельный хост. График потоков позволяет наглядно представить двухсторонний обмен данными между двумя устройствами в сети, а в данном примере — соотношение

обмена данными между несколькими устройствами. Такой график полезен и для понимания обычного потока данных, передаваемых по менее известным из вашего опыта сетевым протоколам.

Экспертная информация

Файл перехвата **download-slow.pcapng**

В дешифраторах каждого протокола в Wireshark определена *экспертная информация*, предупреждающая о конкретных состояниях в пакетах данного протокола. Эти состояния разделяются на следующие категории.

- **Chat (Текстовый диалог).** Основные сведения об обмене данными.
- **Note (Уведомление).** Необычные пакеты, которые могут быть частью обычного обмена данными.
- **Warning (Предупреждение).** Необычные пакеты, которые, вероятнее всего, не являются частью обычного обмена данными.
- **Error (Ошибка).** Ошибка в пакете или интерпретирующем его дешифраторе.

В качестве примера откройте файл перехвата `download-slow.pcapng` и выберите команду Analyze⇨Expert Information (Анализ⇨Экспертная информация) из главного меню, чтобы открыть окно Expert Information. Установите в этом окне флажок Group by summary (Группировать по сводке), чтобы организовать вывод экспертной информации по степени ее важности (рис. 5.23).

В этом окне имеются разделы по каждой категории экспертной информации. В данном случае ошибки отсутствуют, имеются 3 предупреждения, 19 уведомлений и 3 текстовых диалога.

Большинство сообщений из данного файла перехвата связаны с сетевым протоколом TCP просто потому, что к данному протоколу традиционно применялась экспертная информационная система. В то же время в данном окне отображается 29 сообщений с экспертной информацией, настроенных на протокол TCP, и они могут оказаться полезными для диагностики файлов перехвата. Эти сообщения помечают отдельные пакеты, когда они удовлетворяют определенным критериям, как перечислено ниже. (Это означает, что подобные сообщения станут более понятными при изучении сетевого протокола TCP в главе 8, "Протоколы транспортного уровня", и диагностики медленных сетей в главе 11, "Меры борьбы с медленной сетью".

- **Сообщения текстового диалога.**

  Сообщение `"Window Update"` (Изменение размера окна), посылаемое получателем, чтобы уведомить отправителя об изменении размера окна приема в протоколе TCP.

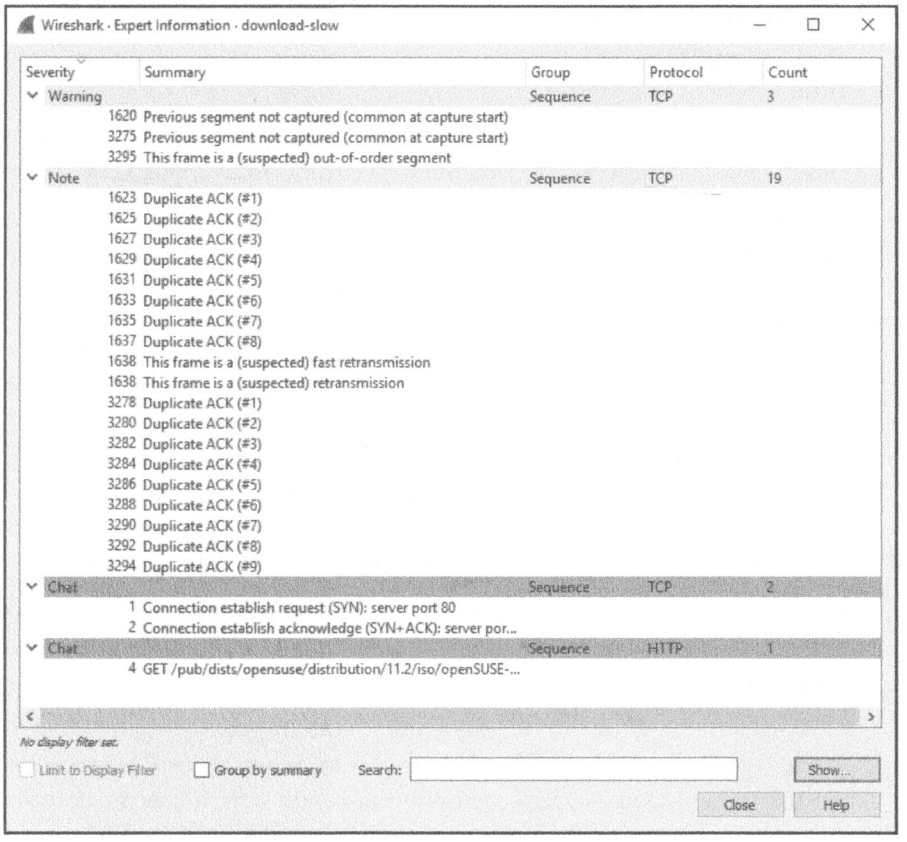


Рис. 5.23. В окне Expert Information отображаются сведения из экспертной системы, запрограммированной в дешифраторах сетевых протоколов

- **Сообщения с уведомлениями.**

  Сообщение "`TCP Retransmission`" (Повторная передача по протоколу TCP), посылаемое в результате потери пакетов. Оно появляется, когда получен дубликат подтверждения приема пакета или сработал таймер времени ожидания повторной передачи пакетов.

  Сообщение "`Duplicate ACK`" (Дубликат подтверждения), посылаемое в том случае, если хост не получает пакет с ожидаемым следующим порядковым номером и формирует дубликат подтверждения последних полученных данных.

  Сообщение "`Zero Window Probe`" (Проба нулевого окна), посылаемое в ходе текущего контроля состояния окна приема в протоколе TCP после передачи пакета с нулевым окном, как поясняется в главе 11, "Меры борьбы с медленной сетью".

Сообщение "Keep Alive ACK" (Подтверждение активности соединения), посылаемое в ответ на пакеты поддержания активным соединения.

Сообщение "Zero Window Probe ACK" (Подтверждение пробы нулевого окна), посылаемое в ответ на пакеты с пробой нулевого окна.

Сообщение "Window Is Full" (Окно заполнено), посылаемое для уведомления о заполнении на стороне получателя окна приема в протоколе TCP.

- **Предупреждающие сообщения.**

  Сообщение "Previous Segment Lost" (Предыдущий сегмент потерян), посылаемое в том случае, если пропущен пакет с ожидаемым порядковым номером в потоке данных.

  Сообщение "ACKed Lost Packet" (Подтверждение потери пакета), посылаемое в том случае, если обнаружен пакет подтверждения ACK, но не пакет, который он подтверждает.

  Сообщение "Keep Alive" (Поддержание активным соединения), посылаемое в том случае, если обнаружен пакет поддержания активным соединения.

  Сообщение "Zero Window" (Нулевое окно), посылаемое в том случае, если достигнут размер окна приема в протоколе TCP и послано уведомление о нулевом окне, запрашивающее отправителя остановить передачу данных.

  Сообщение "Out-of-Order" (Нарушение порядка следования), посылаемое в том случае, если на основании порядковых номеров обнаружено, что пакеты получаются не по порядку следования их номеров.

  Сообщение "Fast Retransmission" (Быстрая повторная передача), посылаемое в том случае, если повторная передача пакета происходит в течение 20 мс после получения дубликата подтверждения.

- **Сообщения об ошибках.**

  Сообщение "No Error Messages" (Сообщения об ошибках отсутствуют).

На первый взгляд может показаться, что некоторые возможности Wireshark, рассматриваемые в этой главе, пригодны лишь в особых, малоизвестных ситуациях, но на самом деле вам придется пользоваться ими чаще, чем вы могли бы предположить. Ознакомиться с рассмотренными здесь возможностями и окнами очень важно потому, что к ним придется еще не раз обращаться в ряде последующих глав.

6

АНАЛИЗ ПАКЕТОВ ИЗ КОМАНДНОЙ СТРОКИ

Во многих случаях анализ пакетов можно проводить, используя графический интерфейс, но иногда инструментальные средства, работающие в режиме командной строки (например, утилиты TShark или `tcpdump`), оказываются просто необходимы и даже предпочтительны. Ниже перечислены ряд ситуаций, в которых инструментальное средство командной строки может быть использовано вместо Wireshark.

- Wireshark одновременно предоставляет немало сведений. Используя инструментальное средство командной строки, можно ограничить отображаемую информацию только нужными данными, например, показывать в строке только IP-адреса.
- Инструментальные средства командной строки лучше всего подходят для фильтрации пакетов в файле перехвата и передачи результатов непосредственно другому инструментальному средству по каналам, применяемым в ОС Unix.
- Обрабатывать очень крупные файлы перехвата в Wireshark зачастую оказывается нелегко, поскольку весь файл должен быть загружен в оперативную память. Благодаря потоковой обработке файлов перехвата

инструментальными средствами командной строки удается быстро отобрать нужные пакеты из файла.

- Если приходится работать с сервером, где недоступны инструментальные средства с графическим интерфейсом, в таком случае остается лишь полагаться на инструментальные средства командной строки.

В этой главе демонстрируются возможности утилит TShark и `tcpdump` — наиболее употребительных инструментальных средств командной строки для анализа пакетов. Ознакомиться с ними полезно, хотя я лично пользуюсь утилитой TShark в системах Windows, а утилитой `tcpdump` — в системах Unix. Если вы пользуетесь только Windows, можете пропустить разделы этой главы, посвященные утилите `tcpdump`.

Установка утилиты TShark

Утилита TShark, являющаяся терминальным аналогом Wireshark, служит для анализа пакетов и предоставляет в основном те же функциональные возможности, что и Wireshark, но только в виде интерфейса командной строки, а не графического интерфейса. Если вы установили Wireshark, то у вас, скорее всего, имеется и TShark, при условии, что вы не исключили эту утилиту из состава устанавливаемых средств Wireshark. Чтобы убедиться, установлена ли утилита TShark, выполните следующие действия.

1. Откройте окно командной строки или терминала, введя команду **`cmd`** и щелкнув на соответствующей кнопке (конкретные действия зависят от применяемой операционной системы).
2. Перейдите к каталогу, где установлено приложение Wireshark. Если вы установили его в стандартном месте, введите в командной строке команду **`cd C:\Program Files\Wireshark`**.
3. Запустите TShark на выполнение и выведите на экран сведения о версии этой утилиты, введя команду **`tshark -v`**. Если утилита TShark не установлена, появится сообщение об ошибке, уведомляющее, что введенная команда не распознана. А если утилита TShark установлена в вашей системе, то появятся следующие сведения о ее версии:

```
C:\Program Files\Wireshark>tshark -v
TShark (Wireshark) 2.6.1 (v2.6.1-0-g860a78b3)
-- фрагмент --
```

Если же вы не установили утилиту TShark, но желаете ею воспользоваться, повторите установку Wireshark, выбрав TShark в составе устанавливаемых

средств, хотя это и делается по умолчанию. А если вы желаете сразу же приступить к изучению возможностей утилиты TShark, то выполните команду **tshark -h**, как показано ниже, чтобы вывести на экран ее команды. Некоторые из команд TShark будут рассмотрены в этой главе.

```
C:\Program Files\Wireshark>tshark -h
```

Как и Wireshark, утилита TShark пригодна для работы в разных операционных системах, но поскольку она не зависит от графических библиотек конкретной операционной системы, то опыт ее использования на разных платформах практически одинаков. В силу этого утилита TShark действует весьма сходным образом в Windows, Linux и Mac OS X, хотя здесь имеются некоторые отличия на каждой платформе. В этой книге основное внимание уделяется работе с TShark в Windows, поскольку это основная операционная система, для которой предназначена данная утилита.

Установка утилиты tcpdump

Если Wireshark является самым распространенным во всем мире графическим приложением для анализа пакетов, то `tcpdump` считается наиболее употребительной утилитой для анализа пакетов, работающей в режиме командной строки. Она предназначена для работы в Unix-подобных операционных системах и легко устанавливается с помощью типичных приложений для управления пакетами и даже входит в состав дистрибутивных пакетов многих разновидностей Linux.

Несмотря на то что бо́льшая часть этой книги ориентирована на работу в среде Windows, в нее включены разделы, посвященные `tcpdump` специально для пользователей Unix и, в частности, версии Ubuntu 14.04 LTS. Если же вы желаете воспользоваться утилитой `tcpdump` в Windows, загрузите и установите ее аналог WinDump для Windows, доступный по адресу `http://www.winpcap.org/windump/`. И хотя опыт использования `tcpdump` и WinDump не совсем одинаков, эти средства анализа пакетов функционируют сходным образом. Следует, однако, иметь в виду, что WinDump и близко не поддерживается так же активно, как и `tcpdump`. В итоге ряд новых возможностей может просто отсутствовать и в то же время присутствовать уязвимости в защите. (В этой книге WinDump не рассматривается.)

Несмотря на то что утилита `tcpdump` не входит в дистрибутивный пакет Ubuntu, установить ее совсем не трудно благодаря системе управления пакетами APT. Чтобы установить `tcpdump`, выполните следующие действия.

1. Откройте окно терминала и выполните команду **sudo apt-get update**, чтобы обновить хранилища пакетов последними их версиями.

2. Выполните команду **sudo apt-get install tcpdump**.
3. Вам будет предложено установить ряд зависимых пакетов, необходимых для работы tcpdump. Разрешите их установку, введя **Y** и нажав клавишу <Enter> по приглашению.
4. По окончании установки введите команду **tcpdump -h**, чтобы запустить утилиту tcpdump на выполнение и вывести на экран сведения о ее версии. Если эта команда будет выполнена успешно, это означает, что вы готовы приступить к работе с tcpdump. А в окне термина появится следующее сообщение:

```
sanders@ppa:~$ tcpdump -h
tcpdump version 4.9.2
libpcap version 1.8.1
Usage: tcpdump [-aAbdDefhHIJKlLnNOpqStuUvxX#] [ -B size ] [ -c count ]
            [ -C file_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]
            [ -i interface ] [ -j tstamptype ] [ -M secret ] [ --number ]
            [ -Q in|out|inout ]
            [ -r file ] [ -s snaplen ] [ --time-stamp-precision precision ]
            [ --immediate-mode ] [ -T type ] [ --version ] [ -V file ]
            [ -w file ] [ -W filecount ] [ -y datalinktype ]
            [ -z postrotate-command ]
            [ -Z user ] [ expression ]
```

Все доступные в tcpdump команды можно вывести на экран, выполнив приведенную ниже команду. Далее будут рассмотрены некоторые из этих команд.

```
sanders@ppa:~$ man tcpdump
```

Перехват и сохранение пакетов

Прежде всего следует перехватить пакеты из сети и отобразить их на экране. Чтобы приступить к перехвату пакетов в TShark, достаточно выполнить команду **tshark**. По этой команде начинается процесс перехвата пакетов из сетевого интерфейса и их вывод на экран в окне терминала аналогично приведенному ниже примеру.

```
C:\Program Files\Wireshark>tshark
  1 0.000000 172.16.16.128 -> 74.125.95.104 TCP 66 1606 80 [SYN]
Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  2 0.030107 74.125.95.104 -> 172.16.16.128 TCP 66 80 1606 [SYN, ACK]
Seq=0 Ack=1 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
  3 0.030182 172.16.16.128 -> 74.125.95.104 TCP 54 1606 80 [ACK]
```

```
Seq=1 Ack=1 Win=16872 Len=0
  4 0.030248 172.16.16.128 -> 74.125.95.104 HTTP 681 GET / HTTP/1.1
  5 0.079026 74.125.95.104 -> 172.16.16.128 TCP 60 80 1606 [ACK]
Seq=1 Ack=628 Win=6976 Len=0
```

А для того чтобы приступить к перехвату пакетов в tcpdump, достаточно выполнить команду **tcpdump**. После выполнения этой команде окно терминала должно выглядеть аналогично следующему:

```
sanders@ppa:~$ tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes

21:18:39.618072 IP 172.16.16.128.slm-api > 74.125.95.104.http: Flags [S],
seq 2082691767, win 8192, options [mss 1460,nop,wscale 2,nop,nop,sackOK], length 0

21:18:39.648179 IP 74.125.95.104.http > 172.16.16.128.slm-api:
Flags [S.], seq 2775577373, ack 2082691768, win 5720, options [mss
1406,nop,nop,sackOK,nop,wscale 6], length 0

21:18:39.648254 IP 172.16.16.128.slm-api > 74.125.95.104.http: Flags [.], ack 1,
win 4218, length 0

21:18:39.648320 IP 172.16.16.128.slm-api > 74.125.95.104.http: Flags [P.],
seq 1:628, ack 1, win 4218, length 627: HTTP: GET / HTTP/1.1

21:18:39.697098 IP 74.125.95.104.http > 172.16.16.128.slm-api: Flags [.], ack 628,
win 109, length 0
```

ПРИМЕЧАНИЕ *Для перехвата пакетов в системах Unix требуются полномочия администратора, и поэтому вам, скорее всего, придется выполнить команду* **tcpdump** *с правами пользователя* root *или же выполнить сначала команду* **sudo**, *а затем упоминаемые здесь и далее команды. Во многих случаях Unix-подобная система будет доступна вам как пользователю с ограниченными полномочиями. И это наиболее вероятная причина возникновения ошибки нарушения прав доступа по ходу дела.*

В зависимости от конфигурации вашей системы утилита TShark или tcpdump может и не выбрать по умолчанию именно тот сетевой интерфейс, из которого вам требуется перехватывать пакеты. В таком случае вам придется указать его вручную. Список доступных для TShark сетевых интерфейсов можно вывести на экран по команде **tshark -D**, как показано ниже.

```
C:\Program Files\Wireshark>tshark -D
1. \Device\NPF_{1DE095C2-346D-47E6-B855-11917B74603A} (Local Area Connection 2)
2. \Device\NPF_{1A494418-97D3-42E8-8C0B-78D79A1F7545} (Ethernet 2)
```

Чтобы воспользоваться конкретным сетевым интерфейсом, выполните команду **tshark -i** с номером, присвоенным сетевому интерфейсу в приведенном выше списке:

```
C:\Program Files\Wireshark>tshark -i 1
```

По этой команде будут перехвачены пакеты только из сетевого интерфейса Local Area Connection 2, которому присвоен номер **1** в списке сетевых интерфейсов. Рекомендуется всегда указывать сетевой интерфейс, из которого требуется перехватывать пакеты. Ведь для инструментальных средств виртуальной машины или виртуальных частных сетей весьма характерно добавлять сетевые интерфейсы, и поэтому следует непременно убедиться, что пакеты поступают из нужного источника.

Если вы работаете с утилитой tcpdump в системе Linux или Mac OS X, выведите список доступных сетевых интерфейсов на экран по команде **ifconfig**, как показано ниже.

```
sanders@ppa:~$ ifconfig
eth0    Link encap:Ethernet HWaddr 00:0c:29:1f:a7:55
        inet addr:172.16.16.139 Bcast:172.16.16.255 Mask:
        255.255.255.0
        inet6 addr: fe80::20c:29ff:fe1f:a755/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:5119 errors:0 dropped:0 overruns:0 frame:0
        TX packets:3088 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:876746 (876.7 KB) TX bytes:538083 (538.0 KB)
```

А указать конкретный сетевой интерфейс можно по приведенной ниже команде. По этой команде будут перехвачены пакеты только из сетевого интерфейса eth0.

```
sanders@ppa:~$ tcpdump -i eth0
```

Итак, настроив все должным образом, можно приступить к перехвату пакетов. Если устройство, из которого перехватывается сетевой трафик, довольно сильно нагружено в сети, вы, скорее всего, заметите, что строки с отдельными пакетами появляются настолько быстро, что их трудно прочитать. Это

положение можно исправить, сначала сохранив пакеты в файл, а затем прочитав лишь небольшое их количество из этого файла.

Чтобы сохранить накопленные пакеты в файле выполните в обеих рассматриваемых здесь утилитах команду (**tshark** или **tcpdump**) с аргументом **-w**, указав после него имя файла. Перехват пакетов продолжится до тех пор, пока вы не остановите его, нажав комбинацию клавиш <Ctrl+C>. Указанный файл будет сохранен в том каталоге, из которого соответствующая утилита была запущена на выполнение, если не указано иное.

Ниже приведен пример выполнения данной команды в TShark.

```
C:\Program Files\Wireshark>tshark -i 1 -w packets.pcap
```

По этой команде все пакеты, перехваченные из сетевого интерфейса, указанного первым в списке интерфейсов, будут записаны в файле перехвата `packets.pcap`. Та же самая команда в `tcpdump` будет выглядеть следующим образом:

```
sanders@ppa:~$ tcpdump -i eth0 -w packets.pcap
```

Чтобы прочитать пакеты из сохраненного файла перехвата, выполните следующую команду с аргументом **-r**, указав также имя этого файла:

```
C:\Program Files\Wireshark>tshark -r packets.pcap
```

По этой команде все пакеты из файла перехвата `packets.pcap` будут прочитаны и выведены на экран. Аналогичная команда в `tcpdump` выглядит следующим образом:

```
sanders@ppa:~$ tcpdump -r packets.pcap
```

А по следующей команде в TShark будут отображены только первые 10 пакетов из файла перехвата:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -c10
```

Аналогичная команда в `tcpdump` выглядит следующим образом:

```
sanders@ppa:~$ tcpdump -r packets.pcap -c10
```

В процессе перехвата можно также указать аргумент **-c** в соответствующей команде. В результате ее выполнения будут перехвачены и выведены для просмотра на экране только 10 пакетов. А если еще добавить аргумент **-w**, то перехваченные пакеты могут быть сохранены в файле. Ниже показано, как подобная команда выглядит в TShark.

```
C:\Program Files\Wireshark>tshark -i 1 -w packets.pcap -c10
```

Аналогичная команда в tcpdump выглядит следующим образом:

```
sanders@ppa:~$ tcpdump -i eth0 -w packets.pcap -c10
```

Манипулирование выводимыми результатами

Преимущество применения инструментальных средств для анализа пакетов в режиме командной строки заключается в более конкретном выводе получаемых результатов. Как правило, в графическом интерфейсе отображается все подряд, и вы сами должны найти именно то, что вам нужно. А инструментальные средства командной строки обычно отображают лишь самый минимум, вынуждая вас пользоваться дополнительными командами для более углубленного анализа пакетов. И в этом отношении утилиты TShark и tcpdump ничем не отличаются. Обе они выводят на экран по одной строке на каждый пакет, требуя дополнительных команд для просмотра более подробных сведений о сетевых протоколах или отдельных байтах из пакетов.

Каждая выводимая в TShark строка представляет один пакет, а ее формат зависит от протоколов, применяемых в этом пакете. В утилите TShark употребляются те же самые интерпретаторы сетевых протоколов, что и в Wireshark, и получаемые в итоге данные анализируются таким же самым способом. Поэтому выводимые в TShark результаты зеркально отображают содержимое панели Packet List в Wireshark, если сравнить их рядом. А поскольку интерпретаторы в утилите TShark позволяют анализировать сетевые протоколы вплоть до седьмого уровня, то она может предоставить намного больше сведений о пакетах, содержащих заголовки, чем утилита tcpdump.

Каждая выводимая в tcpdump строка также представляет один пакет, который форматируется по-разному в зависимости от применяемого сетевого протокола. А поскольку в tcpdump не применяются такие же интерпретаторы протоколов, как и в Wireshark, то сведения о сетевых протоколах седьмого уровня этой утилитой не анализируются. И в этом состоит один из главных недостатков tcpdump. Вместо этого однострочные пакеты форматируются в зависимости от применяемого в них сетевого протокола транспортного уровня, а им может быть протокол TCP или UDP (подробнее об этом речь пойдет в главе 8, "Протоколы транспортного уровня").

В пакетах TCP применяется следующий формат:

```
[Отметка времени] [Протокол третьего уровня]
[IP-адрес отправителя].[Порт отправителя] >
[IP-адрес получателя].[Порт получателя]:
[Флаги протокола TCP], [Порядковый номер в протоколе TCP],
[Номер подтверждения в протоколе TCP],
[Размеры окон в протоколе TCP], [Длина данных]
```

В пакетах UDP применяется такой формат:

```
[Отметка времени] [Протокол третьего уровня]
[IP-адрес отправителя].[Порт отправителя] >
[IP-адрес получателя].[Порт получателя]:
[Протокол четвертого уровня], [Длина данных]
```

Эти основные итоговые сведения в одной строке отлично подходят для быстрого анализа пакетов, но в итоге вам придется выполнять более углубленный анализ пакетов. В приложении Wireshark достаточно щелкнуть на избранном пакете в панели Packet List, чтобы отобразить сведения о нем в панелях Packet Details и Packet Bytes. Получить доступ к той же самой информации из командной строки можно с помощью нескольких параметров.

Чтобы получить дополнительные сведения о каждом пакете, проще всего сделать вывод результатов более многословным. Этой цели в TShark служит параметр **V**, как показано ниже.

```
C:\Program Files\Wireshark>tshark -r packets.pcap -V
```

По этой команде выводятся те же самые сведения о пакетах, прочитанных из файла перехвата packets.pcap, что и в панели Packet Details в Wireshark. Ниже приведены примеры анализа пакета с обычной детализацией (основные итоговые сведения) и расширенной детализацией (подробные итоговые сведения, получаемые с помощью аргумента **-V**).

Сначала приведем стандартный выводимый результат:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -c1
  1    0.000000 172.16.16.172 -> 4.2.2.1 ICMP Echo (ping) request id=0x0001,
seq=17/4352, ttl=128
```

А теперь приведем часть более подробных сведений, получаемых благодаря расширенной детализации.

```
C:\Program Files\Wireshark>tshark -r packets.pcap -V -c1
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
    Interface id: 0 (\Device\NPF_{C30671C1-579D-4F33-9CC0-73EFFFE85A54})
     Encapsulation type: Ethernet (1)
     Arrival Time: Dec 21, 2015 12:52:43.116551000
     Eastern Standard Time
     [Time shift for this packet: 0.000000000 seconds]
-- фрагмент --
```

Для повышения уровня детализации в утилите tcpdump служит аргумент **-v**. Но, в отличие от TShark, в утилите tcpdump допускается несколько уровней детализации для отображения каждого пакета. А если указать в соответствующей команде несколько аргументов **v**, то можно добавить столько же уровней детализации. В приведенном ниже примере демонстрируется расширение отображаемых сведений о пакетах до трех уровней детализации.

```
sanders@ppa:~$ tcpdump -r packets.pcap -vvv
```

Ниже приведен пример отображения сведений об одном и том же пакете как с обычным уровнем детализации, так и с расширением на еще один уровень детализации. Но даже при полной детализации выводимый результат и близко нельзя сравнить с теми подробностями, которые выводятся в TShark.

```
sanders@ppa:~$ tcpdump -r packets.pcap -c1
reading from file packets.pcap, link-type EN10MB (Ethernet)
13:26:25.265937 IP 172.16.16.139 > a.resolvers.level3.net:
ICMP echo request, id 1759, seq 150, length 64

sanders@ppa:~$ tcpdump -r packets.pcap -c1 -v
reading from file packets.pcap, link-type EN10MB (Ethernet)
13:26:25.265937 IP (tos 0x0, ttl 64, id 37322, offset 0,
flags [DF], proto ICMP (1), length 84)
172.16.16.139 > a.resolvers.level3.net: ICMP echo request,
id 1759, seq 150, length 64
```

Доступность уровней детализации зависит от конкретного протокола в анализируемом пакете. Несмотря на всю полезность расширенной детализации, она не позволяет отобразить все, что можно увидеть. Утилиты TShark и tcpdump сохраняют все содержимое каждого пакета, которое можно просматривать также в шестнадцатеричной форме или в коде ASCII.

Так, в TShark можно просматривать представление пакетов в шестнадцатеричной форме и в коде ASCII с помощью аргумента **-x**, который можно сочетать с аргументом **r** для чтения и отображения пакета из файла перехвата, как показано ниже.

```
C:\Program Files\Wireshark>tshark -xr packets.pcap
```

Просмотр перехваченных пакетов по данной команде оказывается таким же, как и в панели Packet Bytes приложения Wireshark (рис. 6.1).

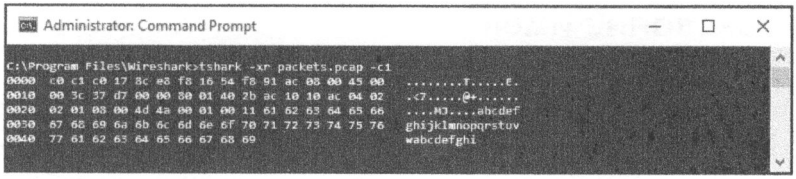


Рис. 6.1. Просмотр в TShark исходных пакетов в шестнадцатеричной форме и в коде ASCII

В утилите tcpdump перехваченные пакеты можно просматривать в шестнадцатеричной форме и в коде ASCII, используя аргумент **-X**. Этот аргумент можно сочетать с аргументом **r** для чтения пакетов из файла перехвата, как в приведенном ниже примере.

```
sanders@ppa:~$ tcpdump -Xr packets.pcap
```

Результат, выводимый по данной команде, приведен на рис. 6.2.

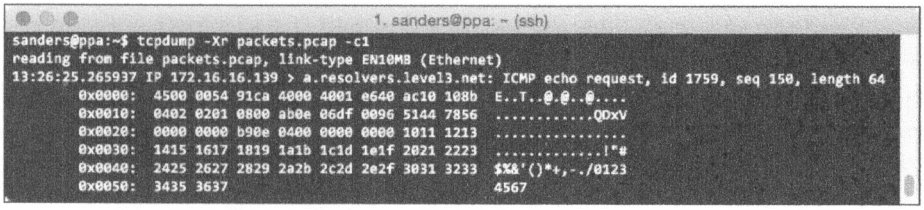


*Рис. 6.2. Просмотр в **tcpdump** исходных пакетов в шестнадцатеричной форме и в коде ASCII*

Утилита tcpdump позволяет также анализировать пакеты более подробно, чем требуется. Выводимые результаты из анализа можно также просматривать только в шестнадцатеричной форме, используя аргумент **-x**, или же только в коде ASCII, используя аргумент **-A**.

Приступая к экспериментированию с данными из перехваченных пакетов, можно очень легко почувствовать подавленность от их объема. На мой взгляд, эффективнее всего пользоваться минимальным объемом информации при анализе пакетов в режиме командной строки. Начните с просмотра пакетов в их стандартном списочном представлении, прибегая к дополнительным уровням детализации выводимых результатов, когда требуется свести анализ до нескольких пакетов, представляющих особый интерес. Такой подход позволит уберечься от подавленности из-за чрезмерных объемов анализируемых данных.

Преобразование имен

Как и в Wireshark, в утилитах TShark и tcpdump предпринимается попытка выполнить преобразование имен, чтобы вместо IP-адресов и номеров портов вывести соответствующие имена. Если вы внимательно следовали предыдущим примерам, то, возможно, заметили, что это происходит по умолчанию. Но, как упоминалось ранее, я обычно предпочитаю отменять режим преобразования имен, чтобы исключить вероятность формирования дополнительных пакетов в сети вследствие анализа уже имеющихся пакетов.

Чтобы отменить преобразование имен в TShark, можно указать аргумент **-n** в командной строке. Этот аргумент, как и многие другие, можно сочетать с другими аргументами для повышения удобочитаемости получаемых результатов, как показано ниже.

```
C:\Program Files\Wireshark>tshark -ni 1
```

Определенные аспекты преобразования имен можно активизировать или дезактивизировать с помощью аргумента **-N**. Если указать этот аргумент в командной строке, то преобразование имен будет полностью отменено, за исключением любых его аспектов, которые вы явно активизируете с помощью соответствующих значений. Например, по следующей команде преобразование номеров портов будет активизировано только на транспортном уровне:

```
C:\Program Files\Wireshark>tshark -i 1 -Nt
```

В одной команде можно объединить несколько значений аргументов. А по следующей команде преобразование имен будет активизировано как на транспортном уровне, так и на канальном уровне MAC-адресов:

```
C:\Program Files\Wireshark>tshark -i 1 -Ntm
```

Ниже приведены значения, которые можно использовать вместе с аргументом **-N**.

- **m** Преобразование MAC-адресов
- **n** Преобразование сетевых адресов
- **t** Преобразование номеров портов на транспортном уровне
- **N** Применение внешних преобразователей сетевых имен
- **C** Выполнение параллельного поиска в службе DNS

В утилите tcpdump можно отменить преобразование IP-адресов с помощью аргумента **-n**, а с помощью аргумента **-nn** — преобразование номеров портов.

Этот аргумент можно также объединять с другими аргументами командной строки, как показано ниже.

```
sanders@ppa:~$ tcpdump -nni eth1
```

В приведенных ниже примерах демонстрируется перехват пакетов сначала с активизированным преобразованием номеров портов, а затем с отмененным (аргумент **-n**).

```
sanders@ppa:~$ tcpdump -r tcp_ports.pcap -c1
reading from file tcp_ports.pcap, link-type EN10MB (Ethernet)
14:38:34.341715 IP 172.16.16.128.2826 > 212.58.226.142.
u http: Flags [S], seq 3691127924, win 8192,
options [mss 1460,nop,wscale 2,nop,nop,sackOK], length 0

sanders@ppa:~$ tcpdump -nr tcp_ports.pcap -c1
reading from file tcp_ports.pcap, link-type EN10MB (Ethernet)
14:38:34.341715 IP 172.16.16.128.2826 > 212.58.226.142.
v 80: Flags [S], seq 3691127924, win 8192,
options [mss 1460,nop,wscale 2,nop,nop,sackOK], length 0
```

По обеим приведенным выше командам читается только первый пакет из файла перехвата tcp_ports.pcap. В первой команде активизируется преобразование номеров портов и порт **80** преобразуется в имя **http** u, а во второй команде просто отображается номер порта v.

Применение фильтров

Фильтрация в утилитах TShark и tcpdump выполняется очень удобно потому, что в них применяются фильтры перехвата типа BPF. Кроме того, в TShark применяются фильтры отображения из Wireshark. Как и в Wireshark, фильтры перехвата можно применять в TShark только во время перехвата, а фильтры отображения – как во время перехвата, так и при отображении уже перехваченных пакетов. Итак, начнем с применения фильтров в TShark.

Фильтры перехвата можно применять в соответствующей команде, указав в ней аргумент **-f** и синтаксис фильтров BPF в двойных кавычках. Так, по следующей команде будут перехвачены и сохранены только те пакеты, где применяется порт получателя **80** и сетевой протокол TCP:

```
C:\Program Files\Wireshark>tshark -ni 1 -w packets.pcap
↳ -f "tcp port 80"
```

Фильтры отображения можно применять в соответствующей команде, указав в ней аргумент **-Y** и синтаксис фильтров Wireshark в двойных кавычках. Их можно применять и во время перехвата следующим образом:

```
C:\Program Files\Wireshark>tshark -ni 1 -w packets.pcap -Y "tcp.dstport == 80"
```

Фильтры отображения можно применять к уже перехваченным пакетам, указав в соответствующей команде тот же самый аргумент. Например, по следующей команде будут отображены только те пакеты из файла перехвата packets.pcap, которые совпадают с критерием, заданным в фильтре:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -Y "tcp.dstport == 80"
```

В утилите tcpdump фильтры можно указывать в конце команды, заключив их в одиночные кавычки. Так, по следующей команде также будут перехвачены и сохранены только пакеты с портом получателя **80** и сетевым протоколом TCP:

```
sanders@ppa:~$ tcpdump -nni eth0 -w packets.pcap 'tcp dst port 80'
```

Фильтры можно применять и при чтении пакетов. Например, по следующей команде будут отображены только те пакеты из файла перехвата packets.pcap, которые совпадают с критерием, заданным в фильтре:

```
sanders@ppa:~$ tcpdump -r packets.pcap 'tcp dst port 80'
```

Однако если файл перехвата был создан без фильтра, то в нем останутся и другие пакеты. Так, по приведенной выше команде ограничение накладывается лишь на то, что отображается на экране при чтении из существующего файла перехвата.

А что, если имеется файл перехвата, содержащий самые разные пакеты, но требуется отсеять лишь их подмножество и сохранить его в отдельном файле? Это можно сделать, указав вместе аргументы **-w** и **-r**, как, например, в следующей команде:

```
sanders@ppa:~$ tcpdump -r packets.pcap 'tcp dst port 80' -w http_packets.pcap
```

По данной команде из файла packets.pcap читается и отсеивается только тот сетевой трафик, который направляется в порт **80** по протоколу TCP, как это происходит в протоколе HTTP, а полученные в итоге пакеты записываются в новый файл http_packets.pcap. Такая методика нередко применяется в том случае, если требуется сохранить крупный исходный файл перехвата, но анализировать пакеты из него лишь небольшими порциями. Я лично пользуюсь этой методикой, чтобы свести сначала содержимое очень крупного файла

перехвата к небольшому подмножеству пакетов средствами tcpdump, а проанализировать его в Wireshark. Ведь обрабатывать небольшие файлы перехвата намного проще.

Помимо указания фильтров непосредственно в командной строке, утилита tcpdump позволяет обращаться к файлу типа BPF, содержащему целые последовательности фильтров. Это очень удобно в том случае, если требуется применить очень крупный или сложный фильтр, редактировать и сопровождать который в командной строке было бы в противном случае очень неудобно. Файл типа BPF можно указать с помощью аргумента **-F** следующим образом:

```
sanders@ppa:~$ tcpdump -nni eth0 -F dns_servers.bpf
```

Если файл типа BPF станет слишком крупным, у вас может возникнуть искушение дополнить его комментариями или примечаниями, чтобы знать назначение каждой его части. Следует, однако, иметь в виду, что комментарии в файле типа BPF не допускаются, и если в нем встретится что-нибудь, кроме операторов фильтрации, это приведет к ошибке. Но поскольку комментарии очень полезны в расшифровке крупных файлов типа BPF, то я обычно храню две копии каждого такого файла: одну без комментариев — для применения вместе с утилитой tcpdump, а другую — с комментариями для справки.

Форматы отображения времени в TShark

Начинающих анализировать пакеты нередко смущает устанавливаемая по умолчанию отметка времени, применяемая в TShark. Отметки времени в пакетах отображаются в этой утилите относительно начала перехвата пакетов. Иногда предпочтительно присваивать отметки времени, но зачастую требуется выяснить время, когда пакет был перехвачен, как это делается с отметками времени в утилите tcpdump. Того же самого результата можно добиться в TShark, указав аргумент **-t** вместе со значением **ad**, обозначающим абсолютную дату, как показано ниже.

```
C:\Program Files\Wireshark>tshark -r packets.pcap -t ad
```

В следующей команде сравниваются те же пакеты, что и прежде, но с относительными (по умолчанию) ❶ и абсолютными ❷ отметками времени:

```
❶ C:\Program Files\Wireshark>tshark -r packets.pcap -c2
    1    0.000000 172.16.16.172 -> 4.2.2.1    ICMP Echo (ping)
request   id=0x0001, seq=17/4352, ttl=128
    2    0.024500 4.2.2.1 -> 172.16.16.172    ICMP Echo (ping)
reply     id=0x0001, seq=17/4352, ttl=54 (request in 1)
```

```
❷ C:\Program Files\Wireshark>tshark -r packets.pcap -t ad -c2
    1    2015-12-21 12:52:43.116551 172.16.16.172 -> 4.2.2.1    ICMP Echo (ping)
request   id=0x0001, seq=17/4352, ttl=128
    2    2015-12-21 12:52:43.141051 4.2.2.1 -> 172.16.16.172    ICMP Echo (ping)
reply     id=0x0001, seq=17/4352, ttl=54 (request in 1)
```

С помощью аргумента **-t** можно указать любой формат отображения времени, доступный в Wireshark. Перечень этих форматов приведен в табл. 6.1.

Таблица 6.1. Форматы отображения времени, доступные в TShark

Значение	Отметка времени	Пример
a	Абсолютное время перехвата пакета (в текущем часовом поясе)	15:47:58.004669
ad	Абсолютное время перехвата пакета с указанием даты (в текущем часовом поясе)	2015-10-09 15:47:58.004669
d	Разница во времени с предыдущим перехваченным пакетом	0.000140
dd	Разница во времени с предыдущим отображенным пакетом	0.000140
e	Время с начала эпохи Unix (1 января 1970 г., UTC)	1444420078.004669
r	Время, прошедшее между первым и текущим пакетом	0.000140
u	Абсолютное время перехвата пакета (в формате UTC)	19:47:58.004669
ud	Абсолютное время перехвата пакета с указанием даты (в формате UTC)	2015-10-09 19:47:58.004669

К сожалению, в утилите tcpdump подобный уровень контроля для манипулирования отображением отметок времени отсутствует.

Сводная статистика в TShark

Еще одной удобной возможностью, которая выгодно отличает утилиту TShark от tcpdump, является генерирование подмножества статистических данных из файла перехвата. Процесс формирования статистических данных во многом подобен тем возможностям, которые имеются в Wireshark, но он легко доступен из командной строки. Чтобы сформировать статистические данные, достаточно указать в соответствующей команде аргумент **-z** вместе с названием варианта вывода данных. Так, полный перечень доступных статистических данных можно просмотреть по следующей команде:

```
C:\Program Files\Wireshark>tshark -z help
```

Многие из уже рассмотренных выше возможностей TShark доступны с помощью аргумента **-z**. К их числу относится возможность выводить статистические данные о конечных точках и сетевых диалогах. Так, по приведенной ниже команде на экран выводится таблица статистических данных о диалогах по сетевому протоколу IP, обнаруженных в файле перехвата `packets.pcap` (рис. 6.3).

```
C:\Program Files\Wireshark>tshark -r packets.pcap -z conv,ip
```

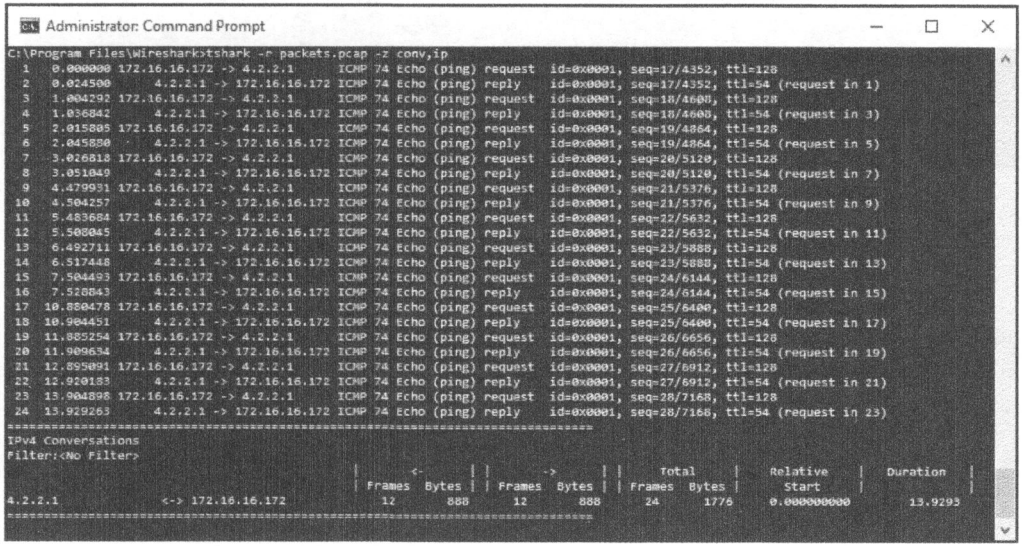


Рис. 6.3. Просмотр статистических данных о сетевых диалогах средствами TShark

С помощью аргумента **-z** можно также просматривать сведения о конкретном сетевом протоколе. Как показано ниже и на рис. 6.4, для просмотра распределения HTTP-запросов и ответов в табличной форме вместе с этим аргументом достаточно указать вариант вывода **http,tree**.

```
C:\Program Files\Wireshark>tshark -r packets.pcap -z http,tree
```

Еще одной полезной возможностью программы TShark является просмотр повторно собранного выходного потока. Эти действия мы проделывали раньше в программе Wireshark, когда щелкали правой кнопкой мыши на пакете и из контекстного меню выбирали команду Follow⇨TCP Stream. Для получения подобного отчета мы должны задать в командной строке параметр `follow` и указать тип потока, режим вывода и какой из номеров потоков мы хотим отобразить. Потоки идентифицируются по тем номерам, которые отображаются в левой колонке при выводе статистических данных о сетевых диалогах (см. рис. 6.3). Команда для получения отчета приведена ниже.

```
C:\Program Files\Wireshark>tshark -r http_google.pcap -z follow,tcp,ascii,0
```

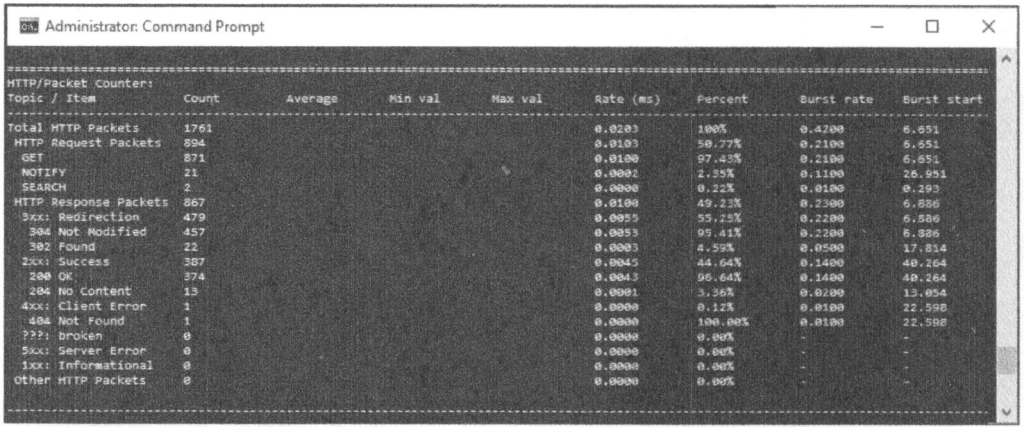


Рис. 6.4. Просмотр статистических данных о HTTP-запросах и ответах средствами TShark

А по приведенной ниже команде нулевой поток по протоколу TCP будет выведен на экран в коде ASCII из файла перехвата `http_google.pcap`. Здесь же показан результат, выводимый по данной команде.

```
C:\Program Files\Wireshark>tshark -r http_google.pcap -z
-- фрагмент --
===================================================================
Follow: tcp,ascii
Filter: tcp.stream eq 0
Node 0: 172.16.16.128:1606
Node 1: 74.125.95.104:80
627
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1;
↳ en-US; rv:1.9.1.7)
Gecko/20091221 Firefox/3.5.7
Accept: text/html,application/xhtml+xml,application/xml;
↳ q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PREF=ID=257913a938e6c248:U=267c896b5f39fb0b:FF=4:LD=e
n:NR=10:TM=1260730654:LM=1265479336:GM=1:S=h1UBGonTuWU3D23L;
NID=31=Z-nhwMjUP63e0tYMTp-3T1igMSPnNS1eM1kN1_
↳ DUrnO2zW1cPM4JE3AJec9b_vG-
↳ FibFXszOApfbhBA1BOX4dKx4L8ZDdeiKwqekgP5_
↳ kzELtC2mUHx7RHx3PIttcuZ
```

```
            1406
HTTP/1.1 200 OK
Date: Tue, 09 Feb 2010 01:18:37 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 4633
X-XSS-Protection: 0
```

Имеется также возможность указать поток, который требуется просмотреть, предоставив подробные сведения об адресе. Например, по приведенной ниже команде из файла перехвата `packets.pcap` будет извлечен поток по протоколу UDP для указанных конечных точек и портов. В частности, по данной команде из указанного файла перехвата будет выведен поток UDP для конечной точки с адресом **192.168.1.5** в порту **23429** ❶ и конечной точки с адресом **4.2.2.1** в порту **53** ❷.

```
C:\Program Files\Wireshark>tshark -r packets.pcap -z
↳ follow,udp,ascii,192.168.1.5:23429 ❶,4.2.2.1:53 ❷
```

Ниже перечислены некоторые из наиболее употребительных (в том числе и мной) вариантов вывода статистических данных из файла перехвата.

- **ip_hosts,tree**. Отображает каждый IP-адрес из файла перехвата наряду с интенсивностью и долями в процентах сетевого трафика, приходящегося на каждый адрес.
- **io,phs**. Отображает иерархию всех сетевых протоколов, обнаруженных в файле перехвата.
- **http,tree**. Отображает статистические данные, касающиеся HTTP-запросов и ответов.
- **http_req,tree**. Отображает статистические данные по каждому HTTP-запросу.
- **smb,srt**. Отображает статистические данные по командам протокола SMB для анализа обмена данными по сети в Windows.
- **endpoints,wlan**. Отображает статистические данные о конечных точках в беспроводной сети.
- **expert**. Отображает экспертную информацию (диаграммы, ошибки и прочее) из файла перехвата.

Имеется немало других полезных вариантов, доступных для применения вместе с аргументом **-z**, но здесь, к сожалению, недостаточно места, чтобы

описать их полностью. Впрочем, если вы собираетесь часто пользоваться утилитой TShark, вам следует ознакомиться с официальной документацией, чтобы изучить все варианты, доступные для вывода статистических данных о перехваченном сетевом трафике. Эту документацию можно найти по следующему адресу:

```
https://www.wireshark.org/docs/man-pages/tshark.html
```

Сравнение утилит TShark и tcpdump

Обе утилиты командной строки, предназначенные для анализа пакетов и рассмотренные ранее в этой главе, вполне пригодны для решения соответствующих задач, и любая из них позволит решить любую насущную задачу с разной степенью прилагаемых усилий. Ниже перечислены некоторые существенные отличия утилит TShark и `tcpdump`, которые следует иметь в виду, выбирая наиболее подходящую из них для решения конкретной задачи.

- **Операционная система.** Утилита `tcpdump` доступна только в Unix-подобных операционных системах, тогда как утилита TShark может функционировать как в Unix-подобных операционных системах, так и в Windows.
- **Поддержка протоколов.** В обеих утилитах поддерживаются сетевые протоколы третьего и четвертого уровня. Хотя поддержка сетевых протоколов седьмого уровня в утилите `tcpdump` ограничена, в утилите TShark она довольно полная, поскольку в ней доступны интерпретаторы сетевых протоколов в Wireshark.
- **Средства анализа пакетов.** Обе утилиты в значительной мере опираются на способности человека анализировать и получать такие же ощутимые результаты, как и в Wireshark. Это может оказать существенную помощь при анализе пакетов в отсутствие графического интерфейса.

Наличие подходящих инструментальных средств для анализа пакетов и личные предпочтения обычно имеют решающее значение при их выборе. Правда, эти инструментальные средства обладают настолько схожими возможностями, что, изучая одно из них, можно узнать немало о другом. Благодаря этому ваши возможности становятся более универсальными и в то же время расширяется арсенал ваших инструментальных средств для анализа пакетов.

7

ПРОТОКОЛЫ СЕТЕВОГО УРОВНЯ

Чтобы обнаружить ненормальный сетевой трафик, включая диагностику сетевых задержек, выявление зловредных приложений или полное устранение угроз безопасности, необходимо знать, что собой представляет нормальный сетевой трафик. В этой и двух последующих главах поясняется, каким образом действует сетевой трафик как на уровне пакетов, так и на всех уровнях модели OSI снизу вверх. С каждой рассматриваемой здесь частью сетевого протокола связан хотя бы один файл перехвата, который можно загрузить и использовать непосредственно.

В этой главе основное внимание уделяется протоколам сетевого уровня, которые являются основным исполнительным механизмом обмена данных по сети. К их числу относятся протоколы ARP, IPv4, IPv6, ICMP и ICMPv6.

Эта и две последующие главы, посвященные сетевым протоколам, безусловно, относятся к самым главным в данной книге. Пропустить материал этих глав — все равно, что пытаться готовить праздничный обед, не разогрев печь. Если вы уже знаете досконально принцип действия сетевых протоколов, просмотрите хотя бы бегло эти главы, чтобы освежить в памяти структуру пакетов в каждом из них.

Протокол преобразования адресов (ARP)

Логические и физические адреса служат для обмена данными по сети. В частности, логические адреса служат для обмена данными по нескольким сетям между косвенно связанными устройствами. А физические адреса упрощают обмен данными в одном сегменте сети между устройствами, непосредственно связанными друг с другом через коммутатор. Как правило, оба эти типа адресации должны взаимодействовать, чтобы произошел нормальный обмен данными по сети.

Рассмотрим в качестве примера сценарий, где требуется передать данные в устройство, подключенное к сети. Этим устройством может быть сервер с определенным портом или рабочая станция, с которой требуется обмениваться данными с устройством в сети. Приложению, применяемому для инициализации обмена данными, уже известно об удаленном IP-адресе хоста (через службу DNS, как поясняется в главе 9). Это означает, что у системы должно быть все, что ей потребуется для формирования данных от третьего до седьмого уровня в пакете, который необходимо передать. И лишь частью данных, требующихся в данный момент, является информация второго канального уровня, содержащая MAC-адрес целевого хоста.

MAC-адреса необходимы потому, что в коммутаторе, соединяющем устройства в сети, используется *таблица ассоциативной памяти*, в которой перечислены MAC-адреса всех устройств и соответствующие им номера портов. Когда коммутатор принимает сетевой трафик, предназначенный для устройства с конкретным MAC-адресом, он использует данную таблицу, чтобы выяснить номер порта, через который следует переслать этот трафик. Если же MAC-адрес получателя неизвестен, то передающее устройство сначала поищет этот адрес в своем кеше. А если искомый адрес отсутствует и в кеше, то он может быть получен путем дополнительного обмена данными по сети.

Процесс, применяемый при организации сети по протоколу TCP/IP (с использованием протокола IPv4) для преобразования IP-адреса в MAC-адрес, называется *протоколом преобразования адресов* (Address Resolution Protocol – ARP). Он определен в стандарте RFC 826. В процессе преобразования адресов по протоколу ARP применяются только два типа пакетов: ARP-запрос и ARP-ответ (рис. 7.1).

ПРИМЕЧАНИЕ *Стандарт RFC (Request for Comments – запрос комментариев) является техническим изданием, выпускаемыми сообществами IETF (Engineering Task Force – Инженерный совет Интернета) и ISOC (Internet Society – Сообщество Интернета) и предназначенным в качестве механизма для определения стандартов, реализуемых для сетевых протоколов. Найти нужную документацию по сетевым протоколам можно на начальной странице RFC Editor, доступной по адресу* **`http://www.rfc-editor.org/`**.

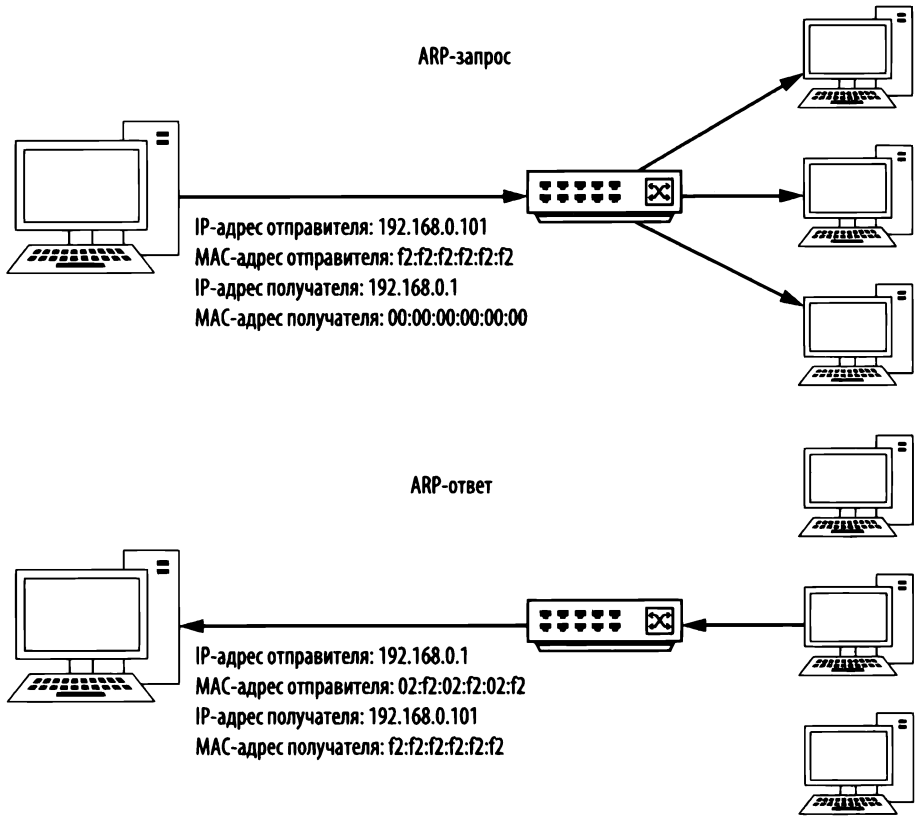


Рис. 7.1. Процесс преобразования адресов по протоколу ARP

Передающий компьютер посылает ARP-запрос, в котором сообщается следующее: "Всем привет. Мой IP-адрес — **`192.168.0.101`**, а мой MAC-адрес — **`f2:f2:f2:f2:f2:f2`**. Мне нужно послать кое-что тому, у кого имеется IP-адрес **`192.168.0.1`**, но я не знаю его физический адрес. Может ли тот, у кого этот IP-адрес, сообщить мне свой MAC-адрес?"

Этот пакет посылается в широковещательном режиме и поэтому принимается на каждом устройстве в сетевом сегменте. Все устройства, не имеющее указанного IP-адреса, должны проигнорировать упомянутый выше запрос. А устройство с указанным IP-адресом должно отправить следующий ARP-ответ: "Привет передающему устройству! Я именно тот, кого вы ищете по IP-адресу **`192.168.0.1`**. Мой MAC-адрес — **`02:f2:02:f2:02:f2`**".

Как только процесс преобразования адресов завершится, передающее устройство обновит свою кеш-память и поместит в нее соответствие MAC- и IP-адресов приемного устройства и начнет передачу данных. Наблюдая этот процесс в действии, можно понять, как именно он действует. Но прежде чем перейти к конкретным примерам, рассмотрим структуру заголовка ARP-пакета.

ПРИМЕЧАНИЕ *Чтобы просмотреть ARP-таблицу хостов в системах Unix и Windows, достаточно ввести команду* **`arp -a`** *в командной строке.*

Структура ARP-пакета

Как показано на рис. 7.2, ARP-пакет состоит заголовка, содержащего описанные ниже поля.

Протокол преобразования адресов (ARP)					
Смещение	Октет	0	1	3	4
Октет	Бит	0–7	8–15	0–7	8–15
0	0	Тип оборудования		Тип протокола	
4	32	Длина физического адреса	Длина протокольного адреса	Операция	
8	64	Физический адрес отправителя			
12	96	Физический адрес отправителя		Протокольный адрес отправителя	
16	128	Протокольный адрес отправителя		Физический адрес получателя	
20	160	Физический адрес получателя			
24+	192+	Протокольный адрес получателя			

Рис. 7.2. Структура заголовка в ARP-пакете

- **Тип оборудования (Hardware Type).** Тип среды, применяемой на втором уровне для передачи данных. Как правило, это сеть Ethernet (т.е. тип 1).
- **Тип протокола (Protocol Type).** Сетевой протокол верхнего уровня, который используется для передачи ARP-запроса. Как правило это IPv4 (`0x0800`).
- **Длина физического адреса (Hardware Address Length).** Длина применяемого физического адреса (в октетах или байтах). Для сети Ethernet ее величина равна 6.
- **Длина протокольного адреса (Protocol Address Length).** Длина применяемого логического адреса по протоколу указанного типа (в октетах или байтах). Для протокола IPv4 ее величина равна 4.
- **Операция (Operation).** Функция пакета : код **`1`** – для запроса или код **`2`** – для ответа.
- **Физический адрес отправителя (Sender Hardware Address).** Физический адрес, присвоенный отправителю пакетов.
- **Протокольный адрес отправителя (Sender Protocol Address).** Протокольный адрес, присвоенный отправителю пакетов.

- **Физический адрес получателя (Target Hardware Address).** Искомый физический адрес получателя. В ARP-запросе указываются все нули.
- **Протокольный адрес получателя (Target Protocol Address).** Предполагаемый адрес получателя пакета по протоколу верхнего уровня.

А теперь откройте файл перехвата `arp_resolution.pcapng`, чтобы наблюдать процесс преобразования адресов в действии. Рассматривая этот процесс, уделим основное внимание каждому пакету в отдельности.

Пакет 1: ARP-запрос

Файл перехвата **`arp_resolution.pcapng`**

Первым пакетом следует ARP-запрос, как показано на рис. 7.3. Чтобы убедиться, что это действительно широковещательный пакет, достаточно проверить заголовок **`Ethernet`** в панели Packet Details приложения Wireshark. Адрес получателя данного пакета — **`ff:ff:ff:ff:ff:ff`** ❶. Это широковещательный адрес сети Ethernet, и все, что по нему посылается, будет приниматься всеми устройствам в текущем сегменте сети. Адрес отправителя данного пакета находится в Ethernet-заголовке и, по существу, является MAC-адресом передающего устройства ❷.

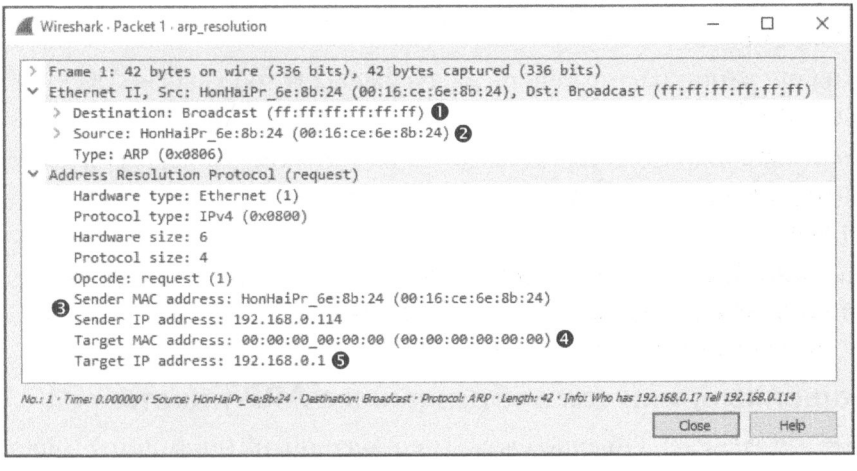


Рис. 7.3. Пакет ARP-запроса

Принимая во внимание рассматриваемую здесь структуру заголовка пакета ARP, можно выяснить, что ARP-запрос на самом деле направляется в сети Ethernet по протоколу IPv4. В нем указан IP-адрес (**`192.168.0.114`**) и MAC-адрес (**`00:16:ce:6e:8b:24`**) отправителя ❸, а также IP-адрес (**`192.168.0.1`**) получателя ❺. В то же время MAC-адрес получателя, который требуется выяснить, неизвестен, и поэтому он задан как **`00:00:00:00:00:00`** ❹.

Пакет 2: ARP-ответ

В ответе на первоначальный запрос (рис. 7.4) Ethernet-заголовок теперь содержит адрес получателя, соответствующий MAC-адресу отправителя из первого пакета. Заголовок такого ARP-ответа выглядит таким же образом, как и у ARP-запроса, за рядом перечисленных ниже исключений.

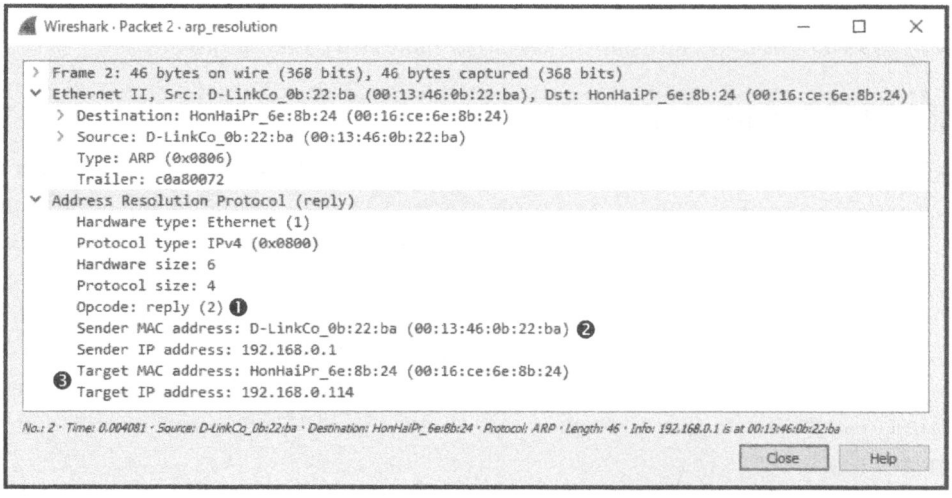


Рис. 7.4. Пакет ARP-ответа

- Код операции пакета теперь равен **0x0002** ❶, обозначая ARP-ответ, а не запрос.
- Адресная информация указывается в обратном порядке: сначала MAC- и IP-адрес отправителя, а затем MAC- и IP-адрес получателя ❸.
- Но самое главное, что теперь у нас имеется вся необходимая информация. Это означает, что мы определили MAC-адрес (**00:13:46:0b:22:ba**) ❷ хоста, находящегося по адресу **192.168.0.1**.

Непрошенные, или самообращенные ARP-пакеты

Файл перехвата **arp_gratuitous.pcapng**

Обычно слово "непрошенный" (gratuitous) имеет отрицательный оттенок. Но порой необоснованное применение протокола ARP все же дает положительный результат, и в этом случае его пакеты называются *самообращенными*.

IP-адрес устройства нередко может меняться. Когда это происходит, соответствие IP- и MAC-адресов в кеш-памяти сетевых узлов может оказаться недействительным. Чтобы предотвратить возникающие в связи с этим ошибки при обмене данными, по сети передается самообращенный ARP-пакет, который вынуждает любое принимающее его устройство обновить кеш-память

новым соответствием IP- и MAC-адресов (рис. 7.5). По сути этот пакет анонсирует изменения по протоколу ARP, произошедшие с конкретным хостом.

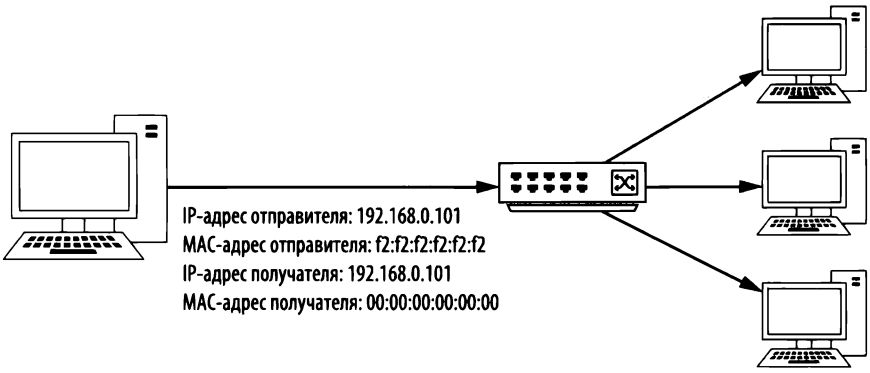


Рис. 7.5. Процесс передачи самообращенного ARP-пакета

Появление самообращенного ARP-пакета может произойти в самых разных случаях. И чаще всего это происходит в связи с изменением IP-адреса. Чтобы убедиться в этом, откройте файл перехвата `arp_gratuitous.pcapng`, который содержит единственный пакет (рис. 7.6), поскольку это все, что передается при самообращенной передаче по протоколу ARP.

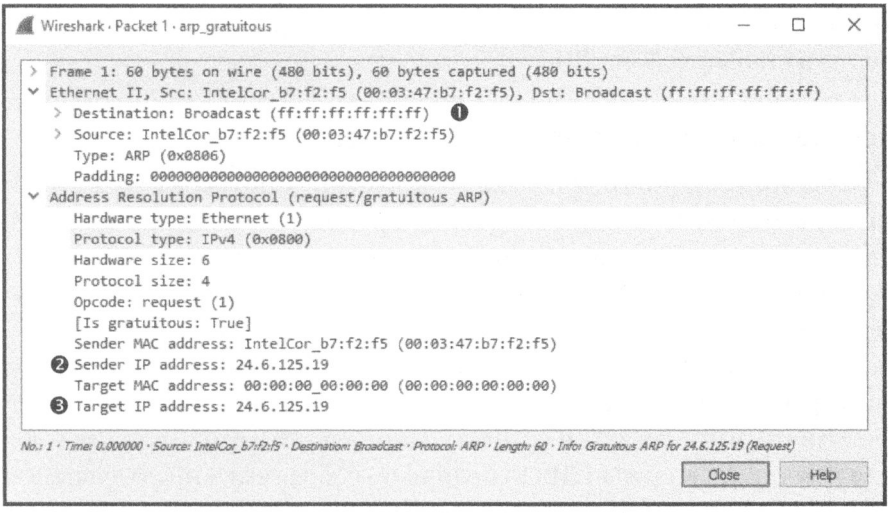


Рис. 7.6. Самообращенный ARP-пакет

Проанализировав Ethernet-заголовок, можно заметить, что самообращенный пакет рассылается в широковещательном режиме всем хостам в сети ❶. ARP-заголовок выглядит так же, как и в ARP-запросе, за исключением того, что IP-адреса отправителя ❷ и получателя ❸ одинаковы. Когда самообращенный

пакет принимается другими хостами в сети, это приводит к обновлению их таблиц ARP новым соответствием IP- и MAC-адресов. Поскольку данный ARP-пакет рассылается сам себе по инициативе отправителя, но в результате его появления все клиенты сети обновляют свои кеш-памяти ARP, его и назвали самообращенным (gratuitous).

Появление самообращенных ARP-пакетов можно обнаружить в ряде случаев. Как упоминалось выше, смена IP-адреса устройства приведет к появлению самообращенного пакета. Кроме того, при запуске некоторых операционных систем происходит передача самообращенных ARP-пакетов. А в ряде систем самообращенные ARP-пакеты служат для поддержки балансировки нагрузки.

Межсетевой протокол (IP)

Основное назначение сетевых протоколов третьего уровня модели OSI состоит в том, чтобы обеспечить связь для обмена данными между узлами, находящихся в разных сетях. Как было показано выше, MAC-адреса служат для обмена данными в одной сети на втором уровне. Аналогично адреса на третьем уровне служат для обмена данными между разными сетями. Для этого можно воспользоваться несколькими сетевыми протоколами, но наиболее распространенным среди них является *протокол IP* (Internet Protocol – межсетевой протокол), который в настоящее время имеется в двух версиях: IP версии 4 и IP версии 6. Итак, начнем с рассмотрения протокола IP версии 4 (IPv4), определенного в стандарте RFC 791.

Межсетевой протокол версии 4 (IPv4)

Чтобы понять принцип действия межсетевого протокола IPv4, необходимо знать, каким образом сетевой трафик проходит между сетями. Протокол IPv4 служит основным исполнительным элементом в процессе обмена данными и в конечном счете отвечает за перенос данных между устройствами независимо от местоположения конечных точек устанавливаемой связи.

Простая сеть, в которой все устройства связаны через концентраторы или коммутаторы, называется *локальной вычислительной сетью* (ЛВС). Если же требуется связать две ЛВС, это можно сделать с помощью маршрутизатора. Сложные сети могут состоять из тысяч ЛВС, соединяемых по всему миру тысячами маршрутизаторов. Сам Интернет состоит из миллионов ЛВС и маршрутизаторов.

Адреса протокола IPv4

В протоколе IPv4 *адреса* представляют собой 32-разрядные числа, присваиваемые сетевыми администраторами, которые однозначно идентифицируют устройства, подключаемые к сети. Вряд ли кому-нибудь будет удобно

запоминать последовательность единиц и нулей длиной 32 бита, и поэтому IP-адреса записываются с помощью *четырехкомпонентной* (или *десятично-точечной*) *системы обозначений через точки.*

В десятично-точечной системе обозначений каждый из четырех наборов единиц и нулей, составляющих IP-адрес, преобразуется в десятичное число в пределах от **0** до **255** в формате **A.B.C.D** (рис. 7.7). Рассмотрим в качестве примера IP-адрес **11000000 10101000 00000000 00000001** в исходной двоичной форме. Очевидно, что в такой форме запомнить или записать IP-адрес очень трудно. А в десятично-точечной системе обозначений этот адрес можно представить как **192.168.0.1**.

Рис. 7.7. IPv4-адрес в десятично-точечной системе обозначений

IP-адрес состоит из двух частей, которые идентифицируют как саму *сеть*, так и *узел* (или хост), находящийся в этой сети. В сетевой части задается номер ЛВС, к которой подключено устройство, а в узловой части — номер самого устройства в данной сети. Однозначно определить, к какой именно части (сетевой или узловой) относятся составляющие IP-адреса, удается далеко не всегда. Поэтому IP-адреса снабжаются вспомогательной адресной информацией, называемой *маской сети*, либо (чаще всего) — *маской подсети.*

ПРИМЕЧАНИЕ *В этой книге под термином IP-адрес мы всегда подразумеваем адрес протокола IPv4. Далее в этой главе будет рассмотрена версия 6 протокола IP, в которой применяются разные правила адресации. Поэтому всякий раз, когда мы будем упоминать об IP-адресе протокола IPv6, мы будем явно на это указывать.*

Маска сети позволяет выяснить, к какой именно части (сетевой или узловой) относятся составляющие IP-адреса. Длина маски сети также равна 32 битам, где каждый бит с установленной единицей обозначает составляющую IP-адреса, относящуюся к сетевой части. А остальные биты с установленными нулями относятся к узловой части.

Рассмотрим в качестве примера IP-адрес **10.10.1.22** и его двоичную форму **00001010 00001010 00000001 00010110**. Чтобы определить расположение каждой части этого IP-адреса, достаточно воспользоваться маской сети. В данном случае применяется маска сети **11111111 11111111 00000000 00000000**. Это

означает, что первая половина IP-адреса (`10.10` или `00001010 00001010`) идентифицирует сетевую часть, а вторая половина (`.1.22` или `00000001 00010110`) – отдельный узел (или хоста) в сети, как показано на рис. 7.8.

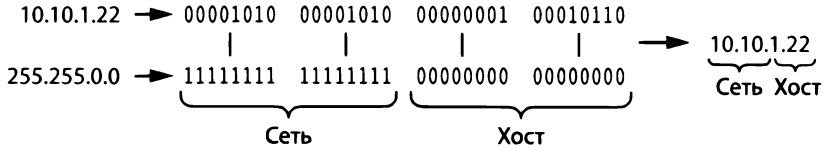


Рис. 7.8. Маска сети определяет расположение отдельных битов в IP-адресе

И как следует из рис. 7.8, маски сети также могут быть записаны в десятично-точечной системе обозначений. Например, маску сети `11111111 11111111 00000000 00000000` можно записать как `255.255.0.0`.

IP-адреса и маски сети обычно записываются в форме CIDR (Classless Inter-Domain Routing – бесклассовая междоменная маршрутизация). В этой форме IP-адрес записывается полностью, а далее следует косая черта (`/`) и количество бит, представляющих сетевую часть IP-адреса. Например, IP-адрес `10.10.1.22` и маску сети `255.255.0.0` можно записать в форме CIDR как `10.10.1.22/16`.

Структура заголовка в пакете IPv4

IP-адреса отправителя и получателя являются главными составляющими заголовка пакета IPv4, но это еще не все сведения о протоколе IP, которые можно обнаружить в пакете. IP-заголовок довольно сложный в сравнении с рассмотренным ранее заголовком пакета ARP. Он содержит немало дополнительных полей с информацией, помогающей протоколу IP выполнять свои функции.

Как показано на рис. 7.9, заголовок пакета IPv4 состоит из следующих полей.

Межсетевой протокол версии 4 (IPv4)							
Смещение	Октет	0		1	2		3
Октет	Бит	0–3	4–7	8–15	16–18	19–23	24–31
0	0	Версия	Длина заголовка	Тип службы	Общая длина		
4	32	Идентификация			Флаги	Смещение фрагмента	
8	64	Время жизни		Протокол	Контрольная сумма заголовка		
12	96	IP-адрес отправителя					
16	128	IP-адрес получателя					
20	160	Параметры					
24+	192+	Данные					

Рис. 7.9. Структура заголовка в пакете IPv4

- **Версия (Version).** Версия применяемого протокола IP (для протокола IPv4 она всегда будет иметь номер 4).
- **Длина заголовка (Header Length).** Длина IP-заголовка.
- **Тип службы (Type of Service).** Состоит из флагов приоритета и класса обслуживания, с помощью которых маршрутизаторы устанавливают приоритетность сетевого трафика.
- **Общая длина (Total Length).** Длина IP-заголовка с учетом данных, включаемых в пакет.
- **Идентификация (Identification).** Идентификационный номер, однозначно обозначающий целый пакет или последовательность фрагментированных пакетов.
- **Флаги (Flags).** Определяют, является ли пакет частью последовательности фрагментированных пакетов.
- **Смещение фрагмента (Fragment Offset).** Если пакет фрагментирован, значение в этом поле служит для расположения пакетов в правильном порядке.
- **Время жизни (Time to Live).** Определяет время жизни пакета, измеряемое в переходах или секундах прохождения через маршрутизаторы.
- **Протокол (Protocol).** Обозначает заголовок протокола транспортного уровня, который инкапсулирован в пакет IPv4.
- **Контрольная сумма заголовка (Header Checksum).** Механизм выявления ошибок, применяемый с целью проверить, не испорчено и не нарушено ли содержимое IP-заголовка.
- **IP-адрес отправителя (Source IP Address).** IP-адрес хоста, пославшего пакет.
- **IP-адрес места получателя (Destination IP Address).** IP-адрес хоста, для которого предназначен пакет.
- **Параметры (Options).** Это поле зарезервировано для дополнительных параметров протокола IP, включая маршрутизацию от источника и отметки времени.
- **Данные (Data).** Конкретные данные, передаваемые по протоколу IP.

Время жизни

Файлы перехвата **ip_ttl_source.pcapng** *и* **ip_ttl_dest.pcapng**

Величина *времени жизни* (Time to Live — TTL) определяет период времени, которое может истечь, или максимальное количество маршрутизаторов, которые может пройти пакет, прежде чем он будет аннулирован по протоколу IPv4. Время жизни определяется при создании

пакета и обычно уменьшается на единицу всякий раз, когда пакет пересылается следующему маршрутизатору. Так, если время жизни пакета равно 2, то первый маршрутизатор, которого он достигнет, уменьшит это время до 1 и пересылает пакет второму маршрутизатору. А тот, в свою очередь, уменьшит время жизни пакета до нуля, и если конечный получатель пакета оказывается не в текущей сети, то такой пакет аннулируется (рис. 7.10).

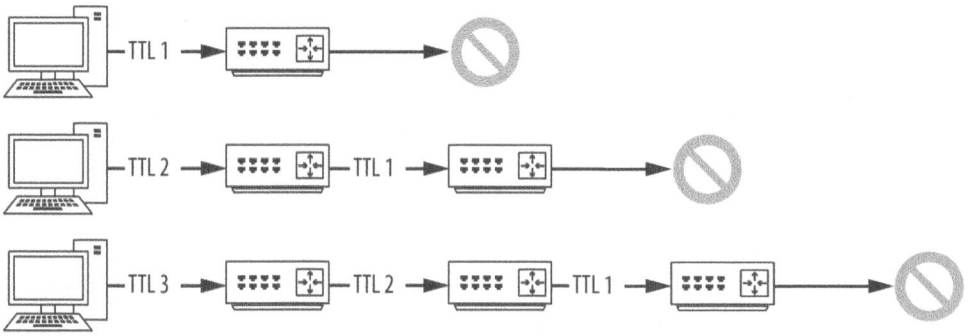


Рис. 7.10. Время жизни пакета (TTL) уменьшается на единицу всякий раз, когда он проходит через маршрутизатор

Почему так важно иметь в виду время жизни пакетов? Как правило, оно представляет интерес только с точки зрения того времени, которое требуется для прохождения пакета от его отправителя до получателя. Представьте, однако, пакет, который должен дойти до хоста через Интернет, пройдя десятки маршрутизаторов. В какой-то точке этого пути может встретиться неверно настроенный маршрутизатор, из-за чего пакет собьется со своего пути к конечному месту своего назначения. В таком случае неверно настроенный маршрутизатор способен сделать немало такого, что могло бы нарушить нормальную работу сети. В частности, он может разослать пакет по сети в бесконечном цикле.

Бесконечный цикл может вызвать самые разные осложнения, но, как правило, это приводит к аварийному сбою как отдельной программы, так и всей операционной системы. Теоретически то же самое может произойти с пакетами в сети, которые будут циклически перемещаться между маршрутизаторами. А по мере роста количества таких пакетов в сети ее пропускная способность будет падать вплоть до возникновения условия отказа в обслуживании. Во избежание этого и было внедрено время жизни пакетов.

Обратимся к конкретному примеру анализа пакетов в Wireshark. В частности, файл перехвата `ip_ttl_source.pcapng` содержит два пакета ICMP. Для доставки пакетов в сетевом протоколе ICMP, рассматриваемом далее в этой главе, применяется протокол IP. Об этом можно судить, развернув соответствующую часть IP-заголовка в панели Packet Details. Как показано на рис. 7.11,

в данном пакете применяется версия 4 протокола IP ❶, длина IP-заголовка составляет 20 байт ❷, общая длина заголовка и полезной информации — 60 байт ❸, а величина времени жизни пакета — 128 ❹.

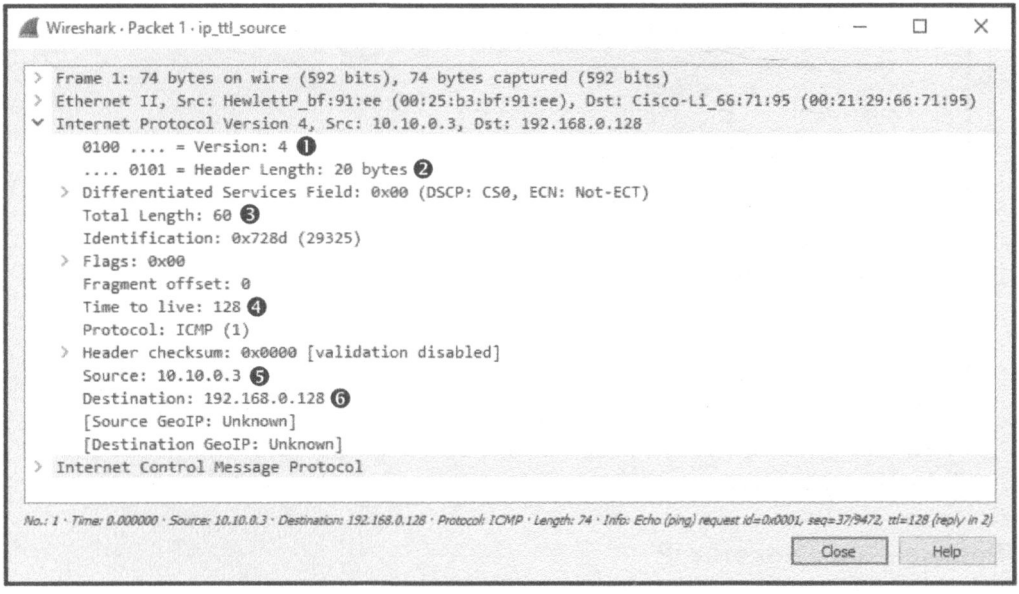


Рис. 7.11. IP-заголовок исходного пакета

Основное назначение эхо-запроса по протоколу ICMP — проверить связь между устройствами. В частности, данные посылаются от одного хоста к другому в виде запроса, а принимающий их хост должен отправить эти данные обратно в качестве ответа. В упомянутом выше файле перехвата одно устройство с адресом **10.10.0.3** ❺ посылает эхо-запрос ICMP другому устройству с адресом **192.168.0.128** ❻. Этот первоначальный файл перехвата был создан в хосте-отправителе по адресу **10.10.0.3**.

А теперь откройте файл перехвата `ip_ttl_dest.pcapng`. В этом файле данные перехвачены в хосте-получателе по адресу **192.168.0.128**. Разверните IP-заголовок первого пакета из этого файла перехвата, чтобы выяснить время жизни данного пакета.

Как следует из рис. 7.12, время жизни первого пакета равно **127** ❶, т.е. оно на **1** меньше первоначального времени жизни данного пакета, которое равно **128**. Даже не зная структуру сети, можно сделать вывод, что оба упомянутых выше устройства разделены маршрутизатором, а следовательно, время жизни пакета при прохождении через маршрутизатор уменьшается на **1**.

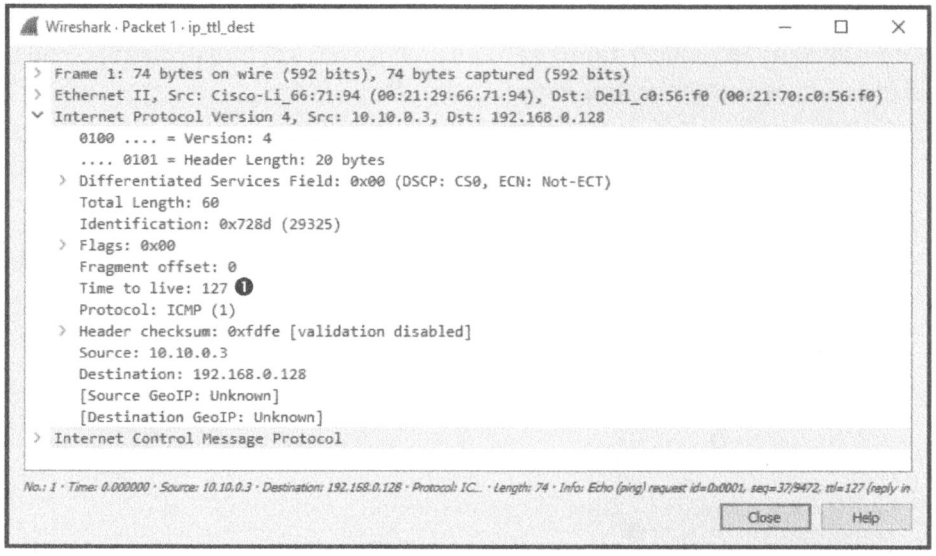


Рис. 7.12. Как следует из IP-заголовка, время жизни пакета уменьшилось на единицу

Фрагментация пакетов IP

Файл перехвата **ip_frag_source.pcapng**

Фрагментация пакетов – это средство протокола IP, обеспечивающее надежную доставку данных по разнотипным сетям путем разбиения потока данных на мелкие фрагменты. Фрагментация пакетов основывается на размере *максимального передаваемого блока* (MTU, Maximum Transmission Unit) для протокола второго канального уровня, а также конфигурации устройств, применяющих этот протокол. Как правило, на втором канальном уровне применяется сетевой протокол Ethernet, где размер блока MTU по умолчанию равен 1500. Это означает, что по сети Ethernet можно передать пакет размером максимум 1500 байт, не включая длину Ethernet-заголовка, равную 14 байт.

ПРИМЕЧАНИЕ *Несмотря на то что имеются стандартные значения размера блока MTU, очень часто размер этого блока на конкретном устройстве может быть изменен вручную. Длина блока MTU устанавливается для каждого отдельного сетевого интерфейса и может изменяться в системах Windows и Linux, а также в интерфейсах управляемых маршрутизаторов.*

Когда устройство готово к передаче IP-пакета, оно определяет, стоит ли разбивать пакет на фрагменты, сравнивая размер данных в пакете с размером блока MTU того сетевого интерфейса, через который будет передаваться этот

пакет. Если размер данных оказывается больше, чем у блока MTU, пакет будет фрагментирован. Процесс фрагментации пакетов разделяется на следующие стадии.

1. Устройство разделяет данные на количество пакетов, требующееся для их успешной передачи.
2. В поле общей длины (Total Length) каждого IP-заголовка задается размер сегмента для каждого фрагмента.
3. Во всех пакетах, кроме последнего в потоке данных, устанавливается флаг More fragments (Дополнительные фрагменты).
4. В поле смещения фрагмента (Fragment offset) IP-заголовка задается конкретное смещение для текущего фрагмента, которое он должен занимать в потоке данных.
5. Далее пакеты передаются по сети.

Файл перехвата `ip_frag_source.pcapng` был создан на компьютере с адресом **`10.10.0.3`**, передавшим эхо-запрос по протоколу ICMP устройству с адресом **`192.168.0.128`**. Обратите внимание на то, что в столбце из панели Packet List перечислены два фрагментированных IP-пакета, после которых следует эхо-запрос по протоколу ICMP. Итак, начните с анализа IP-заголовка первого пакета (рис. 7.13).

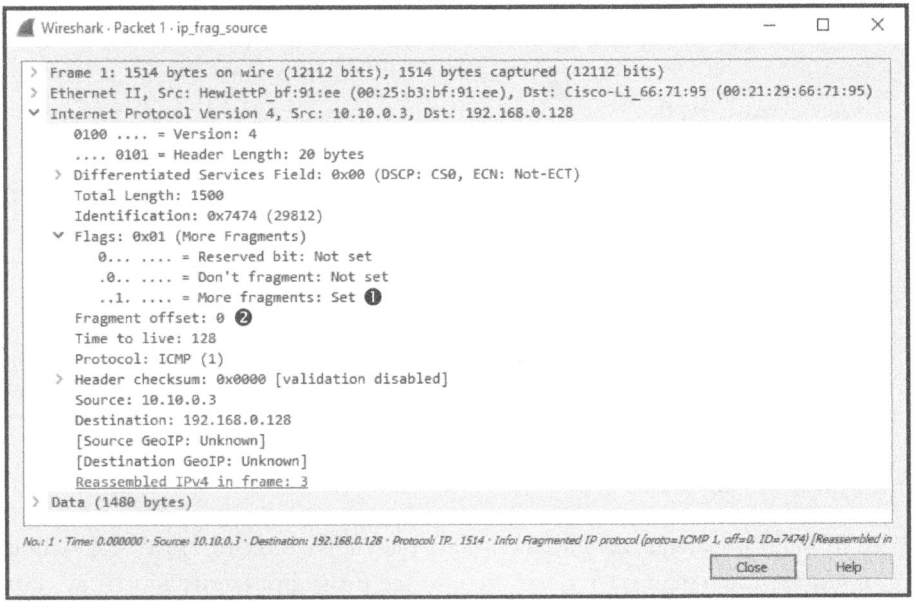


Рис. 7.13. Значения в полях More fragments и Fragment offset из IP-заголовка указывают на то, что это фрагментированный пакет

Как видите, этот пакет составляет часть фрагмента данных, исходя из значений в полях More fragments и Fragment offset. Если пакеты являются фрагментами, то в поле More fragments будет установлен соответствующий флаг или же в поле Fragment offset будет указано ненулевое положительное значение. Так, в поле More fragments из IP-заголовка первого пакета установлен флаг ❶, указывающий на то, что принимающее устройство должно предполагать получение еще одного пакета из данной последовательности. А в поле Fragment offset установлено нулевое значение ❷, указывающее на то, что данный пакет является первым в последовательности фрагментов.

В поле More fragments IP-заголовка второго пакета (рис. 7.14) также установлен флаг ❶, но на этот раз в поле Fragment offset задано значение **1480** ❷. Это свидетельствует о том, что размер первого пакета соответствует размеру блока MTU и составляет 1500 байт, из которых 20 байт занимает IP-заголовок. Поэтому данные из второго полученного пакета должны быть помещены в поток со смещением в 1480 байт.

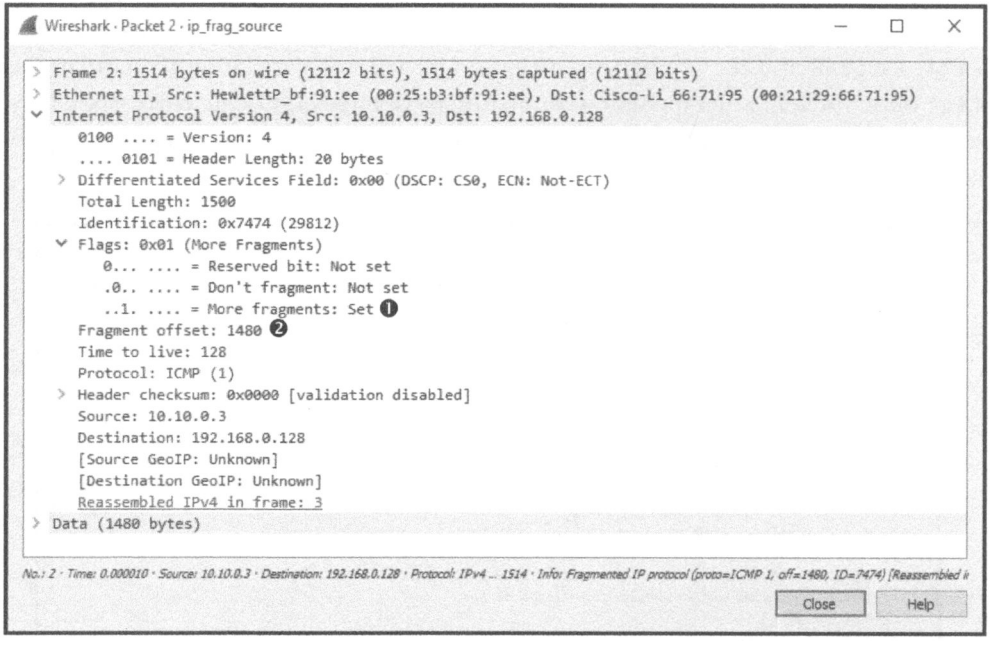


Рис. 7.14. Значение в поле Fragment offset увеличивается в зависимости от размера пакетов

В поле More fragments из IP-заголовка третьего пакета (рис. 7.15) флаг не установлен ❷, указывая на то, что это последний фрагмент в потоке данных, а в поле Fragment offset задано значение **2960** ❸, получаемое в результате сложения размеров второго и первого пакетов **(1500 – 20) * 2**. Все эти фрагменты

могут быть распознаны как часть одной и той же последовательности данных, поскольку в поле идентификации (Identification) их IP-заголовков установлены одинаковые значения ❶.

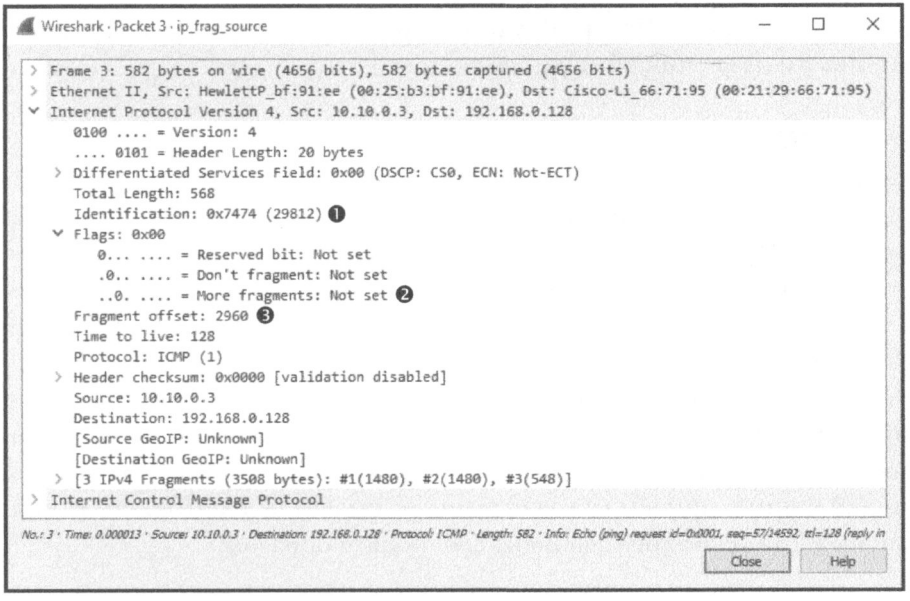


Рис. 7.15. Флаг в поле More fragments не установлен, указывая на то, что это последний фрагмент в потоке данных

Теперь фрагментированные пакеты можно обнаружить в сети не так часто, как это бывало прежде. Тем не менее следует ясно понимать причины фрагментации пакетов, чтобы, обнаружив их, произвести диагностику сети или выявить недостающие фрагменты.

Межсетевой протокол версии 6 (IPv6)

Когда была составлена спецификация сетевого протокола IPv4, никто даже представить себе не мог такое количество устройств, подключенных к Интернету, которые существуют в настоящее время. Максимальное пространство адресов по протоколу IPv4 ограничивалось всего лишь 4,3 миллиарда. Но фактическое пространство адресов сокращается еще больше, если вычесть диапазоны адресов, зарезервированные для специального применения, включая тестирование, широковещательный трафик и внутренние адреса по стандарту RFC 1918. Несмотря на то что было предпринято несколько попыток так или иначе задержать исчерпание пула IPv4-адресов, в конечном итоге остался единственный способ преодолеть данное ограничение: разработать новую версию спецификации на сетевой протокол IP.

Таким образом, была составлена спецификация на сетевой протокол IPv6, а первая ее версия была выпущена как стандарт RFC 2460 в 1998 году. В этой версии был внедрен ряд усовершенствований производительности, включая намного более обширное пространство адресов. В этом разделе мы рассмотрим структуру пакета IPv6 и обсудим отличия протокола IPv6 от его предшественника в отношении обмена данными по сети.

Адреса протокола IPv6

Длина адресов в протоколе IPv4 ограничивалась 32 битами, обеспечивая пространство, измерявшееся миллиардами адресов. А длина адресов в протоколе IPv6 составляет 128 бит, что в 4 раза больше и обеспечивает пространство в 2^{128} адресов, позволяющее охватить ундециллионы хостов (т.е. триллионы триллионов триллионов). И это весьма существенное усовершенствование!

Но поскольку длина адресов протокола IPv6 составляет 128 бит, то манипулировать ими в двоичной форме очень неудобно. Чаще всего IPv6-адрес записывается в виде восьми двухбайтовых групп, представленных в шестнадцатеричной форме, причем каждая группа разделяется двоеточием. Например, самый простой IPv6-адрес выглядит следующим образом:

```
1111:aaaa:2222:bbbb:3333:cccc:4444:dddd
```

Глядя на такой адрес, невольно возникает такая же мысль, как и у многих из тех, кто привык запоминать IPv4-адреса: запомнить IPv6-адреса практически невозможно. И это неизбежный компромисс, на который пришлось пойти ради расширения пространства адресов.

Впрочем, столь неудобное обозначение IPv6-адресов иногда можно упростить, свернув некоторые группы нулей. Рассмотрим в качестве примера следующий IPv6-адрес:

```
1111:0000:2222:0000:3333:4444:5555:6666
```

Группы, состоящие из нулей, можно полностью свернуть, чтобы сделать их невидимыми, как показано ниже.

```
1111::2222:0000:3333:4444:5555:6666
```

Но свернуть можно только одну группу нулей, и поэтому следующий адрес считается недействительным:

```
1111::2222::3333:4444:5555:6666
```

Следует также иметь в виду, что начальные нули можно исключить из IPv6-адресов. Рассмотрим в качестве примера следующий адрес, где в четвертой, пятой и шестой группах имеются начальные нули:

```
1111:0000:2222:0333:0044:0005:ffff:ffff
```

Этот адрес можно представить более эффективно приведенным ниже образом. Безусловно, пользоваться таким адресом не так просто, как IPv4-адресом, но все же намного легче, чем в более длинной записи.

```
1111::2222:333:44:5:ffff:ffff
```

IPv6-адрес также состоит из сетевой и узловой частей, которые нередко называются *сетевым префиксом* и *идентификатором интерфейса* соответственно. Распределение этих частей меняется в зависимости от классификации обмена данными по протоколу IPv6. В частности, сетевой трафик по протоколу IPv6 разделяется на следующие категории: одноадресатный (unicast), многоадресатный (multicast) или альтернативный (anycast). Зачастую анализировать приходится внутриканальный одноадресатный трафик, т.е. обмен данными между устройствами в одной сети. Формат внутриканального одноадресатного IPv6-адреса в таком трафике приведен на рис. 7.16.

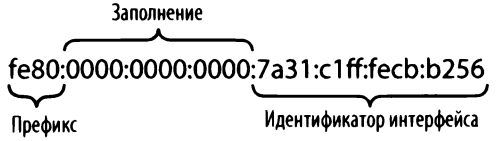


Рис. 7.16. Отдельные части внутриканального одноадресатного IPv6-адреса

Внутриканальные адреса применяются в тех случаях, когда данные требуется передать другому устройству в той же самой сети. Внутриканальный адрес можно распознать по установке двоичного значения **1111111010** в 10 старших его битах и нулей в последующих 54 битах. Таким образом, внутриканальный адрес можно выявить по первой его половине в форме **fe80:0000:0000:0000**.

Вторую половину внутриканального IPv6-адреса составляет идентификатор интерфейса, однозначно определяющий сетевой интерфейс в оконечном хосте. В сетях Ethernet этот идентификатор может основываться на MAC-адресе сетевого интерфейса. Но длина MAC-адреса не превышает 48 бит. Чтобы заполнить все 64 бита, MAC-адрес разделяется надвое, и между образовавшимися половинами вставляется шестнадцатеричное значение **0xfffe** в качестве заполнения, чтобы получить в итоге однозначный идентификатор. И, наконец, обращается седьмой бит в первом байте. Это довольно сложная

операция, поэтому рассмотрим структуру идентификатора интерфейса, приведенную на рис. 7.17, где исходный MAC-адрес устройства, обозначаемого этим идентификатором, был равен `78:31:c1:cb:b2:56`. Затем байты шестнадцатеричного значения `0xfffe` были вставлены посредине и далее обращен седьмой бит в первом байте, сменив шестнадцатеричную цифру `8` на `a`.

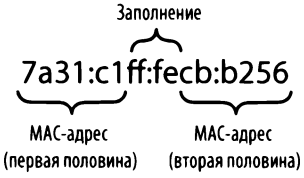


Рис. 7.17. Идентификатор сетевого интерфейса, составляемый из MAC-адреса этого интерфейса и соответствующего заполнения

IPv6-адреса могут быть представлены в упоминавшейся ранее форме CIDR аналогично IPv4-адресам. В следующем примере 64-разрядное пространство адресов представлено внутриканальным адресом:

```
fe80:0000:0000:0000:/64
```

Композиция IPv6-адреса изменятся, когда он используется в глобальном одноадресатном трафике, направляемом через открытый Интернет (рис. 7.18). Если IPv6-адрес применяется в таком трафике, то глобальная одноадресатная передача выявляется по наличию комбинации `001`, установленной в первых трех битах, после чего следует 45-разряный префикс глобальной маршрутизации. Этот префикс присваивается организациям Центром по присвоению номеров Интернета (Internet Assigned Numbers Authority – IANA) и служит для однозначного определения IP-пространства организации. Следующие 16 бит выделяются под идентификатор подсети, который служит для иерархической адресации аналогично той части IPv4-адреса, которая относится к маске сети. И последние 64 бита выделяются для идентификатора сетевого интерфейса, как и во внутриканальных одноадресатных адресах. Размеры префикса маршрутизации и идентификатора подсети могут изменяться.

Рис. 7.18. Отдельные части глобального индивидуального IPv6-адреса

Сетевой протокол IPv6 обеспечивает намного большую эффективность, чем протокол IPv4 с точки зрения маршрутизации пакетов к их получателю и употребления пространства адресов. Такая эффективность обусловлена более обширным диапазоном доступных адресов и применением внутриканальной и глобальной адресации наряду с однозначными идентификаторами хостов.

ПРИМЕЧАНИЕ *Визуально отличить IPv6-адреса от IPv4-адресов человеку совсем не трудно, тогда как многие программы на это неспособны. Чтобы указать IPv6-адрес, в некоторых приложениях, в том числе браузерах или утилитах командной строки, его приходится заключать в квадратные скобки следующим образом:* `[1111::2222:333:44:5:ffff]`. *Данное требование не всегда документировано надлежащим образом, поэтому нередко оно становилось источником недоразумений для многих, когда они изучали протокол IPv6.*

Структура заголовка в пакете IPv6

Файл перехвата `http_ip4and6.pcapng`

Структура заголовка в пакете IPv6 была расширена для поддержки дополнительных возможностей, хотя она и разработана с целью упростить ее синтаксический анализ. Теперь заголовки имеют фиксированную длину 40 байт, а не переменный размер, требовавший ранее проверять поле длины заголовка для его синтаксического анализа. Дополнительные параметры предоставляются благодаря расширению заголовков. Преимущество такого подхода заключается в том, что в большинстве маршрутизаторов достаточно обработать 40-байтовый заголовок, чтобы переслать пакет дальше.

Как показано на рис. 7.19, IPv6-заголовок состоит из следующих полей.

- **Версия (Version).** Это версия применяемого протокола IP (для протокола IPv6 она всегда будет иметь номер 6).
- **Класс трафика (Traffic Class).** Служит для назначения приоритетов определенным классам сетевого трафика.
- **Метка потока (Flow Label).** Употребляется отправителем для отметки ряда пакетов, принадлежащих одному и тому же потоку. Как правило, это поле служит для управления качеством обслуживания (QoS) и гарантии, что пакеты из одного и того же потока будут направлены по тому же самому пути.
- **Длина полезной информации (Payload Length).** Длина полезных данных, следующих после IPv6-заголовка в пакете.

<table>
<tr><td colspan="10">Межсетевой протокол версии 6 (IPv6)</td></tr>
<tr><td>Смещение</td><td>Октет</td><td colspan="2">0</td><td colspan="2">1</td><td colspan="2">2</td><td colspan="2">3</td></tr>
<tr><td>Октет</td><td>Бит</td><td>0–3</td><td>4–7</td><td>8–11</td><td>12–15</td><td colspan="2">16–23</td><td colspan="2">24–31</td></tr>
<tr><td>0</td><td>0</td><td>Версия</td><td colspan="2">Класс трафика</td><td colspan="4">Метка потока</td></tr>
<tr><td>4</td><td>32</td><td colspan="4">Длина полезной информации</td><td colspan="2">Следующий заголовок</td><td colspan="2">Предельное количество переходов</td></tr>
<tr><td>8</td><td>64</td><td colspan="8" rowspan="4">IP-адрес отправителя</td></tr>
<tr><td>12</td><td>96</td></tr>
<tr><td>16</td><td>128</td></tr>
<tr><td>20</td><td>160</td></tr>
<tr><td>24</td><td>192</td><td colspan="8" rowspan="4">IP-адрес получателя</td></tr>
<tr><td>28</td><td>224</td></tr>
<tr><td>32</td><td>256</td></tr>
<tr><td>36</td><td>288</td></tr>
</table>

Рис. 7.19. Структура заголовка в пакете IPv6

- **Следующий заголовок (Next Header).** Обозначает начало заголовка пакета четвертого уровня, который вложен после IPv6-заголовка. Это поле заменяет поле Protocol в IPv4-заголовке.
- **Предельное количество переходов (Hop Limit).** Определяет время жизни пакета, измеряемое в количестве переходов через маршрутизаторы. Это поле заменяет поле TTL в IPv4-заголовке.
- **IP-адрес отправителя (Source IP Address).** IP-адрес хоста, пославший пакет.
- **IP-адрес получателя (Destination IP Address).** IP-адрес хоста, для которого предназначен пакет.

А теперь сравним пакеты IPv4 и IPv6, чтобы выяснить некоторые отличия на примере их анализа из файла перехвата `http_ip4and6.pcapng`. В перехваченном трафике из этого файла веб-сервер был настроен на прослушивание соединений по обоим протоколам, IPv4 и IPv6, на одном и том же физическом хосте. А единственный клиент был настроен как на IPv4-адреса, так и на IPv6-адреса для просмотра сервера независимо по каждому из своих адресов и загрузки страницы из файла `index.php` по протоколу HTTP через приложение библиотеки `curl` (рис. 7.20).

Открыв файл перехвата, вы должны сразу же увидеть, какие именно пакеты относятся к конкретному диалогу, исходя из адресов, указанных в столбцах **Source** и **Destination** подокна Packet List. В частности, пакеты 1–10 представляют IPv4-поток (т.е. поток 0), а пакеты 11–20 — IPv6-поток (т.е. поток 1). Каждый из этих потоков можно отсеять, перейдя в окно Conversations или же введя критерий **`tcp.stream == 0`** или **`tcp.stream == 1`** в строке фильтра.

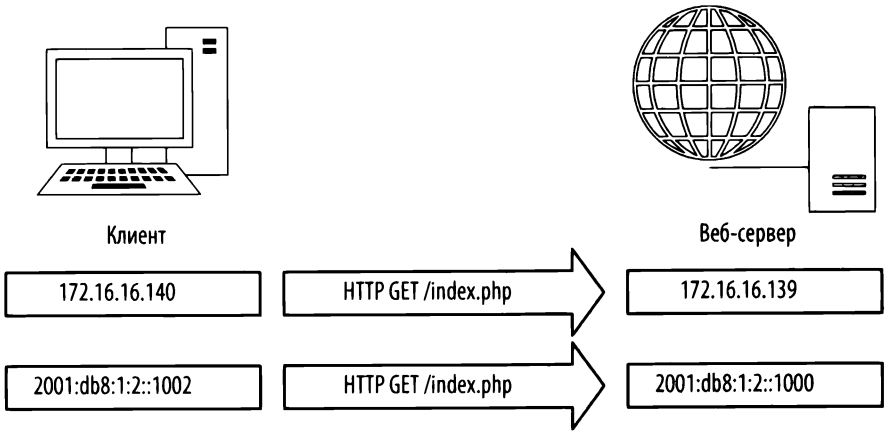


Рис. 7.20. Соединения одних и тех же физических хостов с помощью разных версий протокола IP

Более подробно сетевой протокол HTTP, отвечающий за обслуживание веб-страниц в Интернете, будет рассмотрен в главе 8, "Протоколы транспортного уровня". А в данном примере следует лишь заметить, что обслуживание веб-страниц остается согласованным независимо от применяемого сетевого протокола нижнего уровня. То же самое можно сказать и о сетевом протоколе TCP, который также действует согласованно. И это яркий пример действия инкапсуляции. Несмотря на то что протоколы IPv4 и IPv6 функционируют по-разному, это не оказывает никакого влияния на протоколы, действующие на других уровнях.

На рис. 7.21 непосредственно сравниваются два пакета, 1 и 11, выполняющие одинаковую функцию. Оба пакета относятся к типу TCP SYN и предназначены для инициирования клиентом соединения с сервером. Части этих пакетов, относящиеся к протоколам Ethernet и TCP, почти одинаковы, тогда как части, относящиеся к протоколу IP, совсем разные.

Ниже перечислены отличия, обнаруженные в результате непосредственного сравнения обоих пакетов.

- Форматы адресов отправителя ❻ и получателя ❿ отличаются.
- Длина IPv4-пакета составляет 74 байта, а длина полезных данных — 60 байт ❶, включая IPv4-заголовок, полезную информацию и 14-байтовый Ethernet-заголовок. Длина пакета IPv6 составляет 96 байт, причем 40 байт приходится на полезную информацию ❼, еще 40 байт — на IPv6-заголовок и остальные 14 байт — на Ethernet-заголовок. Длина IPv6-заголовка равна 40 байт, т.е. в два раза больше, чем длина IPv4-заголовка в 20 байт, чтобы учесть размер более крупного пространства адресов.

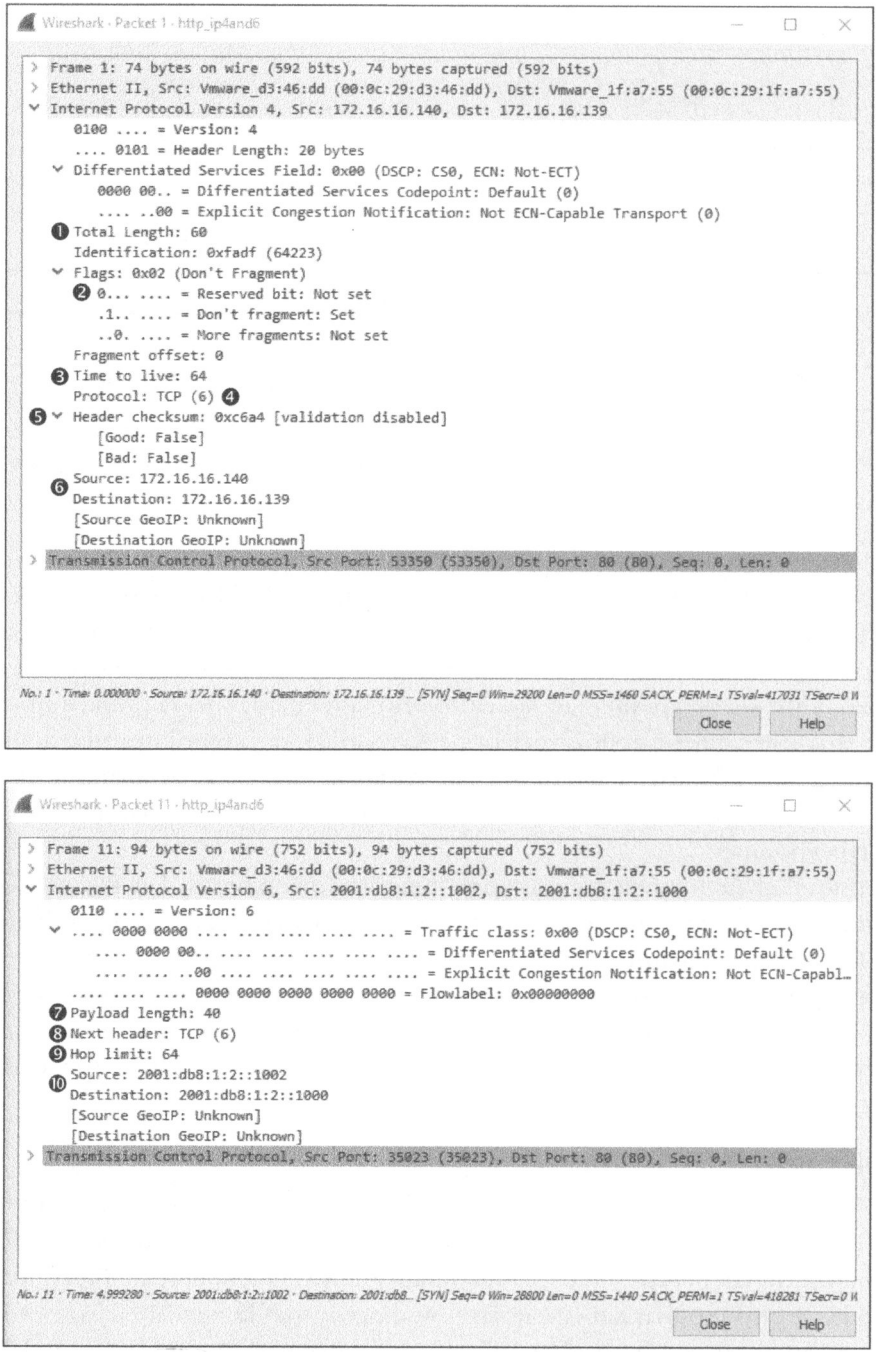


Рис. 7.21. Непосредственное сравнение пакетов IPv4 (вверху) и IPv6 (внизу), выполняющих одну и ту же функцию

- В протоколе IPv4 тип вложенного пакета обозначается в поле Protocol заголовка ❹, тогда как в протоколе IPv6 — в поле Next header заголовка ❽, которое может быть также использовано для указания заголовков расширений.
- В IPv4-заголовке имеется поле TTL ❸, тогда как в IPv6-заголовке ту же функцию выполняет поле Hop limit ❾.
- В IPv4-заголовок включается значение контрольной суммы ❺, тогда как в IPv6-заголовке оно отсутствует.
- Пакет IPv4 не фрагментируется, но все же включает в себя значения параметров фрагментации ❷. А IPv6-заголовок не содержит подобные сведения, поскольку фрагментация, если бы она потребовалась, была бы реализована в заголовке расширения.

Непосредственное сравнение сетевого трафика по протоколам IPv4 и IPv6 дает отличную возможность полностью оценить отличия в функционировании сравниваемых протоколов.

Опрос соседа и протокол ARP

Файл перехвата **icmpv6_neighbor_solicitation.pcapng**

Когда мы ранее обсуждали разные категории сетевого трафика, то перечислили категории одноадресатного, многоадресатного и альтернативного трафика, но не упомянули широковещательный трафик. В сетевом протоколе IPv6 широковещательный трафик не поддерживается, поскольку он считается неэффективным механизмом для передачи данных. Вследствие этого протокол ARP нельзя применять для обнаружения хостами друг друга в сети. Как же тогда IPv6-устройства смогут обнаружить друг друга?

Ответ на этот вопрос дает новая возможность, называемая *опросом соседа (neighbor solicitation)* и являющаяся функцией протокола NDP (Neighbor Discovery Protocol — протокол обнаружения соседей), для выполнения основных функций которого применяется протокол ICMPv6, рассматриваемый в последнем разделе этой главы. С этой целью в протоколе ICMPv6 применяется многоадресатный тип передачи данных, которые смогут получить и обработать только те хосты, которые подписаны на поток этих данных. Многоадресатный трафик можно быстро распознать, поскольку у него имеется свое зарезервированное пространство IP-адресов (**ff00::/8**).

Несмотря на то что процесс преобразования адресов опирается на другой протокол, в нем применяется очень простая последовательность запросов и ответов. Рассмотрим в качестве примера сценарий, в котором хосту с IPv6-адресом **2001:db8:1:2::1003** требуется связаться с другим хостом, который

обозначается адресом **2001:db8:1:2::1000**. Как и в протоколе IPv4, передающее устройство должно быть в состоянии определить канальный (MAC) адрес хоста, с которым ему требуется связаться, поскольку связь устанавливается внутри сети. Этот процесс наглядно показан на рис. 7.22.

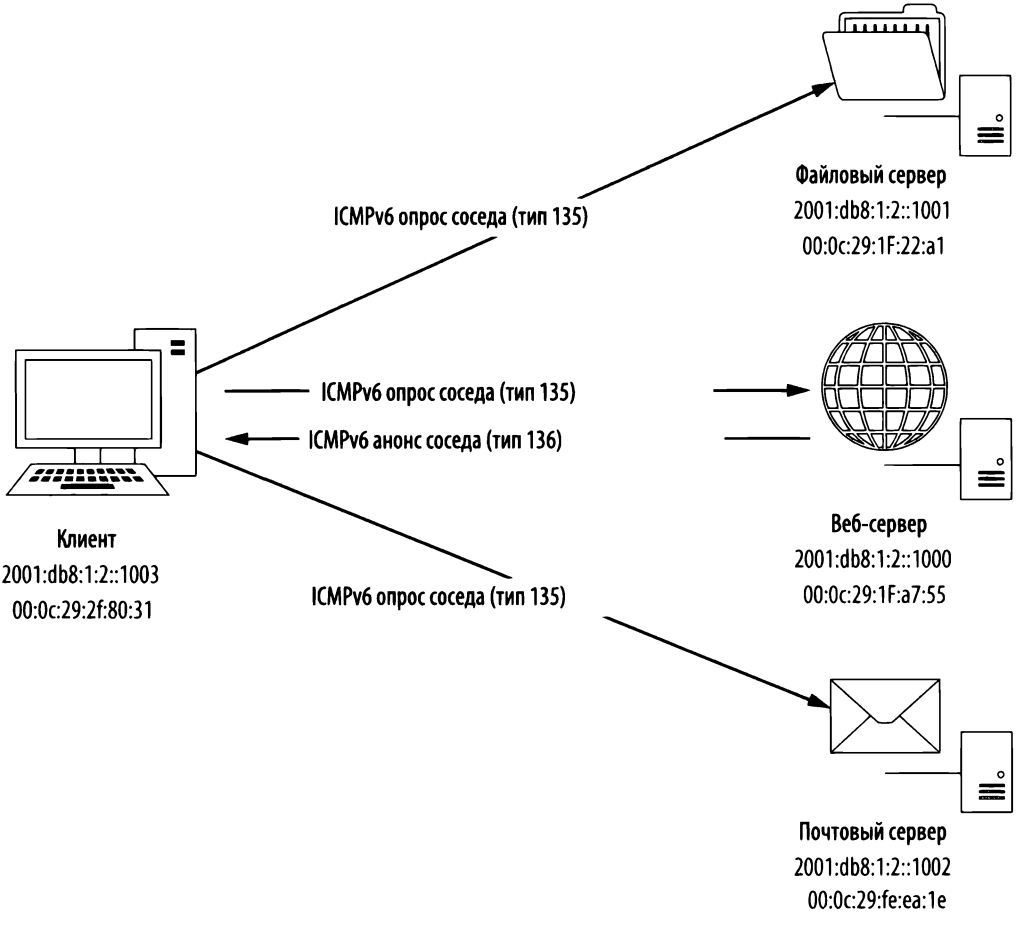


Рис. 7.22. Процесс опроса соседа для преобразования адресов

В данном процессе хост по адресу **2001:db8:1:2::1003** посылает пакет с сообщением Neighbor Solicitation (ICMPv6 type 135), которое означает опрос соседа и относится к типу 135 в протоколе ICMPv6, каждому устройству в сети в режиме многоадресатной передачи, запрашивая следующее: "Какой MAC-адрес устройства, имеющего IP-адрес **2001:db8:1:2::1000**? Мой MAC-адрес — **00:0C:29:2f:80:31**".

Устройство, которому присвоен этот IPv6-адрес, получит данный запрос, отправленный в режиме многоадресатной передачи, и в ответ отправит

запрашивающему хосту пакет с сообщением Neighbor Advertisement (ICMPv6 type 136), которое означает анонс соседа и относится к типу 136 в протоколе ICMPv6. Этот пакет, по существу, содержит такой ответ: "Привет! Мой сетевой адрес – **2001:db8:1:2::1000**, а мой MAC-адрес – **00:0c:29:1f:a7:55**". Как только данный пакет будет получен, можно начинать передачу данных.

Действие данного процесса можно наблюдать на примере файла перехвата `icmpv6_neighbor_solicitation.pcapng`, где хосту, находящемуся по адресу **2001:db8:1:2::1003**, требуется связаться с хостом, расположенным по адресу **2001:db8:1:2::1000**. Рассмотрим первый пакет и развернем его часть, относящуюся к протоколу ICMPv6, в панели Packet Details (рис. 7.23), чтобы обнаружить, что данный пакет относится к протоколу ICMPv6 типа 135 ❷ и отправлен из хоста с адресом **2001:db8:1:2::1003** хосту с многоадресатным адресом **ff02::1:ff00:1000** ❶. Исходный хост предоставил IPv6-адрес хоста-получателя ❸, с которым ему требуется установить связь, наряду с собственным MAC-адресом второго уровня ❹.

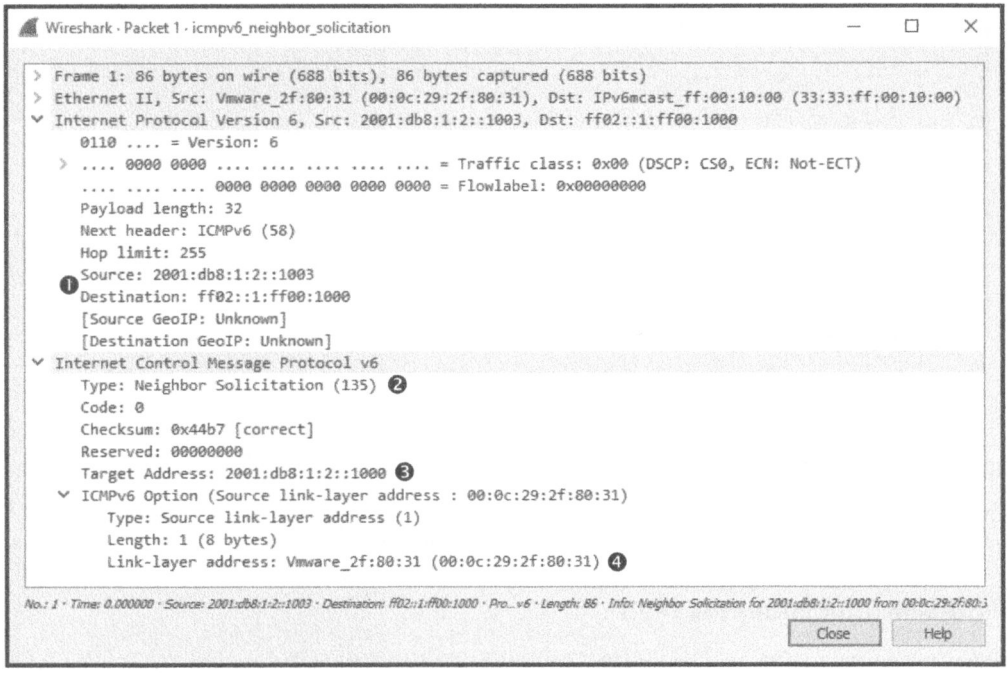


Рис. 7.23. Пакет опроса соседа

Ответ на запрос соседа находится во втором пакете из упомянутого выше файла перехвата. Развернув часть этого пакета, относящуюся к протоколу ICMPv6, в панели Packet Details (рис. 7.24), можно обнаружить, что он относится к протоколу ICMPv6 типа 136 ❷ и отправлен из хоста с

адресом **2001:db8:1:2::1000** обратно хосту с адресом **2001:db8:1:2::1003** ❶. Он также содержит MAC-адрес **00:0c:29:1f:a7:55**, связанный с адресом **2001:db8:1:2::1000**.

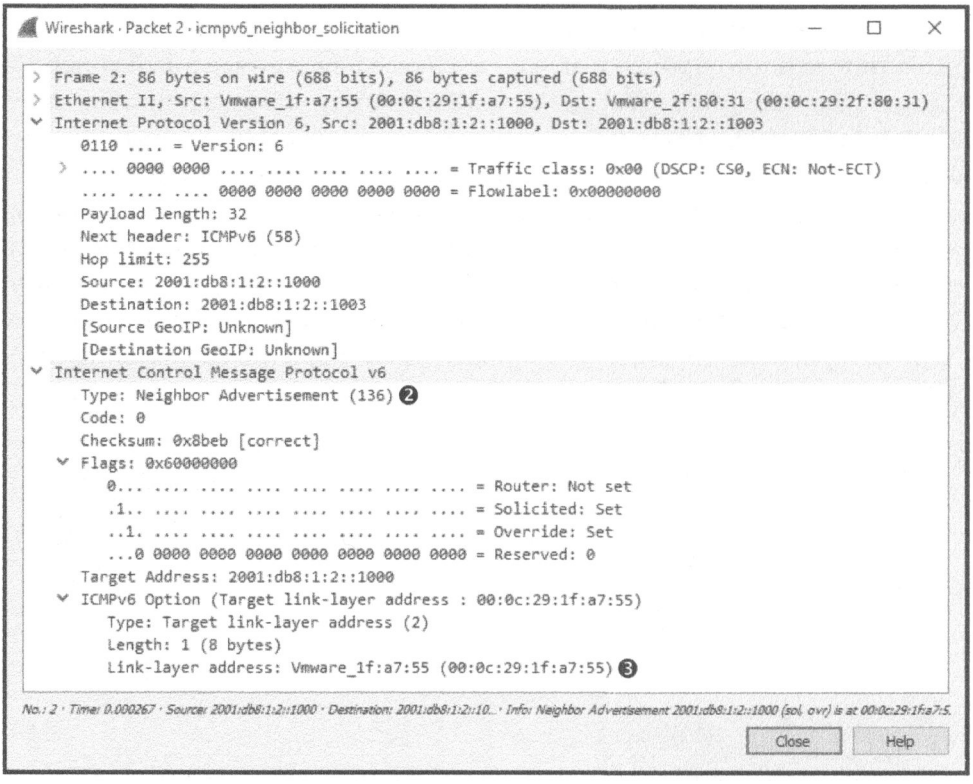


Рис. 7.24. Пакет анонса соседа

По завершении данного процесса хосты по адресам **2001:db8:1:2::1003** и **2001:db8:1:2::1000** начнут обмен пакетами в обычном режиме с эхо-запросами и ответами по протоколу ICMPv6. Это, по существу, означает, что процесс опроса соседа и определения канальных адресов успешно завершен.

Фрагментация пакетов IPv6

Файл перехвата **ipv6_fragments.pcapng**

Поддержка фрагментации пакетов была встроена в заголовок протокола IPv4 для гарантии, чтобы пакеты могли последовательно проходить через самые разные сети, в которых размеры блоков MTU заметно меняются. А в протоколе IPv6 фрагментация пакетов используется в меньшей степени, поэтому параметры ее поддержки не включаются в заголовок IPv6. При этом предполагается, что устройство, передающее пакеты IPv6, должно сначала запустить так называе-

мый процесс *поиска размера блока MTU*, чтобы определить максимальный размер отправляемых пакетов, прежде чем фактически отправить их. Если же маршрутизатор получит пакеты, которые оказываются слишком длинными и превышают размер блока MTU той сети, куда они направляются, то он отбросит их и возвратит передающему устройству пакет с сообщением ICMPv6 Packet Too Big (type 2), т.е. сообщение протокола ICMPv6 типа 2 о том, что пакет слишком велик. Получив это сообщение, передающее устройство попытается снова послать пакет, но уже с меньшим блоком MTU, если только подобное действие поддерживается в протоколе верхнего уровня. Этот процесс будет повторяться до тех пор, пока не будет достигнут достаточно малый размер блока MTU или предел фрагментации полезной информации (рис. 7.25). Сам маршрутизатор вообще не отвечает за фрагментацию пакетов, поскольку ответственность за определение подходящего размера блока MTU в тракте передачи и соответствующая фрагментация пакетов возлагается на передающее устройство.

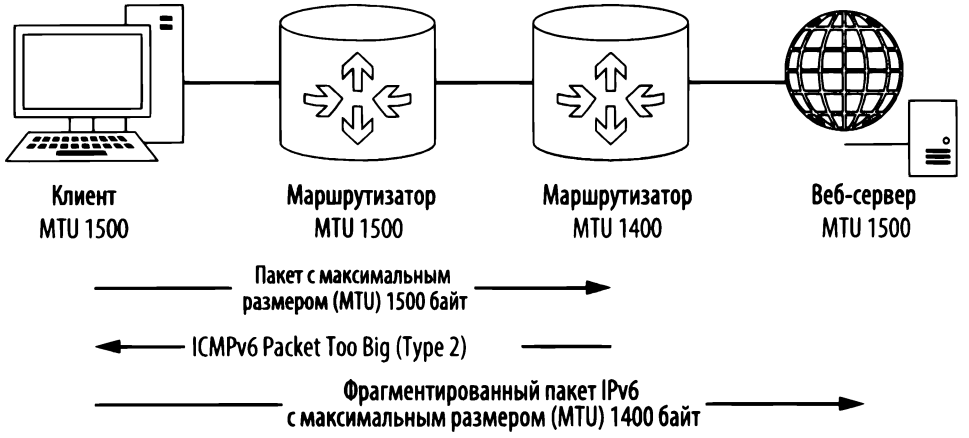


Рис. 7.25. Процесс поиска размера блока MTU в тракте передачи

Если протокол верхнего уровня, применяемый вместе с протоколом IPv6, не в состоянии ограничить полезную информацию в пакете, то фрагментация все еще необходима. В таком случае заголовок расширения для фрагментации может быть введен в заголовок IPv6. Соответствующий пример фрагментации пакетов IPv6 можно найти в файле перехвата `ipv6_fragments.pcapng`.

У принимающего устройства размер блока MTU оказывается меньше, чем у передающего, и поэтому в данном файле перехвата имеются два фрагментированных пакета, представляющих каждый эхо-запрос и ответ по протоколу ICMPv6. Заголовок расширения фрагментации из первого пакета показан на рис. 7.26.

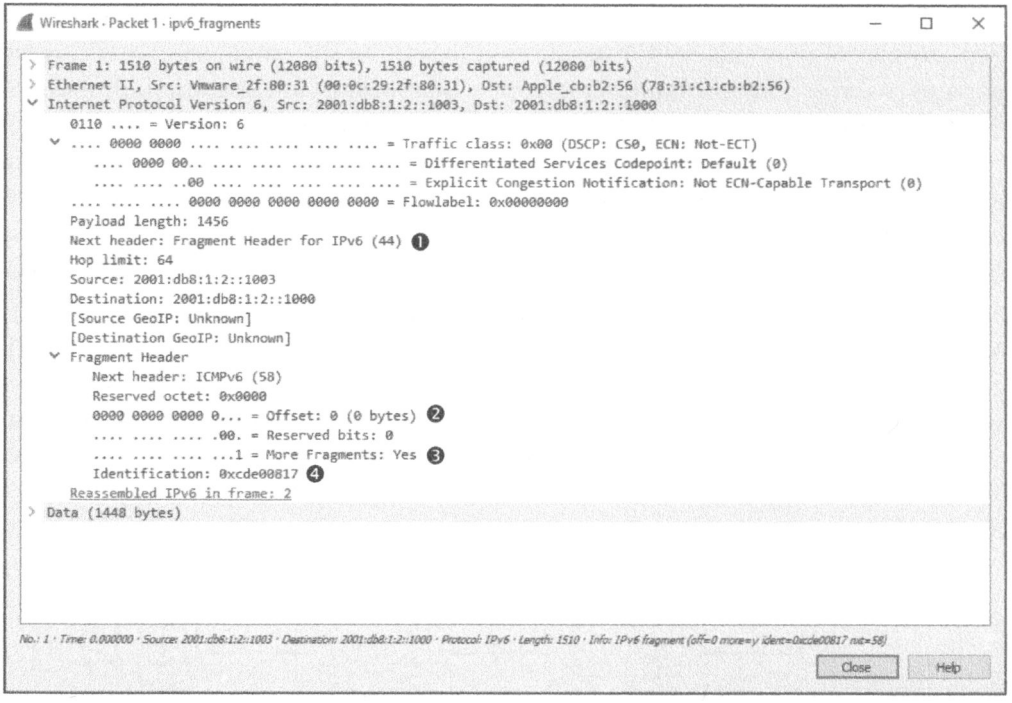


Рис. 7.26. Заголовок расширения для фрагментации пакетов IPv6

Заголовок расширения размером 8 байт содержит те же самые свойства фрагментации, которые обнаруживаются в пакете IPv4, в том числе значение в поле Fragment offset ❷, флаг в поле More Fragments ❸, а также значение в поле Identification ❹. Вместо того чтобы присутствовать в каждом пакете, он добавляется только в конец заголовка пакетов, требующих фрагментации. Этот более эффективный процесс по-прежнему допускает надлежащую повторную сборку фрагментов в принимающей системе. А если имеется данный заголовок расширения, то в поле Next header будет указан именно он, а не протокол инкапсулируемого пакета ❶.

Переходные протоколы к IPv6

IPv6-адреса доставляют немало хлопот, а их внедрение происходило медленно, поскольку для этого требовалось создавать переходную сетевую инфраструктуру. Для облегчения перехода к протоколу IPv6 служит ряд протоколов, допускающих туннелирование обмена данными по протоколу IPv6 через сети, поддерживающие только обмен данными по протоколу IPv4. И в этом смысле туннелирование означает инкапсуляцию обмена данными по протоколу IPv6 в пакетах протокола IPv4 подобно другим протоколам. Такая инкапсуляция обычно реализуется одним из следующих способов.

- **Между маршрутизаторами.** В этом случае туннель применяется для инкапсуляции трафика IPv6 между передающим и принимающим хостами в их сетях через сеть IPv4. Такой способ позволяет целым сетям обмениваться данными по протоколу IPv6 через промежуточные каналы, действующие по протоколу IPv4.
- **Между хостом и маршрутизатором.** В этом случае инкапсуляция применяется на уровне маршрутизатора для передачи сетевого трафика из хоста IPv6 через сеть IPv4. Такой способ позволяет отдельным хостам связываться с сетью IPv6, находясь в сети IPv4.
- **Между хостами.** В этом случае организуется туннель между двумя конечными точками для инкапсуляции трафика IPv6 между хостами, способными обмениваться данными по протоколу IPv4 или IPv6. Такой способ позволяет конечным точкам сети IPv6 обмениваться данными непосредственно через сеть IPv4.

В этой книге недостаточно места, чтобы подробно описать переходные протоколы. Тем не менее об их существовании полезно знать на тот случай, если потребуется их исследовать, выполняя анализ на уровне пакетов. Ниже вкратце описывается ряд наиболее употребительных переходных протоколов.

- **6to4.** Этот переходной протокол иначе называется *IPv6 over IPv4* и позволяет передавать пакеты IPv6 через сеть IPv4. Он поддерживает ретрансляторы и маршрутизаторы для обеспечения обмена данными по протоколу IPv6 между маршрутизаторами, между хостами, а также между хостами и маршрутизаторами.
- **Teredo.** Применяется для одноадресатной передачи по протоколу IPv6 через сеть IPv4 с использованием протокола NAT (Network Address Translation — преобразование сетевых адресов). В этом случае пакеты IPv6 посылаются через сеть IPv4 инкапсулированными в пакеты транспортного протокола UDP.
- **ISATAP.** Протокол внутрисайтовой адресации, допускающий обмен данными исключительно между IPv4- и IPv6-устройствами в сети по образу и подобию, как между отдельными хостами сети.

Протокол межсетевых управляющих сообщений (ICMP)

Протокол ICMP (Internet Control Message Protocol) является служебным протоколом, отвечающим в стеке протоколов TCP/IP за предоставление информации, касающейся доступности устройств, служб или маршрутизаторов в сети TCP/IP. Большинство инструментальных средств и методик диагностики

сетей сосредоточены на общих типах сообщений протокола ICMP. Сетевой протокол ICMP определен в стандарте RFC 792.

Структура заголовка в пакете ICMP

Протокол ICMP является частью протокола IP и использует его для передачи своих сообщений. Протокол ICMP содержит относительно небольшой заголовок, изменяющийся в зависимости от его назначения. Как показано на рис. 7.27, ICMP-заголовок состоит из следующих полей.

Протокол межсетевых управляющих сообщений (ICMP)					
Смещение	Октет	0	1	2	3
Октет	Бит	0–7	8–15	16–23	24–31
0	0	Тип	Код	Контрольная сумма	
4+	32+	Переменная			

Рис. 7.27. Структура заголовка в пакете ICMP

- **Тип (Type).** Тип или категория сообщений в протоколе ICMP на основании спецификации RFC.
- **Код (Code).** Подкатегория сообщений в протоколе ICMP на основании спецификации RFC.
- **Контрольная сумма (Checksum).** Служит для гарантии неизменности ICMP-заголовка и данных, когда они поступят к получателю.
- **Переменная (Variable).** Часть заголовка, изменяющаяся в зависимости от содержимого полей Type и Code.

Типы и коды сообщений протокола ICMP

Как отмечалось ранее, структура пакета ICMP зависит от его назначения, что определяется значениями в полях Type и Code. Содержимое поля Type из заголовка пакета ICMP можно рассматривать как определенную категорию пакета, а содержимое поля Code – как его подкатегорию. Например, значение **3** в поле Type обозначает, что получатель недоступен. Одних только этих сведений явно недостаточно для диагностики неполадок в сети. Но если бы значение **3** было указано и в поле Code, обозначая недоступность порта, то на этом основании можно было бы сделать вывод, что на компьютере получателя недоступен порт, через который предпринимается попытка организовать обмен данными.

ПРИМЕЧАНИЕ *Полный перечень типов и кодов, доступных в протоколе ICMP, можно найти по адресу* **http://www.iana.org/assignments/icmp-parameters/**.

Эхо-запросы и ответы

Файл перехвата **icmp_echo.pcapng**

Особым предметом гордости протокола ICMP является утилита ping, применяемая для тестирования текущего состояния соединения между устройствами сети. И хотя утилита ping не входит в спецификацию протокола ICMP, тем не менее, она применяет этот протокол для реализации своих основных функциональных возможностей.

Чтобы воспользоваться утилитой ping, введите **ping *ip-адрес*** в командной строке, заменив ***ip-адрес*** конкретным IP-адресом устройства в своей сети. Если целевое устройство включено, то у исходного устройства появится путь для передачи ему данных. В итоге вы должны увидеть на экране реакцию устройства на команду **ping**, если только эту передачу не заблокирует брандмауэр.

В примере, приведенном на рис. 7.28, демонстрируются четыре удачных ответа с указанием их длины, времени на передачу и подтверждение приема и времени жизни пакетов. По окончанию работы эта утилита предоставляет также сводку о количестве переданных, принятых и потерянных пакетов. Если при передаче данных произойдет сбой, то должно появиться сообщение о его причине.

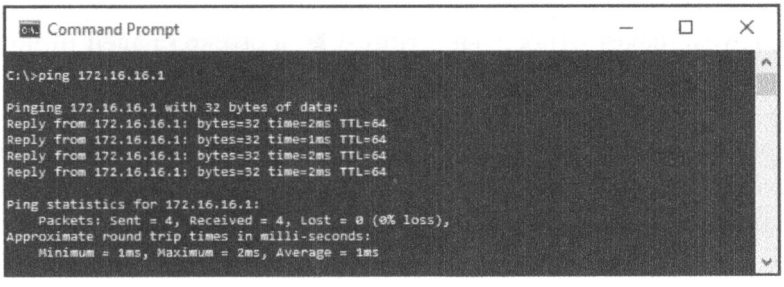


Рис. 7.28. Команда **ping** *предназначена для тестирования связи между узлами сети*

По существу, команда **ping** посылает по очереди пакеты проверяемому устройству и принимает от него ответ, чтобы выяснить, имеется ли связь с этим устройством (рис. 7.29).

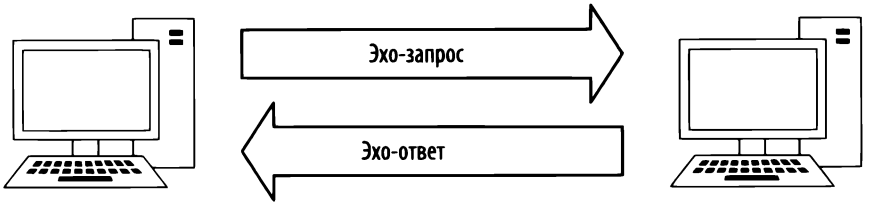


Рис. 7.29. Команда **ping** *выполняется в два этапа*

ПРИМЕЧАНИЕ *Несмотря на то что утилита* `ping` *уже давно применяется в IT-отрасли, на результаты ее работы могут повлиять брандмауэры, запущенные на хостах или маршрутизаторах. Ведь многие современные брандмауэры накладывают ограничения на возможности устройства реагировать на пакеты ICMP. И это совсем не плохо с точки зрения безопасности, поскольку злоумышленники обычно пользуются утилитой* `ping`*, чтобы выяснить доступность хоста для их зловредных действий. Но в то же время наличие брандмауэров иногда затрудняет диагностику сети, поскольку было бы неудовлетворительно запустить утилиту* `ping` *для проверки состояния связи с устройством и не получить от него ответ, заранее зная, что с этим устройством можно поддерживать нормальную связь.*

Действие утилиты `ping` служит отличным примером простого обмена данными по протоколу ICMP. Пакеты из файла перехвата `icmp_echo.pcapng` демонстрируют, что происходит при выполнении утилиты `ping`.

В первом пакете (рис. 7.30) из данного файла демонстрируется, что хост по адресу **192.168.100.138** посылает пакет хосту, находящемуся по адресу **192.168.100.1** ❶. Если развернуть часть этого пакета, относящуюся к протоколу ICMP, то тип пакета ICMP можно определить, глядя на поля Type и Code. В данном случае пакет относится к типу 8 ❷, а его код равен нулю ❸, указывая на эхо-запрос. (Приложение Wireshark должно подсказать, что именно отображается в полях Type и Code.) Но послать эхо-запрос — это лишь полдела. Ведь речь идет о простом пакете ICMP, который посылается с помощью протокола IP и содержит небольшой объем данных. Помимо типа, кода и контрольной суммы, имеется порядковый номер для объединения запросов и ответов в пары, а также произвольная тестовая строка в переменной части пакета ICMP.

ПРИМЕЧАНИЕ *Чтобы не путать термины* ***эхо*** *и* ***ping****, следует иметь в виду, что* ***ping*** *– это название утилиты, применяемой для отправки пакетов с* ***эхо****-запросами по протоколу ICMP.*

Второй пакет в данной последовательности содержит ответ на эхо-запрос (рис. 7.31). В части данного пакета, относящейся к протоколу ICMP, указан нулевой тип ❶ и нулевой код ❷, что, по существу, обозначает эхо-ответ. А поскольку порядковые номера и идентификаторы во втором и в первом пакетах совпадают ❸, то можно сделать вывод, что данный эхо-ответ соответствует эхо-запросу из предыдущего пакета. Значения в соответствующих полях указаны в формате с обратным (BE) и прямым (LE) порядком следования байтов. Иными словами, в этих полях представлены данные в том порядке, в каком

они должны быть обработаны в конкретной конечной точке. Кроме эхо-ответа, второй пакет содержит такую же 32-байтовую символьную сроку данных, как и в переданном эхо-запросе. Как только второй пакет будет получен по адресу **192.168.100.138**, утилита сообщит об удачном завершении проверки связи.

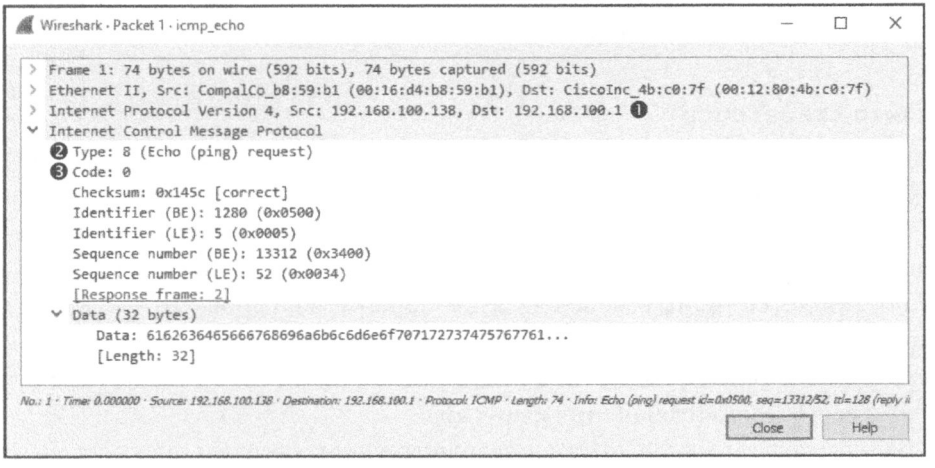


Рис. 7.30. Пакет с эхо-запросом по протоколу ICMP

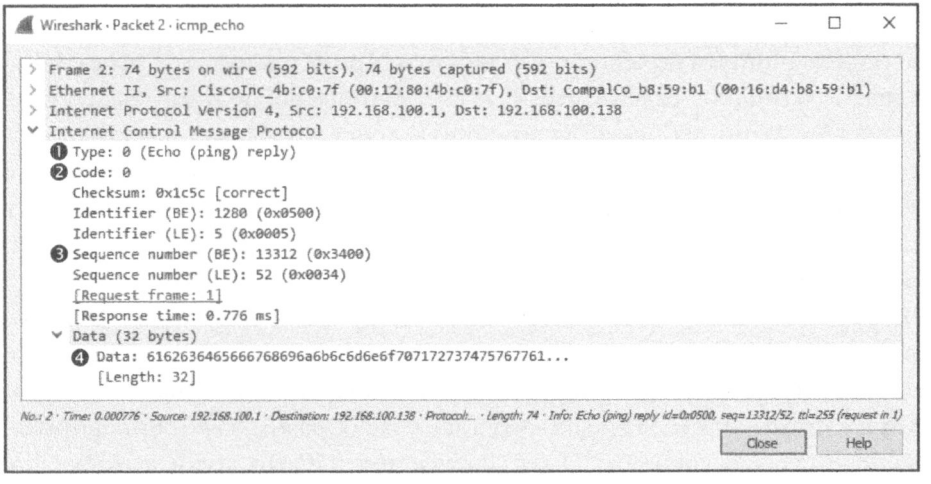


Рис. 7.31. Пакет с эхо-ответом по протоколу ICMP

Чтобы увеличить размер заполняемого данными поля в эхо-запросе и тем самым принудить к фрагментации пакетов, можно воспользоваться соответствующим параметром (`-s` в системах Unix и `-l` в системе Windows) команды **ping**. Такая необходимость может возникнуть при проведении различных видов диагностики сетей, где требуются меньшие размеры фрагментов.

ПРИМЕЧАНИЕ *Произвольный текст, употребляемый в эхо-запросе по протоколу ICMP, может представлять особый интерес для потенциальной атаки со стороны злоумышленников. В частности, злоумышленники могут воспользоваться информацией в заполняемом поле эхо-запроса для определения типа операционной системы, под управлением которой работает проверяемое устройство. Кроме того, злоумышленники могут помещать в это поле небольшие фрагменты данных для скрытой их передачи по сети.*

Утилита traceroute

Файл перехвата **icmp_traceroute.pcapng**

Утилита `traceroute` служит для выявления пути прохождения пакетов от одного устройства к другому. В простой сети этот путь может проходить через единственный маршрутизатор или вообще без него. А в сложной сети пакетам, возможно, придется пройти через десятки маршрутизаторов, чтобы достигнуть конечного получателя. Следовательно, для диагностики связи очень важно иметь возможность проследить конкретный путь, по которому пакеты проходят от одного места назначения к другому.

Используя протокол ICMP (и в незначительной степени протокол IP), утилита `traceroute` может отобразить путь прохождения пакетов. Например, первый пакет из файла перехвата `icmp_traceroute.pcapng` очень похож на эхо-запрос из предыдущего примера (рис. 7.32).

В перехваченном трафике из данного примера пакеты были сформированы по команде **`traceroute 4.2.2.1`**. Чтобы воспользоваться утилитой `traceroute` в Windows, введите команду **`tracert` *`ip-адрес`*** в командной строке, заменив ***`ip-адрес`*** на конкретный IP-адрес того устройства, путь прохождения пакетов к которому требуется выявить и проследить. А для того чтобы воспользоваться утилитой `traceroute` в Linux или Mac OS X, введите команду **`traceroute` *`ip-адрес`***.

На первый взгляд, этот пакет содержит простой эхо-запрос ❸ из хоста, находящегося по адресу **`192.168.100.138`**, к хосту, расположенному по адресу **`4.2.2.1`** ❶, и поэтому все содержимое той части данного пакета, которое относится к протоколу ICMP, выглядит таким же образом, как и при форматировании пакета с эхо-запросом. Но если развернуть IP-заголовок данного пакета, то можно заметить, что в поле TTL этого заголовка установлено значение **`1`** ❷. Это означает, что данный пакет будет аннулирован на первом же маршрутизаторе, которого он достигнет. А поскольку IP-адрес получателя пакета **`4.2.2.1`** является межсетевым, то, как известно, между исходным и целевым устройствами должен находиться хотя бы один маршрутизатор. Следовательно, данный пакет никоим образом не сможет достигнуть места своего назначения.

И это хорошо, поскольку утилита `traceroute` полагается на то, что данный пакет достигнет лишь первого маршрутизатора на своем пути прохождения.

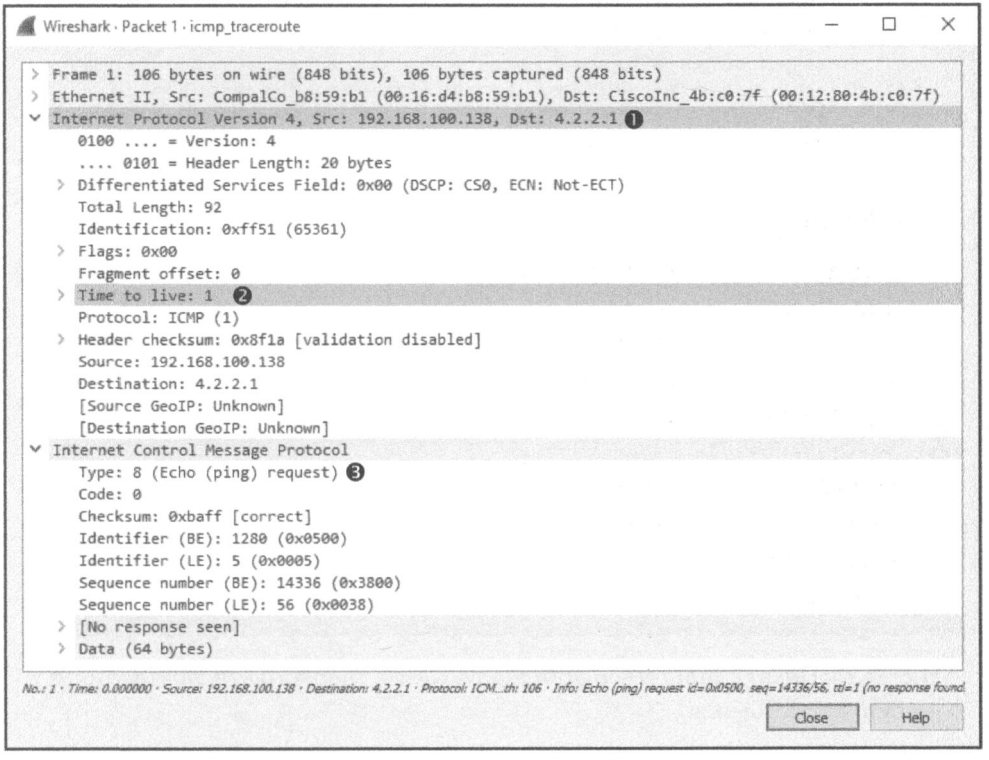


Рис. 7.32. Пакет с эхо-запросом по протоколу ICMP и значением **1**, *установленным в поле TTL*

Второй пакет, как и предполагалось, содержит ответ от первого достигнутого маршрутизатора на пути прохождения к получателю (рис. 7.33). Первый пакет достиг устройства по адресу **`192.168.100.1`**, где значение в поле TTL его IP-заголовка было уменьшено до нуля, соответственно его нельзя передавать дальше. Поэтому маршрутизатор прислал пакет с ответом по протоколу ICMP. Этот пакет относится к типу **11** ❶, и ему присвоен код **0** ❷, а это означает, что получатель недостижим, поскольку время жизни пакета было превышено при переходе через маршрутизатор.

Такой пакет ICMP иногда еще называют *пакетом с двумя заголовками*, поскольку в самом конце той его части, которая относится к протоколу ICMP, содержится копия IP-заголовка ❸ и данные протокола ICMP ❹, которые были отправлены в исходном эхо-запросе. Эти сведения могут принести немалую пользу при диагностике сетей.

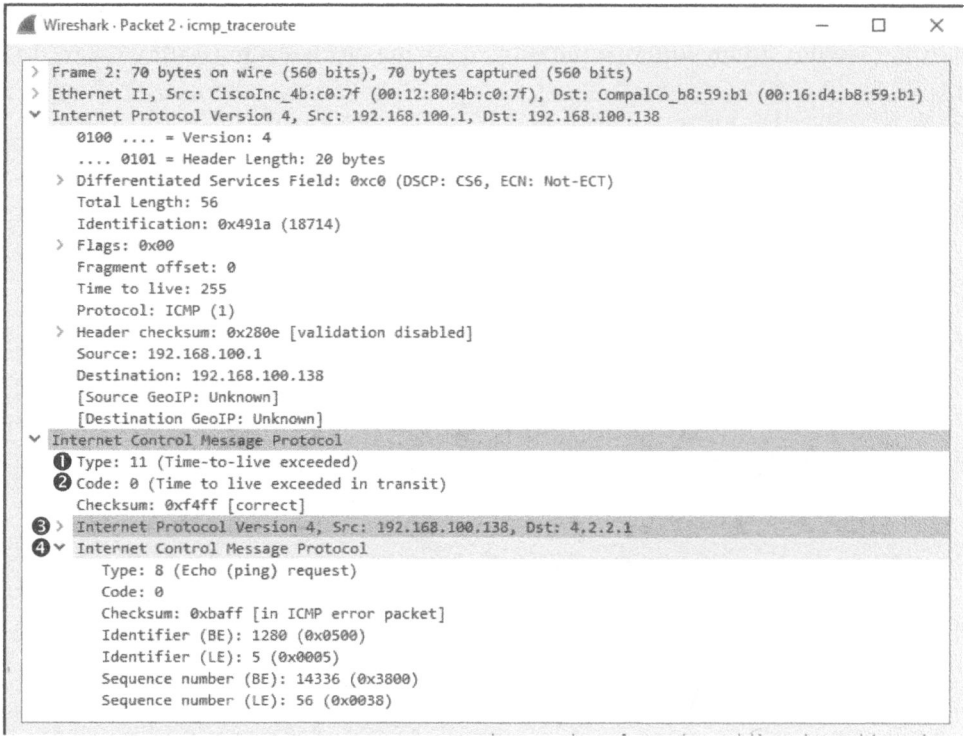


Рис. 7.33. Пакет с ответом по протоколу ICMP от первого маршрутизатора на пути прохождения

Описанный выше процесс отправки пакетов с минимальным значением в поле TTL их IP-заголовка происходит еще несколько раз, прежде чем будет получен пакет 7. И здесь наблюдается то же самое, что и в первом пакете, но на этот раз в поле TTL соответствующего IP-заголовка установлено значение **2**, гарантирующее, что пакет достигнет маршрутизатора на втором переходе, прежде чем он будет аннулирован. Как и следовало ожидать, ответ будет получен от маршрутизатора на следующем переходе по адресу **192.168.1.1** с теми же самыми сообщениями по протоколу ICMP о недостижимости получателя и превышении времени жизни пакета.

Этот процесс с увеличением значения в поле TTL продолжается до тех пор, пока не будет достигнут получатель по адресу **4.2.2.1**. Но прежде чем это произойдет, истечет время ожидания запроса, как следует из строки 8 результатов, выводимых утилитой `traceroute` на экран (рис. 7.34). Как же тогда данный процесс может завершиться успешно, если время ожидания запроса по пути его прохождения истечет? Как правило, это происходит в том случае, если маршрутизатор не отвечает на ICMP-запросы. Такой маршрутизатор по-прежнему получает запрос и пересылает данные следующему

маршрутизатору. Именно поэтому мы можем обнаружить следующий переход в строке 9, как показано на рис. 7.34. На данном переходе просто не был сформирован пакет ICMP с сообщением о превышении времени жизни, как это имело место на других переходах. Если утилита `traceroute` не получает ответ через заданный интервал времени, она увеличивает на 1 значение в поле TTL и просто переходит к следующему запросу.

Таким образом, в ходе описанного выше процесса утилита `traceroute` связывалась с каждым из маршрутизаторов, находящихся на пути прохождения пакетов, строя попутно карту маршрута к месту их назначения. Пример такой карты приведен на рис. 7.34.

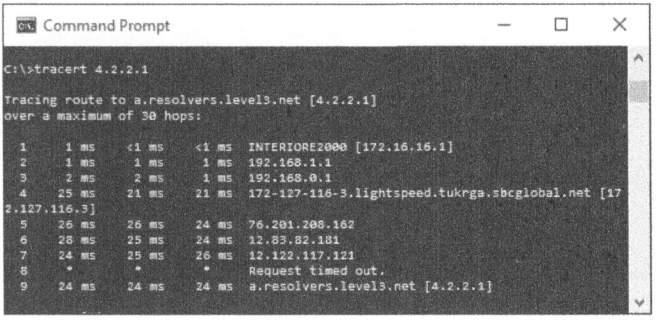


Рис. 7.34. Пример результатов, выводимых на экран утилитой `traceroute`

ПРИМЕР *Приведенное здесь описание утилиты* `traceroute` *относится в основном к Windows, поскольку в этой утилите применяется исключительно протокол ICMP. А в Linux утилите* `traceroute` *присуща несколько большая универсальность. В частности, для прослеживания пути, по которому проходят пакеты, в этой утилите могут применяться и другие сетевые протоколы, например UDP.*

Протокол ICMP версии 6 (ICMPv6)

Обновленная версия протокола IP опирается в значительной степени на протокол ICMP для выполнения таких функций, как опрос соседа или обнаружение пути прохождения пакетов, как демонстрировалось в рассмотренных ранее примерах. Протокол ICMPv6 описан в документе RFC 4443 и предназначен для поддержки ряда возможностей, требующихся для нормального функционирования протокола IPv6, наряду с дополнительными усовершенствованиями. В этой книге протокол ICMPv6 не рассматривается отдельно потому, что в нем применяется такая же структура заголовка пакета, как и в пакетах ICMP.

Как правило, пакеты ICMPv6 разделяются на категории сообщений об ошибках или информационных сообщений. Полный перечень типов и кодов, доступных в протоколе ICMPv6, можно найти на веб-странице организации IANA по адресу `http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml`.

В этой главе были представлены наиболее важные протоколы сетевого уровня, которые вам предстоит исследовать в процессе анализа пакетов. В частности, протоколы ARP, IP и ICMP положены в основу всего обмена данными по сети, и поэтому они играют решающую роль в решении задач, с которыми вы столкнетесь в своей повседневной деятельности. В главе 8 будут рассмотрены наиболее употребительные протоколы транспортного уровня – TCP и UDP.

8

ПРОТОКОЛЫ ТРАНСПОРТНОГО УРОВНЯ

В этой главе мы продолжим рассмотрение отдельных протоколов и их проявление на уровне пакетов. Продвигаясь вверх по модели OSI, мы рассмотрим транспортный уровень и наиболее употребительные на этом уровне протоколы TCP и UDP.

Протокол управления передачей (TCP)

Конечная цель *протокола TCP* (Transmission Control Protocol – протокол управления передачей) – обеспечить надежную доставку данных из одной конечной точки сети в другую. Как определено в стандарте RFC 793, протокол TCP отвечает за отслеживание последовательностей данных и устранение ошибок и в конечном счете обеспечивает доставку данных по месту их назначения. Протокол TCP считается *ориентированным на установку соединения*, поскольку он устанавливает формальное соединение, прежде чем передавать данные, отслеживает доставку пакетов и, как правило, пытается формально закрыть каналы связи по окончании передачи. Во многих из наиболее употребительных протоколов уровня приложений используются протоколы TCP и IP для доставки пакетов получателю.

Структура заголовка в пакете TCP

Протокол TCP предоставляет самые разные функциональные возможности, что и отражается в сложности его заголовка. Как показано на рис. 8.1, заголовок пакета этого протокола состоит из перечисленных ниже полей.

<table>
<tr><td colspan="8">Протокол управления передачей (TCP)</td></tr>
<tr><td>Смещение</td><td>Октет</td><td colspan="2">0</td><td colspan="2">1</td><td>2</td><td>3</td></tr>
<tr><td>Октет</td><td>Бит</td><td>0–3</td><td>4–7</td><td colspan="2">8–15</td><td>16–23</td><td>24–31</td></tr>
<tr><td>0</td><td>0</td><td colspan="3">Порт отправителя</td><td colspan="3">Порт получателя</td></tr>
<tr><td>4</td><td>32</td><td colspan="7">Порядковый номер</td></tr>
<tr><td>8</td><td>64</td><td colspan="7">Номер подтверждения</td></tr>
<tr><td>12</td><td>96</td><td>Смещение данных</td><td>Зарезервировано</td><td colspan="2">Флаги</td><td colspan="2">Размер окна</td></tr>
<tr><td>16</td><td>128</td><td colspan="4">Контрольная сумма</td><td colspan="2">Указатель срочности</td></tr>
<tr><td>20+</td><td>160+</td><td colspan="7">Параметры</td></tr>
</table>

Рис. 8.1. Структура заголовка в пакете TCP

- **Порт отправителя (Source Port).** Порт, используемый для передачи пакета на машине отправителя.
- **Порт получателя (Destination Port).** Порт, в который передается пакет на машине получателя.
- **Порядковый номер (Sequence Number).** Номер, предназначенный для идентификации сегмента протокола TCP. Значение, устанавливаемое в этом поле, служит гарантией, что ни одна из частей потока данных не будет потеряна при передаче.
- **Номер подтверждения (Acknowledgment Number).** Порядковый номер, который ожидается получить в следующем пакете от другого устройства, принимающего участие в обмене данными по сети.
- **Флаги (Flags).** В этом поле устанавливаются флаги `URG`, `ACK`, `PSH`, `RST`, `SYN` и `FIN` для идентификации типа передаваемого пакета TCP.
- **Размер окна (Window Size).** Размер буфера приема пакетов TCP, указываемый в байтах.
- **Контрольная сумма (Checksum).** Служит для гарантии неизменности TCP-заголовка и данных, когда они поступят к получателю.
- **Указатель срочности (Urgent Pointer).** Если установлен флаг `URG`, то содержимое этого поля проверяется на наличие дополнительных инструкций относительно того места, с которого ЦП должен начинать чтение данных в пакете.

- **Параметры (Options).** В этом поле находятся дополнительные параметры, которые могут быть указаны в пакете TCP.

Порты TCP

Файл перехвата **tcp_ports.pcapng**

Весь обмен данными по протоколу TCP происходит между *портами* отправителя и получателя, номера которых можно обнаружить в каждом TCP-заголовке. Порт можно сравнить с гнездом на старой телефонной коммутационной панели (коммутаторе). Оператор телефонной станции (или телефонистка) постоянно контролировала индикаторы и штепсельные разъемы на этой панели. И как только загорался индикатор вызова, телефонистка связывалась с вызывающим абонентом, спрашивала, с кем он желает поговорить, а затем подключала его к другому абоненту, вставив кабель со штепсельным разъемом в соответствующее гнездо коммутатора. Для установки сеанса связи требовалось наличие порта отправителя (т.е. вызывающего абонента) и порта получателя (т.е. принимающего вызов абонента). Аналогичным образом действуют и порты протокола TCP.

Чтобы передать данные конкретному приложению, запущенному на удаленном сервере или устройстве, в пакете TCP должен быть указан известный номер порта удаленной службы, где этот пакет может быть принят. Если же попытаться получить доступ к приложению на удаленной машине через другой порт, на который оно не настроено, то обмен данными не состоится.

Порт отправителя в данной последовательности не столь важен и может быть выбран произвольно. Удаленный сервер определит номер порта, с которым устанавливается связь, из отправленного ему исходного пакета (рис. 8.2).

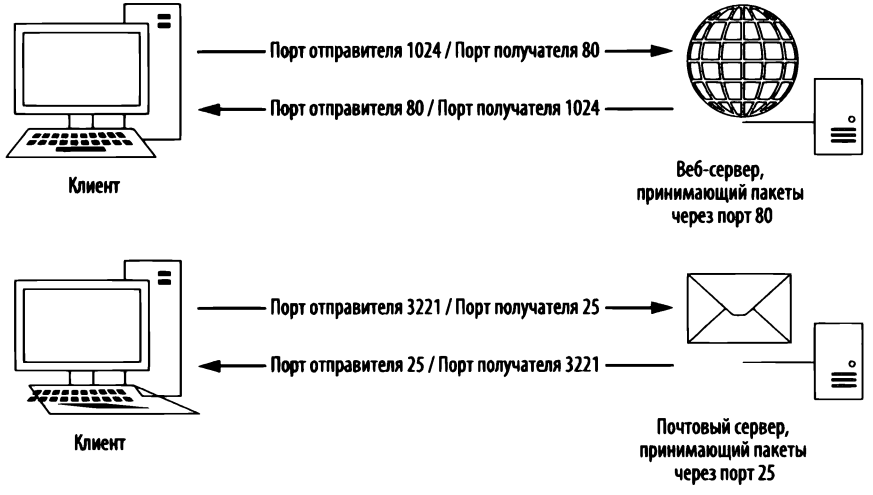


Рис. 8.2. Для передачи данных в протоколе TCP применяются порты

Для обмена данными по протоколу TCP имеется 65535 портов. Все они, как правило, разделяются на следующие категории.

- **Группа системных портов.** Это группа хорошо известных портов, иначе называемых стандартными (или постоянными) с номерами от **1** до **1023**, где номер **0** зарезервирован и поэтому игнорируется. В хорошо известных и вполне упрочившихся службах обычно применяются порты, относящиеся к группе системных портов.
- **Группа временных портов.** Это группа портов с номерами в пределах от **1024** до **65535**, хотя в некоторых операционных системах эти пределы могут определяться иначе. Через порт данной категории одновременно может связываться только одна служба, и поэтому в современных операционных системах порты отправителя выбираются произвольно, чтобы отличать один сеанс связи от другого. Как правило, порты отправителя выбираются из доступного диапазона временных (т.е. динамически назначаемых) номеров портов.

Рассмотрим в качестве примера пару пакетов TCP из файла перехвата `tcp_ports.pcapng`, чтобы выявить используемые в них номера портов. В этом файле содержится сетевой трафик HTTP, перехваченный при просмотре клиентом двух веб-сайтов. Как упоминалось ранее, для обмена данными в протоколе HTTP применяется протокол TCP, что служит характерным примером трафика TCP.

Два первых значения в заголовке первого пакета из данного файла (рис. 8.3) обозначают порт отправителя и получателя этого пакета. В частности, этот пакет посылается из хоста, находящегося по адресу **172.16.16.128**, хосту, расположенному по адресу **212.58.226.142**. Порт отправителя имеет номер **2826** ❶, т.е. это временный порт. (Напомним, что порты отправителя выбираются операционной системой произвольно, хотя их случайно выбранные номера как правило увеличиваются.) А порт получателя имеет номер **80** ❷, т.е. это стандартный системный порт, назначаемый для веб-серверов, работающих по протоколу HTTP.

Обратите внимание на то, что в Wireshark упомянутые выше порты обозначены метками **slc-systemlog (2826)** и **http (80)**. В приложении Wireshark ведется список портов и мест их наиболее частого употребления. И хотя системные порты обозначаются в основном как наиболее употребительные, многие временные порты связаны с часто употребляемыми службами. Такое обозначение портов специальными метками может ввести в заблуждение, и поэтому его, как правило, лучше отменить, отключив преобразование имен на транспортном уровне. С этой целью выберите команду Edit⇨Preferences⇨Name Resolution из главного меню и сбросьте флажок Resolve Transport Names (Преобразовать

транспортные имена). Если вы хотите оставить этот флажок установленным, но желаете изменить порядок обозначения определенного порта в Wireshark, для этого вам придется внести соответствующие изменения в файл `services`, находящийся в системном каталоге Wireshark. Содержимое этого файла основывается на списке общеупотребительных портов (пример редактирования файла, предназначенного для преобразования имен, приведен в разделе "Применение специального файла hosts" главы 5, "Дополнительные возможности Wireshark").

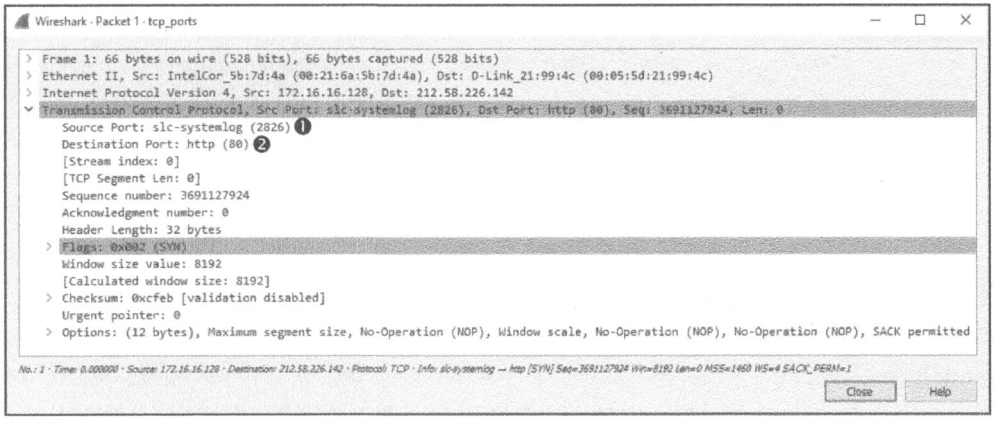


Рис. 8.3. В заголовке протокола TCP указываются порты отправителя и получателя

Второй пакет посылается в обратном направлении из хоста, находящегося по адресу **212.58.226.142**, хосту, расположенному по адресу **172.16.16.128** (рис. 8.4). Как и IP-адреса, порты отправителя и получателя теперь меняются местами ❶.

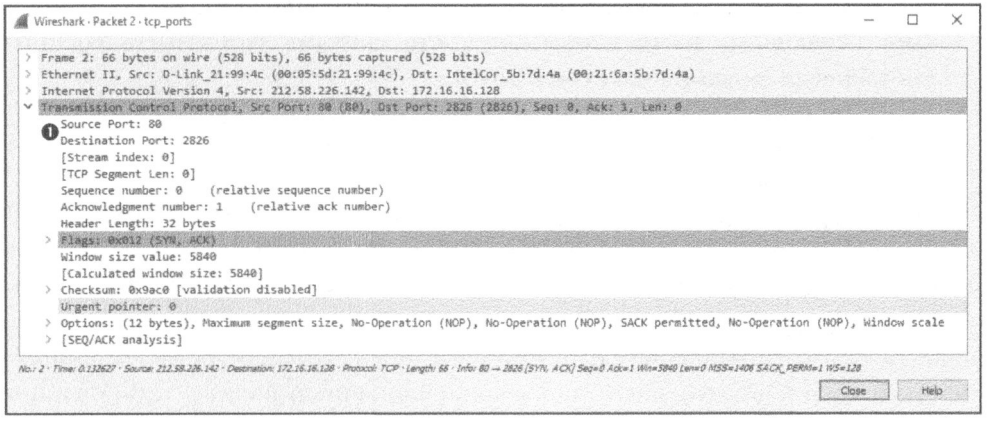


Рис. 8.4. При передаче данных в обратном направлении номера портов отправителя и получателя меняются местами

Как правило, обмен данными по протоколу TCP происходит следующим образом: номер порта отправителя произвольно выбирается для передачи данных получателю по известному порту. И как только будет отправлен первоначальный пакет, удаленное принимающее устройство свяжется с передающим устройством через установленные порты.

Файл перехвата из рассматриваемого здесь примера содержит еще один поток обмена данными. Попробуйте сами определить номера портов, выбираемых в этом случае для обмена данными.

ПРИМЕЧАНИЕ *По мере дальнейшего чтения этой книги вы узнаете еще больше о портах, связанных с общеупотребительными протоколами и службами. И в конечном итоге вы научитесь определять службы и устройства по используемым в них портам. Полный перечень общеупотребительных портов можно найти в упоминавшемся выше файле* **services**, *расположенном в системном каталоге Wireshark.*

Трехэтапный процесс установки связи по протоколу TCP

Файл перехвата **tcp_handshake.pcapng**

Весь обмен данными по протоколу TCP должен начинаться с *установки связи* между двумя хостами. Такой процесс служит нескольким целям, включая следующие.

- Передающий хост может убедиться, что принимающий хост готов к обмену данными.
- Передающий хост может проверить, что принимающий хост готов принимать данные через тот порт, через который передающий хост пытается установить связь.
- Передающий хост может послать свой начальный порядковый номер принимающему хосту, чтобы оба хоста могли поддерживать заданный порядок следования пакетов в потоке.

Процесс установки связи по протоколу TCP происходит в три этапа, как показано на рис. 8.5. На первом этапе устройство, которому требуется передать данные (хост А), посылает пакет TCP получателю (хосту Б). В этом первоначальном пакете не содержится ничего, кроме заголовков пакетов протоколов более низкого уровня. В TCP-заголовке данного пакета установлен флаг SYN, а также указан первоначальный порядковый номер и максимальный размер сегмента (MSS), употребляемый в процессе последующего обмена данными. Хост Б отвечает на этот пакет, посылая аналогичный пакет с установленными флагами SYN и ACK и своим первоначальным порядковым номером. И, наконец, хост А посылает еще один, последний пакет хосту Б с установленным

флагом ACK. Как только этот процесс завершится, оба хоста должны иметь в своем распоряжении все сведения, необходимые для того, чтобы начать обмен данными надлежащим образом.

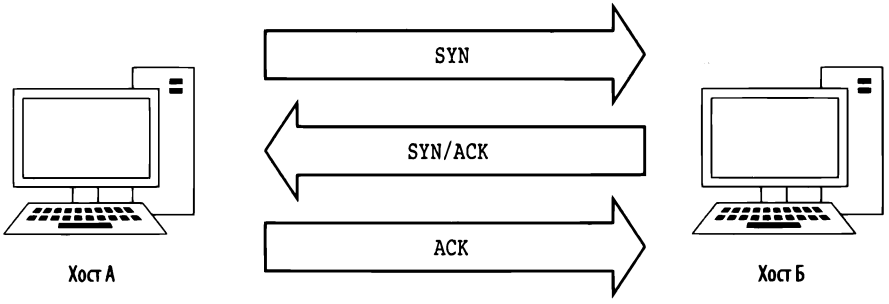


Рис. 8.5. Трехэтапная установка связи по протоколу TCP

ПРИМЕЧАНИЕ *Зачастую пакеты TCP обозначаются по тем флагам, которые установлены в их заголовках. Например, пакет TCP с установленным флагом* SYN *можно для краткости обозначить как пакет* SYN. *Таким образом, пакеты, используемые в процессе установки связи по протоколу TCP, могут быть обозначены как* SYN, SYN/ACK *и* ACK *соответственно.*

Чтобы понаблюдать рассматриваемый здесь процесс в действии, откройте файл перехвата tcp_handshake.pcapng. Для упрощения анализа пакетов в Wireshark имеется режим замены последовательных номеров пакетов TCP относительными номерами. Но для целей данного примера действие этого режима лучше отменить, чтобы наблюдать порядковые номера пакетов. С этой целью выберите команду Edit⇨Preferences из главного меню, разверните заголовок Protocols, выберите вариант TCP, сбросьте флажок Relative Sequence Numbers (Относительные порядковые номера) в открывшемся окне и щелкните на кнопке OK.

Первый пакет в этом перехваченном трафике является первоначально посылаемым пакетом SYN ❷ (рис. 8.6). Этот пакет передается из хоста, находящегося по адресу **172.16.16.128**, через порт **2826** хосту, принимающему переданный пакет по адресу **212.58.226.142** через порт **80**. Как видите, в данном пакете передается порядковый номер **3691127924** ❶.

Вторым в процессе подтверждения связи является пакет SYN/ACK с ответом ❸ от хоста, находящегося по адресу **212.58.226.142** (рис. 8.7). Этот пакет также содержит первоначальный порядковый номер (**233779340**) ❶ данного хоста и номер подтверждения (**3691127925**) ❷. Показанный здесь номер подтверждения на единицу больше порядкового номера из предыдущего пакета, поскольку в данном поле указывается следующий порядковый номер, который ожидает получить хост.

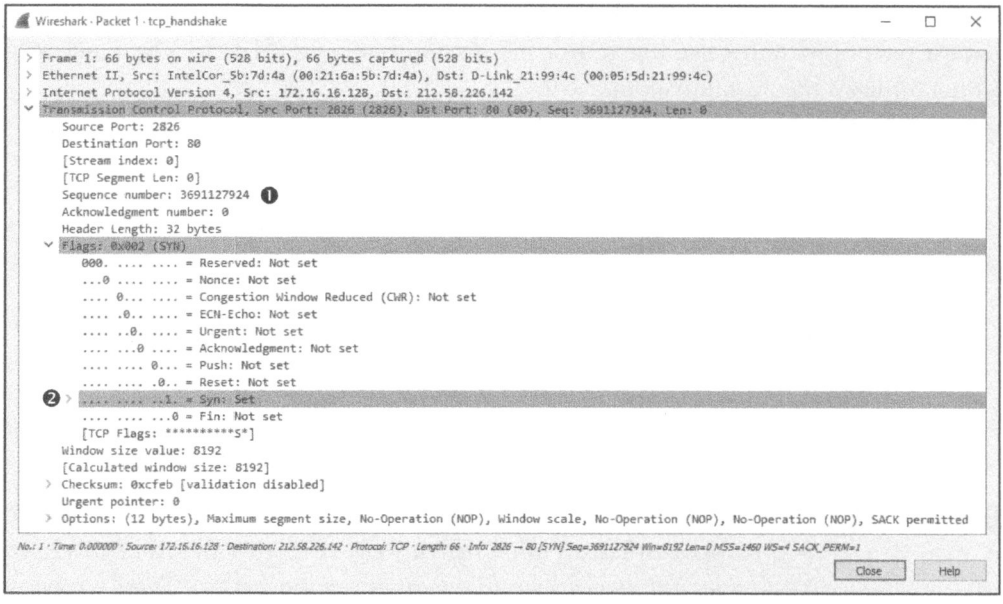


Рис. 8.6. Первоначально посылаемый пакет SYN

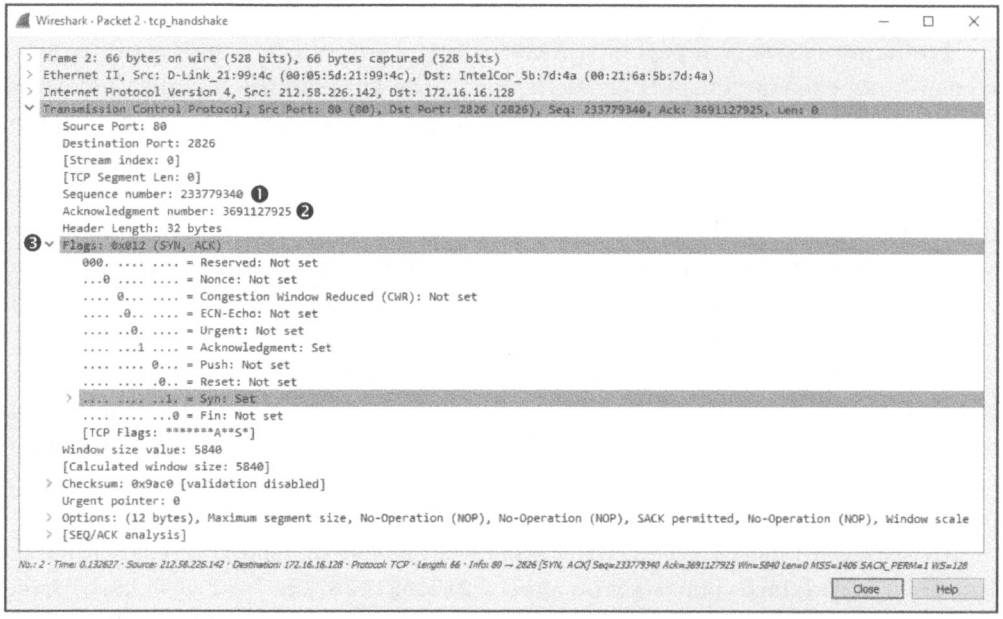


Рис. 8.7. Ответный пакет SYN/ACK

И последним в процессе трехэтапной установки связи по протоколу TCP является пакет ACK ❷, посылаемый из хоста, находящегося по адресу **172.16.16.128** (рис. 8.8). Как и следовало ожидать, этот пакет содержит

порядковый номер **3691127925** ❶, определяемый на основании номера подтверждения в соответствующем поле предыдущего пакета.

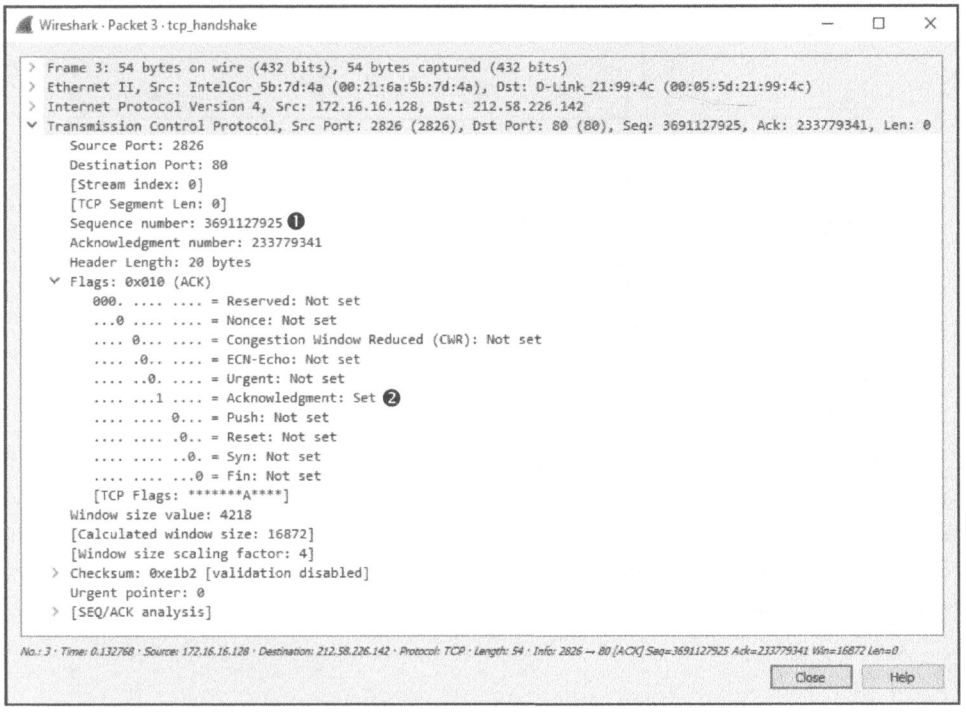


Рис. 8.8. Последний пакет ACK

Установка связи происходит перед каждой последовательностью обмена данными по протоколу TCP. Если вы сортируете крупный файл перехвата в поисках начала последовательности обмена данными, то явным его признаком служит последовательность пакетов `SYN-SYN/ACK-ACK`.

Разрыв связи по протоколу TCP

Файл перехвата **`tcp_teardown.pcapng`**

Большинство приветствий в конечном итоге завершается прощанием, а в протоколе TCP каждый сеанс установки связи — разрывом. В частности, *разрыв связи по протоколу TCP* служит для корректного завершения сеанса связи, установленного между двумя устройствами, после того, как они завершат обмен данными. Этот процесс включает в себя четыре пакета и установку флага `FIN` для обозначения конца связи и разрыва соединения.

В последовательности разрыва связи хост А сообщает об окончании обмена данными посылкой пакета TCP с установленными флагами `FIN` и `ACK`. На это хост Б отвечает пакетом `ACK` и передает свой пакет `FIN/ACK`. В свою очередь,

хост А отвечает пакетом ACK, завершая обмен данными. Этот процесс наглядно показан на рис. 8.9.

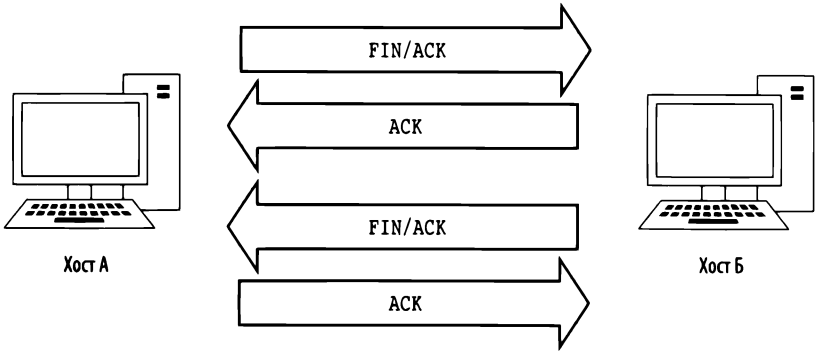


Рис. 8.9. Процесс разъединения по протоколу TCP

Чтобы понаблюдать за процессом разъединения по протоколу TCP в Wireshark, откройте файл перехвата `tcp_teardown.pcapng`. Начиная с первого пакета в перехваченной последовательности (рис. 8.10), вы можете заметить, что устройство, находящееся по адресу **`67.228.110.120`**, инициирует разъединение, посылая пакет с установленными флагами FIN и ACK ❶.

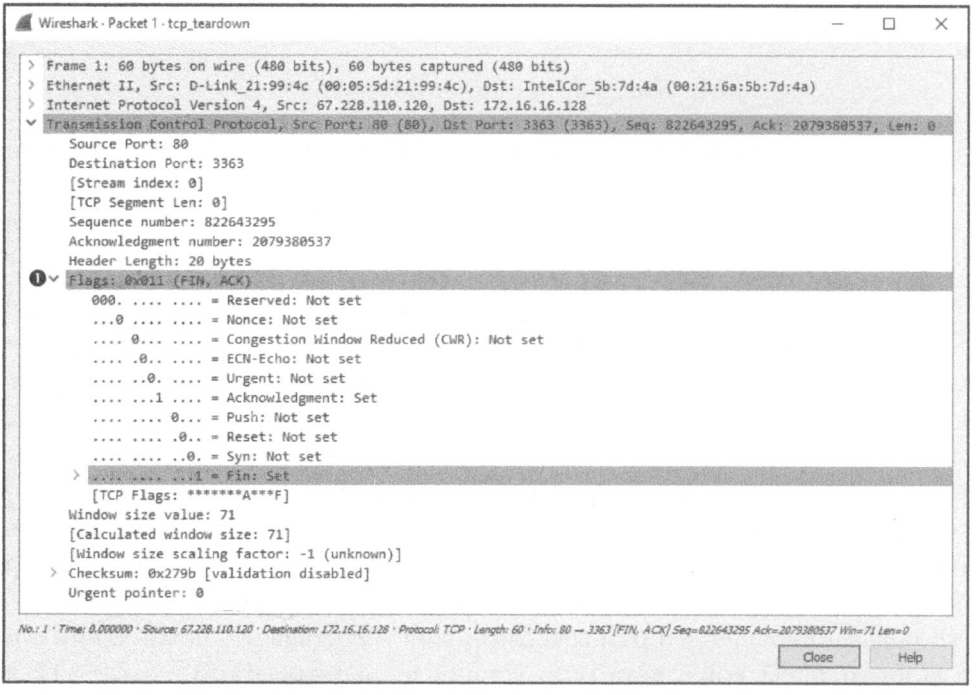


Рис. 8.10. Процесс разрыва связи инициируется отправкой пакета FIN/ACK

Как только первый пакет будет отправлен, хост, находящийся по адресу **172.16.16.128**, ответит пакетом ACK, чтобы подтвердить прием этого пакета, а затем пошлет свой пакет FIN/ACK. Процесс разъединения завершится, когда хост, находящийся по адресу **67.228.110.120**, отправит завершающий пакет ACK. В этот момент обмен данными между двумя устройствами завершится. Если же им потребуется возобновить обмен данными, для этого им придется снова выполнить трехэтапную установку связи по протоколу TCP.

Сбросы соединений по протоколу TCP

Файл перехвата **tcp_refuseconnection.pcapng**

В идеальном случае каждое соединение должно корректно завершаться путем запуска процедуры разрыва связи по протоколу TCP. Однако в реальном мире соединения очень часто разрываются внезапно. Например, хост может быть неверно настроен или же потенциальный злоумышленник может запустить программу сканирования портов. В подобных случаях если устройство не желает принять отправленный ему пакет, оно может отослать пакет TCP с установленным флагом RST. Этот флаг служит для указания на то, что соединение резко разорвано, либо что попытка установки связи отвергнута.

В файле перехвата tcp_refuseconnection.pcapng демонстрируется пример сетевого трафика, включающего в себя пакет RST. Первый пакет в этом файле отправлен хостом, находящимся по адресу **192.168.100.138**, чтобы попытаться связаться с хостом, расположенным по адресу **192.168.100.1**, через порт **80**. Но этому хосту неизвестно, что хост, находящийся по адресу **192.168.100.1**, вообще не принимает пакеты через порт **80**, поскольку это маршрутизатор Cisco, который не поддерживает веб-интерфейс конфигурирования. Таким образом, службы, настроенной на прием соединений через этот порт, не существует. В ответ на подобную попытку установить связь хост, находящийся по адресу **192.168.100.1**, посылает пакет хосту, расположенному по адресу **192.168.100.138**, сообщая, что установить связь через порт **80** невозможно. На рис. 8.11 показано, как в TCP-заголовке второго пакета, можно аварийно завершить данную попытку установки связи. В частности, пакет RST не содержит ничего, кроме установленных в его заголовке флагов RST и ACK ❶, после чего никакого обмена данными больше не происходит.

Пакет RST в любом случае завершает обмен данными, независимо от того поступает ли он вначале попытки установить связь, как в данном примере, или же посылается в ходе обмена данными между хостами.

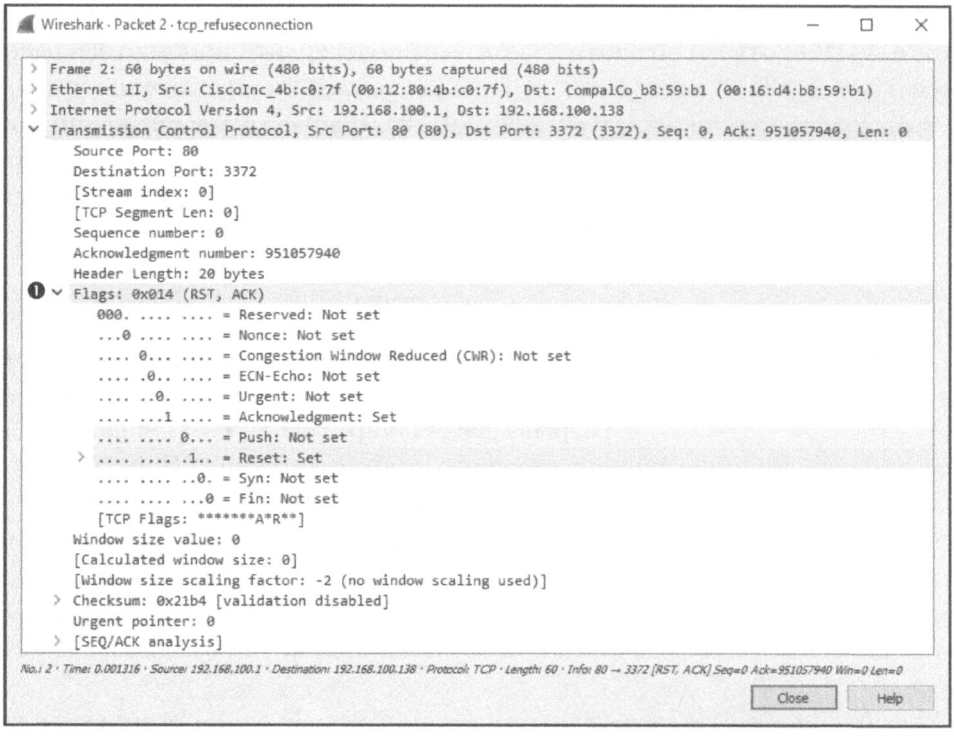


Рис. 8.11. Установленные флаги RST и ACK указывают на конец связи

Протокол пользовательских дейтаграмм (UDP)

Файл перехвата **udp_dnsrequest.pcapng**

Протокол UDP (User Datagram Protocol – протокол пользовательских дейтаграмм) – это еще один протокол четвертого уровня, зачастую применяемый в современных сетях. Если протокол TCP предназначен для надежной доставки данных со встроенной проверкой ошибок, то назначение протокола UDP – обеспечить быструю передачу данных. Именно по этой причине протокол UDP *не требует* предварительной *установки соединения* и, соответственно, *не может гарантировать* доставку пакетов при передаче данных. Протокол UDP формально не устанавливает и не разрывает соединение между хостами, как это делает протокол TCP.

Сетевой трафик с использованием не требующего установки соединения протокола UDP и не предоставляющего надежных услуг по транспортировке данных, может показаться, по меньшей мере, непредсказуемым. И это справедливо! В результате другие протоколы, использующие протокол UDP в качестве транспортного, как правило, имеют встроенную поддержку надежной передачи данных или пользуются определенными средствами протокола

ICMP, чтобы повысить надежность соединения. Например, протоколы уровня приложений DNS и DHCP, функционирование которых сильно зависит от скорости передачи пакетов по сети, пользуются UDP как протоколом транспортного уровня, но в то же время они сами организуют проверку ошибок и установку таймеров повторной передачи.

Структура заголовка в пакете UDP

Файл перехвата **udp_dnsrequest.pcapng**

UDP-заголовок намного меньше и проще, чем TCP-заголовок. Как показано на рис. 8.12, UDP-заголовок состоит из следующих полей.

Протокол пользовательских дейтаграмм (UDP)					
Смещение	Октет	0	1	2	3
Октет	Бит	0–7	8–15	16–23	24–31
0	0	Порт отправителя		Порт получателя	
4	32	Длина пакета		Контрольная сумма	

Рис. 8.12. Структура заголовка в пакете UDP

- **Порт отправителя (Source Port).** Порт, используемый для передачи пакета на машине отправителя.
- **Порт получателя (Destination Port).** Порт, в который передается пакет на машине получателя.
- **Длина пакета (Packet Length).** Длина пакета в байтах.
- **Контрольная сумма (Checksum).** Служит для гарантии неизменности UDP-заголовка и данных, когда они поступят по месту их назначения.

Файл перехвата `udp_dnsrequest.pcapng` содержит один пакет. Этот пакет представляет собой DNS-запрос, в котором применяется протокол UDP. Если развернуть UDP-заголовок, то в нем можно обнаружить четыре поля ❶, как показано на рис. 8.13.

Очень важно не забывать, что протокол UDP не обеспечивает надежную доставку пакетов. Следовательно, в любом приложении, где применяется этот протокол, должны быть приняты дополнительные меры для обеспечения надежной доставки пакетов, если в этом есть необходимость. Это совсем не похоже на протокол TCP, где выполняется формальное установление соединения и его разрыв, а также имеются встроенные средства для проверки удачного исхода передачи пакетов.

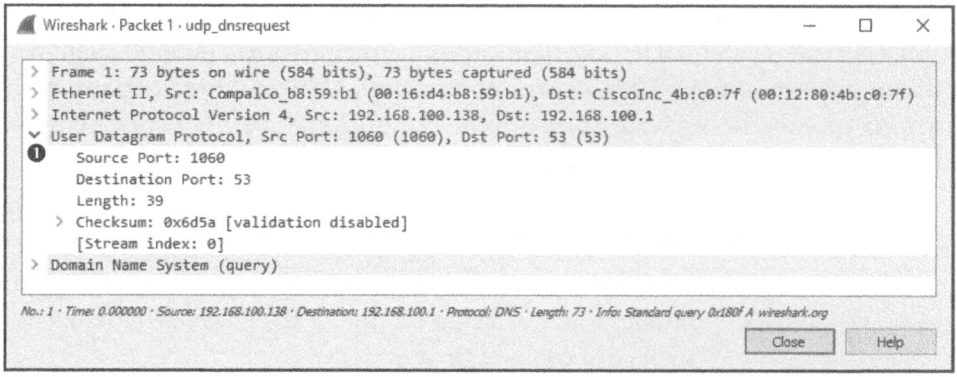


Рис. 8.13. Содержимое пакета UDP довольно простое

В этой главе были представлены протоколы TCP и UDP транспортного уровня. В отличие от протоколов сетевого уровня, протоколы TCP и UDP составляют основу большей части повседневного обмена данными по сети, и поэтому умение эффективно анализировать их играет решающую роль в становлении грамотного аналитика пакетов. В главе 9 будут рассмотрены наиболее употребительные протоколы уровня приложений.

9

РАСПРОСТРАНЕННЫЕ ПРОТОКОЛЫ ВЕРХНЕГО УРОВНЯ

В этой главе мы продолжим исследование функций отдельных протоколов, а также выясним, как выглядят их пакеты при просмотре в Wireshark. Мы также обсудим пять наиболее распространенных протоколов верхнего (седьмого) уровня модели OSI: DHCP, DNS, HTTP и SMTP.

Протокол динамической настройки узла сети (DHCP)

На ранней стадии развития сетей, когда устройству требовалось передать данные по сети, ему приходилось присваивать адрес вручную. По мере разрастания сетей этот ручной процесс быстро стал трудоемким и неудобным. В качестве выхода из столь затруднительного положения был разработан протокол BOOTP (Bootstrap Protocol – протокол начальной самозагрузки), предназначенный для автоматического присваивания адресов устройствам, подключаемым к сети. Впоследствии протокол BOOTP был заменен более развитым протоколом DHCP (Dynamic Host Configuration Protocol – протокол динамической настройки узла сети).

Протокол DHCP относится к уровню приложений и отвечает за возможность для устройства автоматически получать IP-адрес, а также адреса других важных сетевых ресурсов, в том числе DNS-серверов и маршрутизаторов.

Большинство современных DHCP-серверов предоставляют также другие параметры клиентам, в том числе адреса используемого по умолчанию шлюза и DNS-серверы, используемые в сети.

Структура заголовка в пакете DHCP

Пакеты DHCP могут переносить довольно много информации клиенту. Как показано на рис. 9.1, в заголовке пакета DHCP имеются следующие поля.

Протокол динамической настройки узла сети (DHCP)					
Смещение	Октет	0	1	2	3
Октет	Бит	0–7	8–15	16–23	24–31
0	0	Код операции	Аппаратный тип	Аппаратная длина	Переходы
4	32	Идентификатор транзакции			
8	64	Количество истекших секунд		Флаги	
12	96	IP-адрес клиента			
16	128	Ваш IP-адрес			
20	160	IP-адрес сервера			
24	192	IP-адрес шлюза			
28	224	IP-адрес клиента			
32	256	Аппаратный адрес клиента (16 байт)			
36	288				
40	320				
44	352				
48+	384+	Имя сетевого узла сервера (64 байта)			
		Файл начальной загрузки (129 байт)			
		Параметры			

Рис. 9.1. Структура заголовка в пакете DHCP

- **Код операции (OpCode).** Обозначает, содержит ли пакет DHCP-запрос или DHCP-ответ.
- **Аппаратный тип (Hardware Type).** Тип аппаратного адреса (10MB Ethernet, IEEE 802, ATM и т.д.).
- **Аппаратная длина (Hardware Length).** Длина аппаратного адреса.
- **Переходы (Hops).** Применяется агентом ретрансляции в поисках DHCP-сервера.

- **Идентификатор транзакции (Transaction ID).** Произвольное число, применяемое для объединения запросов и ответов в пары.
- **Количество истекших секунд (Seconds Elapsed).** Количество секунд, прошедших с того момента, когда клиент в первый раз запросил адрес у DHCP-сервера.
- **Флаги (Flags).** Типы сетевого трафика (одноадресатного, широковещательного и т.д.), который может принять DHCP-клиент.
- **IP-адрес клиента (Client IP Address).** IP-адрес клиента, полученный из поля Your IP Address.
- **Ваш IP-адрес (Your IP Address).** IP-адрес, предоставляемый DHCP-сервером и в конечном итоге становящийся адресом клиента, помещаемый в поле Client IP Address.
- **IP-адрес сервера (Server IP Address).** IP-адрес DHCP-сервера.
- **IP-адрес шлюза (Gateway IP Address).** IP-адрес используемого по умолчанию сетевого шлюза.
- **Аппаратный адрес клиента (Client Hardware Address).** MAC-адрес клиента.
- **Имя узла сервера (Server Host Name).** Имя сетевого узла, на котором запущен сервер (опционально).
- **Файл начальной загрузки (Boot File).** Файл, применяемый в протоколе DHCP для начальной самозагрузки узла сети (опционально).
- **Параметры (Options).** В этом поле находятся дополнительные параметры, которые применяются для расширения структуры пакета DHCP с целью предоставить дополнительные возможности.

Процесс инициализации по протоколу DHCP

Файл перехвата **dhcp_nolease_initialization.pcapng**

Основное назначение протокола DHCP – присвоить адреса клиентам в процессе инициализации. Этот процесс возобновления связи происходит между одним клиентом и DHCP-сервером, как демонстрируется в примере из файла перехвата dhcp_nolease_initialization.pcapng. Процесс инициализации по протоколу DHCP нередко называют процессом DORA (Discover, Offer, Request, Acknowledgment – обнаружение, предложение, запрос, подтверждение), поскольку в нем используются следующие типы пакетов DHCP: обнаружения, предложения, запроса, подтверждения (рис. 9.2). Рассмотрим по очереди каждый из этих типов пакетов DORA.

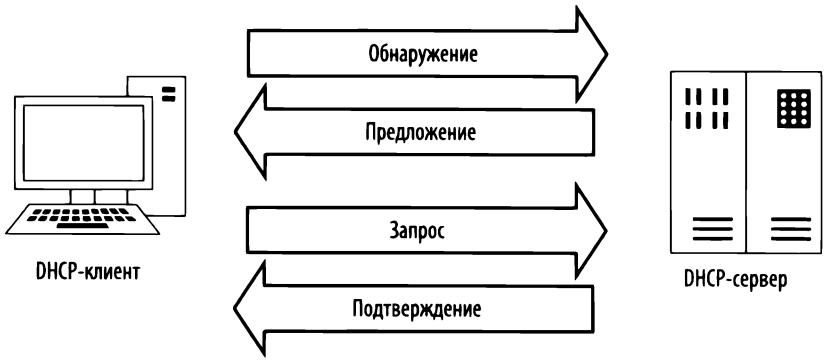


Рис. 9.2. Процесс DORA про протоколу DHCP

Пакет обнаружения

Как следует из упомянутого выше файла перехвата, первый пакет посылается из хоста, находящегося по адресу `0.0.0.0`, через порт **68** хосту, принимающему этот пакет по адресу `255.255.255.255` через порт **67**. Клиент пользуется адресом `0.0.0.0`, поскольку у него пока еще отсутствует свой IP-адрес. Данный пакет посылается по универсальному широковещательному адресу `255.255.255.255`, не зависимому от конкретной сети. Этим гарантируется, что данный пакет будет отправлен каждому устройству в сети. А поскольку устройству неизвестен адрес DHCP-сервера, то этот первый пакет посылается в попытке обнаружить тот DHCP-сервер, который примет его.

Исследуя содержимое панели Packet Details, в первую очередь можно обнаружить, что в протоколе DHCP в качестве протокола транспортного уровня используется UDP. Для протокола DHCP очень важна скорость, с которой клиент принимает запрашиваемую им информацию. Но в протокол DHCP встроены собственные средства, обеспечивающие надежность передачи данных, а это означает, что для него вполне подходит протокол UDP. Более подробно процесс обнаружения можно рассмотреть, исследовав часть первого пакета, относящуюся к протоколу DHCP, в панели Packet Details (рис. 9.3).

ПРИМЕЧАНИЕ *Приложение Wireshark по-прежнему при анализе протокола DHCP называет его BOOTP, и поэтому в панели* Packet Details *имеется раздел* **Bootstrap Protocol** *вместо раздела* **DHCP**. *Тем не менее здесь и далее этот раздел называется частью пакета, относящейся к протоколу DHCP.*

Рассматриваемый здесь пакет содержит запрос, на что указывает значение в поле Message type ❶. Большинство полей в данном пакете обнаружения содержат нули (как можно видеть в полях IP-адресов ❷) или же не требуют особых пояснений, исходя из их назначения, вкратце описанного в предыдущем

разделе, где перечислены поля DHCP-заголовка. Вся суть данного пакета кроется в его четырех перечисленных ниже полях Option ❸.

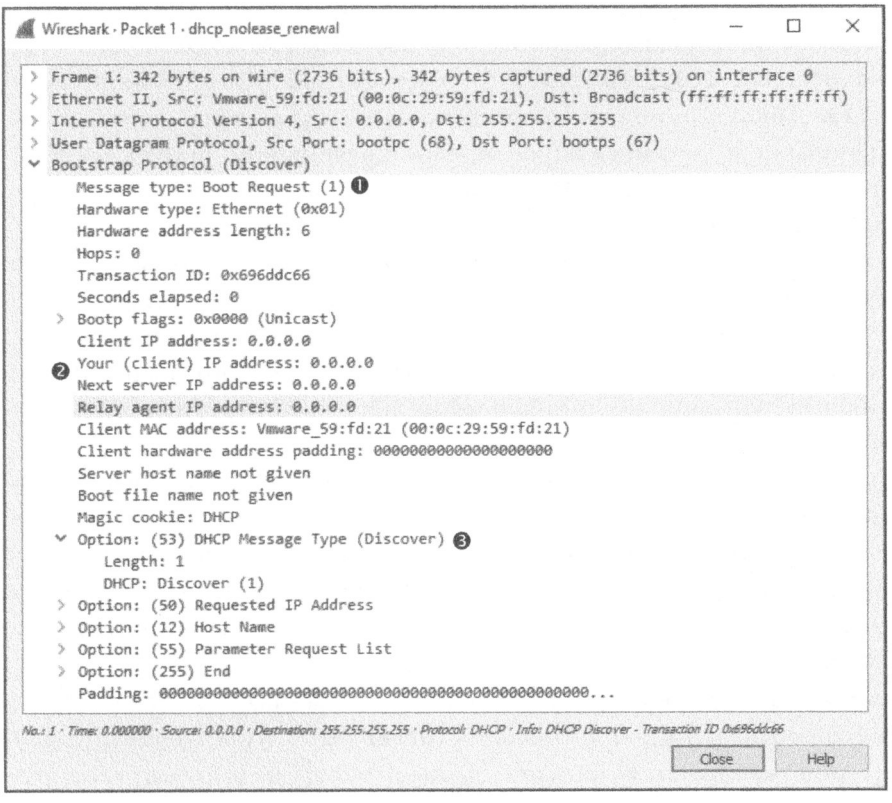


Рис. 9.3. Пакет обнаружения по протоколу DHCP

- **DHCP Message Type (Тип сообщения DHCP).** В этом поле задается тип сообщения **53** длиной **1** и со значением **`Discover (1)`**. Эти значения указывают на то, что данный пакет DHCP относится к категории обнаружения.
- **Client Identifier (Идентификатор клиента).** В этом поле предоставляются дополнительные сведения о клиенте, запрашивающем IP-адрес.
- **Requested IP Address (Запрашиваемый IP-адрес).** В этом поле предоставляется IP-адрес, который требуется получить клиенту. Это может быть применявшийся ранее IP-адрес или же адрес **`0.0.0.0`**, указывающий на отсутствие каких-то предпочтений.
- **Parameter Request List (Список запрашиваемых параметров).** В этом поле предоставляется список различных элементов конфигурации (IP-адресов других важных сетевых устройств и прочих не относящихся к ним элементов), которые клиенту требуется получить от DHCP-сервера.

Пакет предложения

В соответствующем IP-заголовке второго пакета из рассматриваемого здесь файла перехвата перечислены IP-адреса, обозначающие перемещение этого пакета из хоста, находящегося по адресу `192.168.1.5`, к хосту, расположенному по адресу `192.168.1.10` (рис. 9.4). У клиента на самом деле пока еще нет адреса `192.168.1.10`, поэтому сервер попытается сначала связаться с клиентом, используя аппаратный адрес, предоставляемый по протоколу ARP. Если же такая связь невозможна, сервер просто разошлет этот пакет в широковещательном режиме.

Во втором пакете, который называется *пакетом предложения*, также имеется часть, относящаяся к протоколу DHCP, а в ней — поле Message type, где указывается, что речь идет об ответе ❶. Данный пакет содержит тот же самый идентификатор транзакции в поле Transaction ID, что и предыдущий пакет ❷. Это означает, что данный ответ действительно является реакцией на первоначальный запрос.

Пакет предложения посылается DHCP-сервером с целью предложить свои услуги клиенту. Для этого в нем предоставляются сведения о самом сервере и адресах, которыми сервер может снабдить клиента. Как показано на рис. 9.4, IP-адрес `192.168.1.10`, указанный в поле Your IP address, ❸ предлагается клиенту от сервера, находящегося по адресу `192.168.1.5` и указанному в поле Next server IP address ❹.

В первом поле перечисляемых параметров указывается, что данный пакет DHCP относится к типу предложения (`Offer`) ❺. А в следующих далее полях параметров указываются дополнительные сведения, которые сервер может предоставить вместе с IP-адресом клиента. В частности, сервер может предложить следующее.

- Время аренды IP-адреса — 10 минут.
- Маска подсети `255.255.255.0`.
- Широковещательный адрес `192.168.1.255`.
- Адрес маршрутизатора `192.168.1.254`.
- Доменное имя `mydomain.example`.
- Адреса `192.168.1.1` и `192.168.1.2` сервера доменных имен.

Пакет запроса

Как только клиент получит предложение от DHCP-сервера, он должен принять его в пакете DHCP-запроса, как показано на рис. 9.5. Третий пакет из рассматриваемого здесь файла перехвата по-прежнему поступает из сетевого

узла с IP-адресом `0.0.0.0`, поскольку процесс получения IP-адреса еще не завершен ❶. Теперь в пакете известно, с каким именно DHCP-сервером устанавливается связь.

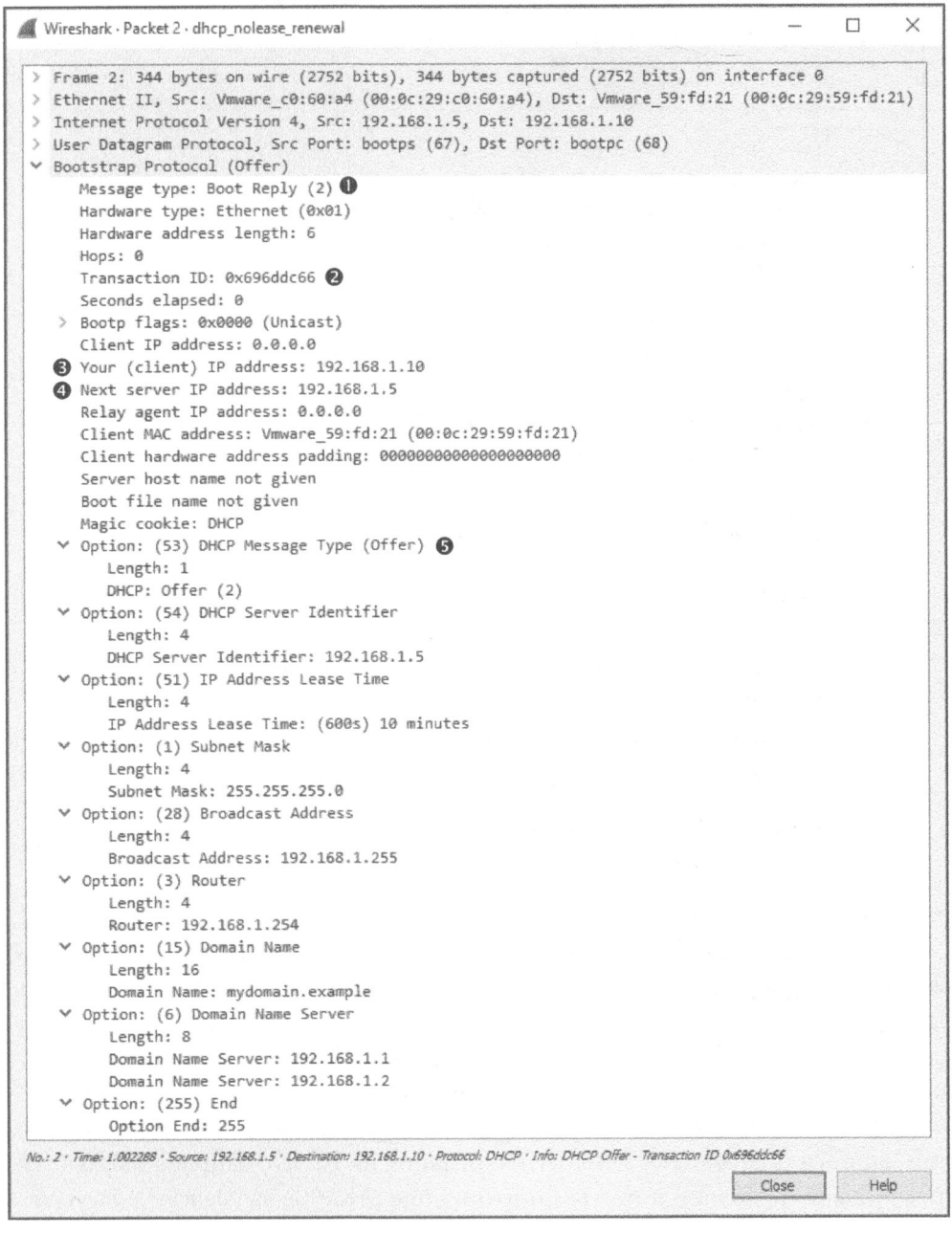


Рис. 9.4. Пакет предложения по протоколу DHCP

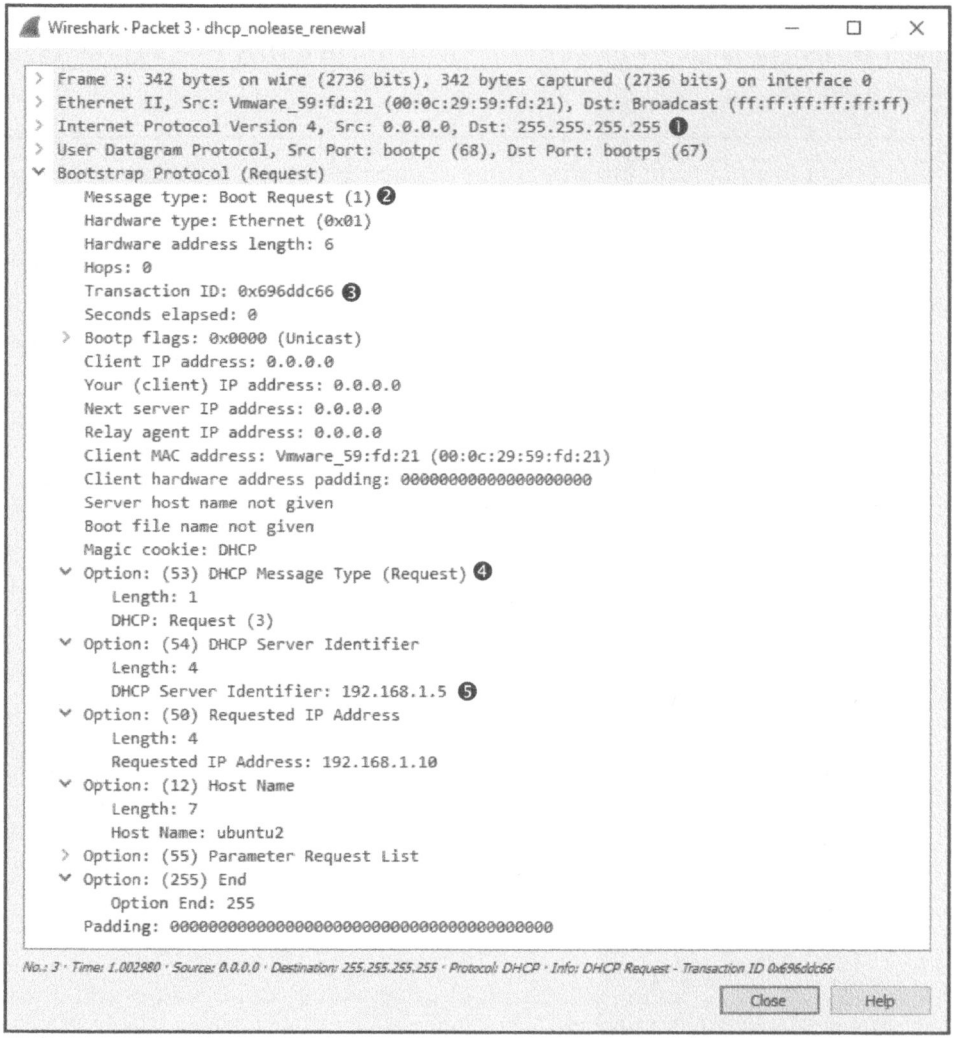


Рис. 9.5. Пакет запроса по протоколу DHCP

В поле Message type указано, что данный пакет относится к типу запроса ❷, а в поле Transaction ID содержится то же самое число, что и в первых двух пакетах ❸, указывая на то, что они являются частью одного и того же процесса. Данный пакет подобен пакету обнаружения в том отношении, что вся содержащаяся в нем информация об IP-адресах обнуляется.

И, наконец, в полях Option указано, что данный пакет DHCP относится к типу запроса (**Request**) ❹. Обратите внимание на то, что запрашиваемый IP-адрес больше не является пустым и что в поле DHCP Server Identifier содержится соответствующий IP-адрес ❺.

Пакет подтверждения

На последней стадии рассматриваемого здесь процесса DHCP-сервер посылает клиенту запрашиваемые IP-адреса в пакете подтверждения, записывая эту информацию в своей базе данных, как показано на рис. 9.6. В итоге клиент получит свой IP-адрес, чтобы начать по нему обмен данными в сети.

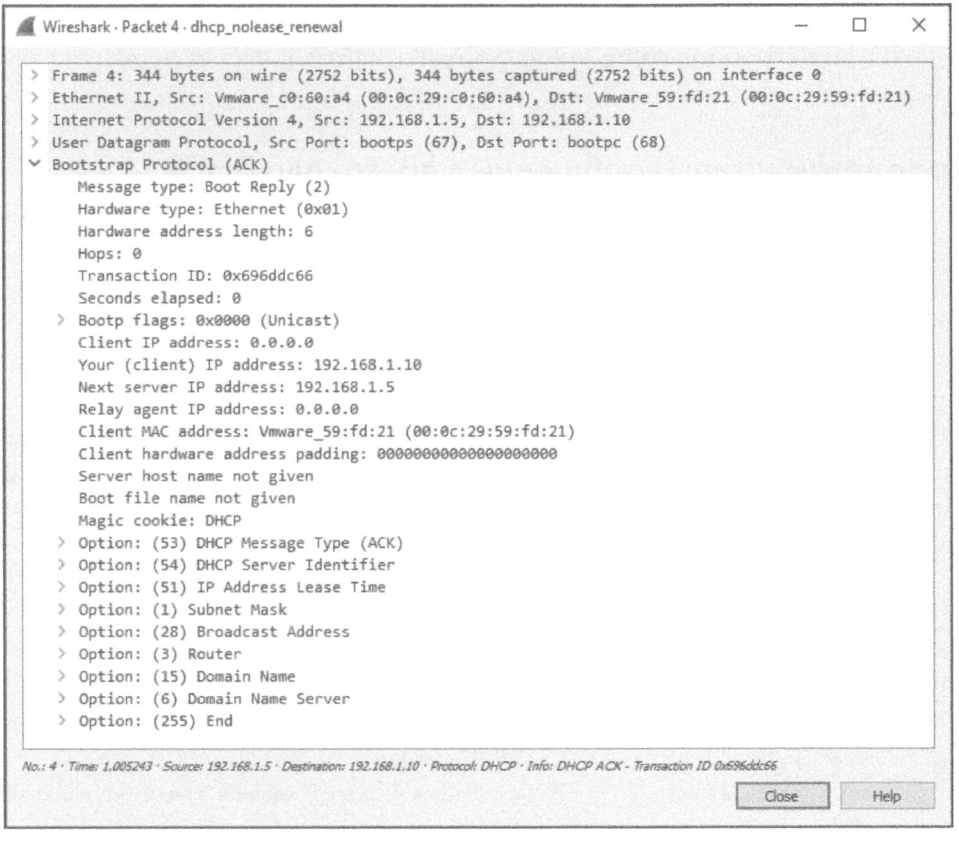


Рис. 9.6. Пакет подтверждения по протоколу DHCP

Возобновление аренды IP-адреса по протоколу DHCP

Файл перехвата **dhcp_inlease_renewal.pcapng**

Когда DHCP-сервер присваивает IP-адрес устройству, он сдает его клиенту в *аренду*. Это означает, что клиенту разрешается пользоваться присвоенным ему IP-адресом только в течение ограниченного периода времени, по истечении которого ему придется возобновить аренду данного адреса. Описанный выше процесс DORA происходит, когда клиент только получает свой IP-адрес или же когда время его аренды истекло. В любом случае считается, что у такого устройства *истек срок аренды*.

Когда клиент с арендованным IP-адресом перезагружается, он должен выполнить укороченный вариант процесса DORA, чтобы затребовать свой IP-адрес обратно. Такой процесс называется *возобновлением аренды*.

Для возобновления арендуемых адресов пакеты обнаружения и предложения не нужны. Возобновление аренды можно рассматривать как такой же самый процесс DORA, что и для устройства с истекшим сроком аренды, только для этого достаточно выполнить стадии запроса и подтверждения. Характерный пример возобновления аренды можно найти в файле перехвата `dhcp_inlease_renewal.pcapng`.

Параметры и типы сообщений в протоколе DHCP

Настоящее удобство протокола DHCP кроется в доступных его параметрах. Как было показано выше, параметры, указываемые в пакете DHCP, могут отличаться длиной и содержимым. Общий размер пакета DHCP зависит от сочетания применяемых в нем параметров. Полный перечень многочисленных параметров протокола DHCP можно найти по адресу `http://www.iana.org/assignments/bootp-dhcp-parameters/`.

Единственный параметр, который требуется во всех пакетах DHCP, указан в поле **Message type** (тип сообщения **53**). Этот параметр обозначает порядок обработки DHCP-сервером и клиентом информации, содержащейся в пакете. Восемь имеющихся типов сообщений перечислены табл. 9.1.

Таблица 9.1. Типы сообщений по протоколу DHCP

Порядковый номер	Тип сообщения	Описание
1	Обнаружение	Применяется клиентом для обнаружения имеющихся DHCP-серверов
2	Предложение	Посылается сервером клиенту в ответ на пакет обнаружения
3	Запрос	Посылается клиентом для запрашивания у сервера предлагаемых им параметров
4	Отклонение	Посылается клиентом серверу для указания на недостоверные параметры в пакете
5	Подтверждение (`ACK`)	Посылается сервером клиенту с запрашиваемыми параметрами конфигурации
6	Неподтверждение (`NACK`)	Посылается клиентом серверу для отказа от запроса параметров конфигурации
7	Освобождение	Посылается клиентом серверу для отмены аренды путем освобождения параметров конфигурации
8	Извещение	Посылается клиентом серверу для запроса параметров конфигурации, когда у клиента уже имеется IP-адрес

Версия 6 протокола DHCP (DHCPv6)

Файл перехвата **dhcp6_outlease_acquisition.pcapng**

Если проанализировать структуру заголовка в пакете DHCP, приведенную на рис. 9.1, то можно заметить, что в нем недостаточно места для размещения адреса протокола IPv6. Вместо модернизации протокола DHCP для этих целей была разработана его версия DHCPv6 по стандарту RFC 3315. Но поскольку версия DHCPv6 не построена по принципу протокола BOOTP, то структура заголовка в пакете становится в данном случае более простой (рис. 9.7).

Версия 6 протокола динамической настройки узла сети (DHCPv6)					
Смещение	Октет	0	1	2	3
Октет	Бит	0–7	8–15	16–23	24–31
0	0	Тип сообщения	Идентификатор транзакции		
4+	32+	Параметры			

Рис. 9.7. Структура заголовка в пакете DHCPv6

Структура заголовка в рассматриваемом здесь пакете DHCPv6 содержит только два статических значения, которые функционируют таким же образом, как и их аналоги в протоколе DHCP. А остальная часть структуры пакета меняется в зависимости от типа сообщения, обозначенного в первом байте. В разделе Параметры (Options) каждый параметр обозначается двухбайтовым кодом этого параметра и двухбайтовым полем длины. Полный перечень типов сообщений и кодов параметров, которые могут появляться в этих полях, приведен по адресу `http://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xhtml`.

Протокол DHCPv6 преследует такую же цель, как протокол DHCP, но чтобы уяснить поток обмена данными по протоколу DHCPv6, сокращенное обозначение процесса DORA следует заменить новым сокращением SARR (Solicit, Advertise, Request, Reply – опрос, анонс, запрос, ответ). Этот процесс наглядно показан на рис. 9.8, где представлен клиент с истекшим сроком аренды.

Процесс SARR состоит из следующих стадий.

1. **Опрос.** Первоначальный пакет посылается клиентом по специальному многоадресатному адресу (**`ff02::1:2`**), чтобы попытаться обнаружить доступные DHCPv6-серверы в сети.
2. **Анонс.** Доступный сервер отвечает непосредственно клиенту, чтобы известить его, что он доступен для предоставления сведений об адресах и конфигурации.

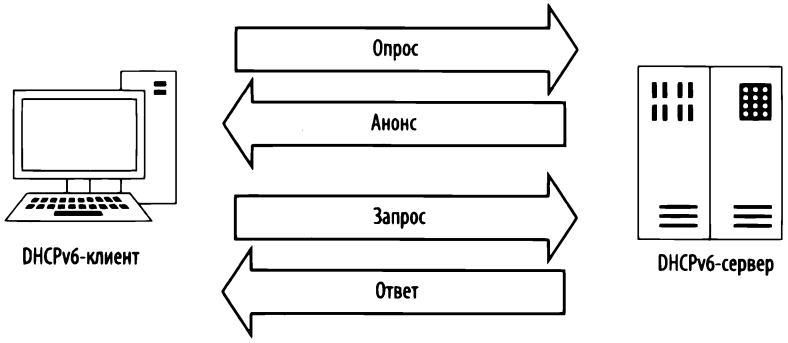


Рис. 9.8. Процесс SARR возобновления адреса клиента с истекшим сроком аренды по протоколу DHCPv6

3. **Запрос.** Клиент посылает формальный запрос сведений о конфигурации доступному серверу по каналу многоадресатной передачи.
4. **Ответ.** Сервер посылает все имеющиеся сведения о конфигурации непосредственно клиенту, и на этом процесс SARR завершается.

Сводка описанного выше процесса SARR, взятая из файла перехвата `dhcp6_outlease_acquisition.pcapng`, приведена на рис. 9.9. В данном примере демонстрируется в действии процесс SARR, в ходе которого новый сетевой узел (**`fe80::20c:29ff:fe5e:7744`**) принимает сведения о конфигурации от DHCPv6-сервера (**`fe80::20c:29ff:fe1f:a755`**). Каждый пакет представляет одну стадию процесса SARR, причем первоначальные пакеты опроса и объявления связываются вместе с помощью идентификатора транзакции **`0x9de03f`**, а пакеты запроса и ответа – с помощью идентификатора транзакции **`0x2d1603`**. И хотя это не показано на рис. 9.9, рассматриваемый здесь обмен данными происходит через стандартные порты **`546`** и **`547`**, используемые в протоколе DHCPv6.

No	Time	Source	Destination	Protocol	Length	Info
1	0.…	fe80::20c:29ff:fe5e:7744	ff02::1:2	DHCPv6	118	Solicit XID: 0x9de03f CID: 000100011def69bd000c295e7744
2	0.…	fe80::20c:29ff:fe1f:a755	fe80::20c:29ff:fe5e:7744	DHCPv6	166	Advertise XID: 0x9de03f CID: 000100011def69bd000c295e7744 IAA: 2001:db8:1:2::1002
3	1.…	fe80::20c:29ff:fe5e:7744	ff02::1:2	DHCPv6	164	Request XID: 0x2d1603 CID: 000100011def69bd000c295e7744 IAA: 2001:db8:1:2::1002
4	1.…	fe80::20c:29ff:fe1f:a755	fe80::20c:29ff:fe5e:7744	DHCPv6	166	Reply XID: 0x2d1603 CID: 000100011def69bd000c295e7744 IAA: 2001:db8:1:2::1002

Рис. 9.9. Клиент, получающий IPv6-адрес по протоколу DHCPv6

В общем, структура пакета в сетевом трафике по протоколу DHCPv6 выглядит совсем иначе, хотя в ней применяются в основном те же самые понятия. Тем не менее в рассматриваемом здесь процессе требуется некоторая форма обнаружения DHCP-сервера и формального извлечения сведений о конфигурации. Все транзакции по-прежнему связываются по своим идентификаторам в каждой паре пакетов, которыми обмениваются клиент и сервер. Адресация по протоколу IPv6 не может поддерживаться традиционными механизмами

DHCP. Если в вашей сети имеются устройства, автоматически получающие IPv6-адреса от сервера, то это означает, что в вашей сети, вероятнее всего, уже действует служба DHCPv6. А если вам требуется продолжить сравнение протоколов DHCP и DHCPv6, то рекомендуется открыть упоминавшиеся ранее в этой главе файлы перехвата по очереди и сравнить их содержимое непосредственно.

Система доменных имен (DNS)

Протокол DNS (Domain Name System – система доменных имен) относится к числу наиболее важных межсетевых протоколов, поскольку он служит своего рода связующим звеном. В частности, протокол DNS связывает доменные имена (например, **www.google.com**) с IP-адресами (например, **74.125.159.99**). Если требуется связаться с сетевым устройством, IP-адрес которого неизвестен, получить доступ к этому устройству можно по его доменному имени в службе DNS.

На DNS-серверах хранится база данных с *записями ресурсов*, содержащих сопоставления IP-адресов и доменных имен, которыми эти серверы делятся с клиентами и другими DNS-серверами.

ПРИМЕЧАНИЕ *Структура DNS-серверов довольно сложна, поэтому здесь рассматриваются лишь самые распространенные типы сетевого трафика DNS. По различным вопросам, связанным с протоколом DNS, обращайтесь за справкой по адресу* **https://www.isc.org/community/rfcs/dns/**.

Структура заголовка в пакете DNS

Как показано на рис. 9.10, структура заголовка в пакете DNS несколько иная, чем в рассматривавшихся ранее типах пакетов. В заголовке пакета DNS имеются перечисленные ниже поля.

Система доменных имен (DNS)					
Смещение	Октет	0	1	2	3
Октет	Бит	0–7	8–15	16–23	24–31
0	0	Идентификационный номер DNS		З/О | Код запроса | АО | УС | ТР | РД	З | Код ответа
4	32	Число запросов		Число ответов	
8	64	Число записей на авторитетном сервере доменных имен		Число дополнительных записей	
12+	96+	Раздел запросов		Раздел ответов	
		Раздел полномочий		Раздел дополнительных сведений	

Рис. 9.10. Структура заголовка в пакете DNS

- **Идентификационный номер DNS (DNS ID Number).** Служит для связывания DNS-запросов с DNS-ответами.
- **Запрос/Ответ, ЗО (Query/Response, QR).** Обозначает, содержит ли пакет DNS-запрос или DNS-ответ.
- **Код запроса (OpCode).** Определяет тип запроса, содержащийся в сообщении.
- **Авторитетный ответ, АО (Authoritative Answers, AA).** Если значение этого поля установлено в ответном пакете, это означает, что ответ поступил от авторитетного сервера доменных имен, обслуживающего данный домен.
- **Усечение, УС (Truncation, TC).** Обозначает, что ответ был усечен, поскольку он оказался слишком большим и не поместился в пакет.
- **Требуется рекурсия, ТР (Recursion Desired, RD).** Если значение этого поля установлено в запросе, это означает, что DNS-клиент делает рекурсивный запрос, если на целевом сервере доменных имен отсутствует запрашиваемая информация.
- **Рекурсия доступна, РД (Recursion Available, RA).** Если значение этого поля установлено в ответе, это означает, что рекурсивные запросы поддерживаются на сервере доменных имен.
- **Зарезервировано, З (Reserved, Z).** По стандарту RFC 1035 в этом поле должны быть установлены все нули. Но оно иногда используется в качестве расширения поля RCode.
- **Код ответа (Response Code, RCode).** Служит в DNS-ответах для указания на наличие любых ошибок.
- **Число запросов (Question Count).** Обозначает число записей в разделе запросов.
- **Число ответов (Answer Count).** Обозначает число записей в разделе ответов.
- **Число записей на авторитетном сервере доменных имен (Name Server (Authority) Record Count).** Обозначает число записей ресурсов на сервере доменных имен в разделе начальной записи зоны (Start of Authority, SOA).
- **Число дополнительных записей (Additional Records Count).** Обозначает число записей ресурсов в разделе дополнительных сведений.
- **Раздел запросов (Questions Section).** Это раздел переменного размера, содержащий один или несколько запросов сведений, посылаемых DNS-серверу.

- **Раздел ответов (Answers Section).** Это раздел переменного размера, содержащий одну или несколько записей ресурсов, посылаемых в ответ на запросы.
- **Раздел полномочий (Authority Section).** Это раздел переменного размера, содержащий записи ресурсов, указывающие на авторитетные серверы доменных имен, которые могут быть использованы для продолжения процесса преобразования имен.
- **Раздел дополнительных сведений (Additional Information Section).** Это раздел переменного размера, содержащий записи ресурсов, хранящие дополнительные сведения, связанные с запросом, хотя их совсем не обязательно посылать в ответ за запрос.

Простой DNS-запрос

Файл перехвата **dns_query_response.pcapng**

Протокол DNS функционирует в режиме "запрос-ответ". В частности, клиент, желающий преобразовать DNS-имя в IP-адрес, посылает *запрос* DNS-серверу, а тот — *ответ* с запрашиваемой информацией. В простейшей форме для выполнения этого процесса требуются два пакета, как демонстрируется в примере из файла перехвата **dns_query_response.pcapng**.

Как показано на рис. 9.11, первый пакет содержит DNS-запрос, посылаемый клиентом, находящимся по адресу **192.168.0.114**, серверу, принимающему этот запрос по адресу **205.152.37.23** через стандартный порт **53**, применяемый в протоколе DNS. Приступив к анализу заголовков в данном пакете, можно обнаружить, что в протоколе DNS в качестве транспортного используется протокол UDP ❶.

В части данного пакета, относящейся к протоколу DNS, можно обнаружить, что мелкие поля, расположенные ближе к началу пакета, сведены средствами Wireshark в единый раздел **Flags**. Разверните этот раздел, чтобы увидеть сообщение, которое на самом деле является стандартным запросом ❷, не усечено и требует рекурсии, о которой речь пойдет далее. В данном случае определен лишь один запрос, который можно обнаружить, развернув раздел **Queries** (Запросы). Там вы увидите, что запрашивается преобразование доменного имени **wireshark.org** в межсетевой адрес (**IN**) хоста (тип **A**) ❸. По существу, в данном пакете запрашивается следующее: "Какой IP-адрес связан с доменом `wireshark.org`?"

Ответ на этот запрос содержится в пакете 2, как показано на рис. 9.12. Этот пакет имеет свой идентификационный номер ❶, по которому можно судить, что он содержит правильный ответ на исходный запрос.

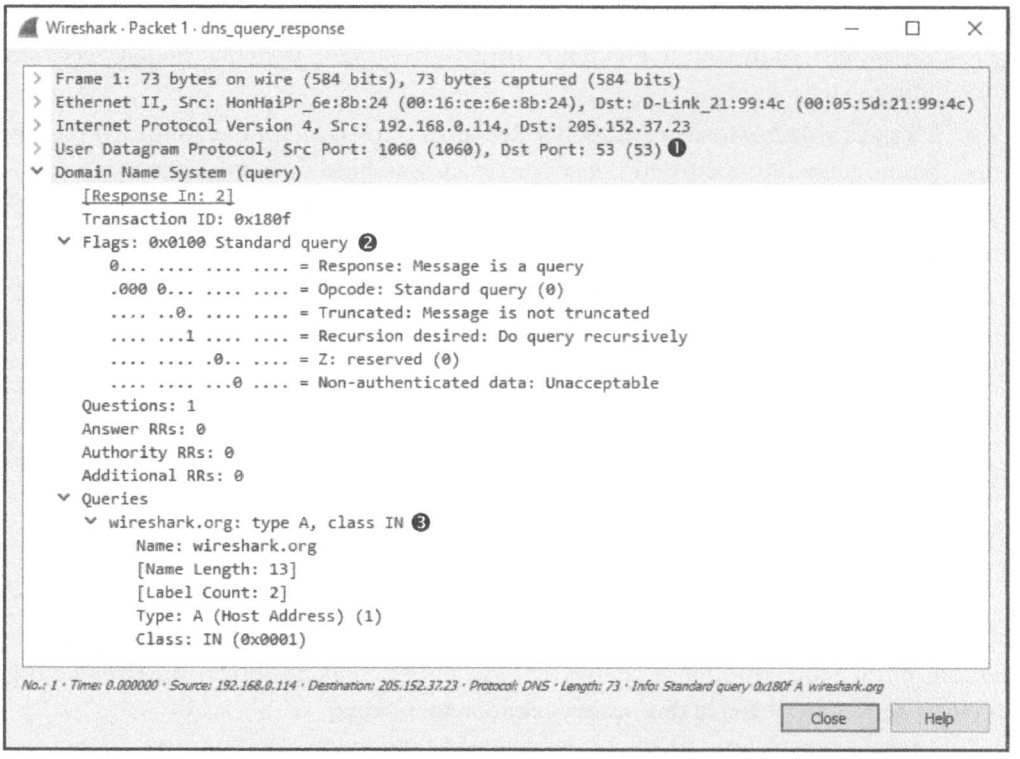


Рис. 9.11. Пакет с DNS-запросом

Заглянув в раздел **Flags**, можно убедиться, что это действительно ответ и что рекурсия доступна, если она требуется ❷. Данный пакет содержит лишь один запрос и одну запись ресурсов ❸, поскольку он включает в себя первоначальный запрос вместе с ответом на него. Доменному имени **wireshark.org** соответствует IP-адрес **128.121.50.122** ❹. Получив эти сведения, клиент может теперь создать IP-пакеты и приступить к обмену данными с сервером, имеющим доменное имя **wireshark.org**.

Типы запросов по протоколу DNS

В полях Type, используемых в запросах и ответах по протоколу DNS, указывается тип записи ресурсов, запрашиваемых или сообщаемых в ответ. Некоторые из наиболее употребительных типов записей ресурсов перечислены в табл. 9.2. Эти типы можно наблюдать как в обычном сетевом трафике, так и в примерах из данной книги. (В табл. 9.2 приведен лишь краткий перечень типов записей ресурсов.) Чтобы ознакомиться со всеми типами записей ресурсов в DNS, посетите веб-страницу по адресу `http://www.iana.org/assignments/dns-parameters/`.)

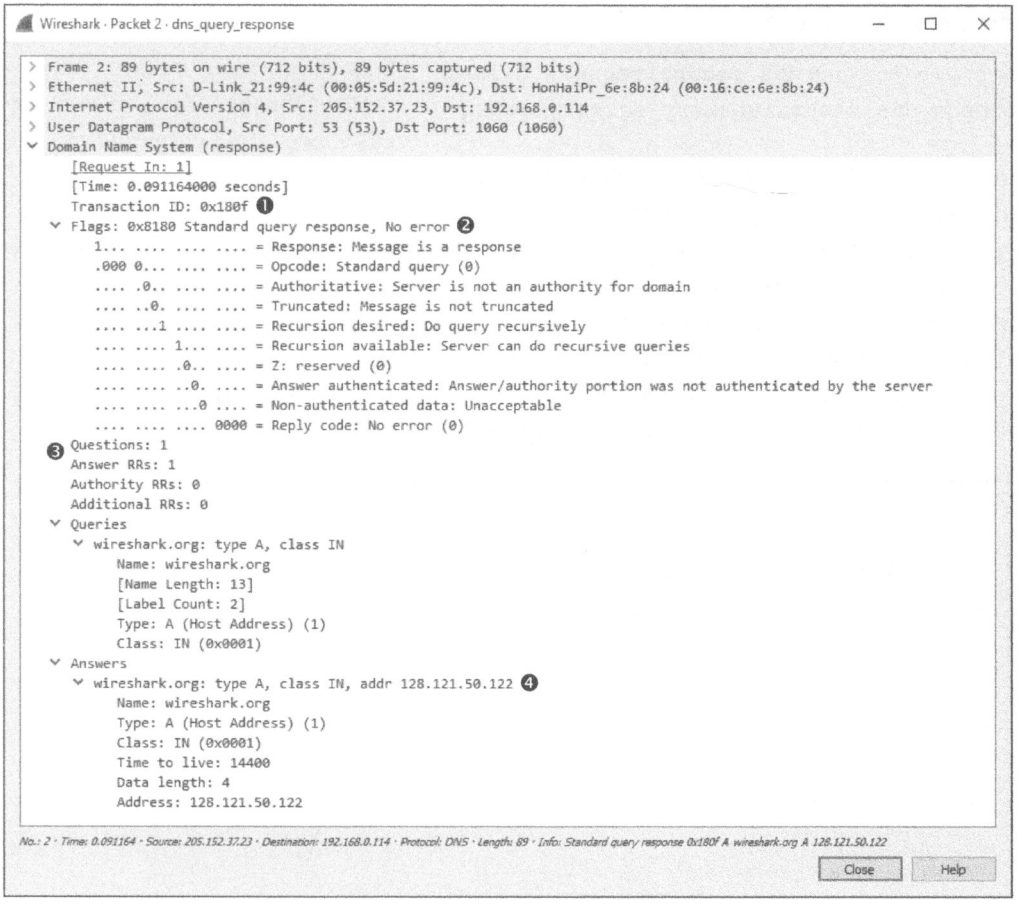


Рис. 9.12. Пакет с DNS-ответом

Таблица 9.2. Наиболее употребительные типы записей ресурсов в DNS

Значение	Тип	Описание
1	A	IPv4-адрес хоста
2	NS	Авторитетный сервер доменных имен
5	CNAME	Каноническое имя псевдонима
15	MX	Обмен почтой
16	TXT	Текстовая строка
28	AAAA	IPv6-адрес хоста
251	IXFR	Инкрементный перенос зоны
252	AXFR	Полный перенос зоны

Рекурсия в DNS

Файлы перехвата **dns_recursivequery_client.pcapng, dns_recursivequery_server.pcapng**

В силу иерархического характера межсетевой структуры DNS все DNS-серверы должны быть в состоянии связываться друг с другом, чтобы получать ответы на запросы, отправляемые клиентами. Если еще можно предположить, что внутреннему DNS-серверу известно, какой именно IP-адрес назначен локальному серверу во внутренней сети предприятия, то вряд ли следует ожидать, что ему известен IP-адрес, назначенный веб-серверу компании Google или Dell.

Когда одному DNS-серверу требуется найти IP-адрес, он запрашивает другой DNS-сервер от имени клиента, делающего первоначальный запрос, действуя, по существу, как клиент. И этот процесс называется *рекурсией*.

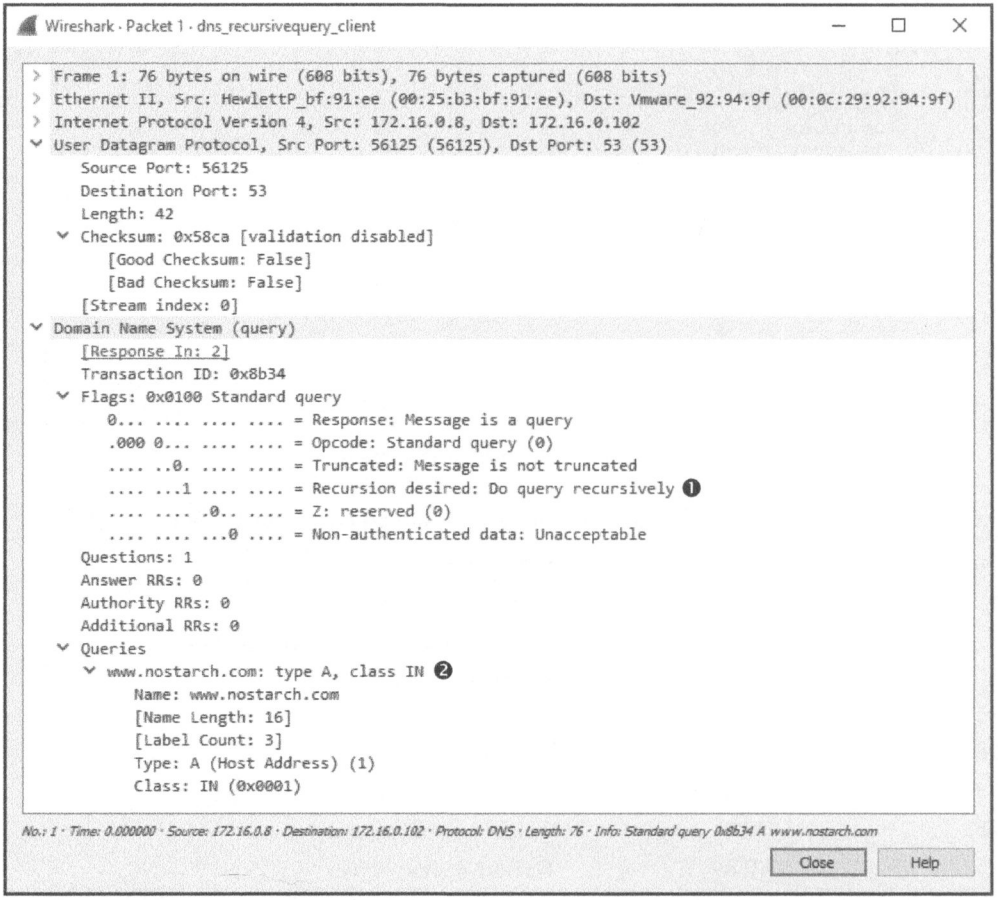


Рис. 9.13. DNS-запрос с установленным флагом запроса рекурсии

Чтобы посмотреть, каким образом происходит процесс рекурсии с точки зрения DNS-клиента и DNS-сервера, откройте сначала файл перехвата `dns_recursivequery_client.pcapng`. Этот файл содержит перехваченный трафик DNS-клиента, состоящий из двух пакетов. Первый пакет содержит первоначальный запрос, отправленный DNS-клиентом, находящимся по адресу **`172.16.0.8`**, DNS-серверу, расположенному по адресу **`172.16.0.102`** (рис. 9.13).

Развернув часть данного пакета, относящуюся к протоколу DNS, вы увидите, что это стандартный запрос записи типа **`A`** о DNS-имени **`www.nostarch.com`** ❷. Чтобы исследовать данный пакет дальше, разверните раздел Flags, где указано, что требуется рекурсия ❶.

Второй пакет, как и следовало ожидать, содержит ответ на первоначальный запрос (рис. 9.14).

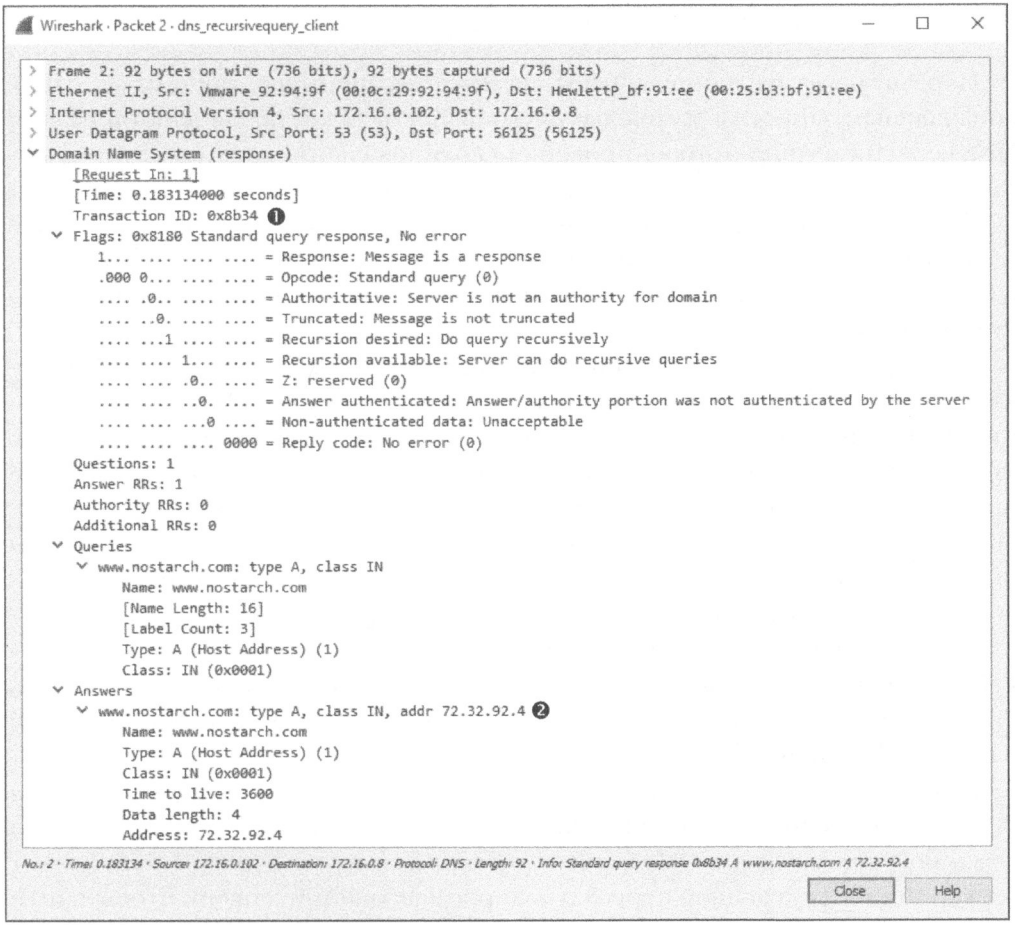


Рис. 9.14. Ответ на DNS-запрос

Идентификатор транзакции этого пакета совпадает с аналогичным идентификатором в первоначальном запросе ❶. А поскольку ошибки отсутствуют, то в конечном итоге клиент получает запись ресурсов типа **A**, связанную с доменным именем **www.nostarch.com** ❷.

Ответ на рассматриваемый здесь запрос был получен посредством рекурсии, которая была зафиксирована при прослушивании трафика DNS-сервера, как демонстрируется в примере из файла перехвата `dns_recursivequery_server.pcapng`. Этот файл содержит сетевой трафик, перехваченный на локальном DNS-сервере, когда был инициирован запрос (рис. 9.15).

No.	Time	Source	Destination	Protocol	Length	Info
→ 1	0.…	172.16.0.8	172.16.0.102	DNS	76	Standard query 0x8b34 A www.nostarch.com
2	0.…	172.16.0.102	4.2.2.1	DNS	76	Standard query 0xf34d A www.nostarch.com
3	0.…	4.2.2.1	172.16.0.102	DNS	92	Standard query response 0xf34d A www.nostarch.com A 72.32.92.4
← 4	0.…	172.16.0.102	172.16.0.8	DNS	92	Standard query response 0x8b34 A www.nostarch.com A 72.32.92.4

Рис. 9.15. DNS-рекурсия с точки зрения сервера

Первый пакет из данного файла оказывается таким же, как и пакет с первоначальным запросом из предыдущего файла перехвата. На данной стадии DNS-сервер получил запрос, проверил свою локальную базу данных и выяснил, что он не знает ответа на запрос, какой именно IP-адрес соответствует DNS-имени **www.nostarch.com**. Но поскольку исходный пакет был отправлен с установленным флагом Recursion Desired, то DNS-сервер может попросить другой DNS-сервер ответить на полученный запрос, как следует из содержимого второго пакета.

Во втором пакете DNS-сервер, находящийся по адресу **172.16.0.102**, передает новый запрос серверу, расположенному по адресу **4.2.2.1** ❶. Первый из них настроен на пересылку запросов восходящего потока второму, как показано на рис. 9.16. Данный запрос зеркально отображает первоначальный запрос, по существу, превращая посылающий его DNS-сервер в клиента. В то же время можно сказать, что это новый запрос, поскольку идентификационный номер транзакции здесь отличается от идентификационного номера в предыдущем файле перехвата ❷.

Как только данный пакет будет принят сервером, находящимся по адресу **4.2.2.1**, локальный DNS-сервер получит от него ответ, как показано на рис. 9.17. Получив этот ответ, локальный DNS-сервер передаст четвертый и пятый пакеты DNS-клиенту с запрашиваемой информацией.

В данном примере демонстрируется лишь один уровень рекурсии, но она может происходить неоднократно по единственному DNS-запросу. И хотя здесь получен ответ от DNS-сервера, находящегося по адресу **4.2.2.1**, этот сервер мог бы рекурсивно переслать запрос еще одному серверу, чтобы найти ответ на запрос. Простой запрос может распространиться по всему миру, прежде чем удастся найти правильный ответ на него. Процесс обработки рекурсивного DNS-запроса наглядно показан на рис. 9.18.

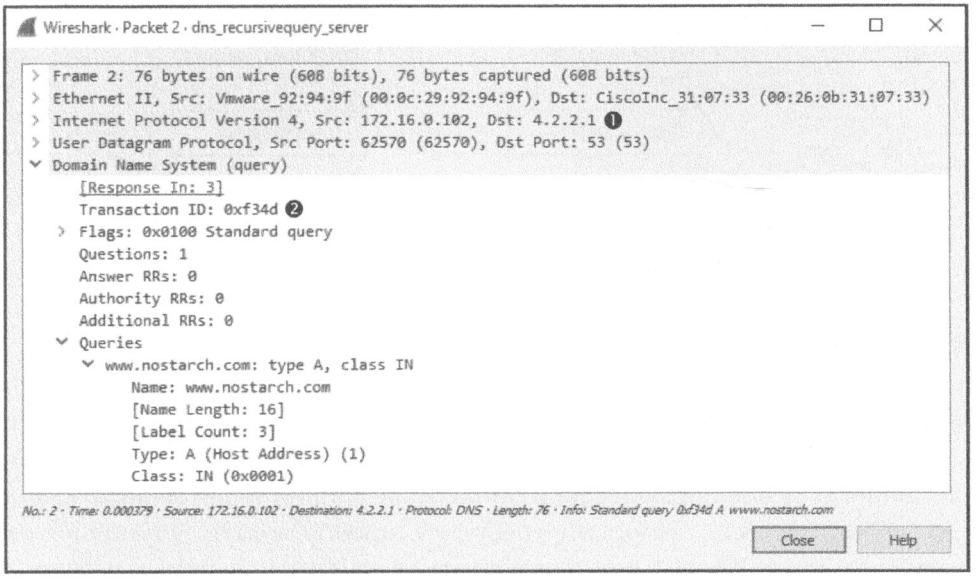


Рис. 9.16. Рекурсивный DNS-запрос

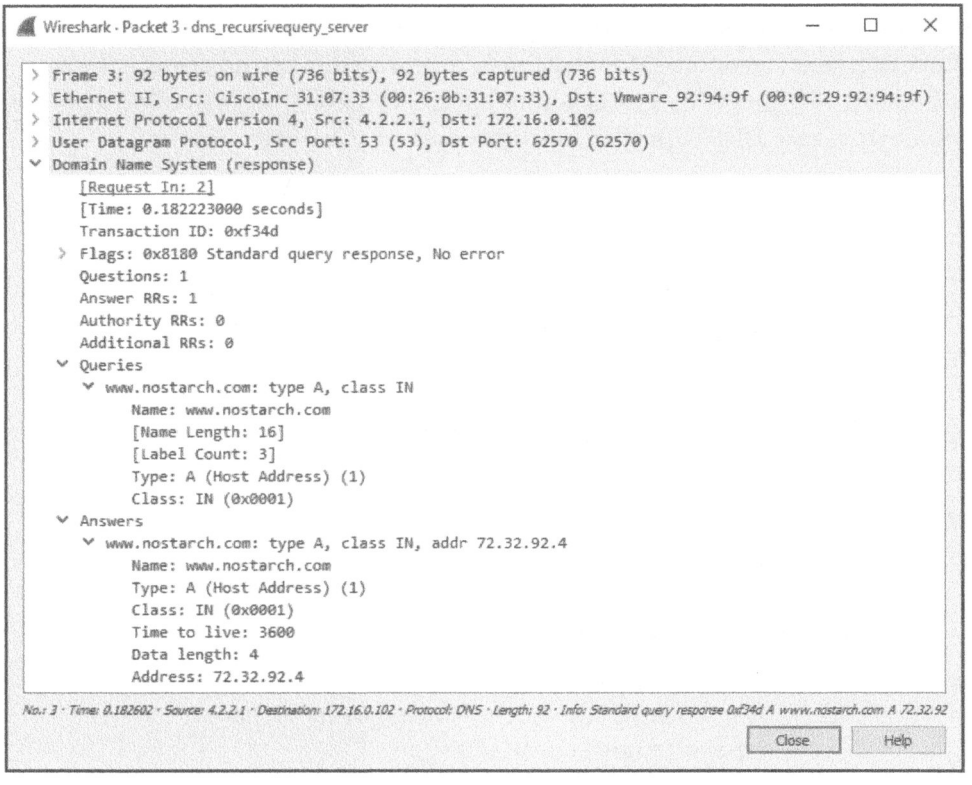


Рис. 9.17. Ответ на рекурсивный DNS-запрос

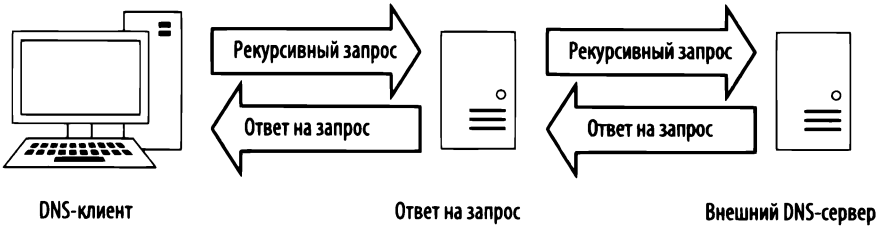


Рис. 9.18. Процесс обработки рекурсивного DNS-запроса

Перенос DNS-зон

Файл перехвата **dns_axfr.pcapng**

DNS-зона – это пространство имен (или группа), вести которое поручено DNS-серверу. Например, у компании Emma's Diner может быть один DNS-сервер, отвечающий за доменное имя **emmasdiner.com**. В этом случае устройства в самой компании и за ее пределами, которым требуется преобразовать доменное имя **emmasdiner.com** в IP-адрес, должны связаться с ее DNS-сервером, отвечающим за данную зону. При расширении компании Emma's Diner может быть введен второй DNS-сервер для ведения только той части пространства имен, которая относится к электронной почте и имеет доменное имя **mail.emmasdiner.com**. Таким образом, второй DNS-сервер будет отвечать за почтовый субдомен. По мере необходимости для ведения других субдоменов могут быть введены дополнительные DNS-серверы, как показано на рис. 9.19.

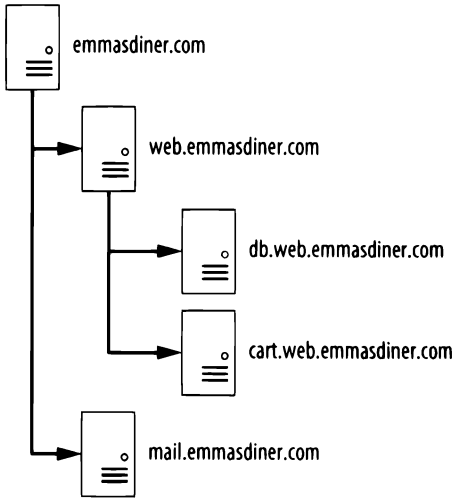


Рис. 9.19. Разделение ответственности за пространства имен среди DNS-зон

Перенос зоны происходит в том случае, если данные зоны передаются между двумя устройствами — как правило, из соображений резервирования. Например, в организациях с несколькими DNS-серверами сетевые администраторы зачастую настраивают вспомогательный DNS-сервер на хранение копии данных зоны из основного DNS-сервера на тот случай, если основной сервер окажется недоступным. Имеются следующие виды переноса зон.

- **Полный перенос зоны (AXFR).** К этому виду относятся переносы, при которых вся зона пересылается между устройствами.
- **Инкрементный перенос зоны (IXFR).** К этому виду относятся переносы, при которых пересылается лишь часть данных зоны.

В файле перехвата `dns_axfr.pcapng` содержится пример полного переноса зоны между хостами, находящимися по адресам **`172.16.16.164`** и **`172.16.16.139`**. На первый взгляд, содержимое этого файла может удивить вас, поскольку вместо пакетов UDP вы обнаружите в нем пакеты TCP. И хотя в протоколе DNS используется протокол UDP, для решения некоторых задач, в том числе переноса зон, в нем применяется и протокол TCP, поскольку он обеспечивает более надежную передачу данных. Поэтому первые три пакета из данного файла перехвата служат для трехэтапного процесса установки связи по протоколу TCP.

Перенос зоны между хостами, находящимися по адресам **`172.16.16.164`** и **`172.16.16.139`**, начинается с четвертого пакета. Этот пакет не содержит никакой информации о протоколе DNS. Он обозначен как "TCP-сегмент восстановленного блока PDU", поскольку данные, пересылаемые по запросу на перенос зоны, на самом деле пересылаются в нескольких пакетах и, в частности, в пакетах 4 и 6. А пакет 5 содержит подтверждение приема пакета 4. Эти пакеты анализируются и отображаются в Wireshark именно в таком порядке ради большей их удобочитаемости. Для целей данного примера пакет 6 можно отнести к запросу на полный перенос DNS-зоны, как показано на рис. 9.20.

Запрос на перенос зоны является стандартным запросом ❶, но вместо единственного типа записи в нем запрашивает тип AXFR ❷. Это означает, что по данному запросу требуется получить всю DNS-зону из сервера. На это сервер ответит записями зоны в пакете 7, как показано на рис. 9.21. Как видите, перенос зоны включает в себя немало данных, а ведь это едва ли не самый простой его пример! Как только перенос зоны завершится, начнется процесс разрыва связи по протоколу TCP, представленный в самом конце данного файла перехвата.

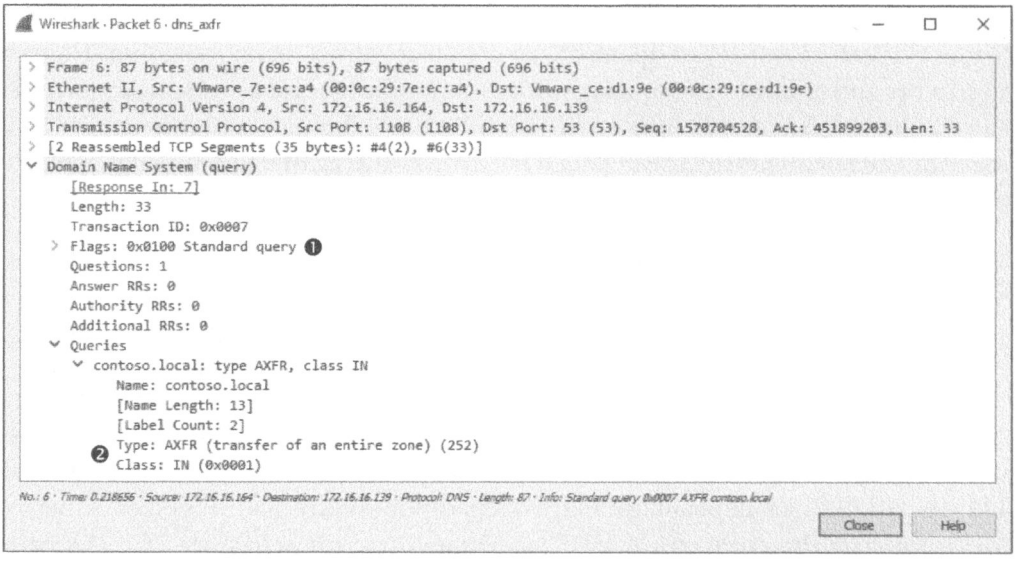


Рис. 9.20. Запрос на полный перенос DNS-зоны

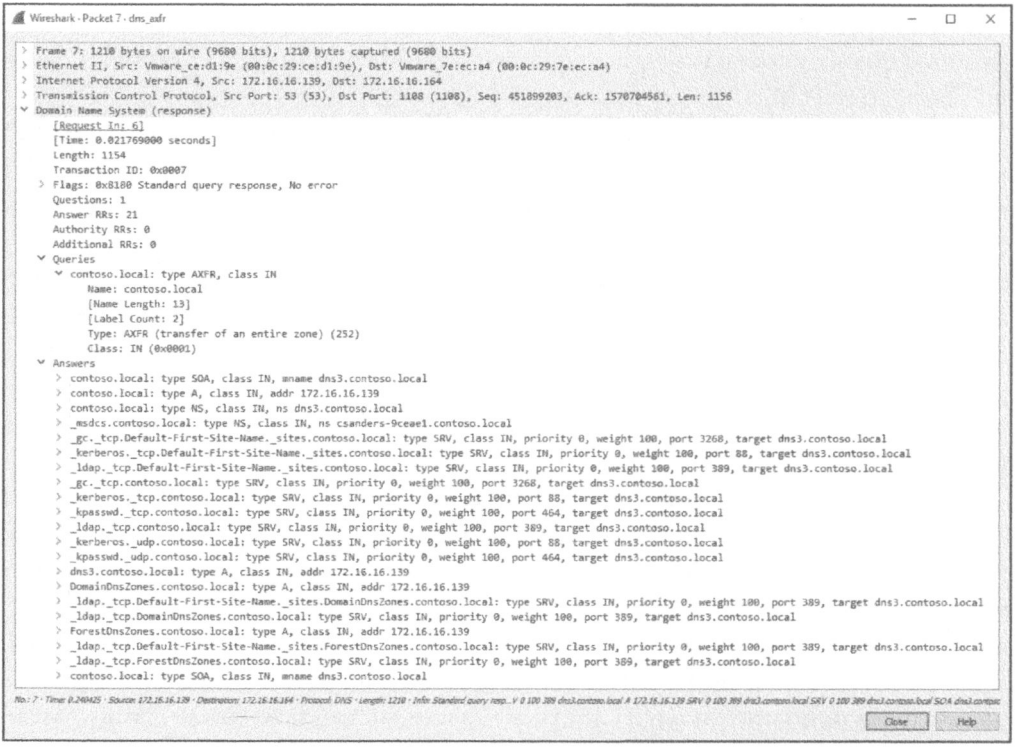


Рис. 9.21. Процесс полного переноса DNS-зоны

ПРЕДУПРЕЖДЕНИЕ *Данные, содержащиеся в переносимой зоне, могут представлять немалую опасность, если они окажутся в руках злоумышленников. Например, сделав запрос единственному DNS-серверу, можно получить в ответ всю инфраструктуру сети.*

Протокол передачи гипертекста (HTTP)

Протокол HTTP (Hypertext Transfer Protocol – протокол передачи гипертекста) служит в качестве механизма доставки содержимого во Всемирной паутине, давая веб-браузерам возможность подключаться к веб-серверам для просмотра веб-страниц. В большинстве организаций протокол HTTP представляет наибольшую долю трафика, передаваемого по сети. Всякий раз, когда вы осуществляете поиск в Google, посылаете сообщение в Твитер или проверяете турнирную таблицу НБА на веб-сайте по адресу `http://www.espn.com/`, вы, по существу, пользуетесь протоколом HTTP.

Мы не будем рассматривать структуры заголовков в пакетах для передачи данных по протоколу HTTP, поскольку имеется так много различных реализаций этого протокола, что структура заголовков в этих пакетах может существенно отличаться. В силу подобных различий оставляем вам в качестве упражнения самостоятельно разобраться в структуре HTTP-заголовков. Вместо этого мы рассмотрим примеры практического применения протокола HTTP и, в частности, извлечение и публикацию содержимого.

Просмотр веб-страниц с помощью протокола HTTP

Файл перехвата **`http_google.pcapng`**

Протокол HTTP чаще всего применяется для просмотра веб-страниц, получаемых от веб-сервера, с помощью браузера. В примере из файла перехвата `http_google.pcapng` демонстрируется подобная разновидность передачи данных по протоколу HTTP с использованием TCP в качестве протокола транспортного уровня. Обмен данными между клиентом, находящимся по адресу **`172.16.16.128`**, и веб-сервером компании Google, расположенным по адресу **`74.125.95.104`**, начинается с трехэтапного процесса установки связи по протоколу TCP.

Как только связь будет установлена, первый пакет, передаваемый клиентом серверу, будет помечен как пакет протокола HTTP (рис. 9.22).

Этот пакет HTTP доставляется по протоколу TCP через порт **`80`** сервера ❶. Он считается стандартным портом для обмена данными по протоколу HTTP, хотя для этой цели нередко применяются и другие порты, в том числе с номерами **`8080`**, **`8888`** и т.п.

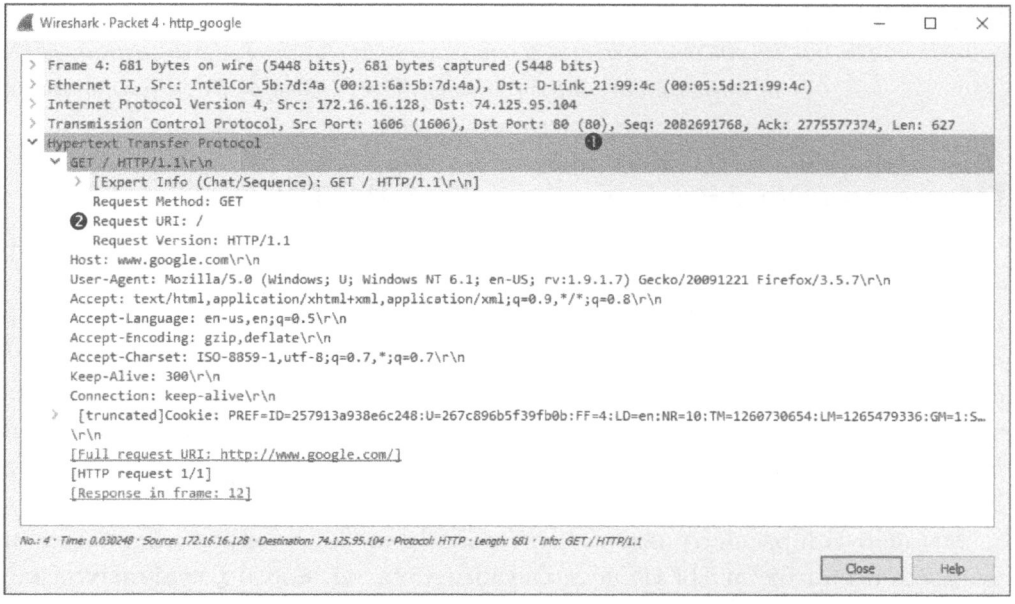


Рис. 9.22. Первоначальный пакет с HTTP-запросом по методу **GET**

Пакеты HTTP распознаются по одному из восьми методов запроса, определенных в спецификации протокола HTTP версии 1.1 (подробнее об этом см. по адресу `http://www.iana.org/assignments/http-methods/http-methods.xhtml`). Метод запроса обозначает действие, которое требуется выполнить стороне, передающей пакеты, на стороне их приема. Как показано на рис. 9.22, в данном пакете указан метод `GET` для запроса URI (Uniform Resource Indicator – универсальный указатель ресурсов) в виде **/**, а также версия протокола **`HTTP/1.1`** для данного запроса ❷. Эти сведения сообщают, что клиент посылает запрос для загрузки (по методу `GET`) корневого каталога (**/**) на веб-сервере, где применяется версия 1.1 протокола HTTP.

Далее хост посылает сведения о себе веб-серверу. Эти сведения включают в себя применяемый браузер (поле **`User-Agent`** – пользовательский агент), принятые в браузере языки (поле **`Accept-Languages`**) и сведения из cookie-файла, которые отображаются в нижней части окна перехвата. Сервер может воспользоваться этими сведениями с целью выяснить, какие именно данные следует возвратить клиенту, чтобы обеспечить совместимость.

Когда сервер принимает HTTP-запрос по методу `GET` в пакете 4, он отвечает подтверждающим пакетом TCP `ACK` и начинает передавать запрашиваемые данные в пакетах 6–11. Протокол HTTP служит только для обмена командами уровня приложений между клиентом и сервером. Почему же все эти HTTP-пакеты появляются как пакеты TCP под заголовком протокола в списке пакетов? Когда начинается передача данных, эти пакеты будут обозначены в окне

Wireshark со списком пакетов как TCP, а не HTTP, поскольку в них отсутствуют заголовки с HTTP-запросами и ответами. Следовательно, когда происходит передача данных в столбце **Protocol**, появится обозначение TCP вместо HTTP. Тем не менее это часть процесса обмена данными по протоколу HTTP.

Данные посылаются сервером в пакетах 6 и 7, а подтверждение от клиента — в пакете 8. Дополнительные данные посылаются в пакетах 9 и 10, а очередное подтверждение — в пакете 11, как показано на рис. 9.23. Все эти пакеты представлены в Wireshark в виде сегментов TCP, а не пакетов HTTP, хотя протокол HTTP по-прежнему отвечает за их передачу.

No.	Time	Source	Destination	Protocol	Length	Info
6	0...	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
7	0...	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
8	0...	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082692395 Ack=2775580186 Win=16872 Len=0
9	0...	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
10	0...	74.125.95.104	172.16.16.128	TCP	156	[TCP segment of a reassembled PDU]
11	0...	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082692395 Ack=2775581694 Win=16872 Len=0

Рис. 9.23. Передача данных между браузером клиента и веб-сервером по протоколу TCP

Как только данные будут переданы, поток данных будет воссоздан в Wireshark для просмотра, как показано на рис. 9.24.

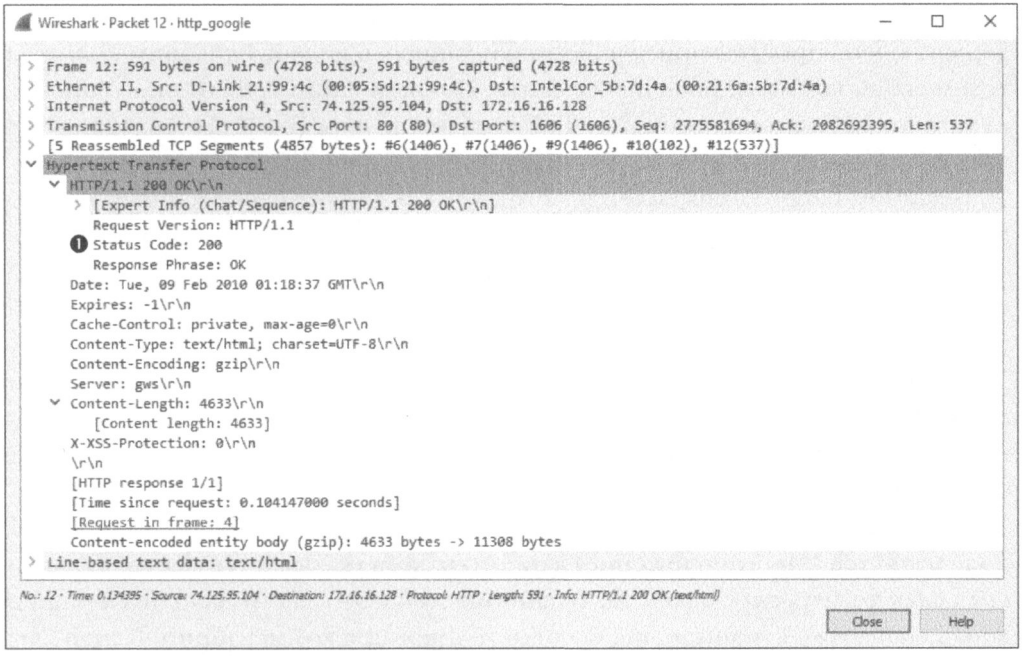


Рис. 9.24. Окончательный HTTP-пакет с кодом состояния **200**

ПРИМЕЧАНИЕ *Во многих примерах при просмотре списка пакетов вам не удастся увидеть удобочитаемые данные в формате HTML, поскольку эти данные уплотнены в формате* `gzip` *ради повышения пропускной способности. На это указывается в поле* **`Content-Encoding`** *(Кодирование содержимого) в HTTP-ответе от веб-сервера. Эти данные становятся удобочитаемыми лишь в том случае, если просматривается поток полностью декодированных данных.*

В протоколе HTTP применяется целый ряд предопределенных кодов ответа для обозначения результатов, получаемых в после выполнения метода запроса. В данном примере наблюдается пакет с кодом состояния **200** ❶, который обозначает успешное завершение метода запроса. В этот пакет входят также отметка времени, дополнительные сведения о кодировании содержимого и параметры настройки веб-сервера. Как только клиент получит этот пакет, транзакция завершится.

Публикация данных по протоколу HTTP

Файл перехвата **`http_post.pcapng`**

Итак, рассмотрев процесс получения данных от веб-сервера, обратим внимание на процесс отправки данных серверу. В частности, файл перехвата `http_post.pcapng` содержит очень простой пример выгрузки данных в виде публикации пользователем своих комментариев на веб-сайте. По окончании трехэтапного процесса установки связи по протоколу TCP клиент (по адресу **`172.16.16.128`**) посылает HTTP-пакет веб-серверу (**`69.163.176.56`**), как показано на рис. 9.25.

В данном пакете применяется метод `POST` ❶ для отправки данных веб-серверу на обработку. В применяемом здесь методе `POST` указывается URI (`/wp-comments-post.php`) ❷ и версия протокола HTTP (`HTTP/1.1`). Чтобы увидеть содержимое публикуемых данных, разверните раздел HTML Form URL Encoded (Закодированные данные HTML-формы) пакета ❸.

Как только данные будут переданы в описанном выше пакете с запросом по методу `POST`, далее будет отправлен пакет с подтверждением `ACK`. Как показано на рис. 9.26, сервер ответит пакетом 6, передавая код ответа **`302`** ❶, который обозначает "найдено".

Код ответа **`302`** обычно обозначает переадресацию по протоколу HTTP. В поле **`Location`** (Местоположение) данного пакета указывается место, куда клиента следует переадресовать ❷. В данном случае это место находится на первоначальной веб-странице, где были опубликованы комментарии. Клиент выполняет новый запрос по методу `GET`, чтобы извлечь содержимое из нового места, откуда они посылаются сервером в ряде следующих пакетов. И, наконец, сервер передает код состояния **`200`**, и на этом обмен данными завершается.

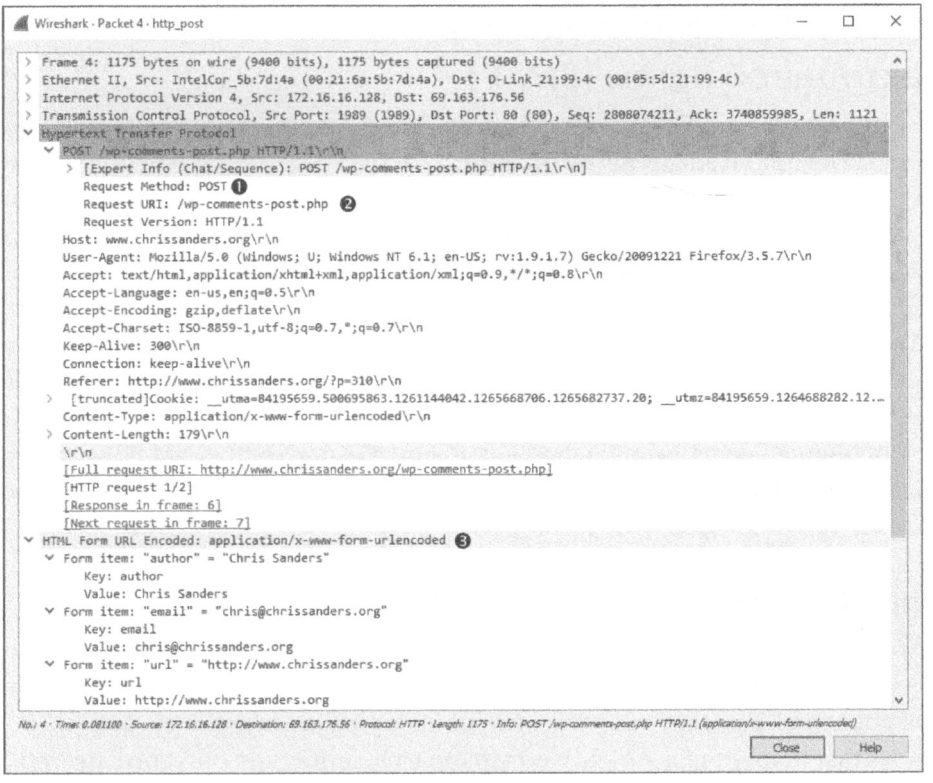


Рис. 9.25. HTTP-пакет с запросом по методу **POST**

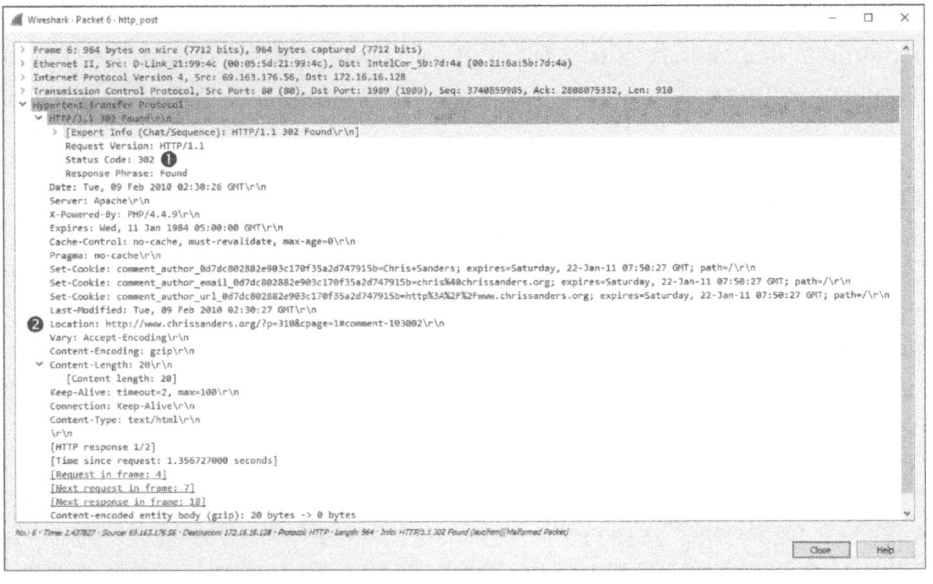


Рис. 9.26. HTTP-ответ с кодом состояния **302** *применяется для переадресации*

Простой протокол передачи электронной почты (SMTP)

Если просмотр веб-страниц считается наиболее распространенным видом деятельности пользователя в Интернете, то отправка и получение электронной почты, вероятно, стоит на втором месте. В качестве стандартного для передачи электронной почты на платформах вроде Microsoft Exchange и Postfix служит протокол SMTP (Simple Mail Transfer Protocol — простой протокол передачи электронной почты).

Аналогично протоколу HTTP, структура заголовка в пакете SMTP может меняться в зависимости от конкретной реализации ряда функциональных средств, поддерживаемых клиентом и сервером. В этом разделе рассматриваются основные функциональные возможности протокола SMTP, а также показывается, каким образом выглядит отправка электронной почты на уровне пакетов.

Отправка и получение электронной почты

Архитектура, лежащая в основе системы обмена электронной почтой, подобна структуре почтовой службы США. Написав письмо, вы запечатываете его в конверт и бросаете в почтовый ящик, почтальон извлекает его из почтового ящика и доставляет в свое почтовое отделение, где оно сортируется. Оттуда письмо доставляется в другой почтовый ящик того же почтового отделения, отвечающего за его доставку адресату. Письмо может пройти несколько местных почтовых отделений и даже центральные почтовые отделения, предназначенные исключительно для распределения почты по региональным почтовым отделениям. Такой поток почтовой информации наглядно показан на рис. 9.27.

Доставка электронной почты действует по тому же самому принципу, хотя при этом применяется несколько иная терминология. На уровне отдельного пользователя физический почтовый ящик заменяется цифровым почтовым ящиком, отвечающим за хранение и позволяющий отправлять и получать электронную почту. Доступ к этому почтовому ящику осуществляется с помощью *почтового агента пользователя* (Mail User Agent — MUA), представляющего собой клиентскую программу для работы с электронной почтой вроде Microsoft Outlook или Mozilla Thunderbird.

Когда вы отправляете сообщение, почтовый агент пользователя посылает его *агенту пересылки почты* (Mail Transfer Agent — MTA). Этот агент нередко называется почтовым сервером, наиболее распространенными примерами которого служат приложения Microsoft Exchange и Postfix. На платформе

Unix чаще всего используются серверные приложения Sendmail или Exim. Если электронная почта предназначается для того же самого домена, откуда она отправляется, то агент MTA может непосредственно доставить ее в почтовый ящик получателя без дополнительного обмена данными. Если же электронная почта посылается в другой домен, агенту MTA придется прибегнуть к услугам DNS, чтобы найти IP-адрес почтового сервера получателя, а затем передать ему сообщение. Следует, однако, иметь в виду, что почтовый сервер нередко состоит из других компонентов, наподобие агента по доставке почты (Mail Delivery Agent – MDA) или агента отправки почты (Mail Submission Agent – MSA). Но с точки зрения сетевого анализа нас больше интересуют понятия клиента и сервера. На рис. 9.28 наглядно показано, каким образом происходит пересылка электронной почты по протоколу SMTP. Ради простоты будем называть далее агента MUA почтовым клиентом и агента MTA почтовым сервером.

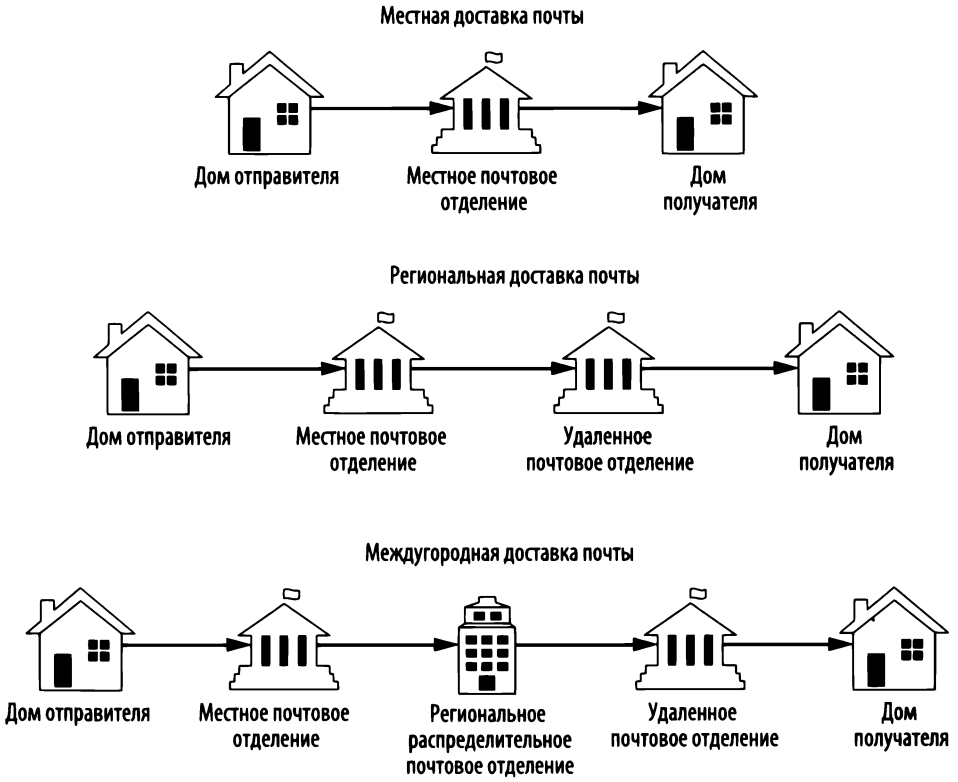


Рис. 9.27. Отправка письма через почтовое отделение

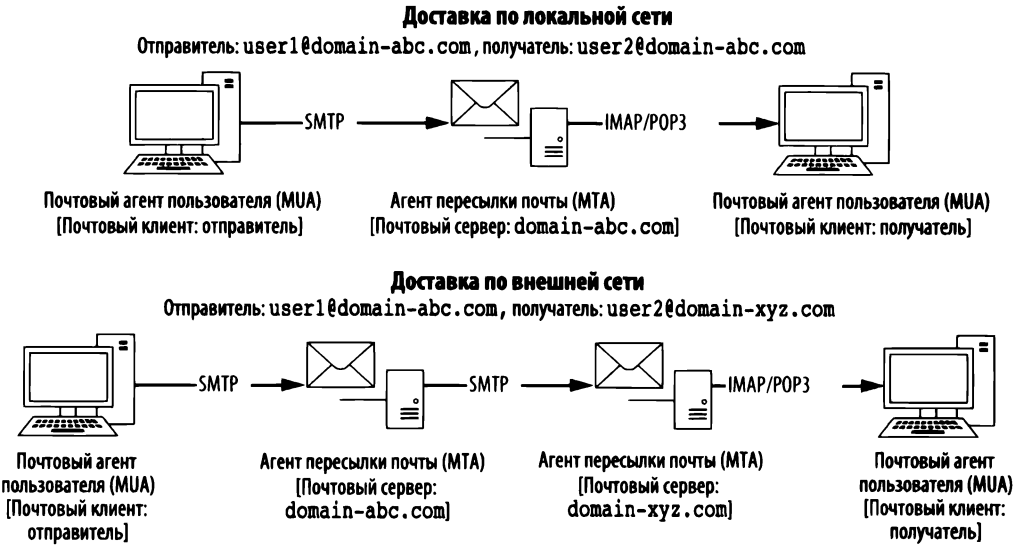


Рис. 9.28. Пересылка электронной почты по протоколу SMTP

Отслеживание сообщений электронной почты

Итак, имея основное представление о том, каким образом передаются сообщения, можно приступить к анализу пакетов, представляющих данный процесс. Начнем со сценария, наглядно показанного на рис. 9.29.

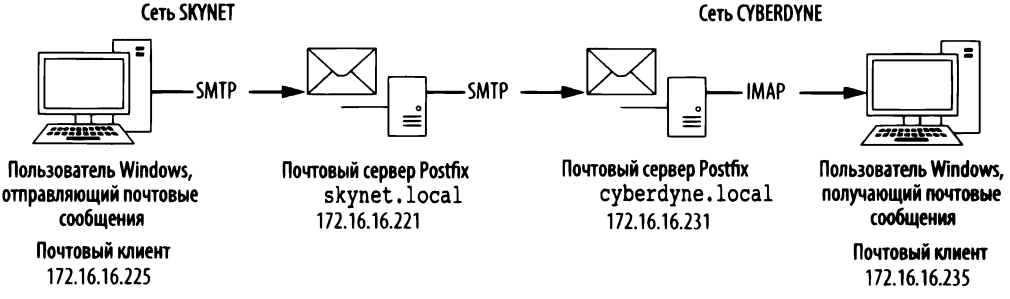


Рис. 9.29. Отслеживание сообщений электронной почты от их отправителя к получателю

Процесс отправки сообщений электронной почты в данном сценарии происходит в три этапа, как описано ниже.

1. Пользователь посылает сообщение со своего рабочего компьютера, находящегося по адресу **172.16.16.225**. Клиент электронной почты передает это сообщение по протоколу SMTP локальному серверу

электронной почты, расположенному по адресу **172.16.16.221** в домене **skynet.local**.

2. Локальный сервер электронной почты принимает сообщение и передает его по протоколу SMTP удаленному серверу электронной почты, расположенному по адресу **172.16.16.231** в домене **cyberdyne.local**.
3. Удаленный сервер электронной почты принимает сообщение и помещает его в соответствующий почтовый ящик. Клиент электронной почты на рабочем компьютере пользователя, находящейся по адресу **172.16.16.235**, извлекает данное сообщение из почтового ящика по протоколу IMAP.

Первый этап: от клиента к локальному серверу

Файл перехвата **mail_sender_client_1.pcapng**

Начнем постепенное рассмотрение данного процесса с анализа пакетов на первом этапе, представленном в файле перехвата mail_sender_client_1.pcapng. Сетевой трафик, перехваченный в этом файле, начинается с того момента, когда пользователь щелкает на кнопке Send (Отправить) в своем клиенте электронной почты, инициируя тем самым процедуру трехэтапного процесса установки связи по протоколу TCP между своим рабочим компьютером и локальным сервером электронной почты в пакетах 1–3.

ПРИМЕЧАНИЕ *Анализируя перехваченные пакеты, упоминаемые в этом разделе, не обращайте особого внимания на ошибки вроде* **ETHERNET FRAME CHECK SEQUENCE INCORRECT** *(Неправильная последовательность фреймов Ethernet). Они возникают вследствие перехвата сетевого трафика в лабораторной среде.*

Как только соединение будет установлено, в действие вступит протокол SMTP, начав передачу сообщения от пользователя к серверу. Каждый SMTP-запрос можно просмотреть на панели Packet Details, хотя для этого имеется и более простой способ. В частности, SMTP считается простым транзакционным протоколом, а в рассматриваемом здесь примере передается открытый текст, и поэтому можно легко проследить весь поток данных по протоколу TCP, чтобы просмотреть всю транзакцию в одном окне. С этой целью щелкните правой кнопкой мыши на любом пакете и выберите команду Follow⇨TCP Stream (Отслеживать⇨Поток TCP) из контекстного меню. Полученный в итоге поток данных приведен на рис. 9.30.

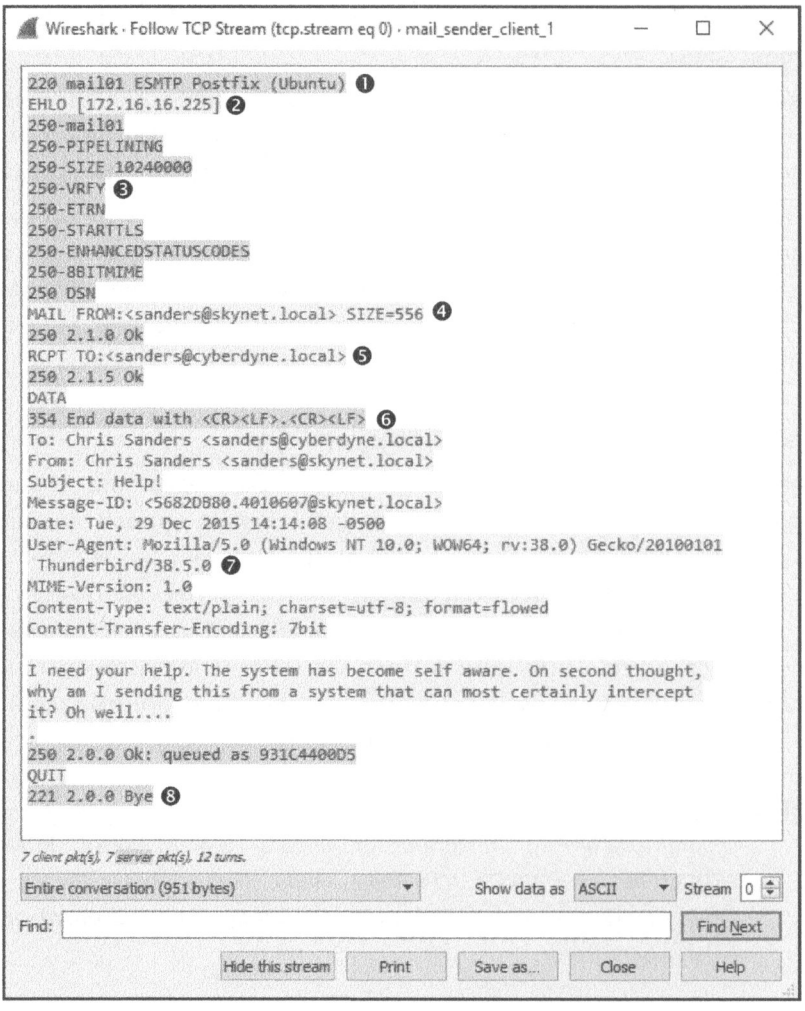


Рис. 9.30. Просмотр потока данных, передаваемых по протоколу TCP от клиента электронной почты к локальному серверу

Как только соединение будет установлено, сервер электронной почты отправит клиенту довольно длинный заголовок (так называемый баннер) почтовой службы в пакете 4, чтобы указать на свою готовность к получению команды. В данном случае он обозначает себя как почтовый сервер Postfix, работающий в версии Ubuntu операционной системы Linux ❶. Он также указывает на свою способность принимать команды *расширенного протокола SMTP* (ESMTP). Протокол ESMTP — это расширение спецификации SMTP, позволяющее выполнять дополнительные команды при пересылке почты.

Клиент электронной почты отвечает на это выдачей команды **EHLO** в пакете 5 ❷. Это приветственная команда, предназначенная для идентификации

хоста отправителя, когда поддерживается протокол ESMTP. Если же протокол ESMTP недоступен, клиент выдаст команду **HELO**, чтобы идентифицировать себя. В данном примере отправитель идентифицируется по его IP-адресу, хотя для этой цели может быть также использовано доменное имя.

В пакете 7 сервер отвечает списком команд, включая **VRFY**, **STARTTLS** и **SIZE 10240000** ❸. В этом списке отражены те команды, которые поддерживаются SMTP-сервером, и он предоставляется клиенту с целью известить его, какими командами разрешается пользоваться при передаче сообщения. Такое согласование функциональных средств происходит в начале каждой транзакции по протоколу SMTP перед отправкой сообщения. А передача сообщения начинается с пакета 8 и занимает почти весь остальной перехваченный трафик.

В протоколе SMTP используется простой набор команд и значений их параметров, посылаемых клиентом, после чего следует код ответа от сервера. Это очень похоже на работу таких протоколов, как HTTP и TELNET, и сделано ради простоты. Примеры запроса и ответа можно наблюдать в пакетах 8 и 9, где клиент посылает команду **MAIL** с параметром **FROM: <sanders@skynet.local> SIZE=556** ❹, на что сервер отвечает кодом состояния **250** с параметром **2.1.0 Ok**, обозначающим успешное завершение запрашиваемого почтового действия. Здесь клиент обозначает адрес электронной почты отправителя и длину сообщения, а в ответ сервер извещает, что эти данные получены и вполне приемлемы. Аналогичная транзакция происходит снова в пакетах 10 и 11, где клиент выдает команду **RCPT** с параметром **TO: <sanders@cyberdyne.local>** ❺, а сервер отвечает очередным кодом состояния **250** с параметром **2.1.5 Ok**.

ПРИМЕЧАНИЕ *Если желаете ознакомиться со всеми доступными в SMTP командами и их параметрами, обратитесь по адресу* **https://www.iana.org/assignments/mail-parameters/mail-parameters.xhtml**. *А если вы хотите ознакомиться со всеми кодами ответа, то обратитесь адресу* **https://www.iana.org/assignments/smtp-enhanced-status-codes/smtp-enhanced-status-codes.xml**.

Остается лишь передать само сообщение. Клиент инициирует этот процесс в пакете 12, выдав команду **DATA**. На это сервер ответит кодом **354** с сообщением ❻, что он создал буфер для приема почтового сообщения и предлагает клиенту начать его передачу. В строке, содержащей код ответа **354** с сообщением, клиенту предлагается послать знак точки в отдельной строке (**<CR><LF>.<CR><LF>**), чтобы обозначить конец передачи. Почтовое сообщение передается открытым текстом, на что посылается код ответа, обозначающий удачный исход передачи. Обратите также внимание на дополнительные сведения, включаемые в текст почтового сообщения, в том числе дату, тип

содержимого и его кодировку, а также почтовый агент пользователя, связанный с передачей данного сообщения. Эти сведения сообщают, что для отправки данного сообщения конечный пользователь воспользовался почтовым приложением Mozilla Thunderbird ❼.

По окончании передачи клиент электронной почты разрывает соединение по протоколу SMTP, выдавая команду **QUIT** без параметров в пакете 18. На это сервер отвечает в пакете 19 кодом состояния **221** с параметром **2.0.0 Bye**, обозначающим, что канал передачи данных закрывается ❽. В итоге соединение по протоколу TCP корректно разрывается в пакетах 20–23.

Второй этап: от локального к удаленному серверу

Файл перехвата **mail_sender_server_2.pcapng**

Рассмотрим далее тот же самый сценарий с точки зрения локального сервера электронной почты, обслуживающего домен **skynet.local** по адресу **172.16.16.221**. Соответствующий сетевой трафик перехвачен в файле mail_sender_server_2.pcapng непосредственно на данном сервере. Как и следовало ожидать, первые 20 пакетов или около того зеркально отображают перехват сетевого трафика на первом этапе рассматриваемого здесь процесса, поскольку это такие же самые пакеты, перехваченные из другого источника.

Если бы отправляемое почтовое сообщение предназначалось для другого почтового ящика в домене **skynet.local**, то мы бы не увидели больше никакого сетевого трафика по протоколу SMTP. Вместо этого мы обнаружили бы только извлечение почтового сообщения клиентом электронной почты по протоколу POP3 или IMAP. Но поскольку данное сообщение предназначено для домена **cyberdyne.local**, локальный SMTP-сервер должен передать полученное сообщение удаленному SMTP-серверу, ответственному за этот домен. И этот процесс начинается в пакете 22 с процедуры трехэтапного процесса установки связи по протоколу TCP между локальным почтовым сервером, находящимся по адресу **172.16.16.221**, и удаленным почтовым сервером, расположенным по адресу **172.16.16.231**.

ПРИМЕЧАНИЕ *В реальном сценарии сервер электронной почты обнаруживает другой почтовый сервер, используя специальный тип записи DNS* MX *(mail exchange), ответственной за обмен почтой. Но данный сценарий был воссоздан в лабораторной среде, где IP-адрес удаленного сервера электронной почты был предварительно настроен на локальном сервере, и поэтому данный сетевой трафик здесь не будет виден. Если вы проводите диагностику доставки электронной почты, то должны принимать во внимание не только проблемы, связанные с конкретным протоколом электронной почты, но и с протоколом DNS.*

Как только будет установлено соединение, в панели Packet List можно обнаружить, что для доставки почтового сообщения к удаленному серверу применяется протокол SMTP. Этот диалог лучше наблюдать, проследив поток данных по протоколу TCP для выполнения транзакции, как показано на рис. 9.31. Если потребуется помощь в изоляции данного соединения, примените фильтр, указав выражение **`tcp.stream == 1`** на панели фильтров.

Рис. 9.31. Просмотр потока данных по протоколу TCP между локальным и удаленным серверами электронной почты

Данная транзакция очень похожа на ту, что приведена на рис. 9.30. По существу, почтовое сообщение передается между серверами. Удаленный сервер идентифицирует себя как **`mail02`** ❶, а локальный сервер – как **`mail01`** ❷. Они обмениваются списком поддерживаемых команд ❸, после чего сообщение

полностью передается вместе с дополнительными заголовками из предыдущей транзакции, добавляемыми в начале почтового сообщения выше строки **To** (Кому) ❹. И все это происходит в промежутке между пакетами 27 и 35, после чего происходит разрыв связи по протоколу TCP для закрытия канала передачи данных.

В конечном счете для сервера вообще не имеет значения, поступает ли сообщение от клиента электронной почты или другого SMTP-сервера, поэтому здесь применяются те же самые правила и процедуры, кроме установленных ограничений на управление доступом. В действительности на локальном и удаленном серверах электронной почты могут и не поддерживаться одни и те же функциональные средства, а возможно, эти серверы базируются на совершенно разных платформах. Именно поэтому так важен первоначальный обмен данными по протоколу SMTP. Ведь он позволяет серверу получателя передать отправителю поддерживаемый на нем перечень функциональных средств. Если SMTP-клиенту или серверу известны функциональные средства, поддерживаемые на сервере получателя, то команды SMTP могут быть откорректированы таким образом, чтобы наиболее эффективно передать почтовое сообщение. Это дает возможность широко применять протокол SMTP для пересылки электронной почты при использовании любых клиентских и серверных решений, не особенно разбираясь в структуре сети получателя.

Третий этап: от удаленного сервера к удаленному клиенту

Файл перехвата **mail_receiver_server_3.pcapng**

Итак, почтовое сообщение достигло удаленного сервера, ответственного за доставку электронной почты в почтовые ящики, находящиеся в домене **cyberdyne.local**. Проанализируем теперь перехваченные пакеты из файла `mail_receiver_server_3.pcapng` с точки зрения удаленного сервера (рис. 9.32).

Первые 15 пакетов из анализируемого здесь перехваченного трафика очень похожи на рассмотренные ранее пакеты, поскольку они представляют тот же самый обмен сообщениями, где адрес отправителя относится к локальному серверу электронной почты ❶, а адрес получателя — к удаленному серверу электронной почты ❷. Как только этот обмен завершится, SMTP-сервер может поместить полученное сообщение в соответствующий почтовый ящик, чтобы получатель, для которого оно предназначается, смог извлечь его оттуда с помощью своего клиента электронной почты.

Как упоминалось ранее, протокол SMTP применяется, главным образом, для пересылки электронной почты, и поэтому он чаще всего употребляется именно для этой цели. Извлечение электронной почты из почтового ящика на сервере нельзя ограничить определенными рамками. Поэтому, вследствие

возникающих здесь различий для решения данной задачи предназначено несколько протоколов. К числу наиболее употребительных относятся протоколы POP3 (Post Office Protocol version 3 – почтовый протокол версии 3) и IMAP (Internet Message Access Protocol – протокол доступа к сообщениям в Интернете). В рассматриваемом здесь примере удаленный клиент электронной почты извлекает сообщения из почтового сервера, используя протокол IMAP в пакетах 16–34.

Рис. 9.32. Просмотр потока данных по протоколу TCP между локальным и удаленным серверами электронной почты

Протокол IMAP в этой книге не рассматривается, но в данном примере этого и не требуется, поскольку обмен данными зашифрован. Если проанализировать пакет 21, то можно заметить, что клиент, находящийся по

адресу **172.16.16.235**, посылает команду **STARTTLS** серверу электронной почты, расположенному по адресу **172.16.16.231**, в пакете 21 ❶, как показано на рис. 9.33.

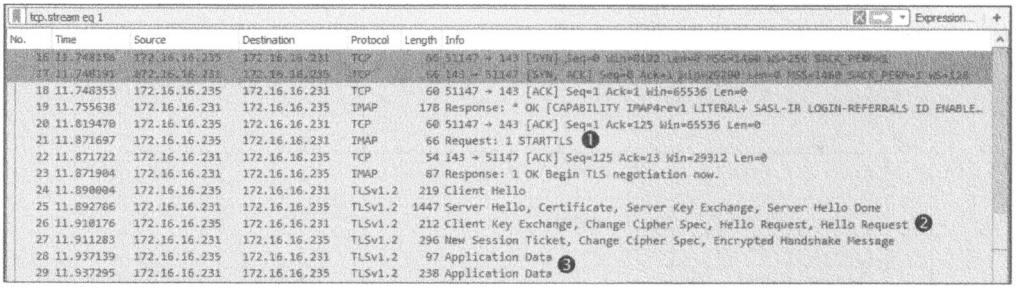


Рис. 9.33. Команда **STARTTLS** *обозначает, что сетевой трафик по протоколу IMAP будет зашифрован*

Данная команда извещает сервер, что клиенту требуется безопасно извлечь сообщения, используя шифрование по протоколу TLS. Между отдельными конечными точками в пакетах 24–27 согласуется защищенный канал связи ❷, а в остальных пакетах сообщение безопасно извлекается по протоколу TLS (Transport Layer Security – протокол защиты транспортного уровня) ❸. Если щелкнуть на любом из этих пакетов, чтобы просмотреть данные или попытаться проследить их поток по протоколу TCP (рис. 9.34), то в конечном итоге можно обнаружить неудобочитаемое содержимое, поскольку электронная почта защищена от перехвата злоумышленниками, которые могут попытаться похитить или проанализировать сетевой трафик в злонамеренных целях. Как только эти завершающие пакеты будут получены, процесс отправки почтового сообщения пользователем в одном домене пользователю в другом домене завершится.

Отправка вложений по протоколу SMTP

Файл перехвата **mail_sender_attachment.pcapng**

Протокол SMTP никогда не предназначался в качестве механизма для передачи файлов, но простота отправки файлов по электронной почте означает, что он стал для многих основным механизмом обмена. Рассмотрим краткий пример отправки файла на уровне пакетов с помощью протокола SMTP.

Как демонстрируется в примере из файла перехвата пакетов `mail_sender_attachment.pcapng`, пользователь посылает сообщение электронной почты из своего клиента, находящегося по адресу **172.16.16.225**, другому клиенту в той же самой сети через локальный почтовый SMTP-сервер, расположенный по адресу **172.16.16.221**. Это сообщение содержит мало текста и включает в себя вложенный файл изображения.

Рис. 9.34. Сетевой трафик по протоколу IMAP шифруется, когда клиент загружает почтовое сообщение

Отправка вложения по протоколу SMTP практически ничем не отличается от отправки текста. Все эти данные предназначаются для сервера, и хотя они обычно подвергаются специальной кодировке, отправной точкой для их анализа по-прежнему является команда **DATA**. Чтобы посмотреть, как это происходит на практике, откройте упомянутый выше файл перехвата и проследите поток данных по протоколу TCP для выполнения отдельной транзакции по протоколу SMTP. Этот поток приведен на рис. 9.35.

24 client pkt(s), 7 server pkt(s), 12 turns.

Entire conversation (77 kB) Show data as ASCII Stream 0

Find: Find Next

Hide this stream Print Save as... Close Help

Рис. 9.35. Отправка пользователем вложения по протоколу SMTP

Данный пример начинается, аналогично предыдущим примерам, с идентификации службы и обмена поддерживаемыми протоколами. Как только клиент будет готов передать почтовое сообщение, он сделает это, предоставив адреса отправителя и получателя и послав команду **DATA**, предписывающую серверу открыть буфер для получения информации. А дальше рассматриваемый здесь процесс несколько отличается.

В предыдущем примере клиент передавал текст непосредственно серверу, а в данном примере клиент должен отправить сообщение в текстовом виде наряду с двоичными данными, связанными с вложением изображения. Чтобы это произошло, он обозначает в поле **Content-Type** тип своего содержимого как **multipart/mixed** с заданной границей **----------050407080301000500070000** ❶. Этим серверу сообщается, что ему передаются разнотипные данные со своим особым типом MIME и кодировкой и что каждый тип данных будет отделен от другого границей с указанным значением. Следовательно, когда эти данные получит другой почтовый клиент, он будет знать, как интерпретировать их, исходя из заданных границ, особого типа MIME и кодировки, указанных в каждом фрагменте данных.

Почтовое сообщение в рассматриваемом здесь примере состоит из двух отдельных частей. В первой части содержится сам текст сообщения, поэтому она обозначается типом содержимого **text/plain** ❷. После этого следует маркер границы и начинается новая часть сообщения ❸. В этой части содержится файл изображения, поэтому она обозначается типом содержимого **image/jpeg** ❹. Следует также иметь в виду, что в поле **Content-Transfer-Encoding** (Кодировка передаваемого содержимого) установлено значение **base64** ❺. Это означает, что полученные данные должны быть преобразованы перед анализом их представления из формата Base-64 в JPEG. А в остальную часть передаваемых данных входит закодированный файл изображения ❻.

Эту кодировку не следует путать с шифрованием для защиты данных. Кодировка Base-64 почти мгновенно обратима, поэтому любой злоумышленник, перехвативший подобный сетевой трафик, сумеет извлечь из него файл изображения без особых усилий. Если же файл изображения требуется извлечь из перехваченного пакета самостоятельно, обратитесь за справкой к разделу "Троянская программа удаленного доступа" главы 12, "Анализ пакетов на безопасность", где рассматривается аналогичный сценарий извлечения изображения из передаваемого по протоколу HTTP файла. Прочитав этот раздел, вернитесь к рассматриваемому здесь файлу изображения, чтобы попытаться выявить нового таинственного соратника данного пользователя.

Заключительные соображения

В этой главе были представлены наиболее распространенные протоколы, с которыми приходится иметь дело, анализируя сетевой трафик на уровне приложений. А в последующих главах будут исследованы новые протоколы и дополнительные возможности упомянутых здесь протоколов при рассмотрении целого ряда реальных сценариев работы в сети.

Для более углубленного изучения отдельных протоколов рекомендуется ознакомиться с их спецификациями в соответствующих документах RFC или прочитать книгу Дугласа Камера *Сети TCP/IP, том 1. Принципы, протоколы и структура, 4-е издание* (пер. с англ. ИД "Вильямс", 2003 г.). См. также перечень учебных ресурсов в приложении А.

10

ОСНОВНЫЕ РЕАЛЬНЫЕ СЦЕНАРИИ

Начиная с этой главы, мы обратимся непосредственно к анализу пакетов в Wireshark для выявления настоящих затруднений, возникающих в сети. На примерах реальных сценариев подобных затруднений будет описан их контекст и предоставлены сведения, доступные для анализа в момент возникновения этих затруднений.

Заложив прочное основание, мы перейдем к анализу, описав сначала методику перехвата подходящих пакетов, а затем весь процесс работы в направлении диагностики сети. Как только анализ будет завершен, мы укажем возможные решения и вкратце рассмотрим извлеченные уроки.

Не следует, однако, забывать, что анализ пакетов является весьма динамичным процессом. Поэтому методики, применяемые для анализа пакетов в каждом рассматриваемом здесь сценарии, могут отличаться от тех, которыми, возможно, пользуетесь вы. Ведь каждый подходит к осмыслению задачи и находит ее решение по-своему. Самое главное, чтобы анализ привел к разрешению возникшего затруднения. Но даже если он и не привел к удаче, очень важно учиться и на неудачах. Ведь опыт приобретается именно тогда, когда мы получаем не то, что нам нужно.

Кроме того, большинство затруднений, обсуждаемых в этой главе, могут быть разрешены и по методикам, не подразумевающим анализ пакетов. Что же тогда в них интересного? Когда я только начинал изучать анализ пакетов,

я нашел полезным исследовать типичные затруднения нетипичными способами, используя методики анализа пакетов. Именно поэтому здесь представлены вашему вниманию реальные сценарии работы в сети.

Отсутствие веб-содержимого

Файл перехвата **http_espn_fail.pcapng**

В первом из рассматриваемых здесь сценариев пользователь Пит Пакет является большим поклонником университетского баскетбола. Он не привык вставать и ложиться поздно, из-за чего нередко пропускает игры конференции Западного Побережья. Поэтому каждое утро он прежде всего садится за свою домашний компьютер, чтобы посетить веб-страницу спортивного телеканала ESPN по адресу `http://www.espn.com/` и посмотреть результаты игр, проведенных накануне поздно вечером. Просматривая как-то утром упомянутую выше страницу, Пит обнаружил, что она долго загружается, а когда наконец-то загрузится, большинство изображений и прочего содержимого на ней отсутствует (рис. 10.1). Итак, поможем Питу продиагностировать возникшее затруднение.

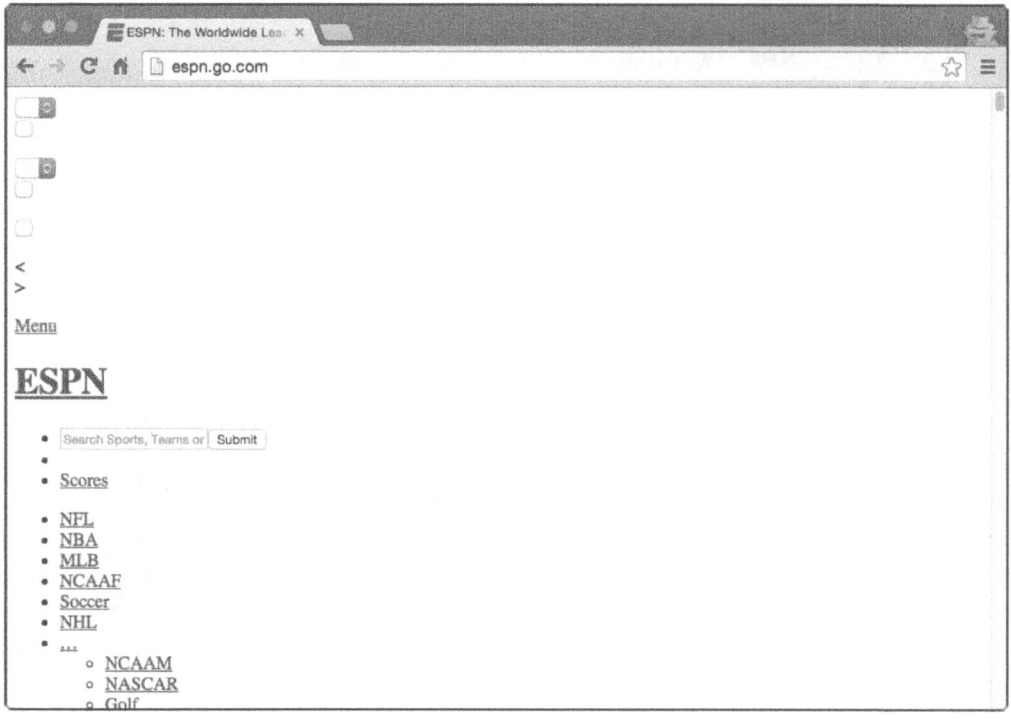


Рис. 10.1. Неудачно загруженная веб-страница спортивного телеканала ESPN

Подключение к сети

Данное затруднение возникло только на компьютере Пита и не оказало влияния на других пользователей, поэтому начнем с перехвата пакетов непосредственно в этой конечной точке сети. С этой целью установим Wireshark и перехватим пакеты при просмотре веб-сайта ESPN. Эти пакеты находятся в файле перехвата `http_espn_fail.pcapng`.

Анализ

Нам известно, что Пит не в состоянии просматривать содержимое веб-страницы, которую он посещает, и поэтому мы должны проанализировать протокол HTTP. Если вы читали предыдущую главу, то должны иметь основное представление о том, как выглядит сетевой трафик по протоколу HTTP между клиентом и сервером. Начать его анализ лучше всего с HTTP-запросов, посылаемых удаленному серверу. С этой целью можно применить фильтр запросов по методу `GET`, введя выражение **`http.request.method == "GET"`** на панели фильтров. Хотя это можно сделать проще, выбрав команду Statistics⇨HTTP⇨Requests (Статистика⇨HTTP⇨Запросы) из главного меню.

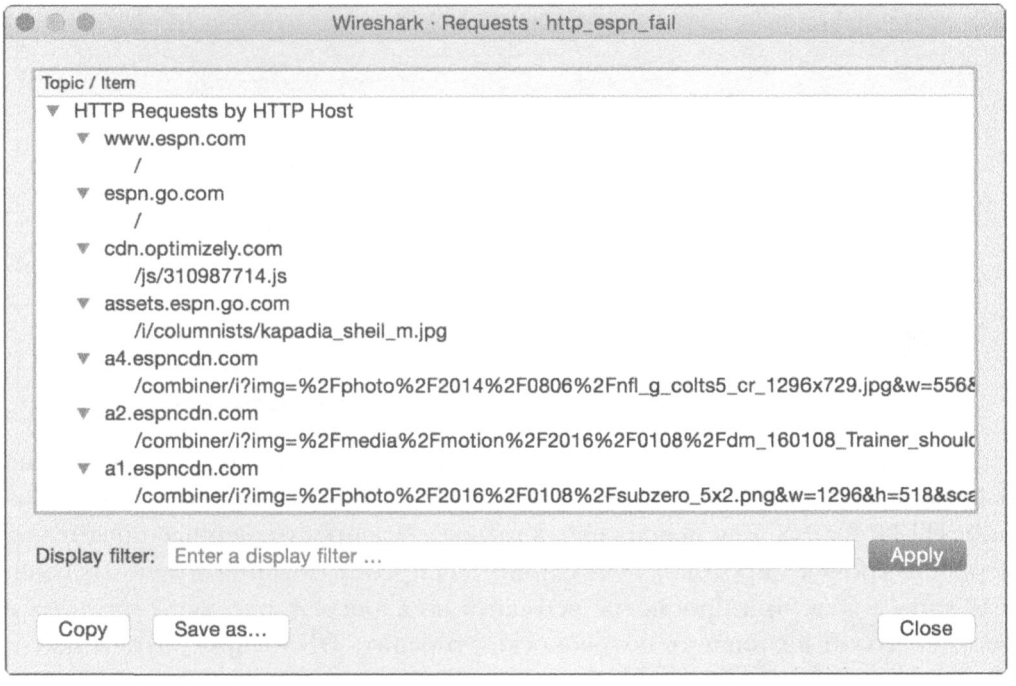


Рис. 10.2. Просмотр HTTP-запросов на веб-сайт ESPN

Как следует из краткого обзора перехваченного трафика, приведенного на рис. 10.2, он ограничивается семью различными HTTP-запросами, причем все они выглядят так, как будто они связаны с веб-сайтом ESPN. Каждый HTTP-запрос содержит символьную строку `espn` в имени указанного домена, за исключением домена **`cdn.optimizely.com`**, который относится к *сети доставки содержимого* (Content Delivery Network, CDN), применяемой для доставки рекламы на многие сайты. Просматривая веб-сайты, на которых размещается реклама или другое внешнее содержимое, можно нередко увидеть запросы к различным сетям CDN.

В отсутствие явных признаков для прослеживания возникшего затруднения перейдем к следующей стадии, которая заключается в просмотре иерархии протоколов в файле перехвата. С этой целью выберите команду Statistics⇨Protocol Hierarchy из главного меню. Это даст возможность выявить непредвиденные протоколы или особые распределения сетевого трафика по протоколам (рис. 10.3). Следует, однако, иметь в виду, что отображение иерархии протоколов на экране зависит от применяемого в настоящий момент фильтра. Поэтому очистите применявшийся ранее фильтр, чтобы получить предполагаемые результаты по всем перехваченным пакетам.

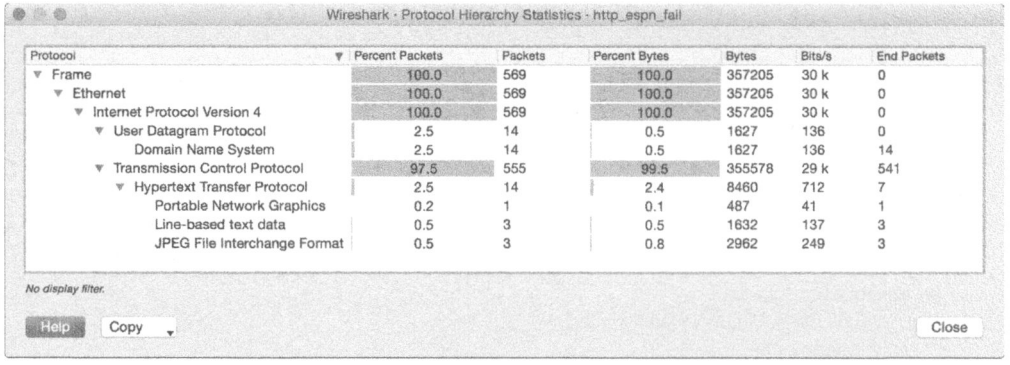


Рис. 10.3. Просмотр иерархии протоколов в текущем сеансе

Данная иерархия протоколов не особенно сложна, и поэтому можно очень быстро выяснить, что в ней действуют лишь два протокола уровня приложений: HTTP и DNS. Как пояснялось в главе 9, "Распространенные протоколы верхнего уровня", протокол DNS служит для преобразования доменных имен в IP-адреса. Так, при просмотре веб-сайта по адресу `http://www.espn.com/` в системе может возникнуть потребность отправить DNS-запрос, чтобы найти IP-адрес удаленного веб-сервера, если он еще неизвестен. И как только будет получен DNS-ответ, содержащаяся в нем информация может быть введена в локальный кеш, после чего может быть начата передача данных по протоколу HTTP (с помощью протокола TCP).

Несмотря на то что здесь ничего вроде бы не выглядит необычно, особого внимания все же заслуживают 14 пакетов DNS. В частности, DNS-запрос единственного доменного имени, как правило, содержится в одном пакете, как, впрочем, и ответ на этот запрос, если только он не очень большой, и тогда в протоколе DNS будет употреблен протокол TCP. А поскольку в данном случае имеется 14 пакетов DNS, то вполне возможно, что из них было сформировано семь DNS-запросов и семь ответов на них, что в итоге составляет 14 пакетов. На рис. 10.2 были показаны HTTP-запросы к семи разным доменам, но ведь Пит ввел единственный URL в окне своего браузера. Почему же были сделаны все остальные, лишние запросы?

В идеальном случае для посещения веб-страницы было бы достаточно запросить один сервер, чтобы извлечь ее содержимое в течение одного HTTP-диалога. Но в действительности отдельная веб-страница может предоставить содержимое, размещаемое на нескольких серверах. Все текстовое содержимое может быть размещено в одном месте, графика — в другом, а встроенные видеозаписи — в третьем. И это не считая рекламу, которая может размещаться на десятках серверов многих поставщиков. Всякий раз, когда HTTP-клиент выполняет синтаксический анализ HTML-кода и обнаруживает ссылку на содержимое из другого хоста, он попытается запросить содержимое у этого хоста, что приведет к формированию дополнительных DNS-запросов и соответствующих HTTP-запросов. Именно это и произошло, когда Пит посетил веб-сайт ESPN. И хотя он намеревался просмотреть содержимое только из одного источника, в коде HTML были обнаружены ссылки на дополнительное содержимое, и браузер Пита автоматически запросил это содержимое из многих других доменов.

Итак, выяснив причину появления всех лишних запросов, исследуем далее отдельные диалоги, связанные с каждым запросом, выбрав команду Statistics⇨ Conversations из главного меню. Просмотр окна Conversations (рис. 10.4) дает главный ключ к разгадке.

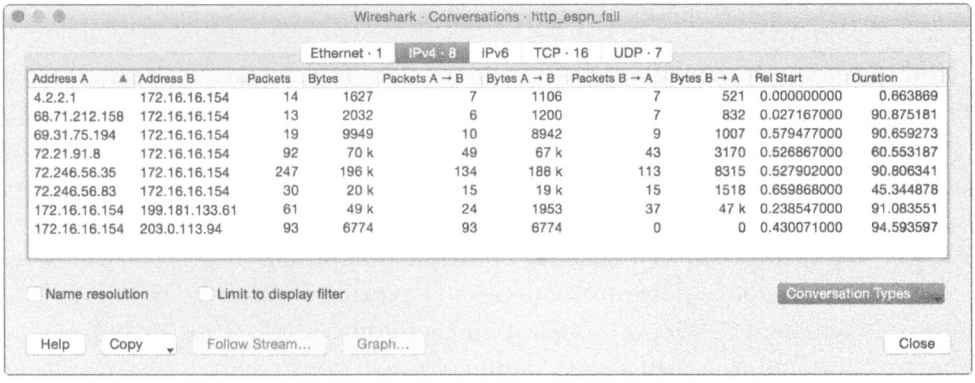


Рис. 10.4. Просмотр IP-диалогов

Как было обнаружено ранее, имеется семь DNS-запросов и семь соответствующих HTTP-запросов. Принимая во внимание это обстоятельство, было бы вполне оправданно предположить, что имеется также семь соответствующих IP-диалогов, но это не так, поскольку их восемь. Чем же это объяснить?

Это можно было бы объяснить, в частности, тем, что перехваченный трафик был "загрязнен" дополнительным диалогом, не связанным с рассматриваемым здесь затруднением. Безусловно, следует ясно отдавать себе отчет, что анализ не должен страдать от неуместного трафика, хотя это и не касается данного диалога. Если исследовать каждый HTTP-запрос и отметить IP-адрес, по которому он был отправлен, то в конечном итоге останется один диалог, у которого отсутствует соответствующий HTTP-запрос. А конечными точками этого диалога являются компьютер Пита (по адресу **`172.16.16.154`**) и удаленный хост (по адресу **`203.0.113.94`**). Этот диалог представлен в нижней строке на рис. 10.4. Обратите внимание, что этому неизвестному хосту было послано 6774 байта, а в ответ — ни одного байта. И это обстоятельство заслуживает особого внимания.

Если отсеять рассматриваемый здесь диалог, щелкнув на нем правой кнопкой мыши и выбрав команду Apply As Filter⇨Selected⇨A↔B (Применить как фильтр⇨Выбрано⇨A↔B) из контекстного меню, то, зная принцип действия протокола TCP, можно выяснить, что пошло не так (рис. 10.5).

No.	Time	Source	Destination	Protocol	Length	Info
25	0.430071	172.16.16.154	203.0.113.94	TCP	78	64862 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093668 TSec…
26	0.430496	172.16.16.154	203.0.113.94	TCP	78	64863 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093668 TSec…
27	0.431050	172.16.16.154	203.0.113.94	TCP	78	64864 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093669 TSec…
39	0.500663	172.16.16.154	203.0.113.94	TCP	78	64865 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093737 TSec…
40	0.500873	172.16.16.154	203.0.113.94	TCP	78	64866 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093737 TSec…
70	0.553964	172.16.16.154	203.0.113.94	TCP	78	64869 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093787 TSec…
456	1.460006	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64863 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 …
457	1.460086	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64862 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 …
458	1.461238	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64864 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 …
459	1.530278	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64866 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 …
460	1.530278	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64865 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 …
461	1.580145	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64869 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 …
462	2.461157	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64863 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 …
463	2.461157	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64862 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 …

Рис. 10.5. Просмотр непредвиденного соединения

При обычном обмене данными по протоколу TCP можно ожидать появления стандартной последовательности установки соединения `SYN-SYN/ACK-ACK`. В данном случае компьютер Пита отправил пакет `SYN` хосту, находящемуся по адресу **`203.0.113.94`**, но мы не видим ответного пакета `SYN/ACK`. Мало того, компьютер Пита отправил несколько пакетов `SYN` без всякой пользы, что в конечном счете вынудило эту машину повторить передачу пакетов TCP. Более подробно особенности повторных передач пакетов по протоколу TCP рассматриваются в главе 11, "Меры борьбы с медленной сетью", а до тех пор можно сделать следующий важный вывод: один хост посылает пакеты, на которые он не получает ответа. Глядя на столбец **`Time`**, можно заметить, что повторные

передачи пакетов продолжаются в течение 95 секунд без ответа. Для передачи данных по сети это крайне медленно.

Таким образом, мы выявили семь DNS-запросов, семь HTTP-запросов и восемь IP-диалогов. А поскольку нам известно, что перехваченный сетевой трафик не "загрязнен" лишними данными, то мы можем с полным основанием предположить, что таинственный восьмой IP-диалог, вероятнее всего, и послужил причиной медленной и неполной загрузки веб-страницы на компьютер Пита. По какой то причине компьютер Пита попытался связаться с устройством, которое вообще не существует или просто не принимает пакеты. Чтобы понять, почему это происходит, рассмотрим не то, что находится в файле перехвата, а то, чего в нем нет.

Когда Пит просматривал веб-сайт по адресу `http://www.espn.com/`, его браузер выявил ресурсы, расположенные в других доменах. Чтобы извлечь данные из этих ресурсов, компьютер Пита сформировал DNS-запросы с целью найти соответствующие IP-адреса, а затем установил с ними связь по протоколу TCP, чтобы оправить HTTP-запрос на доставку содержимого. Но для диалога с хостом, находящимся по адресу **`203.0.113.94`**, соответствующий DNS-запрос не был обнаружен. Откуда же этот адрес известен компьютеру Пита?

Как обсуждалось в главе 9, "Распространенные протоколы верхнего уровня", при рассмотрении протокола DNS, в большинстве систем в той или иной форме реализовано кеширование DNS-запросов. Это дает им возможность обращаться к извлеченному ранее локальному соответствию DNS-имени IP-адресу, не формируя DNS-запрос при каждом посещении домена, с которым часто приходится обмениваться данными. В конечном итоге срок действия такого соответствия истечет, после чего должен быть сформирован новый DNS-запрос. Но если соответствие DNS-имени IP-адресу изменится, а устройство не сформирует соответствующий DNS-запрос на получение нового адреса при очередном посещении домена, то устройство попытается установить соединение по адресу, который уже недействителен.

Именно это и произошло с компьютером Пита, на котором уже было кешировано соответствие DNS-имени IP-адресу домена, где размещалось содержимое веб-сайта ESPN. А поскольку эта запись в кеше существовала, то DNS-запрос не был сформирован и компьютер Пита попытался установить соединение по старому IP-адресу. Но хост по этому адресу уже не отвечал на запросы. В итоге время ожидания ответов на запросы истекло, а содержимое так и не было загружено.

К счастью для Пита, очистить кеш DNS-запросов вручную совсем не трудно. Для этого нужно ввести в командной строке окна терминала всего несколько символов. С другой стороны, можно было бы повторить попытку связи через несколько минут, когда запись в кеше DNS-запросов, вероятнее всего, окажется просроченной и будет сформирован новый DNS-запрос.

Усвоенные уроки

Чтобы узнать, что баскетбольная команда университета штата Кентукки обыграла команду университета Дьюка с отрывом в 90 очков, нам пришлось немало потрудиться. Но из этого мы вынесли более глубокое понимание взаимодействия сетевых узлов. В данном сценарии нам удалось выработать решение благодаря доступу к нескольким информационным точкам, связанным с запросами и диалогами, происходящими в перехваченном трафике. И там нам удалось выявить несколько несогласованностей, которые в конечном итоге привели нас к неудавшемуся обмену данными между клиентом и одним из серверов доставки содержимого для веб-сайта ESPN.

В действительности диагностика неисправностей в сети редко сводится к простой прокрутке списка пактов и поиску среди них пакетов, которые выглядят необычно. Для диагностики даже самых простых неполадок в сети может потребоваться перехват довольно крупного трафика, опирающийся на средства анализа и статистической обработки Wireshark для выявления отклонений от нормы. Поэтому очень важно освоить такой стиль анализа, чтобы успешно проводить диагностику на уровне пакетов.

Если вы хотите посмотреть, каким образом происходит нормальный обмен данными между веб-браузером и веб-сайтом ESPN, попробуйте посетить этот сайт, самостоятельно перехватывая сетевой трафик. Постарайтесь сами выявить все серверы, ответственные за доставку содержимого для этого веб-сайта.

Не реагирующая метеорологическая служба

Файлы перехвата **weather_broken.pcapng** *и* **weather_working.pcapng**

И во втором сценарии главным действующим лицом снова оказывается уже знакомый нам пользователь Пит Пакет. К числу многих увлечений Пита относится любительская метеорология. Он не может прожить и нескольких часов, чтобы не проверить текущие метеорологические условия и получить прогноз погоды. И в этом он не полагается только на местные новости с прогнозом погоды, а держит вне своего дома небольшую метеостанцию, сообщающую данные на веб-сайт по адресу `https://www.wunderground.com` для накапливания и просмотра. Однажды утром Пит вышел из дому, чтобы проверить на своей метеостанции, насколько упала температура воздуха с вечера, но обнаружил, что его метеостанция не сообщала данные на веб-сайт Wunderground в течение девяти часов после полуночи (рис. 10.6).

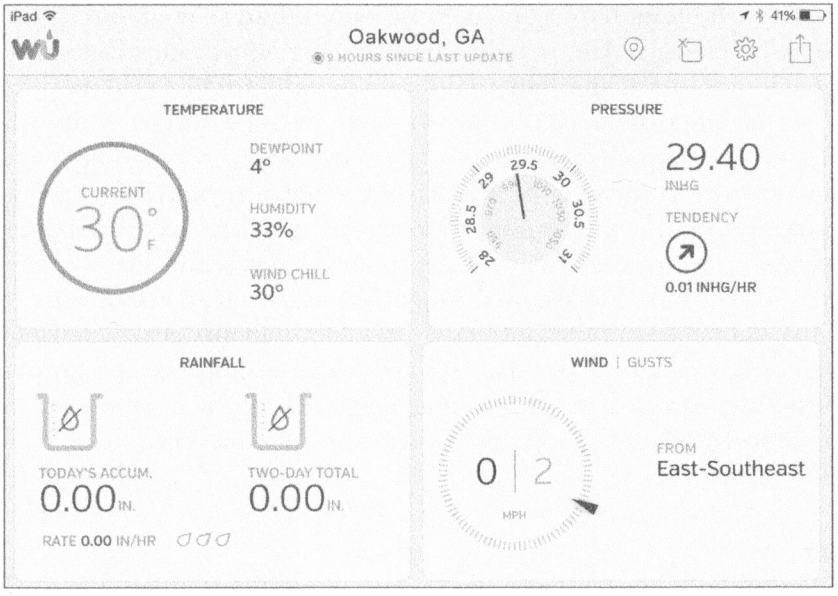


Рис. 10.6. Метеостанция не посылала статистический отчет в течение девяти часов

Подключение к сети

Метеостанция Пита, установленная на крыше его дома, подключена к приемнику внутри дома по радиоканалу. А этот приемник подключен к сети через коммутатор для передачи статистических отчетов на веб-сайт Wunderground через Интернет. Такая схема сети наглядно показана на рис. 10.7.

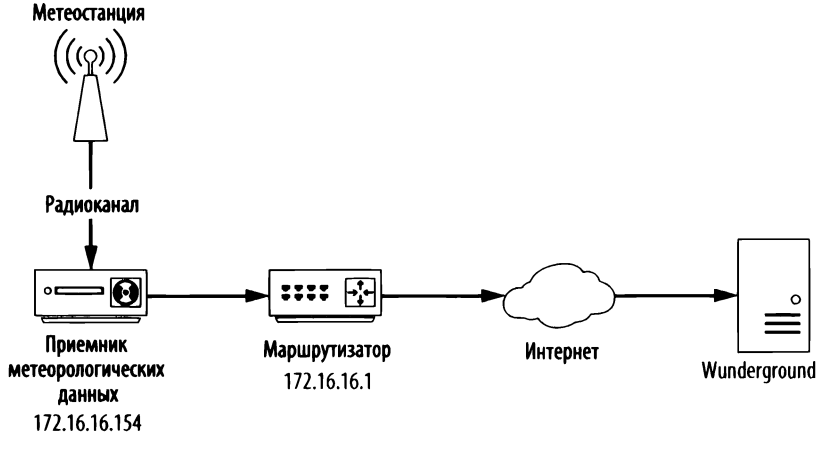


Рис. 10.7. Схема сети для подключения метеостанции

Управление приемником данных от метеостанции выполняется с помощью простой веб-страницы. Но, перейдя на нее, Пит обнаружил лишь таинственное сообщение о последнем времени синхронизации без всяких рекомендаций по устранению неполадок, потому что соответствующее программное обеспечение не выполняет подробного ведения журнала ошибок. А поскольку приемник служит концентратором связи с инфраструктурой метеостанции, то для диагностики неисправности в сети имеет смысл перехватывать пакеты, входящие и исходящие из этого устройства. Рассматриваемая здесь сеть относится к категории домашних, и поэтому на SOHO-коммутаторе, предназначенном для небольших учрежденческих и домашних сетей, вряд ли будет поддерживаться режим зеркального отображения портов. В данном случае для перехвата пакетов лучше всего подходит недорогой сетевой ответвитель или заражение ARP-кеша. Перехваченные в итоге пакеты сохранены в файле `weather_broken.pcapng`.

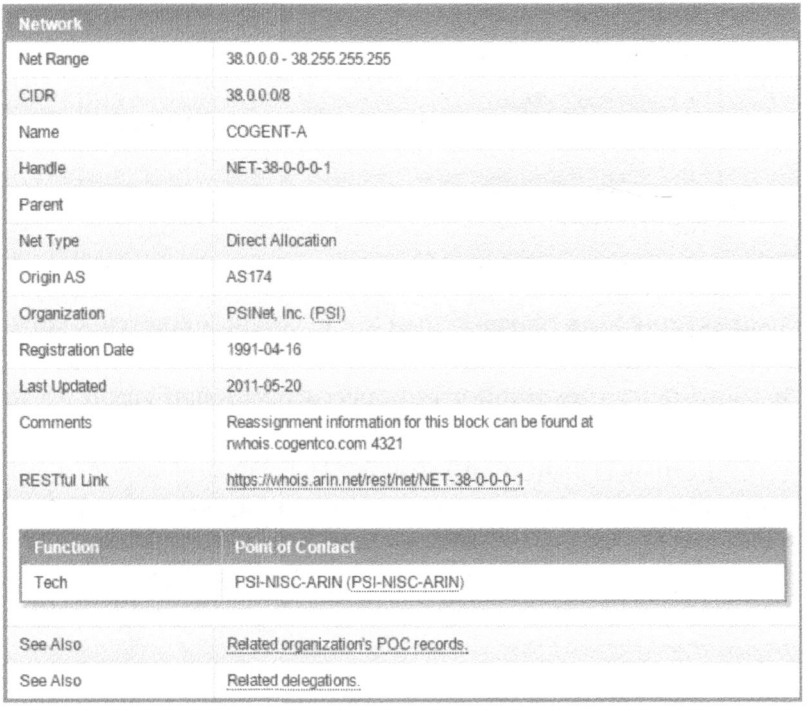
Анализ

Открыв упомянутый выше файл перехвата, вы обнаружите, что и в данном случае мы имеем дело с обменом данными по протоколу HTTP. Перехват пакетов ограничивается единственным диалогом между локальным приемником метеорологических данных, находящимся по адресу **`172.16.16.154`**, и неизвестным удаленным устройством, расположенным в Интернете по адресу **`38.102.136.125`** (рис. 10.8).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.154	38.102.136.125	TCP	78	53904 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1015236041 TSecr=0 SACK_PERM=1
2	0.087018	38.102.136.125	172.16.16.154	TCP	60	80 → 53904 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1360
3	0.087108	172.16.16.154	38.102.136.125	TCP	54	53904 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
4	0.087178	172.16.16.154	38.102.136.125	HTTP	571	GET /weatherstation/updateweatherstation.php?ID=KGADAWJO2&PASSWORD=00000000&tempf=43.0&humidity=30…
5	0.176462	38.102.136.125	172.16.16.154	HTTP	237	HTTP/1.0 200 OK (text/html)
6	0.176567	172.16.16.154	38.102.136.125	TCP	54	53904 → 80 [ACK] Seq=518 Ack=184 Win=65535 Len=0
7	0.176714	172.16.16.154	38.102.136.125	TCP	54	53904 → 80 [FIN, ACK] Seq=518 Ack=184 Win=65535 Len=0
8	0.262587	38.102.136.125	172.16.16.154	TCP	60	80 → 53904 [FIN, ACK] Seq=184 Ack=519 Win=7673 Len=0
9	0.262656	172.16.16.154	38.102.136.125	TCP	54	53904 → 80 [ACK] Seq=519 Ack=185 Win=65535 Len=0

Рис. 10.8. Обособленный обмен данными между приемником метеорологических данных и неизвестным удаленным устройством

Прежде чем исследовать характеристики сетевого диалога, выясним, можно ли распознать неизвестный IP-адрес. Без обширного исследования нам вряд ли удастся выяснить, тот ли это IP-адрес, по которому приемник должен передавать данные из метеостанции Пита. Но, по крайней мере, мы можем проверить, что он относится к инфраструктуре веб-сайта Wunderground, выполнив запрос WHOIS. Такой запрос можно выполнить через веб-сайты регистрации доменов или региональных регистраторов в Интернете, например, по адресу `http://whois.arin.net/`. В данном случае IP-адрес относится к Cogent — *поставщику услуг Интернета* (ISP; рис. 10.9). Здесь упоминается также компания PSINet Inc., но в результате быстрого поиска обнаруживается, что большинство ее ресурсов было приобретено компанией Cogent еще в начале 2000-х годов.



Network	
Net Range	38.0.0.0 - 38.255.255.255
CIDR	38.0.0.0/8
Name	COGENT-A
Handle	NET-38-0-0-0-1
Parent	
Net Type	Direct Allocation
Origin AS	AS174
Organization	PSINet, Inc. (PSI)
Registration Date	1991-04-16
Last Updated	2011-05-20
Comments	Reassignment information for this block can be found at rwhois.cogentco.com 4321
RESTful Link	https://whois.arin.net/rest/net/NET-38-0-0-0-1

Function	Point of Contact
Tech	PSI-NISC-ARIN (PSI-NISC-ARIN)

See Also	Related organization's POC records.
See Also	Related delegations.

Рис. 10.9. Данные по запросу WHOIS обозначают владельца данного IP-адреса

Иногда по запросу WHOIS возвращается имя организации, если IP-адрес зарегистрирован непосредственно самой организацией. Но зачастую организация просто пользуется пространством IP-адресов, выделяемым провайдером, даже не регистрируя его непосредственно. В подобных случаях в качестве еще одной полезной тактики можно произвести поиск информации по *номеру автономной системы* (ASN), к которой относится анализируемый IP-адрес.

Просматривать соответствие IP-адресов номерам ASN можно самыми разными способами (в результате некоторых видов поиска по запросам WHOIS такое соответствие предоставляется автоматически), но я лично предпочитаю пользоваться инструментальным средством автоматического поиска Team Cymru (`https://asn.cymru.com/`). С помощью этого средства можно обнаружить, что адрес **`38.102.136.125`** связан с именем **`AS 36347`**, которое, в свою очередь, связано с названием организации "Wunderground – The Weather Channel, LLC, US" (рис. 10.10). Это означает, что устройство, с которым связывается метеостанция, находится где-то в текущей организации. Если же узнать конкретного владельца адреса не удается, то, возможно, стоит предположить, что приемник метеостанции Пита обменивается данными с некорректным устройством, хотя в нашем случае правильность адреса подтверждается.

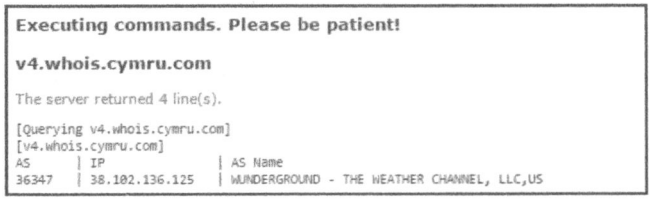


Рис. 10.10. Поиск соответствия внешнего IP-адреса конкретному номеру ASN

Итак, охарактеризовав неизвестный хост, можно перейти к углубленному анализу обмена данными с ним. Сетевой диалог с этим хостом относительно краткий и состоит из трехэтапного процесса установки связи по протоколу TCP, передачи одного HTTP-запроса по методу `GET`, получения соответствующего ответа и разрыва связи по протоколу TCP. Установка соединения и разрыв связи происходят успешно, поэтому искомая проблема, вероятнее всего, кроется в самом HTTP-запросе. Чтобы проанализировать его более подробно, проследим поток данных по протоколу TCP (рис. 10.11).

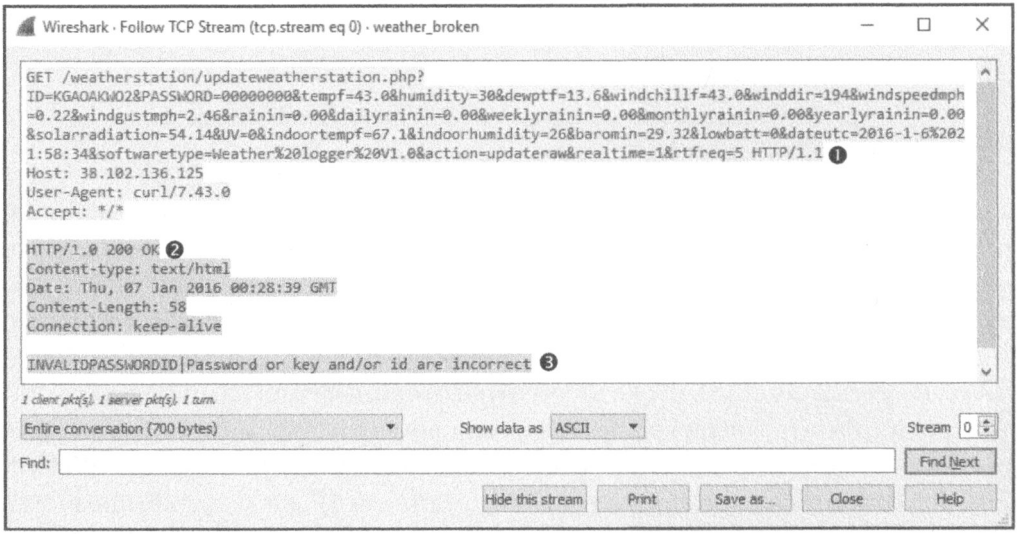


Рис. 10.11. Прослеживание потока данных по протоколу TCP, исходящего из приемника метеорологических данных

Обмен данными по протоколу HTTP начинается с запроса по методу `GET`, посылаемого приемником данных от метеостанции Пита на веб-сайт Wunderground. Но вместо HTTP-содержимого значительный массив данных был передан в URL ❶. Передача данных через символьную строку запроса в URL весьма характерна для веб-приложений, и похоже, что приемник

передает обновленные метеорологические данные, используя именно этот механизм. Это наблюдается в таких полях, как `tempf=43.0`, `dewptf=13.6` и `windchillf=43.0`. Сервер Wunderground, занимающийся сбором данных, выполняет синтаксический анализ списка полей и параметров из URL, сохраняя их в базе данных.

На первый взгляд, все в запросе сервера Wunderground по методу `GET` выглядит верно, но проанализируем полученный ответ, где сообщается об ошибке. В частности, сервер прислал код ответа **`HTTP/1.0 200 OK`** ❷, указывающий на то, что запрос по методу `GET` был получен успешно, но в теле ответа ❸ содержится следующее сообщение об ошибке:

```
INVALIDPASSWORDID|Password or key and/or id are incorrect
(Неверный идентификатор и пароль | Пароль и/или ключ неверен)
```

Если вернуться к URL с запросом, то можно заметить, что первыми в нем были переданы параметры `ID` и `PASSWORD`. Они служат для обозначения позывного метеостанции и ее аутентификации на сервере Wunderground.

В данном случае идентификатор метеостанции Пита указан верно, чего нельзя сказать о его пароле. По какой-то неизвестной причине он был заменен нулями. Последний успешный обмен данными был осуществлен в полночь, и поэтому вполне возможно, что было выполнено обновление или произведена перезагрузка приемника, в результате чего конфигурация пароля могла быть утрачена.

ПРИМЕЧАНИЕ *Несмотря на то что многие разработчики предпочитают передавать параметры непосредственно в URL, делать это в отношении паролей не рекомендуется. Дело в том, что параметры в URL передаются в виде открытого текста, если протокол HTTP применяется без дополнительного шифрования, как в протоколе HTTPS. Следовательно, злонамеренный пользователь, которому случилось прослушивать сеть, может перехватить пароль, открыто передаваемый в URL.*

Имея доступ к странице настройки своего приемника, Пит может ввести новый пароль, после чего его метеостанция снова начнет синхронно передавать свои данные. Пример успешной передачи метеорологических данных можно найти в файле перехвата `weather_working.pcapng`. А соответствующий поток обмена данными приведен на рис. 10.12. Теперь пароль указан верно ❶, поэтому сервер Wunderground ответит сообщением `success` (успешно) в теле HTTP-ответа ❷.

Рис. 10.12. Успешный обмен данными с метеостанцией

Усвоенные уроки

В данном сценарии нам пришлось иметь дело со сторонней службой, облегчавшей обмен данными по сети с помощью функциональных средств другого протокола (HTTP). Устранение неполадок при обмене данными со сторонними службами — довольно распространенная задача, и для ее решения вполне пригодны методики анализа пакетов, если имеется подходящая документация или подробное протоколирование ошибок. Подобные диагностические задачи все чаще приходится решать с появлением таких устройств IoT (Internet of Things — Интернет вещей), как данная метеостанция.

Чтобы устранить подобную проблему нам пришлось проанализировать последовательность обмена данными с неизвестным устройством и прикинуть, как это все должно работать. Ряд приложений, в которых используется протокол HTTP для передачи данных, как в случае с нашей метеостанцией, имеют довольно простую структуру. Однако существуют довольно сложные устройства, в которых данные передаются в виде нескольких транзакций, да еще в зашифрованном виде, либо в них может применяться никому не известный протокол собственной разработки. В подобных случаях Wireshark не сможет корректно проанализировать передаваемые данные и отобразить их в виде удобного диалога.

По мере исследования работы сетевых служб, созданных сторонними разработчиками, вы в конечном итоге поймете те общие шаблоны, которые применяли разработчики для упрощения обмена данными по сети. И эти знания позволят вам с высокой эффективностью разрешить возникшие затруднения.

Отсутствие доступа к Интернету

Во многих случаях может возникнуть потребность продиагностировать или разрешить проблемы, возникающие при подключении к Интернету. В этом разделе рассматриваются некоторые из наиболее распространенных затруднений подобного рода.

Трудности конфигурирования шлюза

Файл перехвата **nowebaccess1.pcapng**

В следующем сценарии рассматривается проблема, возникающая в связи с отсутствием доступа к Интернету. При этом удалось убедиться, что у пользователя имеется доступ ко всем внутренним ресурсам в сети, включая общие сетевые ресурсы на других рабочих станциях и приложения, размещаемые на локальных серверах.

Архитектура такой сети довольно проста, поскольку все клиенты и серверы подключены к последовательному ряду простых коммутаторов. Доступ к Интернету осуществляется через единственный маршрутизатор, служащий в качестве стандартного шлюза, а сведения об IP-адресах предоставляются по протоколу DHCP. Это довольно типичный сценарий для мелких компаний.

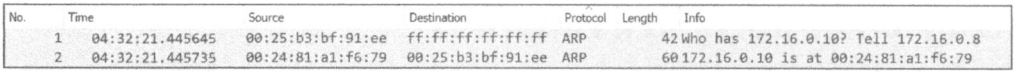
Подключение к сети

Чтобы выяснить причину подобной проблемы, придется предложить пользователю попробовать подключиться к Интернету для просмотра веб-контента, в то время как анализатор пакетов будет прослушивать сеть. Воспользуемся блок-схемой, приведенной на рис. 2.15 в главе 2, "Подключение к сети", чтобы выбрать подходящий метод для размещения анализатора пакетов.

Коммутаторы в данной сети не поддерживают зеркальное отображение портов. Мы уже вмешались в работу пользователя, чтобы провести свое испытание, и поэтому его можно еще раз попросить перейти в автономный режим. И хотя в данном сценарии не идет речь о высокой пропускной способности, для перехвата сетевого трафика вполне подойдет сетевой ответвитель, если таковой имеется. Полученный в итоге результат сохранен в файле перехвата `nowebaccess1.pcapng`.

Анализ

Перехват сетевого трафика начинается с ARP-запроса и ответа, как показано на рис. 10.13. В пакете 1 компьютер пользователя с MAC-адресом **`00:25:b3:bf:91:ee`** и IP-адресом **`172.16.0.8`** посылает широковещательный пакет по протоколу ARP всем компьютерам в сегменте сети с целью обнаружить MAC-адрес, связанный с IP-адресом его стандартного шлюза по адресу **`172.16.0.10`**.



No.	Time	Source	Destination	Protocol	Length	Info
1	04:32:21.445645	00:25:b3:bf:91:ee	ff:ff:ff:ff:ff:ff	ARP	42	Who has 172.16.0.10? Tell 172.16.0.8
2	04:32:21.445735	00:24:81:a1:f6:79	00:25:b3:bf:91:ee	ARP	60	172.16.0.10 is at 00:24:81:a1:f6:79

Рис. 10.13. ARP-запрос компьютера и ответ на него стандартного шлюза

Ответ получается в пакете 2, из которого компьютер пользователя узнает, что IP-адресу **`172.16.0.10`** соответствует MAC-адрес **`00:24:81:a1:f6:79`**. Как только этот ответ будет получен, в распоряжении компьютера появится маршрут к шлюзу, через который он сможет получить прямой доступ к Интернету.

Прослеживая ARP-ответ, компьютер должен попытаться преобразовать DNS-имя веб-сайта в IP-адрес, используя протокол DNS в пакете 3. Как показано на рис. 10.14, компьютер посылает с этой целью пакет с DNS-запросом своему основному DNS-серверу, находящемуся по адресу **`4.2.2.2`** ❶.

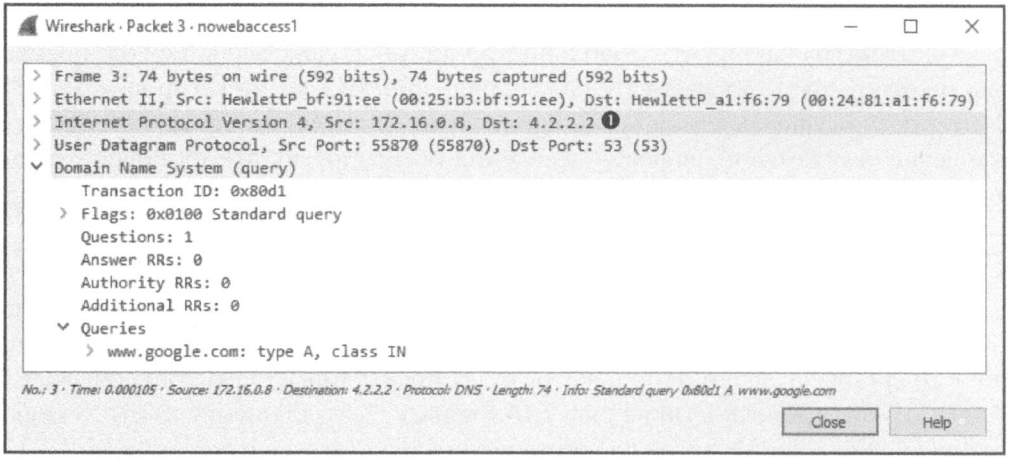


Рис. 10.14. DNS-запрос, посылаемый по адресу **`4.2.2.2`**

При обычных обстоятельствах DNS-сервер очень быстро ответит на полученный DNS-запрос, но в данном случае это совсем не так. Вместо ответа мы наблюдаем тот же самый DNS-запрос, посылаемый во второй раз, но уже по другому адресу получателя. Как показано на рис. 10.15, в пакете 4 второй DNS-запрос посылается резервному DNS-серверу, адрес которого указан на данном компьютере как **`4.2.2.1`** ❶.

Но и в этом случае ответа от DNS-сервера так и не последовало, и поэтому запрос по адресу **`4.2.2.1`** был послан еще раз через секунду. Этот процесс повторяется с чередованием адресов основного ❶ и резервного ❷ DNS-серверов в течение нескольких последующих секунд, как показано на рис. 10.16. А в целом этот процесс занимает около 8 секунд ❸ или продолжается до тех пор, пока браузер пользователя не сообщит, что искомый веб-сайт недоступен.

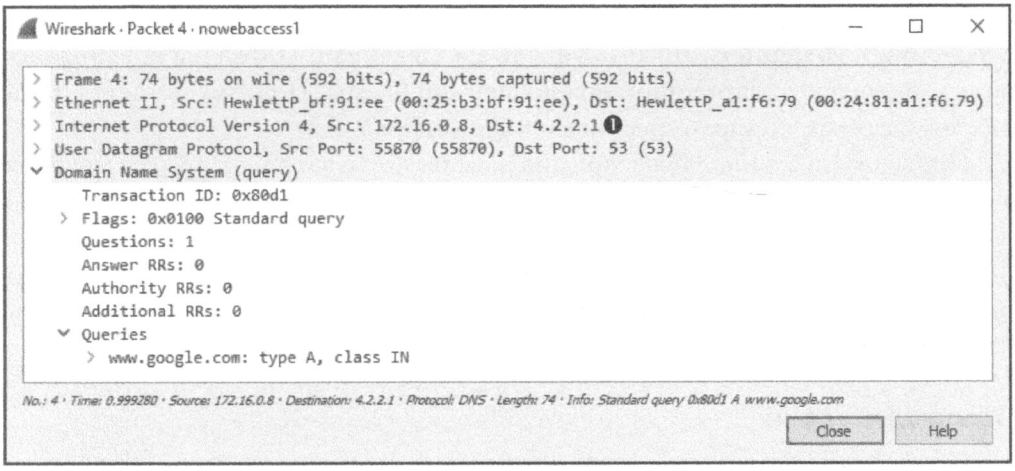
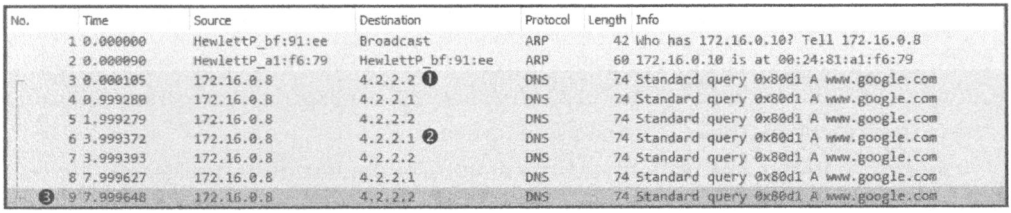


Рис. 10.15. Второй DNS-запрос, посылаемый по адресу **4.2.2.1**



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	HewlettP_bf:91:ee	Broadcast	ARP	42	Who has 172.16.0.10? Tell 172.16.0.8
2	0.000090	HewlettP_a1:f6:79	HewlettP_bf:91:ee	ARP	60	172.16.0.10 is at 00:24:81:a1:f6:79
3	0.000105	172.16.0.8	4.2.2.2 1	DNS	74	Standard query 0x80d1 A www.google.com
4	0.999280	172.16.0.8	4.2.2.1	DNS	74	Standard query 0x80d1 A www.google.com
5	1.999279	172.16.0.8	4.2.2.2	DNS	74	Standard query 0x80d1 A www.google.com
6	3.999372	172.16.0.8	4.2.2.1 2	DNS	74	Standard query 0x80d1 A www.google.com
7	3.999393	172.16.0.8	4.2.2.2	DNS	74	Standard query 0x80d1 A www.google.com
8	7.999627	172.16.0.8	4.2.2.1	DNS	74	Standard query 0x80d1 A www.google.com
3 9	7.999648	172.16.0.8	4.2.2.2	DNS	74	Standard query 0x80d1 A www.google.com

Рис. 10.16. DNS-запросы повторяются до тех пор, пока не будет остановлен процесс их отправки

На основании проанализированных ранее пакетов можно приступить к уточнению причины рассматриваемой здесь проблемы. Прежде всего, мы обнаружим успешный ARP-запрос к стандартному, как предполагается, шлюзу (маршрутизатору) сети, а следовательно, делаем вывод, что данное устройство подключено к сети и находится на связи. Мы знаем также, что компьютер пользователя фактически передает пакеты в сеть, и поэтому можем предположить, что анализируемая проблема не связана со стеком протоколов на самом компьютере, а начинает проявляться, когда делается DNS-запрос.

Что касается рассматриваемой здесь сети, то DNS-запросы обрабатываются внешним DNS-сервером в Интернете (по адресу **4.2.2.2** или **4.2.2.1**). Это означает, что для правильной обработки DNS-запросов шлюз, ответственный за маршрутизацию пакетов в Интернет, должен успешно пересылать DNS-запросы соответствующему серверу, а тот – отвечать на них. И все это должно произойти прежде, чем протокол HTTP будет использован для запроса самой веб-страницы.

Если у других пользователей не возникает никаких трудностей при подключении к Интернету, то сетевой шлюз и удаленный DNS-сервер, вероятнее всего, не являются причиной возникшей проблемы. Поэтому остается лишь исследовать сам компьютер данного пользователя.

Проведя углубленное исследование компьютера, затронутого рассматриваемой здесь проблемой, мы обнаружим, что вместо назначаемого по протоколу DHCP IP-адреса кто-то вручную задал параметры настройки протокола TCP/IP, и поэтому адрес стандартного шлюза указан неверно. Это был адрес какого-то другого устройства, а не маршрутизатора Интернета, поэтому пакеты с DNS-запросами не уходили за пределы локальной сети.

Усвоенные уроки

Проблема в данном сценарии возникла вследствие неверной конфигурации клиента. И хотя она оказалась простой, тем не менее, значительно повлияла на работу пользователя. Диагностика столь простых случаев неверной конфигурации может отнять немало времени у тех, кто слабо разбирается в сетях и не умеет оперативно анализировать пакеты, как в данном сценарии. Как видите, анализ пакетов не ограничивается только разрешением крупных и сложных затруднений, возникающих в сети.

Следует, однако, иметь в виду, что в данном сценарии нам заранее не был известен IP-адрес сетевого шлюза, и поэтому выявить источник проблемы с помощью приложения Wireshark не удалось, хотя оно и подсказало, где его искать, сэкономив нам немало драгоценного времени. Вместо того чтобы анализировать трафик, проходящий через сетевой шлюз, звонить провайдеру Интернета или искать ресурсы для диагностики удаленного DNS-сервера, мы сумели сосредоточить свои усилия на самом компьютере, который, по существу, и оказался причиной данной проблемы.

ПРИМЕЧАНИЕ *Если бы мы лучше знали схему IP-адресации в данной конкретной сети, то смогли бы провести свой анализ быстрее. Проблему можно было бы выявить немедленно, как только обнаружилось бы, что ARP-запросы были посланы не по IP-адресу сетевого шлюза. Подобные ошибки в конфигурации нередко становятся причиной проблем в работе сети, которые, как правило, разрешаются быстро в результате беглого анализа пакетов.*

Нежелательная переадресация

Файл перехвата
`nowebaccess2.pcapng`

В данном сценарии главным действующим лицом снова оказывается пользователь, испытывающий трудности при доступе к Интернету со своего рабочего

компьютера. Но, в отличие от пользователя из предыдущего сценария, он все же может подключиться к Интернету. А затруднение возникает у него в связи с тем, что он не может перейти на свою начальную страницу `https://www.google.com/`. Когда этот пользователь пытается достичь любого домена, хостинг которого выполняется в Google, он направляется на страницу браузера с сообщением `"Internet Explorer cannot display the web page"` (Internet Explorer не может отобразить указанную веб-страницу). Возникающее затруднение затрагивает только данного конкретного пользователя. Как и в предыдущем сценарии, неполадка возникает в небольшой сети с несколькими простыми коммутаторами и одним маршрутизатором, служащим в качестве стандартного шлюза.

Подключение к сети

Чтобы приступить к анализу, придется предложить пользователю попробовать подключиться к Интернету для просмотра начальной страницы по адресу `https://www.google.com/` и в то же время воспользоваться сетевым ответвителем для прослушивания формируемого при этом сетевого трафика. Полученный в итоге результат сохранен в файле перехвата `nowebaccess2.pcapng`.

Анализ

Перехваченный сетевой трафик начинается с ARP-запроса и ответа, как показано на рис. 10.17. В пакете 1 компьютер пользователя с MAC-адресом **`00:25:b3:bf:91:ee`** и IP-адресом **`172.16.0.8`** посылает широковещательный пакет по протоколу ARP всем компьютерам в данном сегменте сети, чтобы попытаться найти MAC-адрес, связанный с IP-адресом хоста **`172.16.0.102`**. Мы не смогли сказать сходу, что это за адрес.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:25:b3:bf:91:ee	ff:ff:ff:ff:ff:ff	ARP	42	Who has 172.16.0.102? Tell 172.16.0.8
2	0.000334	00:21:70:c0:56:f0	00:25:b3:bf:91:ee	ARP	60	172.16.0.102 is at 00:21:70:c0:56:f0

Рис. 10.17. ARP-запрос и ответ другого устройства в сети

Из пакета 2 компьютеру пользователя становится известно, что MAC-адресу **`00:21:70:c0:56:f0`** соответствует IP-адрес **`172.16.0.102`**. Опираясь на опыт из предыдущего сценария, можно предположить, что это адрес сетевого шлюза, используемый для того, чтобы пакеты можно было (как и в прошлом сценарии) пересылать на внешний DNS-сервер. Но, как показано на рис. 10.18, следующий пакет содержит не DNS-запрос, а TCP-запрос, передаваемый из сетевого узла, находящегося по адресу **`172.16.0.8`**, в сетевой узел, расположенный по адресу **`172.16.0.102`**. В заголовке этого пакета установлен флаг `SYN` ❸, указывающий на то, что это первый пакет в процессе установки соединения по протоколу TCP между двумя сетевыми узлами (или хостами).

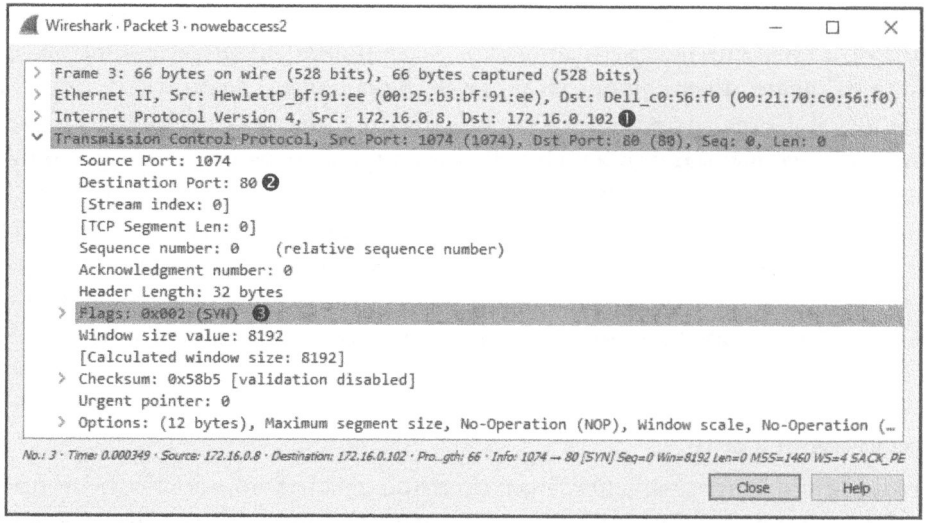


Рис. 10.18. Пакет SYN, посылаемый по протоколу TCP из одного хоста к другому

Примечательно, что соединение по протоколу TCP устанавливается с портом **80** ❷ по адресу **172.16.0.102** ❶, который, как правило, связан с HTTP-трафиком. Как показано на рис. 10.19, попытка такого соединения неожиданно отвергается, когда хост, находящийся по адресу **172.16.0.102**, посылает в ответ пакет TCP с установленными флагами RST и ACK ❶ (т.е. пакет 4).

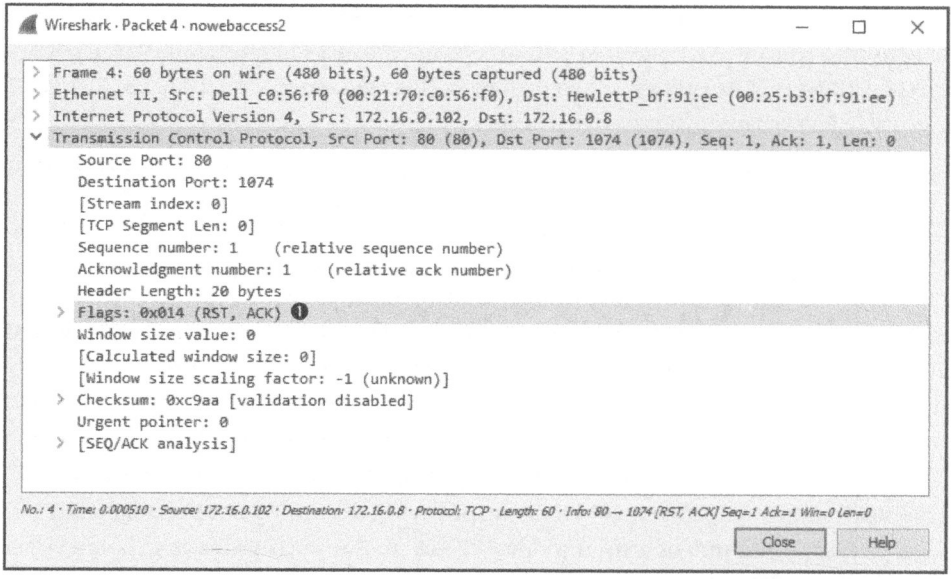


Рис. 10.19. Пакет TCP с установленными флагами RST и ACK, посылаемый в ответ на пакет TCP с установленным флагом SYN

Как упоминалось в главе 8, "Протоколы транспортного уровня", флаг RST служит для разрыва соединения, устанавливаемого по протоколу TCP. В данном случае хост, находящийся по адресу **172.16.0.8**, попытался установить соединение по протоколу TCP с хостом, расположенным по адресу **172.16.0.102**, через порт **80**. Но в ответ, к сожалению, он получил пакет TCP с установленным флагом RST, указывающим на разрыв соединения. Этот процесс повторялся три раза, прежде чем связь между хостами окончательно прекратилась (рис. 10.20). И в этот момент пользователь получил в окне своего браузера сообщение, извещавшее, что нельзя отобразить запрашиваемую страницу.

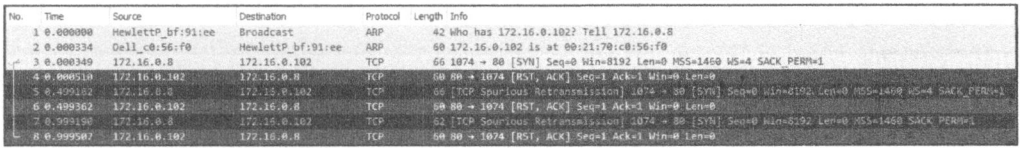


Рис. 10.20. Обмен пакетами SYN и RST по протоколу TCP происходит три раза подряд

После анализа конфигурации другого правильно работающего сетевого устройства обратим внимание на ARP-запрос и ответ в пакетах 1 и 2, поскольку ARP-запрос делается не с целью получения MAC-адреса маршрутизатора сетевого шлюза, а для определения MAC-адреса какого-то неизвестного устройства. Прослеживая ARP-запрос и ответ, мы ожидаем увидеть DNS-запрос, отправленный по указанному нами адресу DNS-сервера с целью найти IP-адрес, связанный с доменным именем **https://www.google.com/**, но мы этого не находим. Имеются следующие два условия, способные помешать сделать такой DNS-запрос.

- Устройство, инициирующее соединение, уже содержит в своем кеше DNS-запросов отображение DNS-имени на IP-адрес, как и в первом сценарии из этой главы.
- Устройство, с которым устанавливается соединения по DNS-имени, уже содержит отображение DNS-имени на IP-адрес, указанное в его файле hosts.

Дальнейшее исследование конфигурации клиентского компьютера показывает, что в его файле hosts имеется запись для доменного имени **https://www.google.com**, связанного с внутренним IP-адресом **172.16.0.102**. Именно эта неверная запись и является главной причиной затруднений, возникших у данного пользователя.

Как правило, компьютер использует свой файл hosts в качестве авторитетного источника соответствий DNS-имен и IP-адресов, и поэтому он

обращается именно к этому файлу, прежде чем запрашивать внешний источник. В данном сценарии компьютер пользователя проверил сначала свой файл hosts, обнаружил в нем запись для доменного имени **https://www.google.com** и решил, что хост под этим именем находится в его локальном сегменте сети. Затем он послал этому хосту ARP-запрос, получил ответ и попытался инициировать соединение по протоколу TCP через порт **80** по адресу **172.16.0.102**. Но поскольку удаленная система не была сконфигурирована как веб-сервер, то она отвергла все попытки установить такое соединение. И как только соответствующая запись была удалена из файла hosts, компьютер пользователя начал правильно устанавливать связь и сумел получить доступ к веб-серверу по адресу https://www.google.com/.

ПРИМЕЧАНИЕ *Чтобы проверить содержимое своего файла* **hosts** *в системе Windows, откройте его для просмотра, воспользовавшись полным именем* **C:\Windows\System32\drivers\etc\hosts**, *а в системе Linux –* **/etc/hosts**.

Это весьма распространенный сценарий, которым годами использовался во вредоносном программном обеспечении, чтобы переадресовать пользователей на веб-сайты с вредоносным кодом. Так, если атакующий злоумышленник внесет изменения в файл hosts, то всякий раз, когда пользователь собирается выполнить онлайновые банковские операции, он будет переадресован на поддельный веб-сайт, специально предназначенный для кражи учетных данных этого пользователя!

Усвоенные уроки

Продолжая анализировать сетевой трафик, вы можете не только узнать, каким образом действуют различные сетевые протоколы, но и как можно нарушить их нормальную работу. В данном сценарии DNS-запрос не был послан из-за неверной конфигурации клиента, а не вследствие каких-то внешних ограничений или конфигураций.

Анализируя данную проблему на уровне пакетов, мы сумели быстро выявить неизвестный IP-адрес и выяснить отсутствие DNS-запроса на преобразование имени – главной составляющей подобного процесса обмена данными по сети. Используя эти сведения, нам удалось установить, что главной причиной возникшей проблемы был сам клиент.

Проблемы с обратным потоком данных

Файл перехвата **nowebaccess3.pcapng**

Как и в двух предыдущих сценариях, в данном сценарии пользователь жалуется на отсутствие доступа к Интернету из его рабочего компьютера. Этот пользователь локализовал возникшую проблему к единственному веб-сайту по адресу `https://www.google.com/`. Но в дальнейшем исследование показало, что данная проблема возникала у всех сотрудников организации: никому из них не удавалось получить доступ к доменам компании Google. Сеть в этой организации имеет такую же архитектуру, как и в двух предыдущих сценариях, и состоит из нескольких простых коммутаторов и одного маршрутизатора, через который происходит подключение к Интернету.

Подключение к сети

Чтобы выявить причину возникшей проблемы, попытаемся обратиться по адресу `https://www.google.com/` для просмотра содержимого веб-сайта и формирования сетевого трафика. Данная проблема распространена по всей сети, и поэтому в идеальном случае она должна воспроизводиться на любом устройстве сети с помощью большинства методик перехвата. Результат перехвата пакетов через сетевой ответвитель сохранен в файле `nowebaccess3.pcapng`.

Анализ

Рассматриваемый здесь перехват пакетов начинается с сетевого трафика по протоколу DNS, а не ARP, как мы уже привыкли наблюдать. Первый пакет в этом перехвате посылается по внешнему адресу, а второй содержит ответ от сетевого узла, находящегося по этому адресу, поэтому мы можем предположить, что ARP-процесс уже произошел и соответствие MAC-адреса IP-адресу маршрутизатора сетевого шлюза уже существует в ARP-кеше сетевого узла, расположенного по адресу **`172.16.0.8`**.

Как показано на рис. 10.21, первый пакет в рассматриваемом здесь перехвате посылается хостом, находящимся по адресу **`172.16.0.8`**, хосту, расположенному по адресу **`4.2.2.1`** ❶, и относится к протоколу DNS ❷. Анализируя содержимое этого пакета, можно обнаружить, что в нем содержится запрос записи типа **`A`** для преобразования доменного имени **`www.google.com`** ❸ в соответствующий ему IP-адрес.

Ответ на данный запрос от хоста, расположенного по адресу **`4.2.2.1`**, находится во втором пакете из рассматриваемого здесь файла перехвата, как показано на рис. 10.22. Здесь мы видим, что сервер доменных имен, откликнувшийся на данный запрос, прислал несколько ответов на него ❶. Пока что все вроде бы хорошо, и обмен данными происходит должным образом.

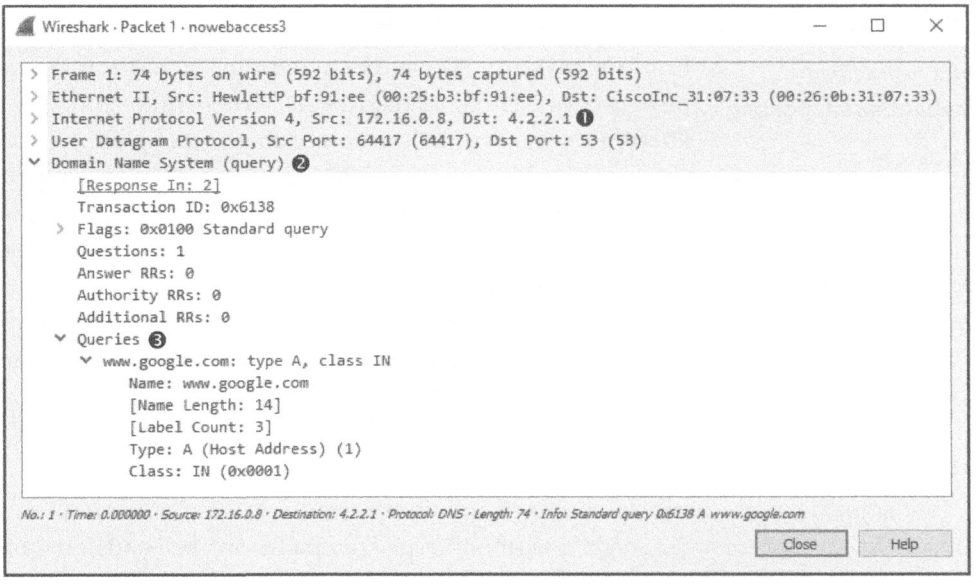


Рис. 10.21. DNS-запрос записи типа **A** *для преобразования доменного имени* **www.google.com** *в IP-адрес*

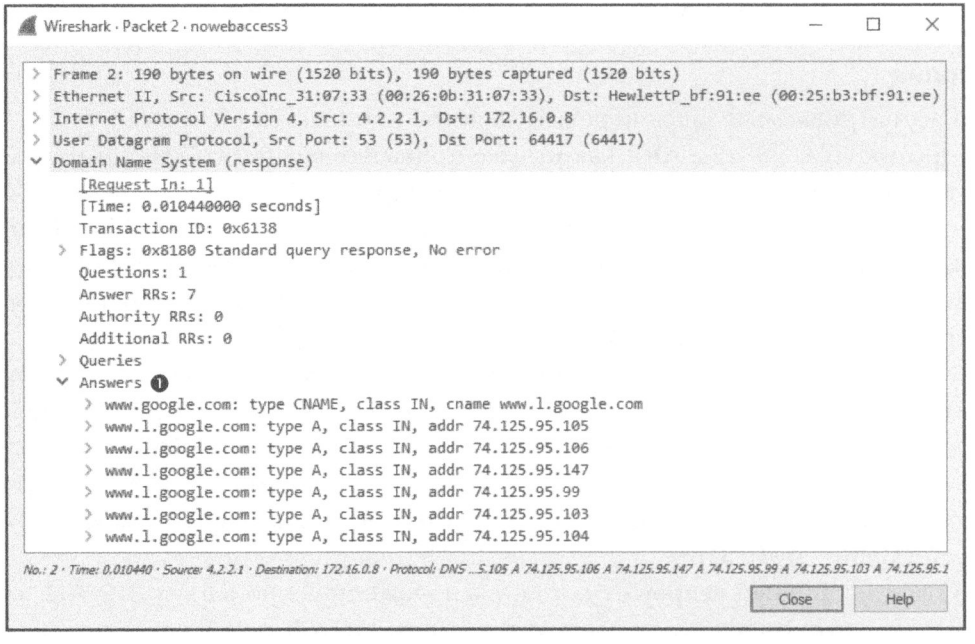


Рис. 10.22. DNS-ответ с несколькими записями типа **A**

А теперь, когда компьютер пользователя определил IP-адрес веб-сервера, он может попытаться установить связь с ним. Как показано на рис. 10.23,

этот процесс инициируется в пакете 3, посылаемом по протоколу TCP из хоста, находящегося по адресу **172.16.0.8**, хосту, расположенному по адресу **74.125.95.105** ❶. Этот адрес получателя взят из первой записи типа **A** в DNS-ответе, который был прислан в рассмотренном выше пакете 2. А в данном пакете TCP установлен флаг SYN ❸ и предпринимается попытка связаться с удаленным сервером через порт **80** ❷.

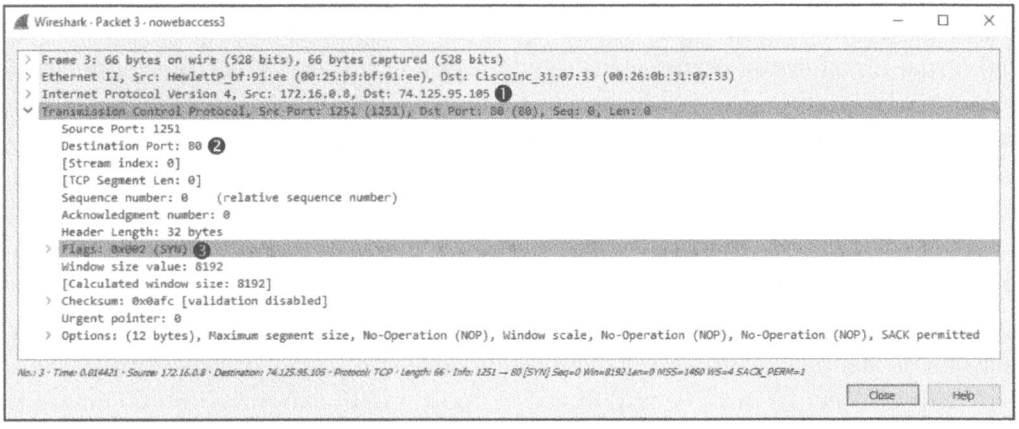


Рис. 10.23. Пакет SYN, посылаемый с целью попытаться установить соединение с удаленным сервером через порт **80**

В данном случае происходит процесс установки связи по протоколу TCP, и поэтому мы вправе ожидать появления пакета SYN/ACK, посылаемого в ответ по протоколу TCP. Но вместо этого некоторое время спустя отправитель посылает еще один пакет SYN своему адресату. Этот процесс повторяется еще раз приблизительно через одну секунду, как показано на рис. 10.24. И в этот момент все попытки установить связь прекращаются, а браузер сообщает, что найти веб-сайт по указанному адресу не удалось.

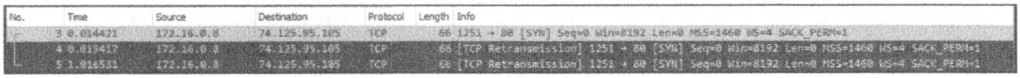


Рис. 10.24. Пакет SYN посылался по протоколу TCP три раза подряд, чтобы попытаться установить связь с удаленным сервером, но ответ на него так и не был получен

Проводя диагностику сети в данном сценарии, следует иметь в виду, что рабочий компьютер в этой сети может подключаться к внешней сети благодаря получению успешного ответа на DNS-запрос к внешнему DNS-серверу, расположенному по адресу **4.2.2.1**. На этот запрос DNS-сервер ответил корректным адресом, и поэтому хосты в диагностируемой здесь сети пытаются связаться по одному из адресов, полученных в ответ. Кроме того, локальный

компьютер, из которого предпринимается попытка подключиться к Интернету, функционирует нормально.

Но проблема состоит в том, что удаленный сервер просто не отвечает на запросы установки соединения и не присылает в ответ пакет RST по протоколу TCP. И это может произойти по нескольким причинам: неверно сконфигурированный веб-сервер, нарушена работа стека протоколов на веб-сервере или включена фильтрация пакетов неким устройством (например, брандмауэром) в удаленной сети. Если предположить, что в локальной сети устройство фильтрации пакетов отсутствует, то все остальные возможные решения относятся к удаленной сети, которая нам не подвластна. В данном случае веб-сервер функционировал неверно, и любые попытки получить к нему доступ оказались безуспешными. Но как только затруднение было устранено на стороне веб-сервера компании Google, связь с ним удалось восстановить.

Усвоенные уроки

В данном сценарии решить проблему оказалось не в наших силах. Как показал наш анализ, возникшее затруднение не было связано ни с хостами в диагностируемой сети, ни с ее маршрутизатором или предоставлением внешним DNS-сервером услуг по преобразованию имен в IP-адреса. Данная проблема возникла за пределами инфраструктуры диагностируемой сети.

Иногда выявление проблемы, решить которую не в нашей власти, не только снимает напряжение в коллективе, но и не роняет наше достоинство, когда начальство требует ответа. Мне лично приходилось немало воевать с поставщиками услуг Интернета и программного обеспечения, которые заявляли, что возникшая проблема не является следствием их оплошности, но, как видите, пакеты не лгут.

Испорченный принтер

В следующем сценарии у администратора диспетчерской службы технической поддержки в отделе информационных технологий возникли трудности с выводом на печать. Пользователи из отдела продаж сообщают, что высокопроизводительный принтер не работает. Когда пользователь посылает этому принтеру крупное задание на печать, он сначала напечатает несколько страниц, а затем прерывает печать еще до завершения данного задания. Было предпринято несколько попыток изменить параметры конфигурации драйвера принтера, но все они оказались безуспешными. Сотрудники диспетчерской службы технической поддержки предположили, что скорее всего это не сетевая проблема.

Подключение к сети

Файл перехвата **inconsistent_printer.pcapng**

Главной причиной данной проблемы является принтер, поэтому начнем с размещения анализатора пакетов как можно ближе к принтеру. И хотя мы не можем установить Wireshark на самом принтере, коммутаторы, применяемые в сети, к которой подключен диагностируемый принтер, относятся к категории продвинутых коммутаторов третьего уровня. Следовательно, мы можем воспользоваться зеркальным отображением портов, чтобы отобразить порт, используемый принтером, на пустой порт коммутатора и подключить к нему ноутбук с установленным приложением Wireshark. Как только это будет сделано, нам придется предложить пользователю отправить на диагностируемый принтер крупное задание на печать, чтобы мы могли проконтролировать этот вывод. Полученный в итоге результат сохранен в файле перехвата `inconsistent_printer.pcapng`.

Анализ

В процессе установки связи по протоколу TCP между рабочим компьютером (с адресом **172.16.0.8**), посылающим принтеру (с адресом **172.16.0.253**) ❶ задание на печать по сети, инициируется запрос на соединение, как зафиксировано вначале файла перехвата. Прослеживая подтверждение связи по протоколу TCP, можно обнаружить, что в пакете 4 порция данных объемом 1460 байт посылается принтеру по протоколу TCP ❷ (рис. 10.25). Объем передаваемых данных представлен в столбце **Info** справа в панели Packet List или под TCP-заголовком в панели Packet Details.

Прослеживая далее пакет 4, можно также обнаружить очередную порцию данных объемом 1460 байт, посылаемых принтеру ❶, как показано на рис. 10.26. Прием этих данных подтверждается принтером в пакете 6 ❷.

Поток передаваемых данных продолжается до тех пор, пока не будут отправлены последние несколько пакетов, зафиксированные в рассматриваемом здесь файле перехвата. Признак проблемы обнаруживается в пакете 121, отправляемом по протоколу TCP, как показано на рис. 10.27.

Повторная передача пакета по протоколу TCP осуществляется в том случае, если одно устройство посылает пакет TCP другому, удаленному устройству, а то, в свою очередь, не подтверждает прием переданного ему пакета. И как только будет достигнут порог повторной передачи, передающее устройство посчитает, что удаленное устройство не получило переданные ему данные, и повторит передачу пакета. Этот процесс продолжается несколько раз, прежде чем обмен данными прекратится.

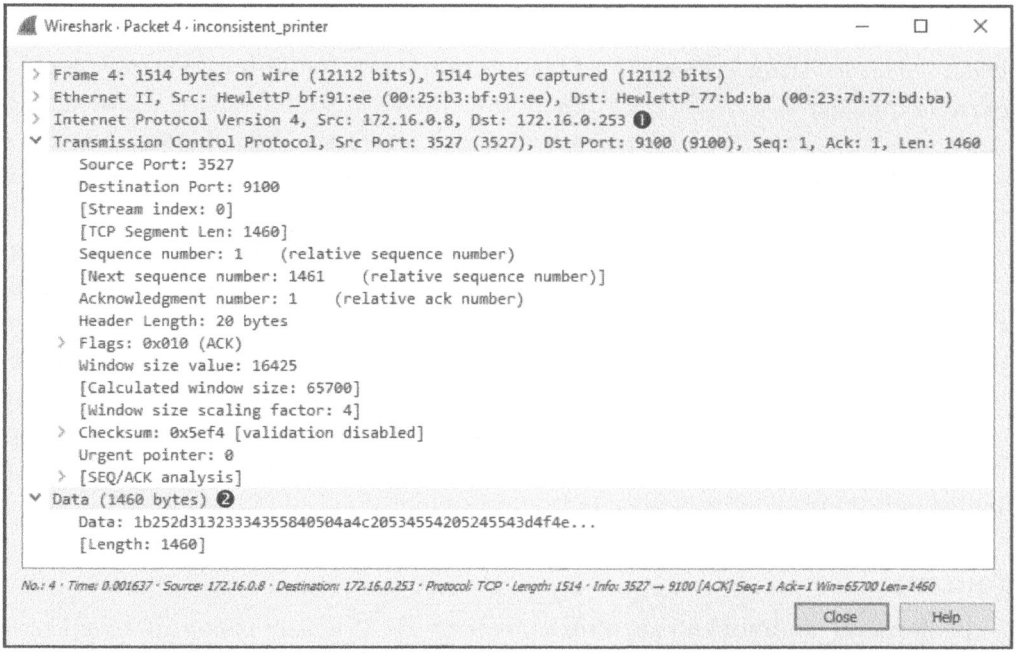


Рис. 10.25. Данные, передаваемые принтеру по протоколу TCP

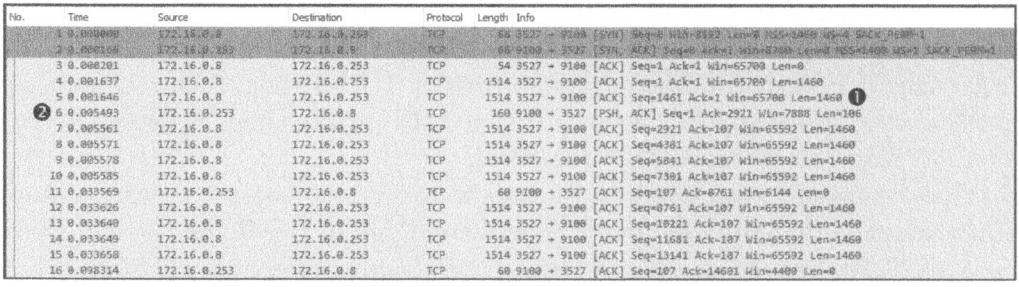


Рис. 10.26. Обычный режим передачи данных и подтверждения их приема по протоколу TCP

В данном сценарии повторно передаваемые данные посылаются из компьютера клиента на принтер, поскольку принтеру не удалось подтвердить прием переданных ему данных. Как показано на рис. 10.27, причины для такой повторной передачи можно выяснить более подробно, если развернуть раздел **`SEQ/ACK analysis`** из TCP-заголовка ❶ вместе с указанными под ним сведениями. В соответствии с подобными результатами обработки перехваченных пакетов в Wireshark пакет 121 является повторно переданным пакетом 120 ❸. Кроме того, время ожидания до повторной передачи (RTO) пакета составило около 5,5 с ❷.

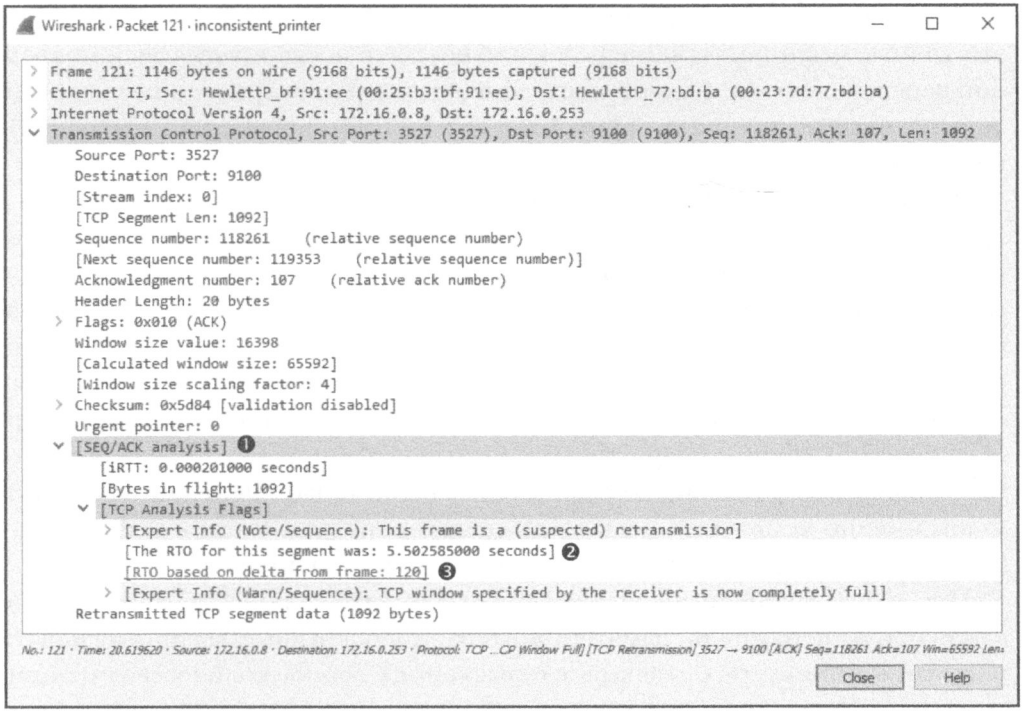


Рис. 10.27. Эти пакеты, переданные по протоколу TCP, служат признаком потенциальной проблемы в сети

Анализируя задержку между пакетами, можно изменить формат отображения времени применительно к конкретной ситуации. В данном случае требуется выяснить, как долго продолжались повторные передачи данных после отправки предыдущего пакета, поэтому измените этот формат, выбрав команду View⇨Time Display Format⇨Seconds Since Previous Captured Packet (Вид⇨Формат времени отображения⇨Количество секунд, прошедших после предыдущего перехваченного пакета) из главного меню. И тогда можно ясно увидеть (рис. 10.28), что повторная передача данных в пакете 121 произошла через 5,5 с после отправки первоначального пакета 120 ❶.

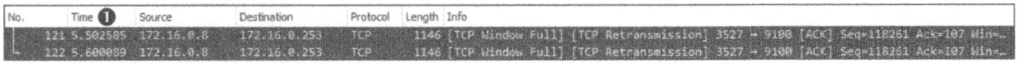


Рис. 10.28. Для диагностики сети очень удобно просматривать промежуток времени, прошедший между пакетами

В следующем пакете происходит очередная повторная передача пакета 120. Время ожидания до повторной передачи этого пакета составляет уже 11,10 с, включая время ожидания до повторной передачи предыдущего пакета, равное

5,5 с. Как показывает анализ содержимого столбца **Time** в панели Packet List, эти данные были повторно переданы через 5,6 с после предыдущей повторной передачи. А поскольку это последний пакет в рассматриваемом здесь файле перехвата, то именно в данный момент принтер прекращает печать, что нельзя считать чистым совпадением.

В данном сценарии мы можем анализировать работу лишь двух устройств в локальной сети, поэтому нам достаточно выяснить, кто является виновником возникшей проблемы: компьютер клиента или принтер. Как видите, поток данных правильно проходит по сети в течение определенного промежутка времени, но затем наступает момент, когда принтер перестает отвечать клиентскому компьютеру. Со своей стороны компьютер клиента делает все возможное, чтобы доставить данные получателю, о чем свидетельствуют повторные передачи, но его усилия не находят отклика в принтере. Анализируемая здесь проблема неизменно воспроизводится и возникает независимо от того, какой именно компьютер посылает задание на печать. Из этого можно сделать вывод, что причиной данной проблемы является принтер.

Проведя дальнейший анализ, мы обнаружим, что в процессе работы возникают сбои в оперативной памяти принтера. Когда крупные задания на печать посылаются принтеру, он печатает только определенное количество страниц, вероятно, до тех пор, пока не будут достигнуты сбойные участки оперативной памяти. И в этот момент сбои в работе оперативной памяти приведут к тому, что принтер будет неспособен принять любые новые данные и поэтому прекратит обмен данными с хостом, передавшим ему задание на печать.

Усвоенные уроки

Несмотря на то что возникшая с принтером проблема не связана с неполадками в сети, нам все же удалось выявить ее с помощью Wireshark. В отличие от предыдущих сценариев, анализ в данном случае был сосредоточен исключительно на сетевом трафике по протоколу TCP. А поскольку протокол TCP служит для надежной передачи данных, то он нередко предоставляет нам немало полезных сведений, когда два устройства перестают обмениваться данными по сети.

В данном случае нам удалось выявить момент внезапного прекращения обмена данными с принтером, опираясь только на информацию о повторной передаче данных, которая встроена в протокол TCP. И в последующих сценариях мы будем нередко полагаться на функциональные возможности сетевых протоколов, чтобы выявить и устранить более сложные неполадки в сети.

Отсутствие связи с филиалом

Файлы перехвата **stranded_clientside.pcapng** *и* **stranded_branchdns.pcapng**

В данном сценарии речь идет о компании с центральной штаб-квартирой и недавно организованным удаленным филиалом. Информационно-технологическая инфраструктура этой компании сосредоточена, главным образом, в ее центральной штаб-квартире, где используется домен на основе сервера под управлением системы Windows. Эта инфраструктура поддерживается контроллером домена, DNS-сервером и сервером приложений, где размещается веб-ориентированное программное обеспечение, которым ежедневно пользуются работники компании. Филиал подключается к центральной штаб-квартире через маршрутизаторы для организации канала связи через глобальную сеть. А в самом филиале находятся пользовательские компьютеры и резервный DNS-сервер, который должен получать информацию о записях ресурсов от основного DNS-сервера, расположенного в центральной штаб-квартире. Блок-схема расположения каждого подразделения компании со связями между ними приведена на рис. 10.29.

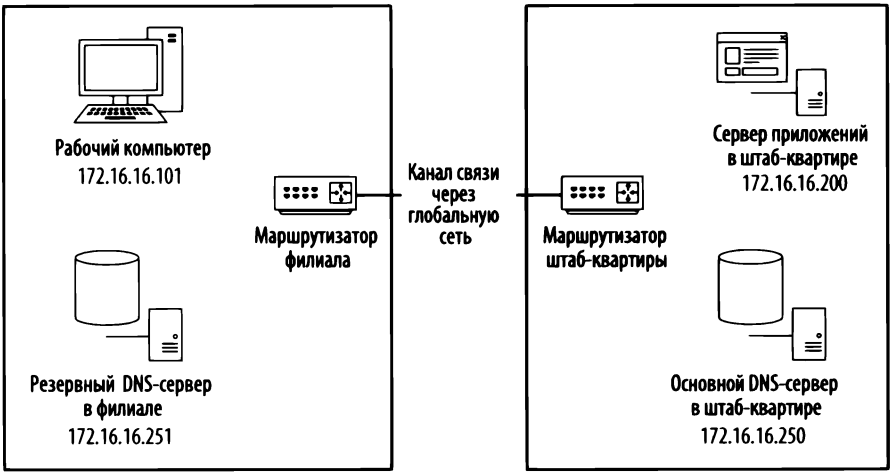


Рис. 10.29. Схема подключения соответствующих компонентов филиала, с которым потеряна связь

Развертывая новую инфраструктуру в филиале, команда специалистов неожиданно обнаруживает, что никому из пользователей сети этого филиала недоступен сервер веб-приложений, находящийся во внутренней корпоративной сети. Этот сервер расположен в центральной штаб-квартире и подключается через канал связи, проходящий через глобальную сеть. Рассматриваемая здесь проблема, возникшая из-за отсутствия связи с центральным офисом, оказала влияние на всех пользователей, работающих в данном филиале. При

этом все пользователи филиала могут получать доступ к любым ресурсам своей локальной сети, а также без проблем выходить в Интернет.

Подключение к сети

Данная проблема заключается в том, что штаб-квартира не может обмениваться данными с филиалами, а следовательно, нетрудно найти хотя бы пару мест, где можно было бы собрать данные, чтобы приступить к отслеживанию возникшего затруднения. Проблему обнаружили клиенты в филиале, и поэтому мы начали с зеркального отображения портов на компьютере одного из этих клиентов, чтобы проверить, что именно ему доступно в сети. Собрав всю необходимую информацию, мы можем с ее помощью перейти в другие места сбора данных, чтобы разрешить возникшее затруднение. Сетевой трафик, перехваченный на компьютере одного из клиентов, в конечном итоге сохранен в файле `stranded_clientside.pcapng`.

Анализ

Как показано на рис. 10.30, сетевой трафик в первом из упоминаемых здесь файлов перехвата начинается в тот момент, когда пользователь компьютера, находящегося по адресу **`172.16.16.101`**, пытается получить доступ к приложению, размещающемуся на сервере приложений, который развернут в центральной штаб-квартире с присвоенным ему адресом **`172.16.16.200`**. Этот файл перехвата содержит лишь два пакета. Их анализ показывает, что по адресу **`172.16.16.251`** ❶ посылается DNS-запрос на запись **`appserver`** ❷ типа **`A`** ❸ в первом пакете. В нем указано DNS-имя сервера приложений, расположенного в центральной штаб-квартире по адресу **`172.16.16.200`**.

Как показано на рис. 10.31, в качестве реакции на этот пакет происходит отказ сервера ❶, который указывает на нечто, мешающее успешному преобразованию доменного имени, указанного в DNS-запросе. Обратите внимание на то, что в данном пакете отсутствует ответ на запрос ❷, поскольку произошла ошибка (отказ сервера).

Теперь нам известно, что трудности обмена данными по сети связаны с некоторой проблемой, возникающей в работе службы DNS. Обработка всех DNS-запросов из филиала выполняется на локальном DNS-сервере, расположенном по адресу **`172.16.16.251`**, и поэтому займемся далее именно им.

Чтобы перехватить подходящий сетевой трафик из DNS-сервера, развернутого в филиале, мы оставим свой анализатор пакетов на месте и просто изменим назначение зеркального отображения портов, чтобы на наш анализатор пакетов зеркально отображался трафик из интересующего нас DNS-сервера, а не компьютера клиента. Полученный в итоге результат сохранен в файле `stranded_branchdns.pcapng`.

Рис. 10.30. Обмен данными начинается с DNS-запроса записи типа **A** *с именем* **appserver**

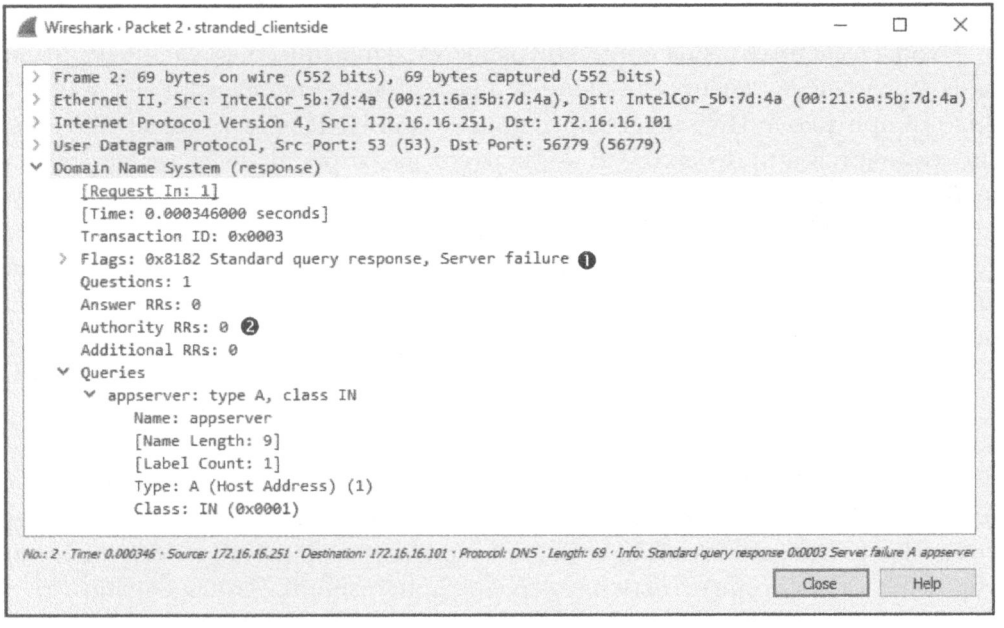


Рис. 10.31. Реакция на DNS-запрос указывает на проблему вне филиала

Как показано на рис. 10.32, анализируемый здесь перехваченный сетевой трафик начинается с рассмотренного ранее запроса и соответствующего

ответа наряду с еще одним дополнительным пакетом. Этот дополнительный пакет выглядит несколько странно, поскольку в нем предпринимается попытка связаться с основным DNS-сервером, развернутым в центральной штаб-квартире, с присвоенным ему адресом **172.16.16.250** ❶ через его стандартный порт **53** ❸, но это происходит совсем не по протоколу UDP ❷, как мы к тому привыкли.

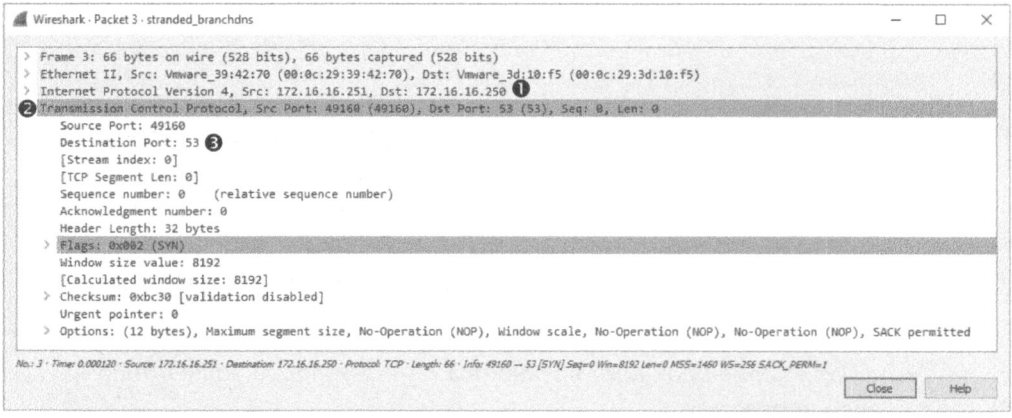


Рис. 10.32. В этом пакете SYN применяется стандартный порт **53**, *но не протокол UDP*

Чтобы выяснить назначение этого пакета, напомним обсуждение протокола DNS в главе 9, "Распространенные протоколы верхнего уровня". Как правило, в протоколе DNS используется протокол UDP, но в то же время в нем употребляется и протокол TCP, когда ответ на запрос превышает определенный размер. В данном случае мы видим первоначальный сетевой трафик по протоколу UDP, после чего инициируется трафик по протоколу TCP. Кроме того, протокол TCP используется в протоколе DNS во время переноса зоны, когда записи ресурсов передаются между DNS-серверами, что вполне возможно и в данном случае.

DNS-сервер, развернутый в филиале, является резервным по отношению к основному DNS-серверу, размещенному в центральной штаб-квартире. Это означает, что для получения записей ресурсов он должен запросить их у основного DNS-сервера. А сервер приложений, доступ к которому пользователи из филиала пытаются получить, также расположен в центральной штаб-квартире. И это означает, что DNS-сервер, находящийся в центральной штаб-квартире, является авторитетным для сервера приложений. Чтобы обработать на сервере, развернутом в филиале, DNS-запрос на получение IP-адреса сервера приложений, запись ресурсов для этого сервера необходимо перенести из DNS-сервера в центральной штаб-квартире на DNS-сервер в филиале. И это наиболее вероятная причина появления пакета SYN в рассматриваемом здесь файле перехвата.

Отсутствие ответа на этот пакет SYN свидетельствует о том, что проблемы в службе DNS возникли в результате неудачного переноса зоны из DNS-сервера в центральной штаб-квартире на DNS-сервер в филиале. И теперь мы можем перейти непосредственно к выяснению причины неудачного переноса зоны. Наиболее вероятными виновниками этой неудачной операции могут быть маршрутизаторы, размещаемые между DNS-серверами в филиалах и центральной штаб-квартире. Чтобы выявить среди них отказавший сервер, можно проанализировать сетевой трафик DNS-сервера в центральной штаб-квартире и определить, доходит ли пакет SYN до этого сервера.

Сетевой трафик DNS-сервера в центральной штаб-квартире не включен в рассматриваемый здесь файл перехвата потому, что он просто отсутствует, а следовательно, пакет SYN вообще не доходит до этого сервера. После распределения обязанностей технических специалистов для анализа конфигурации маршрутизаторов, соединяющих два подразделения компании, было обнаружено, что порт **53**, через который входящий трафик поступает на маршрутизатор в центральной штаб-квартире, был настроен на прохождение только трафика по протоколу UDP и блокировку входящего трафика по протоколу TCP. И эта простая ошибка в конфигурации помешала переносу зоны из одного DNS-сервера на другой, а следовательно, и клиентам из филиала выполнить запросы на преобразование имен устройств, находящихся в центральной штаб-квартире.

Усвоенные уроки

Чтобы научиться расследовать неполадки в сети, рекомендуется внимательно смотреть детективные фильмы. Когда совершается преступление, детективы начинают расследование с допросов тех, кого оно большего всего затронуло. Это дает путеводную нить для дальнейшего расследования, которое продолжается до тех пор, пока преступник не будет найден.

В данном сценарии мы начали свой анализ с изучения целевого объекта (рабочего компьютера), получив путеводную нить, которая и привела нас к причине затруднения, возникшего при обмене данными по протоколу DNS. В частности, путеводная нить привела нас сначала к DNS-серверу в филиале, затем к DNS-серверу в центральной штаб-квартире и, наконец, к маршрутизатору, который и послужил причиной возникшего затруднения.

Выполняя анализ пакетов, старайтесь рассматривать их как ключи к разгадке. И хотя такие ключи не всегда сообщают о том, кто совершил преступление, зачастую они приводят в конечном итоге к самому виновнику.

Повреждение данных программы

Файл перехвата
tickedoffdeveloper.pcapng

К числу наиболее частых споров среди специалистов из отдела информационных технологий относятся перепалки между разработчиками и сетевыми администраторами. В частности, разработчики всегда сваливают вину за программные ошибки на неудачное проектирование сетей и неудовлетворительно работающее сетевое оборудование. А сетевые администраторы обвиняют программистов в том, что они написали плохой код, ставший причиной возникновения ошибок в сети и ее медленной работы.

В данном сценарии программист разработал приложение для отслеживания продаж в сети магазинов и сообщения полученных результатов в центральную базу данных. Чтобы сильно не нагружать сеть в рабочее время это приложение не обновляет данные в реальном режиме. Вместо этого данные накапливаются в течение всего рабочего дня и затем передаются ночью в виде текстового файла формата CSV (значения, разделяемые запятыми), для последующего ввода в центральную базу данных.

Это вновь разработанное приложение функционирует не совсем верно. В частности, сервер получает файлы, посылаемые из магазинов, но данные, вводимые в центральную базу данных, оказываются неверными. Так, в них отсутствуют некоторые разделы или порции данных, а иногда они оказываются не на своем месте. К немалому разочарованию сетевого администратора программист винит во всем сеть: он убежден, что файлы повреждаются в процессе их переноса из магазинов в центральное информационное хранилище. Поэтому наша цель – выяснить, насколько он прав.

Подключение к сети

Чтобы собрать необходимые данные, мы можем перехватить пакеты в одном из магазинов или в центральном офисе. Рассматриваемая здесь проблема оказывает влияние на функционирование всех магазинов, поэтому она должна возникать в центральном офисе, если это имеет какое-то отношение к сети. И это единственная нить, связующая все магазины, помимо самого программного обеспечения.

Сетевые коммутаторы поддерживают зеркальное отображение портов, поэтому мы отобразим зеркально порт, к которому подключен сервер, и перехватим его сетевой трафик. Этот трафик ограничивается единственным магазином, выгружающим свой файл формата CSV на сервер сбора данных. Полученный в итоге результат сохранен в файле перехвата `tickedoffdeveloper.pcapng`.

Анализ

О приложении, разработанном программистом, нам неизвестно ничего, кроме основного потока информации в сети. Рассматриваемый здесь файл перехвата начинается с сетевого трафика по протоколу FTP, поэтому исследуем его, чтобы выяснить, действительно ли он является основным механизмом для переноса файла формата CSV.

Заглянув сначала в список пакетов (рис. 10.33), мы можем заметить, что хост, находящийся по адресу **172.16.16.128** ❶, инициирует обмен данными с хостом, расположенным по адресу **172.16.16.121** ❷, начиная с установки соединения по протоколу TCP. Таким образом, мы можем предположить, что хост, находящийся по адресу **172.16.16.128**, является клиентом, передающим данные на обработку, а хост, расположенный по адресу **172.16.16.121**, — сервером, где собираются и обрабатываются данные, полученные от клиента. Прослеживая процедуру установки связи по протоколу TCP, мы начинаем замечать FTP-запросы клиента и ответы сервера ❸.

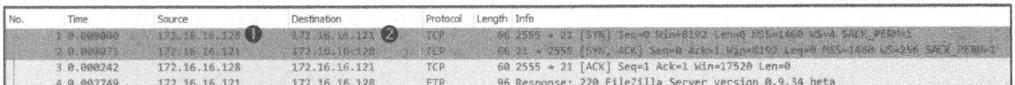


Рис. 10.33. Первоначальный обмен данными помогает отличить клиента от сервера

Нам известно, что в данном случае должна происходить передача некоторых данных, поэтому, используя свои знания протокола FTP, попробуем найти пакет, с которого начинается эта передача. Соединение и передача данных по протоколу FTP инициируются клиентом, а следовательно, от хоста, находящегося по адресу **172.16.16.128**, следует ожидать выдачу команды **STOR**, применяемой в протоколе FTP для выгрузки данных на FTP-сервер. Чтобы обнаружить эту команду, проще всего составить фильтр.

В рассматриваемом здесь файле перехвата имеется немало команд FTP-запросов, и мы можем составить нужный нам фильтр непосредственно в панели Packet List вместо того, чтобы перебирать сотни протоколов и их опций в построителе выражений. Для этого необходимо сначала выбрать пакет с доступной в нем командой FTP-запроса. Выберите пакет 5, поскольку он находится ближе к началу списка. Затем разверните разделы **FTP** и **USER** из выбранного пакета в панели Packet Details. Щелкните правой кнопкой мыши на поле **Request Command: USER** и выберите команду Prepare a Filter⇨Selected (Подготовить фильтр⇨Выбранный) из контекстного меню.

В итоге фильтр для всех пакетов, содержащих команду **USER** для FTP-запроса, будет подготовлен и размещен в диалоговом окне фильтрации. Далее отредактируйте этот фильтр, заменив слово **USER** словом **STOR** ❶, как показано на рис. 10.34.

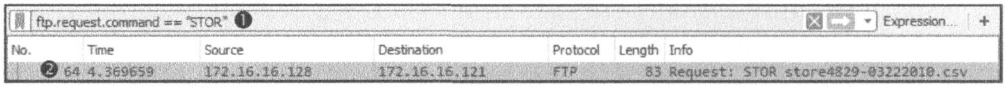


Рис. 10.34. Этот фильтр помогает выяснить, где начинается передача данных

Этот фильтр можно было бы конкретизировать еще больше, выбрав IP-адрес клиента и указав его в качестве отправителя в дополнительном условии **&& ip.src == 172.16.16.128**. Но поскольку рассматриваемый здесь файл перехвата и так ограничен одним клиентом, то особой необходимости в таком дополнении фильтра нет.

А теперь примените этот фильтр, нажав клавишу <Enter>. В итоге появится единственный экземпляр команды **STOR**, имеющийся в пакете 64 ❷ из рассматриваемого здесь файла перехвата. Зная теперь, где начинается передача данных, щелкните на пакете 64, чтобы выбрать его, а затем очистите фильтр, щелкнув на кнопке X, расположенной выше панели Packet List. На экране появятся все пакеты из перехваченного трафика, среди которых будет выбран пакет 64.

Анализируя данный файл перехвата, начиная с пакета 64, можно заметить, что в этом пакете указана передача файла `store4829-03222010.csv` ❶ (рис. 10.35).

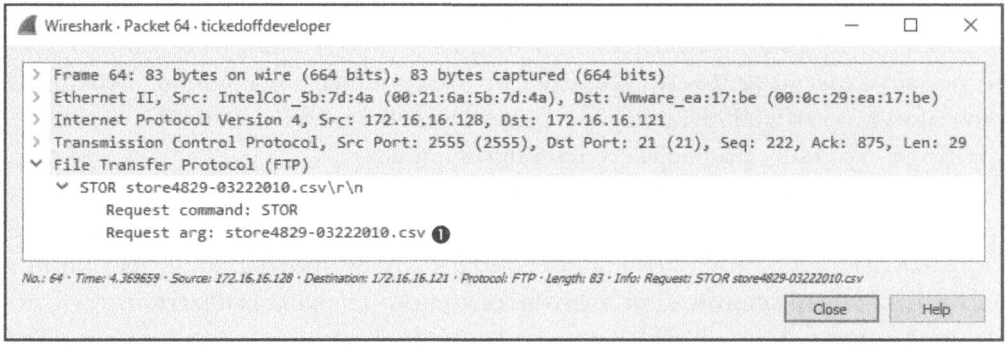


Рис. 10.35. Файл формата CSV, передаваемый по протоколу FTP

В пакетах, следующих после команды **STOR**, указан другой порт, хотя они и обозначаются как часть передачи данных типа `FTP-DATA`. Итак, мы убедились, что данные передаются, но еще не выяснили, был ли программист прав. Для этого необходимо убедиться, остается ли содержимое переданного файла

невредимым после переноса по сети. Перейдем далее к извлечению этого файла из перехваченных пакетов.

Когда файл передается по сети в незашифрованном формате, он разбивается на сегменты и собирается снова в месте его получения. В данном сценарии мы перехватывали пакеты по мере их поступления к получателю, но прежде, чем они собирались снова. Таким образом, в нашем распоряжении имеются все данные, и нам остается лишь собрать их вместе, извлекая файл в виде потока данных. С этой целью выберите любой пакет из потока данных типа `FTP-DATA` (например, пакет 66), щелкните на нем правой кнопкой мыши и выберите команду Follow⇨TCP Stream (Отслеживать⇨Поток TCP) из контекстного меню. Полученный в итоге результат показан на рис. 10.36. Он похож на обычный текстовый файл формата CSV, содержащий данные о продажах.

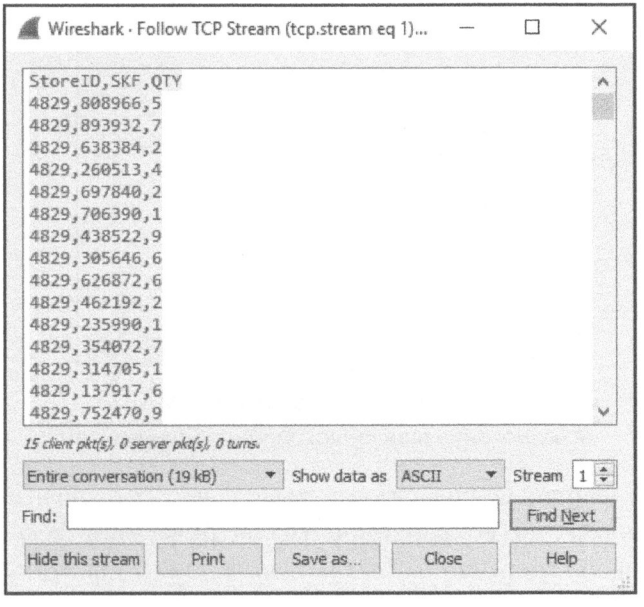


Рис. 10.36. В TCP-потоке можно увидеть, какие именно данные передаются

Данные можно увидеть в анализируемом потоке потому, что они передаются в незашифрованном виде по протоколу FTP. Но, основываясь только на потоке, нельзя с уверенностью судить, что переданный по сети файл остался невредимым. Чтобы собрать данные снова и сохранить их в виде исходного файла, щелкните на кнопке Save As и укажите то имя файла, которое обозначено в пакете 64, а затем щелкните на кнопке Save.

В результате операции сохранения будет создан файл формата CSV, являющийся точной копией на уровне отдельных байтов того файла, который

был первоначально передан из магазина. В этом можно убедиться, сравнив хеш-суммы исходного и извлеченного файлов, получаемые по алгоритму шифрования MD5. Сравниваемые хеш-суммы по алгоритму MD5 обоих файлов должны быть одинаковы, как показано на рис. 10.37.

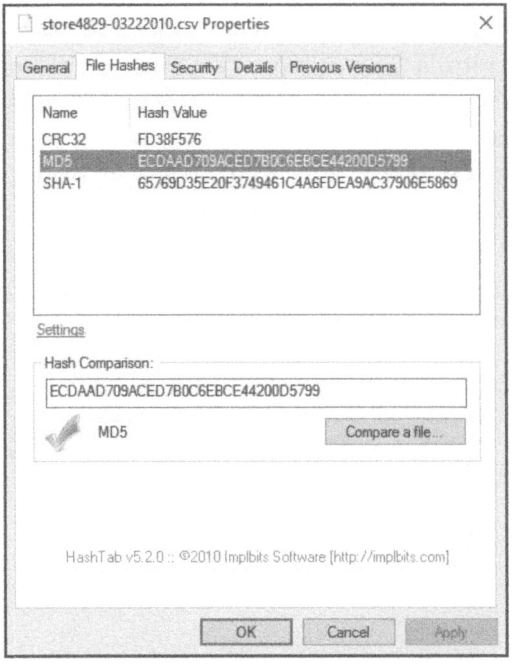


Рис. 10.37. Хеш-суммы по алгоритму MD5 исходного и извлеченного файлов одинаковы

Сравнив оба файла, можно сделать вывод, что сеть нельзя винить в повреждении информации, сохраняемой рассматриваемым здесь приложением в базе данных. Файл, переданный из магазина для сбора данных на сервере, остается невредимым, доходя до сервера. Поэтому любое повреждение, скорее всего, происходит в процессе обработки этого файла в приложении на стороне сервера.

Усвоенные уроки

Анализ на уровне пакетов примечателен, в частности, тем, что он избавляет от необходимости разбираться в нагромождении различных приложений. Неудачно разработанных приложений намного больше, чем разработанных удачно, но на уровне пакетов это не имеет никакого значения. В данном случае программист использовал в своем приложении некие загадочные сетевые компоненты. Но в конечном счете весь сложный алгоритм передачи данных,

реализованный в сотнях строк кода, может быть сведен к сетевым протоколам FTP, TCP и IP. Зная особенности функционирования этих основных протоколов, мы смогли убедиться, что процесс передачи данных по сети проходил верно, и даже сумели извлечь файл из потока данных, чтобы доказать исправность сети. В этой связи очень важно помнить, что какой бы сложной не оказалась возникшая в сети проблема, она все равно сводится к пакетам.

Заключительные соображения

В этой главе был рассмотрен ряд сценариев, в которых анализ пакетов позволяет лучше понять и разобраться в трудностях обмена данными по сети. Применяя элементарный анализ общеупотребительных сетевых протоколов, мы смогли проследить и своевременно разрешить различные затруднения, возникающие в сети. И хотя рассмотренные здесь сценарии могут отличаться от тех, с которыми вам придется иметь дело на практике, представленные здесь методики анализа пакетов должны принести вам пользу в разрешении затруднений, возникающих в вашей сети.

11

МЕРЫ БОРЬБЫ С МЕДЛЕННОЙ СЕТЬЮ

Большую часть времени сетевому администратору приходится уделять устранению неполадок на компьютерах и серверах, которые работают медленнее, чем обычно должны. Но если кто-нибудь пожалуется на медленную работу сети, то это еще не означает, что виной тому сама сеть.

Прежде чем начинать бороться с медленной сетью, необходимо выяснить, действительно ли она работает медленно. Из этой главы вы узнаете, как это делается.

И начнем мы с обсуждения функциональных средств, доступных в протоколе TCP для устранения ошибок и управления потоками данных, передаваемых по сети. Затем мы покажем, как обнаруживаются характерные признаки медленно работающей сети. И, наконец, мы рассмотрим способы сравнения с исходными характеристиками сетей и действующих в них служб и устройств. Проработав материал этой главы, вы сможете намного лучше выявлять, диагностировать и устранять неполадки в медленно работающих сетях.

ПРИМЕЧАНИЕ *Для диагностики медленно работающих сетей имеется немало методик. Здесь выбрана методика, опирающаяся в основном на протокол TCP, потому что именно этого чаще всего оказывается достаточно. В отличие от протокола ICMP, протокол TCP дает возможность проводить пассивный ретроспективный анализ вместо того, чтобы генерировать дополнительный сетевой трафик.*

Функциональные средства устранения ошибок в протоколе TCP

Функциональные средства устранения ошибок, доступные в протоколе TCP, лучше всего подходят для обнаружения, диагностики и в конечном счете устранения больших задержек при передаче данных в сети. С точки зрения организации компьютерных сетей *задержка* – это мера, определяющая разность между моментами времени передачи и приема пакетов.

Задержки бывают односторонние (измеряется время передачи пакетов от отправителя к получателю) и двухсторонние (измеряется время передачи пакетов от отправителя к получателю и обратно к отправителю). Если обмен данными между устройствами происходит быстро и время доставки пакета из одной конечной точки в другую невелико, то такой обмен данными происходит с *малой задержкой*. А если доставка пакетов от отправителя к получателю отнимает немало времени, то такой обмен данными происходит с *большой задержкой*. Большая задержка – это первый враг всех сетевых администраторов, которые очень ценят исправность сети. Ведь именно в ней заключается цель их работы.

В главе 8, "Протоколы транспортного уровня", пояснялось, каким образом номера подтверждения и порядковые номера применяются в протоколе TCP для обеспечения надежной доставки пакетов. А в этой главе эти номера будут рассмотрены снова, чтобы выяснить, каким образом протокол TCP реагирует на нарушение последовательности приема (или вообще неприема) пакетов с этими номерами из-за большой задержки в сети.

Повторная передача данных в протоколе TCP

Файл перехвата **tcp_retransmissions.pcapng**

Способность хоста повторно передавать пакеты относится к самым основным функциональным средствам устранения ошибок в протоколе TCP. Оно предназначено для борьбы с потерями пакетов.

Для потери пакетов существуют несколько возможных причин, включая неверно функционирующие приложения, сильно перегруженные маршрутизаторы или временно выходящие из строя службы. В связи с тем что операции на уровне пакетов выполняются очень быстро, потеря пакетов зачастую носит временный характер. Поэтому крайне важно, чтобы в протоколе TCP можно было своевременно обнаруживать и устранять потерю пакетов.

Главным механизмом, позволяющим определить необходимость повторной передачи пакета, является *таймер повторной передачи*, который отвечает за отсчет *времени ожидания до повторной передачи* (RTO, или Retransmission Timeout). Всякий раз, когда пакет передается по протоколу TCP, запускается

таймер повторной передачи. Этот таймер останавливается, когда получен пакет ACK. Промежуток времени между передачей и приемом пакета ACK называется *временем на передачу и подтверждение приема* (RTT, или Round-Trip Time). Некоторые из этих временных характеристик усредняются, чтобы определить окончательную величину времени ожидания до повторной передачи.

До тех пор, пока время ожидания до повторной передачи не будет определено, операционная система передающего устройства полагается на устанавливаемое в ней по умолчанию время на передачу и подтверждение приема, которое выдается при первоначальном обмене данными между хостами. Затем его величина корректируется, исходя из времени на передачу и подтверждение приема полученных пакетов, чтобы определить время ожидания до повторной передачи.

Как только величина времени ожидания до повторной передачи будет определена, по каждому передаваемому пакету будет срабатывать таймер повторной передачи, если произошла потеря этого пакета. Процесс повторной передачи пакетов по протоколу TCP наглядно показан на рис. 11.1.

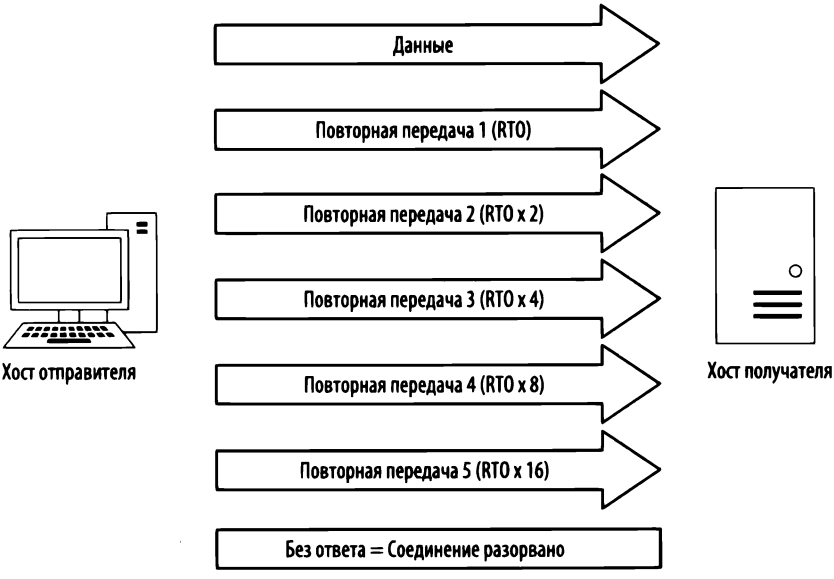


Рис. 11.1. Концептуальное представление процесса повторной передачи пакетов по протоколу TCP

Когда пакет посылается, а получатель не отправляет в ответ подтверждающий пакет ACK по протоколу TCP, отправитель предполагает, что исходный пакет был потерян, и передаст его еще раз. При повторной передаче пакета величина времени ожидания до повторной передачи (RTO) удваивается. И если подтверждающий пакет ACK не будет получен до истечения этого времени, то

произойдет еще одна повторная передача пакета. А если и после этого подтверждающий пакет ACK не будет получен в ответ, то величина времени ожидания до повторной передачи удвоится снова. Этот процесс будет продолжаться до тех пор, пока отправитель не достигнет максимального количества попыток повторной передачи, на которое он был настроен. Более подробно этот процесс описан в документе RFC 6298.

Максимальное количество попыток повторной передачи зависит от величины, установленной в операционной системе передающего устройства. По умолчанию хосты под управлением системы Windows совершают максимум 5 попыток повторной передачи, а большинство хостов под управлением Linux – до 15 таких попыток. Этот параметр настраивается в любой операционной системе.

В качестве примера повторной передачи по протоколу TCP откройте файл перехвата tcp_retransmissions.pcapng, состоящего из шести пакетов. Первый из этих пакетов приведен на рис. 11.2.

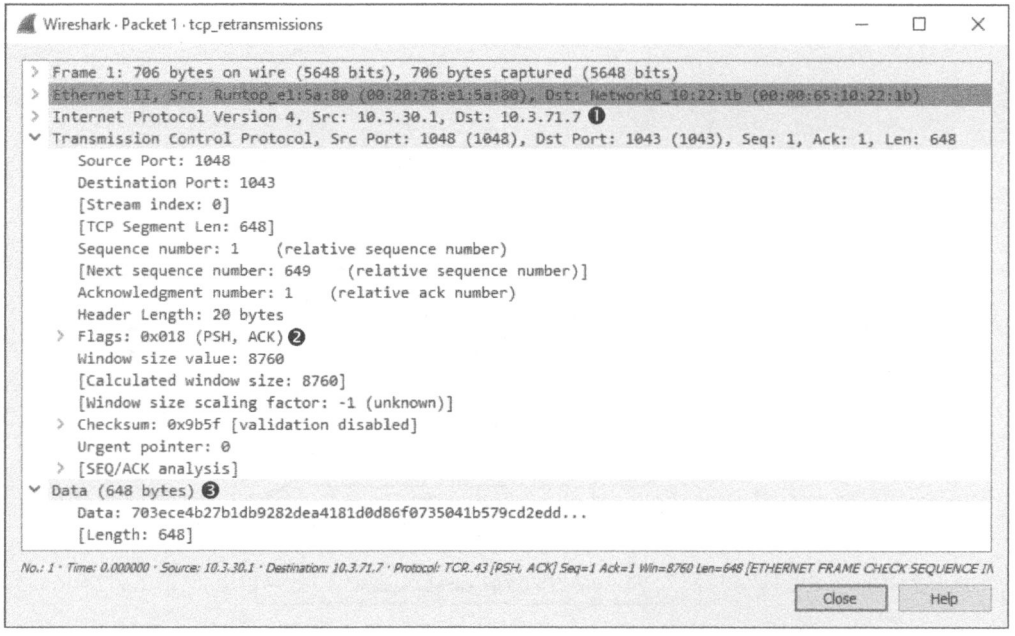
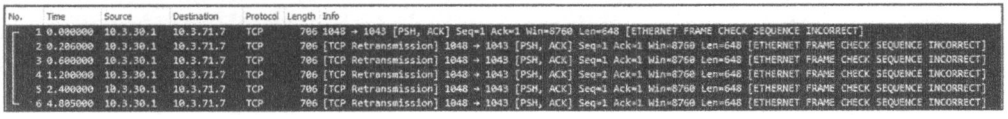


Рис. 11.2. Простой пакет TCP, содержащий данные

Это, по существу, пакет PSH/ACK ❷, содержащий 648 байтов данных ❸ и передаваемый по протоколу TCP из хоста, находящегося по адресу **10.3.30.1**, хосту, расположенному по адресу **10.3.71.7** ❶. И это типичный пакет для передачи данных.

При обычных обстоятельствах следует ожидать, что подтверждающий пакет ACK должен посылаться в ответ по протоколу TCP через довольно короткий промежуток времени после отправки первого пакета. Но в данном случае повторно передается следующий пакет. Об этом можно судить, глядя на пакет в панели Packet List, где в столбце **Info** ясно указана повторная передача [TCP Retransmission], а текст из этого пакета выделен красным цветом на черном фоне. Некоторые примеры повторных передач, наблюдаемые в панели Packet List, приведены на рис. 11.3.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.3.30.1	10.3.71.7	TCP	706	1048 → 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
2	0.206000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 → 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
3	0.600000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 → 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
4	1.200000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 → 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5	2.400000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 → 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
6	4.805000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 → 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]

Рис. 11.3. Примеры повторных передач, наблюдаемые в панели Packet List

Чтобы выяснить, передается ли пакет повторно, можно также проанализировать его в панели Packet Details, как показано на рис. 11.4.

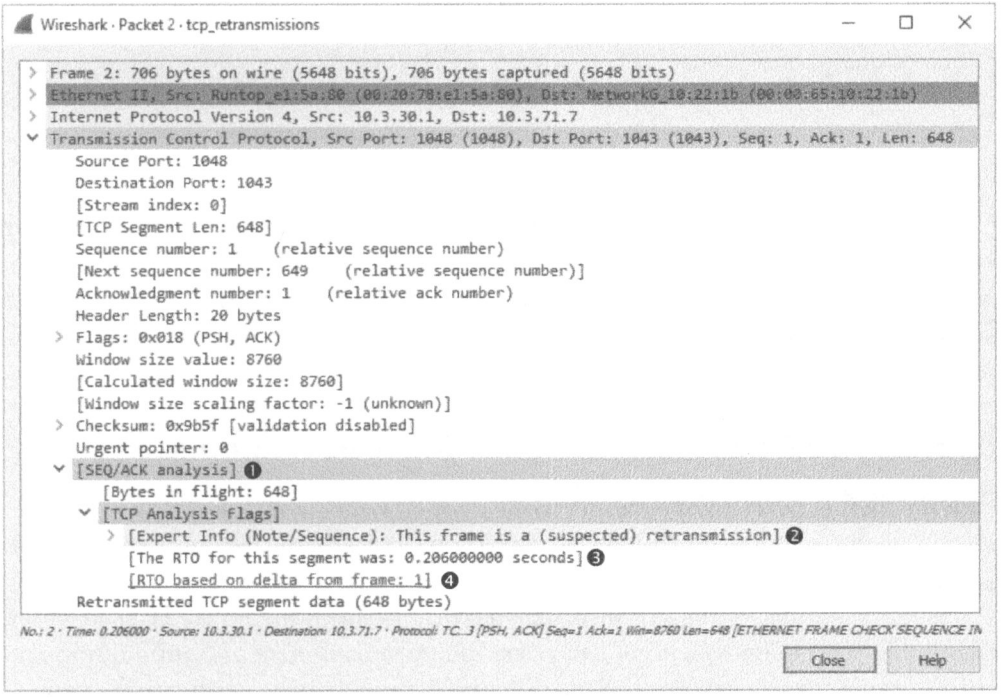


Рис. 11.4. Отдельный повторно передаваемый пакет

Проводя анализ пакета в панели Packet Details, обратите внимание на то, что в повторно передаваемый пакет включены дополнительные сведения под заголовком **SEQ/ACK analysis** ❶. Эти полезные сведения предоставляются

приложением Wireshark и не входят в сам пакет. Они, в частности, сообщают, что пакет действительно передается повторно ❷, а время ожидания до повторной передачи (RTO) составляет 0,206 с ❸ и отсчитывается как прирост времени от момента передачи пакета 1 ❹.

Обратите также внимание, что данный пакет такой же, как и первоначальный пакет (вплоть до контрольной суммы). Можете убедиться в этом сами, сравнив повторно переданный пакет с первоначальным в панели Packet Bytes.

К аналогичным результатам приводит анализ остальных пакетов, которые могут отличаться контрольными суммами и временем ожидания до повторной передачи. Чтобы увидеть промежутки времени, протекающего между пакетами, загляните в столбец **Time** в панели Packet List (рис. 11.5). Здесь вы обнаружите экспоненциальный рост времени ожидания до повторной передачи по мере удвоения его величины после каждой повторной передачи.

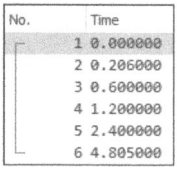


Рис. 11.5. В столбце **Time** *из панели Packet List показан экспоненциальный рост времени ожидания до повторной передачи*

Возможности протокола TCP для повторной передачи пакетов используются передающим устройством для выявления и устранения потерь пакетов. А далее мы рассмотрим *дублирующие подтверждения* по протоколу TCP. Такой возможностью на этот раз пользуется получатель данных для выявления и устранения потерь пакетов.

Дублирующие подтверждения и быстрые повторные передачи по протоколу TCP

Файл перехвата **tcp_dupack.pcapng**

Дублирующее подтверждение передается получателем в пакете ACK по протоколу TCP в том случае, если получатель принимает пакеты не по порядку. В TCP-заголовке специально выделяются поля для указания номера подтверждения и порядкового номера пакета, чтобы данные можно было надежно принять и снова собрать в том же самом порядке, в каком они были переданы.

ПРИМЕЧАНИЕ *Пакет TCP было бы правильнее называть сегментом TCP, но большинство специалистов в данной области привыкли называть его пакетом.*

После установки нового соединения по протоколу TCP, происходит передача первоначального порядкового номера (ISN, или Initial Sequence Number) – одного из самых важных фрагментов информации, обмениваемой в процессе подтверждения связи. После установки первоначального порядкового номера на обеих сторонах соединения этот номер постепенно увеличивается в каждом передаваемом далее пакете на размер полезной информации, переносимой в данном пакете.

Рассмотрим хост отправителя, который имеет первоначальный порядковый номер **5000** и посылает пакет размером 500 байт хосту получателю. Как только этот пакет будет получен, хост получателя ответит пакетом ACK по протоколу TCP с номером подтверждения **5500** на основании следующей формулы:

```
Исходный порядковый номер + Количество байтов полученных данных =
        = Выходной номер подтверждения
```

В результате расчета по этой формуле передающему хосту возвращается номер подтверждения, равный следующему порядковому номеру, который предполагает получить хост получателя. Характерный тому пример приведен на рис. 11.6.

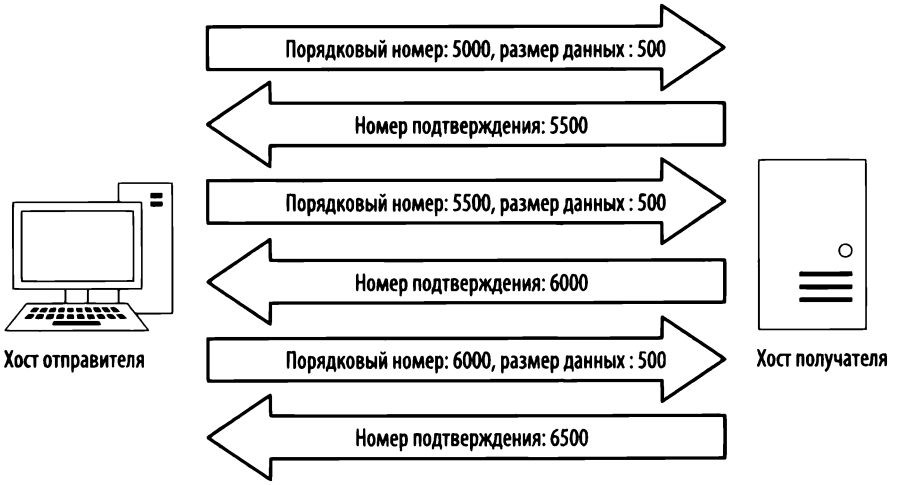


Рис. 11.6. Последовательность обмена номерами подтверждения по протоколу TCP

Выявление факта потери пакетов получателем данных становится возможным благодаря порядковым номерам. Отслеживая принимаемые порядковые номера пакетов, получатель может выяснить, когда он получил их не по порядку.

Когда получатель принимает пакет с непредвиденным порядковым номером, он предполагает, что пакет был потерян во время передачи. Чтобы правильно собрать данные снова, получатель должен иметь в своем распоряжении

отсутствующий пакет, и поэтому он снова посылает подтверждающий пакет ACK, содержащий предполагаемый порядковый номер потерянного пакета, чтобы добиться повторной передачи этого пакета у отправителя.

Получив три дублирующих подтверждения ACK от принимающего хоста (получателя), передающий хост (отправитель) посчитает, что пакет был действительно потерян при передаче, и сразу же начнет его *быструю повторную передачу*. Как только быстрая повторная передача потерянного пакета будет начата, все остальные передаваемые пакеты будут поставлены в очередь до тех пор, пока быстрая повторная передача не завершится. Этот процесс наглядно показан на рис. 11.7.

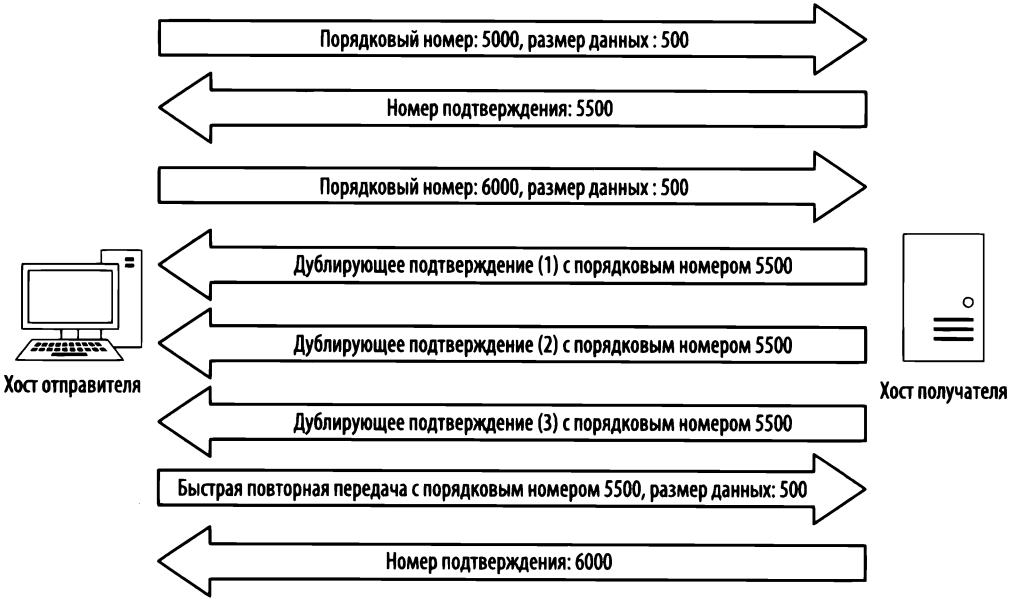


Рис. 11.7. Дублирующее подтверждение от получателя приводит к быстрой повторной передаче потерянного пакета

Пример дублирующих подтверждений и быстрых повторных передач можно найти в файле перехвата `tcp_dupack.pcapng`. Первый пакет из этого файла перехвата приведен на рис. 11.8.

Это, по существу, пакет ACK, передаваемый по протоколу TCP хостом, находящимся по адресу **`172.31.136.85`**, хосту, расположенному по адресу **`195.81.202.68`** ❶. Он содержит подтверждение о получении данных, отправленных в предыдущем пакете, не включенном в рассматриваемый здесь файл перехвата.

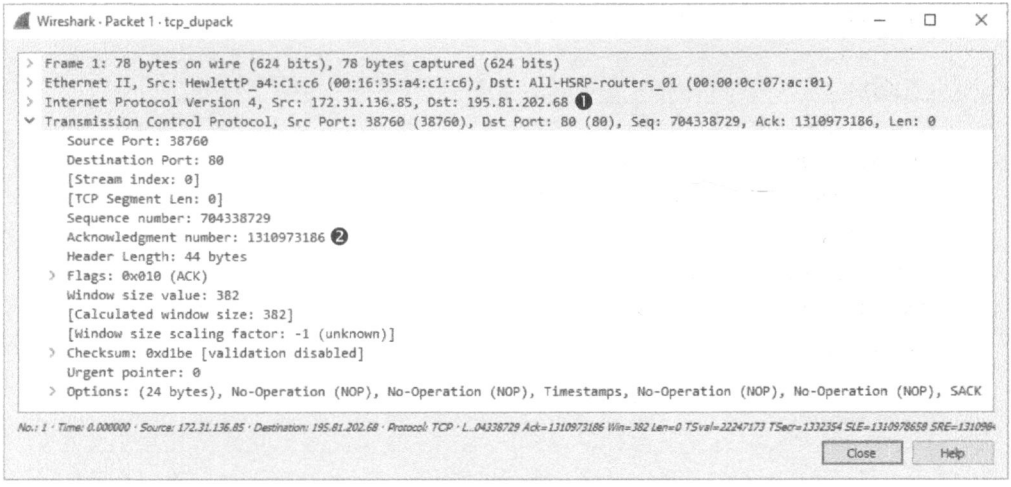


Рис. 11.8. Пакет ACK, где указан следующий ожидаемый порядковый номер

ПРИМЕЧАНИЕ *По умолчанию в Wireshark применяются относительные порядковые номера пакетов для упрощения их анализа, но в примерах и копиях экранов, приведенных в ряде последующих разделов, этот режим отображения не используется. Чтобы отменить этот режим, выберите сначала команду* Edit⇨Preferences *из главного меню, затем раздел* Protocols *в открывшемся окне, далее раздел* TCP *и, наконец, сбросьте флажок* Relative sequence numbers *(Относительные порядковые номера).*

В данном пакете указан номер подтверждения **1310973186** ❷. Он должен совпадать с порядковым номером в следующем полученном пакете, как показано на рис. 11.9.

К сожалению для нас и получателя пакетов, порядковый номер следующего пакета равен **1310984130** ❶ и не соответствует ожидаемому. Этот полученный не по порядку пакет указывает на то, что ожидаемый пакет **1310973186** был каким-то образом потерян при передаче. Хост получателя обратит внимание на то, что данный пакет получен не по порядку, и отправит дублирующее подтверждение в третьем пакете из анализируемого файла перехвата, как показано на рис. 11.10.

Чтобы выяснить, содержит ли данный пакет дублирующее подтверждение, проанализируйте одно из двух.

- Столбец **Info** в панели Packet Details. Текст из этого пакета должен быть выделен красным цветом на черном фоне.
- Заголовок **SEQ/ACK** в панели Packet Details (см. рис. 11.10). Если развернуть этот заголовок, то в нем можно обнаружить, что данный пакет указан как пакет с первым дублирующим подтверждением ❶.

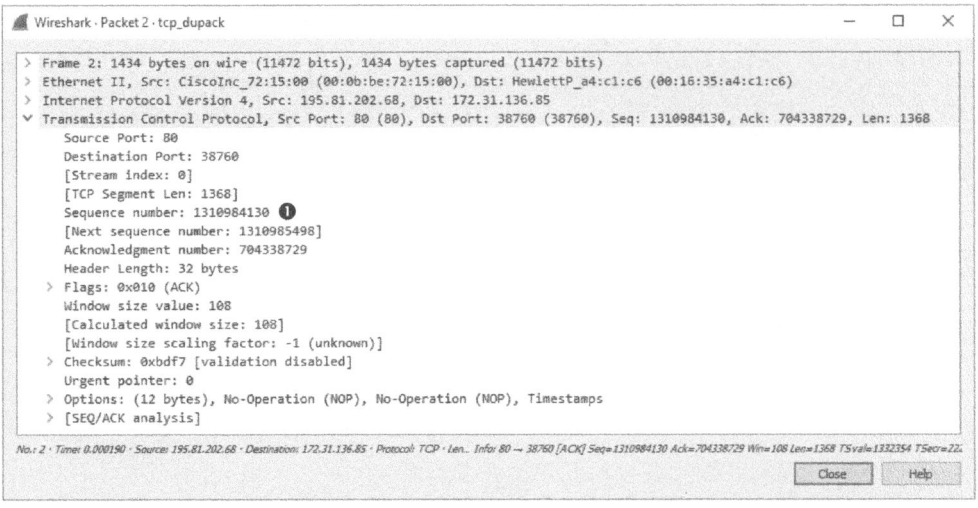


Рис. 11.9. Порядковый номер данного пакета не соответствует ожидаемому

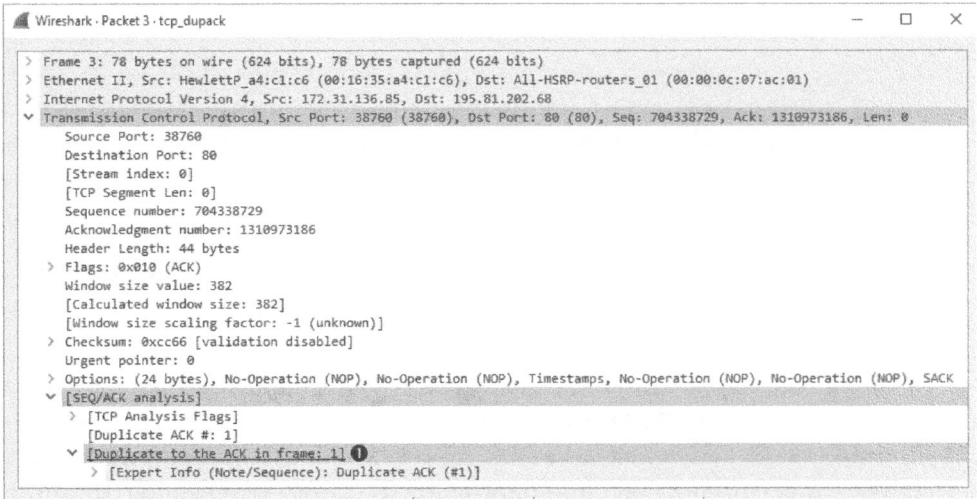


Рис. 11.10. Пакет с первым дублирующим подтверждением

Описываемый здесь процесс продолжается в нескольких последующих пакетах, как показано на рис. 11.11.

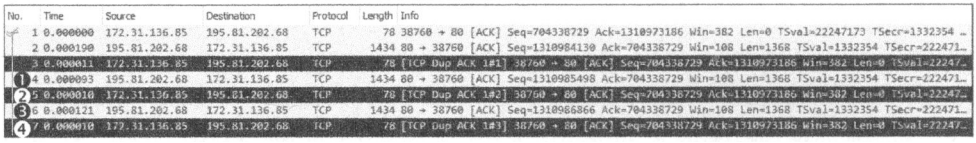


Рис. 11.11. Дополнительные дублирующие подтверждения формируются из-за того, что отправляемые пакеты принимаются не по порядку

Четвертый пакет из рассматриваемого здесь файла перехвата содержит очередной фрагмент данных, посылаемых передающим хостом с неверным порядковым номером ❶. В итоге принимающий хост посылает второе дублирующее подтверждение ❷. Далее получатель принимает еще один пакет с неверным порядковым номером ❸. И это приводит к отправке третьего и заключительного дублирующего подтверждения ❹.

Как только передающий хост получит пакет с третьим дублирующим подтверждением от принимающего хоста, он будет вынужден остановить весь процесс передачи пакетов и снова послать потерянный пакет. Пример быстрой повторной передачи потерянного пакета приведен на рис. 11.12.

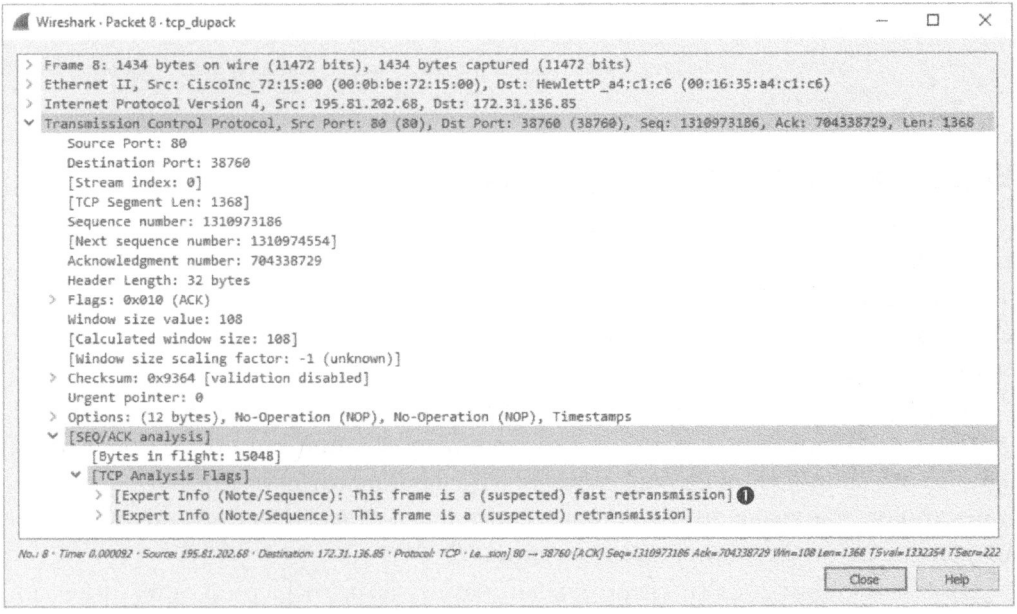


Рис. 11.12. Три дублирующих подтверждения приводят к показанной здесь быстрой повторной передаче потерянного пакета

Повторно передаваемый пакет может быть снова обнаружен в столбце **`Info`** на панели Packet List. Как и в предыдущих примерах, текст из этого пакета явно выделен красным цветом на черном фоне. В разделе **`SEQ/ACK analysis`** этого пакета (см. рис. 11.12) сообщается, что он подозревается в быстрой повторной передаче ❶. (И в этом случае сведения, которые указывают на то, что данный пакет относится к категории быстрой повторной передачи, не входят в сам пакет, а предоставляются приложением Wireshark для упрощения анализа.) И последний пакет в рассматриваемом здесь перехваченном трафике подтверждает получение быстрой повторной передачи потерянного пакета.

ПРИМЕЧАНИЕ *На обмен потоком данных по протоколу TCP, где может иметь место потеря пакетов, может оказывать влияние режим SACK (Selective Acknowledgment – выборочное подтверждение). В рассмотренном выше примере перехваченного трафика была согласована активизация этого режима в процессе трехэтапной установки связи по протоколу TCP. Таким образом, всякий раз, когда пакет теряется и в ответ присылается дублирующее подтверждение, повторно должны передаваться только потерянные пакеты, даже если после потерянного пакета были успешно получены другие пакеты. Если бы режим SACK не был активизирован, пришлось бы повторно передавать и каждый пакет, появляющийся после потерянного пакета. Выборочное подтверждение намного повышает эффективность восстановления потерянных пакетов. Оно нередко применяется, поскольку поддерживается в большинстве современных реализаций стека протоколов TCP/IP.*

Управление потоками данных в протоколе TCP

Повторные передачи и дублирующие подтверждения относятся к реагирующим функциям протокола TCP, предназначенным для восстановления потерянных пакетов. Протокол TCP вряд ли бы смог справиться со своими задачами, если бы в него не была включена некоторая форма упреждения, препятствующая потере пакетов.

В протоколе TCP реализован *механизм скользящего окна (sliding-window mechanism)* для обнаружения момента, когда может возникнуть потеря пакетов и изменения скорости передачи данных с целью предотвратить такую потерю. В механизме скользящего окна используется *окно приема* на стороне получателя пакетов для управления потоком данных.

Принимающий хост (получатель данных) указывает размер окна приема в байтах, сохраняет его в TCP-заголовке и сообщает передающему хосту (отравителю данных), сколько данных он готов сохранить в *памяти* своего *TCP-буфера*. Именно в этой памяти данные временно хранятся до тех пор, пока они не будут переданы вверх по стеку протоколов на уровень приложений для последующей обработки. В итоге передающий хост может одновременно послать только то количество данных, которое указано в поле **Window size** (Размер окна) заголовка полученного в ответ пакета TCP. Чтобы передающий хост мог послать больше данных, принимающий хост должен отправить ему подтверждение о получении предыдущей порции данных. Он должен также очистить память своего TCP-буфера, обработав данные, занимающие эту память. Принцип действия окна приема наглядно показан на рис. 11.13.

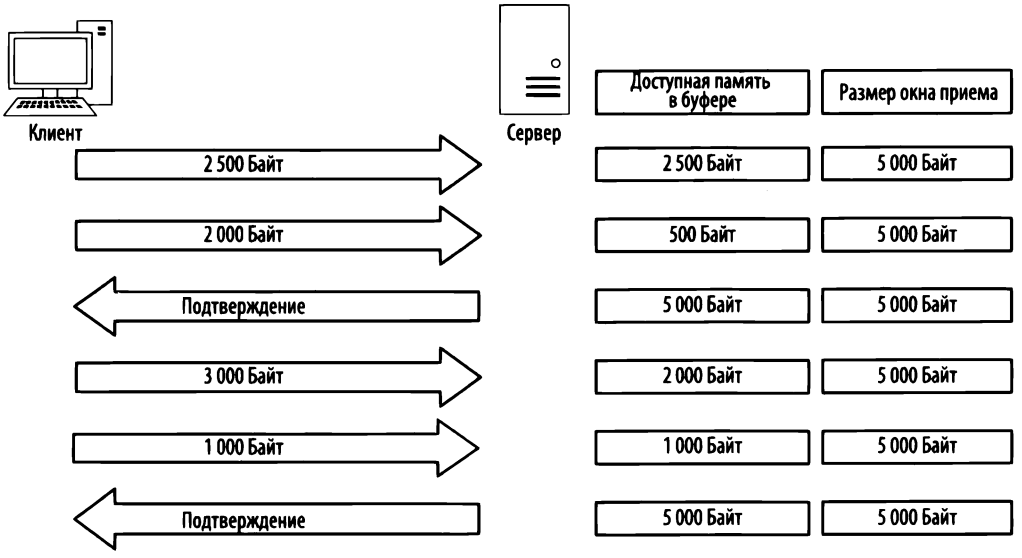


Рис. 11.13. Окно приема ограничивает объем данных, которые готов принять получатель

Как показано на рис. 11.13, клиент посылает данные серверу, который предварительно сообщил ему размер окна приема равный 5000 байт. Но клиент посылает сначала 2500 байт, сокращая тем самым размер памяти в буфера на сервере до 2500 байт, а затем еще 2000 байт, сокращая размер пространства буфера до 500 байт. На это сервер посылает подтверждение о приеме этих данных, и после их обработки буфер очищается и снова готов к приему очередной порции данных. Весь процесс повторяется, но на этот раз клиент посылает сначала 3000 байт, а затем еще 1000 байт, сокращая размер памяти в буфере на сервере до 1000 байт. На это сервер снова посылает подтверждение о приеме данных и обрабатывает содержимое своего буфера.

Изменение размера окна приема

Описываемый здесь процесс, в ходе которого корректируется размер окна приема, довольно прост, хотя и не всегда идеален. Всякий раз, когда данные принимаются стеком TCP, формируется подтверждение и посылается ответ. Но данные, размещаемые в буфере получателя, не всегда обрабатываются немедленно.

Когда сервер занят обработкой пакетов, полученных от многих клиентов, он вполне может повременить с очисткой своего буфера, а следовательно, и с освобождением места для новых данных. Без надлежащих средств управления потоком данных переполнение буфера может привести к потере пакетов

и повреждению данных. Правда, когда сервер слишком занят, чтобы обрабатывать данные в объеме, указанном в окне приема, он может откорректировать его размер. С этой целью сервер может уменьшить величину размера окна приема в TCP-заголовке пакета ACK, посылаемого обратно тем клиентам, которые передают ему свои данные. Характерный тому пример приведен на рис. 11.14.

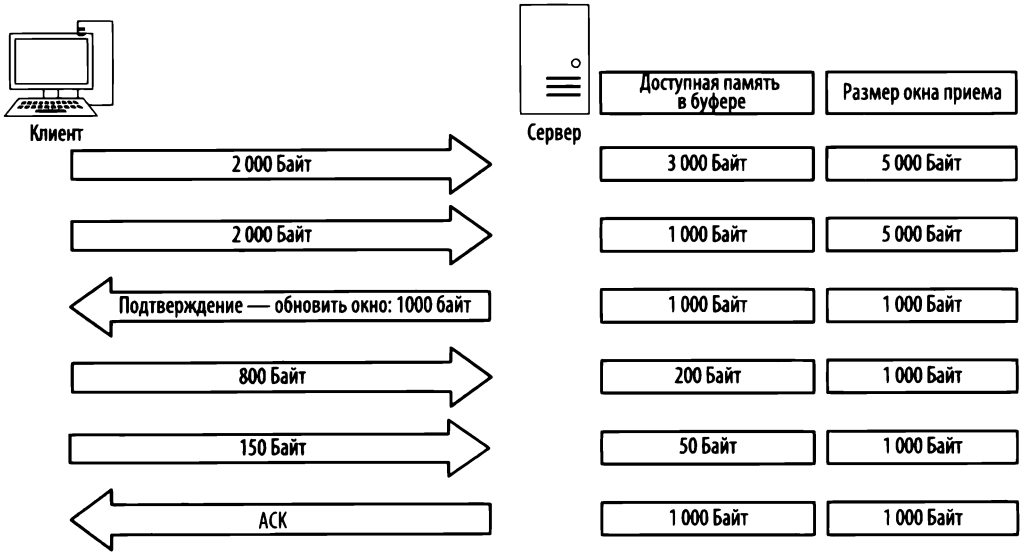


Рис. 11.14. Размер окна приема может быть изменен, когда сервер оказывается занят

Как показано на рис. 11.14, сервер начинает обмен данными с анонсированного размера окна приема в 5000 байт. На это клиент посылает сначала 2000 байт, затем еще 2000 байт, оставляя доступной память в буфере размером всего 1000 байт. Но это пространство заполнится очень быстро, и если данные будут и дальше поступать с той же скоростью, то пакеты окажутся вскоре потерянными. Во избежание подобного осложнения сервер посылает клиенту подтверждение с уменьшенным до 1000 байт размером окна приема. На это клиент отправит меньше данных, и теперь скорость, с которой сервер способен обрабатывать полученные данные, позволит установить постоянный поток данных.

Процесс изменения размеров окна приема действует как в сторону уменьшения, так и увеличения. Так, если сервер способен обработать данные с большей скоростью, он может послать подтверждающий пакет ACK с бо́льшим размером окна приема

Прекращение потока данных с помощью установки нулевого окна приема

Вследствие нехватки памяти, вычислительных ресурсов для обработки данных или иных затруднений сервер может оказаться временно не в состоянии обрабатывать данные, поступающие от клиента. Такая пауза может привести к пропуску пакетов и нарушению процесса обмена данными. Впрочем, окно приема позволяет свести к минимуму отрицательные последствия такой ситуации.

Когда возникает подобная ситуация, сервер может послать пакет, содержащий нулевой размер окна приема. Получив такой пакет, клиент прервет дальнейшую передачу данных, но сохранит установленное соединение с сервером, посылая время от времени *пакеты поддержания активным соединения*. Такие пакеты клиент может посылать через регулярные промежутки времени, чтобы проверить состояние окна приема на сервере. И как только сервер снова начнет обработку данных, он отправит клиенту в ответ пакет с ненулевым размером окна приема, возобновив тем самым обмен данными с клиентом. Пример уведомления о нулевом размере окна приема приведен на рис. 11.15.

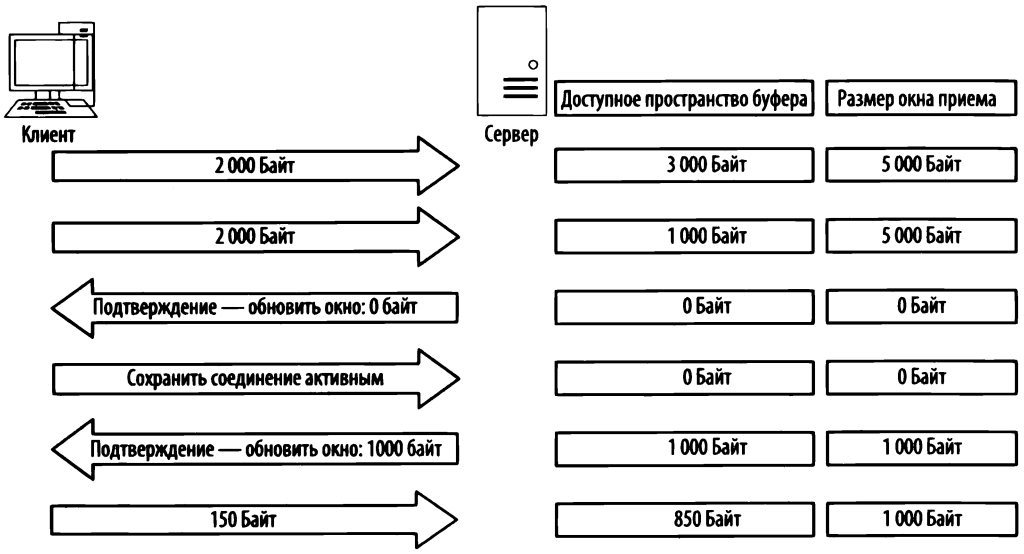


Рис. 11.15. Передача данных прекращается, когда размер окна приема становится равным 0 байт

Как показано на рис. 11.15, сервер начинает принимать данные в окне размером 5000 байт. Приняв в общем 4000 байт данных от клиента, сервер начинает испытывать чрезмерную нагрузку на свой процессор и поэтому больше не в состоянии обрабатывать дальше любые данные, поступающие от клиента.

И тогда сервер посылает пакет с нулевым значением в поле **Window size** его заголовка. Получив такой пакет, клиент прервет передачу данных и будет периодически отправлять пакет поддержания активным соединения. Получив, в свою очередь, такой пакет, сервер ответит пакетом, уведомляющим клиента, что он может теперь принимать данные в окне размером 1000 байт. В таком случае клиент возобновит передачу данных, но с меньшей скоростью, чем прежде.

Применение механизма скользящего окна на практике

Файлы перехвата **tcp_zerowindowrecovery.pcapng** *и* **tcp_zerowindowdead.pcapng**

Итак, изложив теорию, положенную в основу механизма скользящего окна в протоколе TCP, рассмотрим его действие на практике, обратившись к примеру, взятому из файла перехвата tcp_zerowindowrecovery.pcapng. Сетевой трафик, зафиксированный в этом файле, начинается с отправки нескольких подтверждающих пакетов ACK по протоколу TCP из хоста, находящегося по адресу **192.168.0.20**, хосту, расположенному по адресу **192.168.0.30**. Наибольший интерес для нас представляет значение, устанавливаемое в поле **Window size** заголовка этих пакетов. Его можно посмотреть как в столбце **Info** на панели Packet List, так и в заголовке **TCP** на панели Packet Details. Но в любом случае можно сразу заметить, что значение в данном поле постепенно уменьшается в первых трех пакетах, как показано на рис. 11.16.

No.	Time ❶	Source	Destination	Protocol	Length	Info ❷
1	0.000000	192.168.0.20	192.168.0.30	TCP	60	2235 → 1720 [ACK] Seq=1422793785 Ack=2710996659 Win=8760 Len=0
2	0.000237	192.168.0.20	192.168.0.30	TCP	60	2235 → 1720 [ACK] Seq=1422793785 Ack=2710999579 Win=5840 Len=0
3	0.000193	192.168.0.20	192.168.0.30	TCP	60	2235 → 1720 [ACK] Seq=1422793785 Ack=2711002499 Win=2920 Len=0

Рис. 11.16. Размер окна приема в этих пакетах постепенно уменьшается

Размер окна приема постепенно уменьшается от 8760 байт в первом пакете до 5840 байт во втором пакете и далее до 2920 байт в третьем пакете ❶. Такое уменьшение размера окна приема служит явным признаком увеличения задержки в сети. Обратите внимание на столбец **Time**, где ясно видно, насколько быстро это происходит ❷. Когда размер окна приема уменьшается настолько быстро, то его величина обычно падает до нуля, что и происходит в четвертом пакете, как показано на рис. 11.17.

Кроме того, четвертый пакет посылается из хоста, находящегося по адресу **192.168.0.20**, хосту, расположенному по адресу **192.168.0.30**, хотя он и предназначен лишь для уведомления о том, что хост получателя больше не в состоянии принимать данные. Нулевое значение размера окна приема видно из TCP-заголовка ❶. О том, что это пакет с нулевым размером окна приема, сообщается в столбце **Info** на панели Packet List, а также в разделе **SEQ/ACK analysis** его TCP-заголовка ❷.

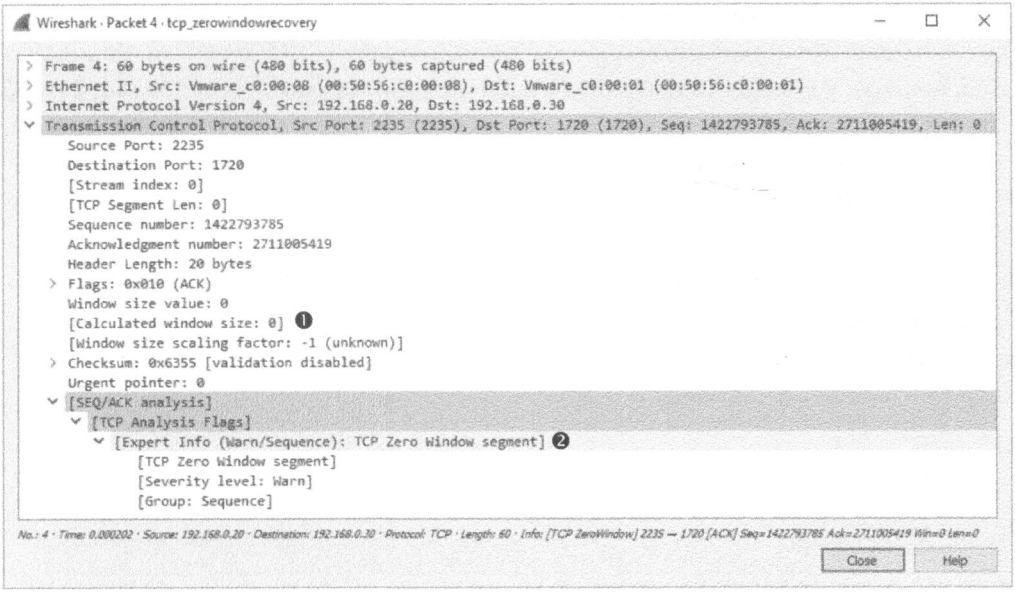


Рис. 11.17. Пакет с нулевым размером окна приема явно свидетельствует, что хост больше не в состоянии принимать данные

Как только пакет с нулевым размером окна приема будет отправлен, устройство, находящееся по адресу **192.168.0.30**, перестанет посылать данные до тех пор, пока оно не получит пакет с обновленным размером окна приема от устройства, расположенного по адресу **192.168.0.20** и уведомляющего, что размер окна приема увеличился. К счастью, причина, по которой возникло условие нулевого размера окна приема, носила лишь временный характер. Поэтому в следующем пакете был послан обновленный размер окна приема, как показано на рис. 11.18.

В данном случае размер окна приема заметно увеличивается до 64240 байт ❶. И в этом случае анализ в Wireshark показывает, что этот пакет содержит анонс обновления размера окна приема, как следует из раздела **SEQ/ACK analysis** его заголовка.

Как только будет получен пакет с обновленным размером окна приема, хост, находящийся по адресу **192.168.0.30**, может снова начать передачу данных, что и происходит в пакетах 6 и 7. Весь промежуток времени, в течение которого передача данных была приостановлена, проходит очень быстро. Но если бы эта задержка длилась дольше, она привела бы к перебоям в работе сети, а следовательно, к замедлению передачи данных или неудачному исходу данной операции.

В качестве еще одного, последнего примера употребления механизма скользящего окна рассмотрим файл перехвата `tcp_zerowindowdead.pcapng`.

Первым в сетевом трафике, перехваченном в данном файле, оказывается пакет, передаваемый обычным образом по протоколу HTTP из хоста, находящегося по адресу **172.31.136.85**, хосту, расположенному по адресу **195.81.202.68** (рис. 11.19).

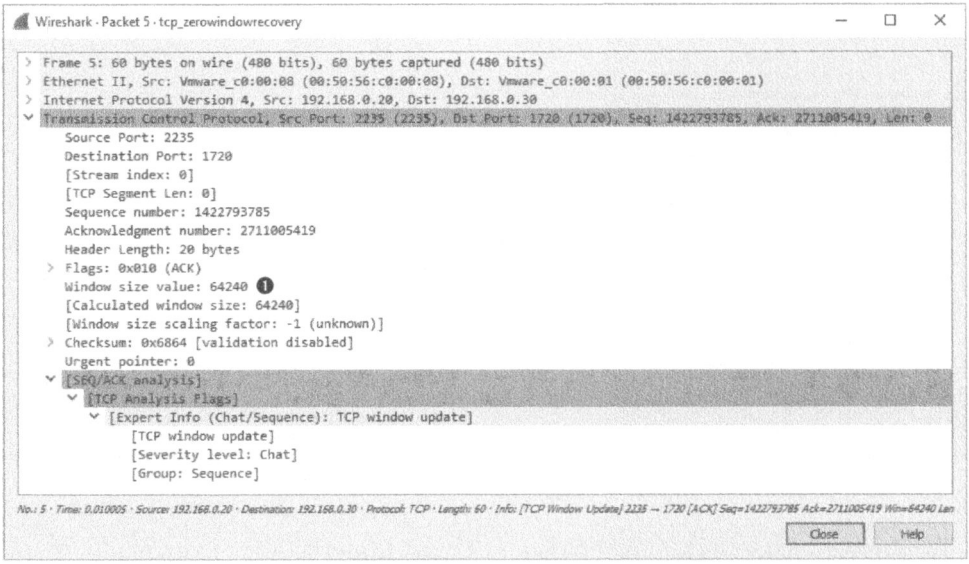


Рис. 11.18. Пакет TCP с обновленным размером окна приема уведомляет другой хост, что он может снова начать передачу данных

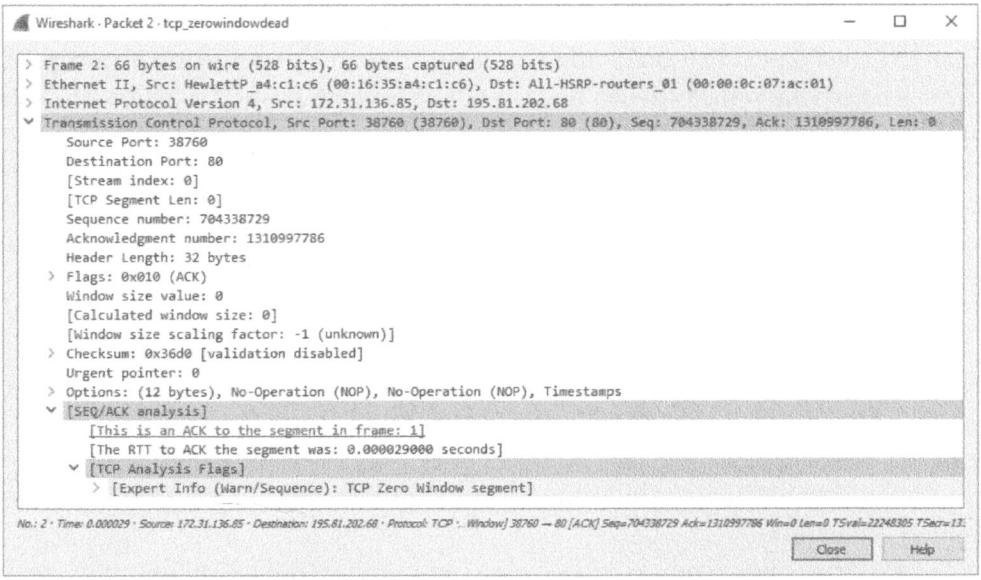


Рис. 11.19. Пакет с нулевым размером окна приема прерывает передачу данных

Этот пакет очень похож на пакет с нулевым размером окна приема, приведенный на рис. 11.17, хотя результат его отправки совсем другой. Вместо пакета с обновленным размером окна приема, а следовательно, разрешением от хоста, расположенного по адресу **172.31.136.85**, на возобновление передачи данных мы наблюдаем в данном случае пакет поддержания активным соединения (рис. 11.20).

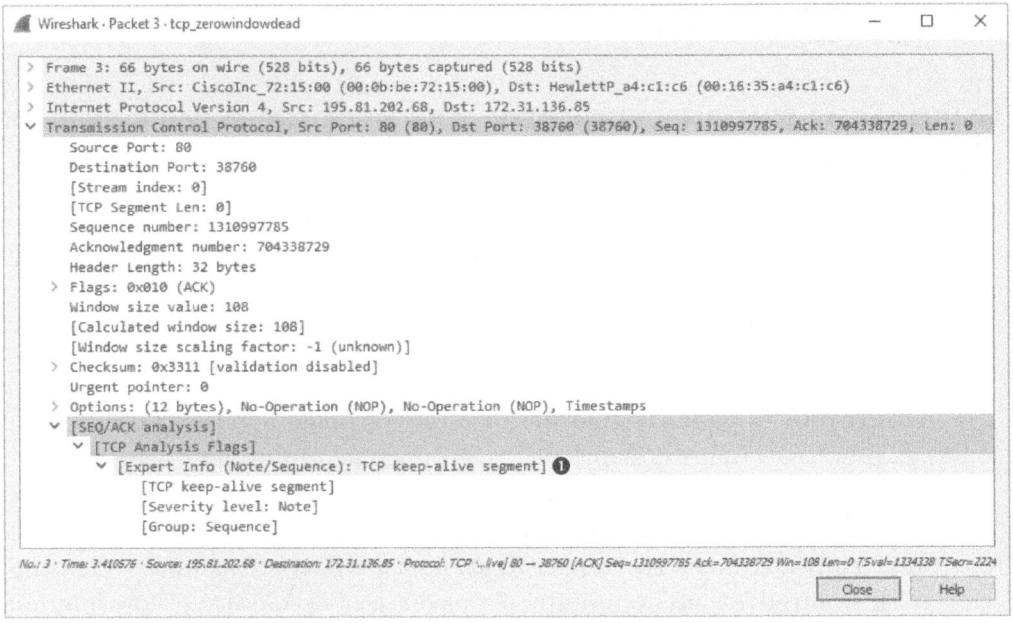


Рис. 11.20. Этот пакет поддержания активным соединения гарантирует, что связь с хостом, пославшим нулевой размер окна, по-прежнему существует

В результате анализа, проведенного в Wireshark, данный пакет помечен в разделе **SEQ/ACK analysis** его TCP-заголовка как поддерживающий активным соединение ❶. И как показано в столбце **Time**, данный пакет отправлен через 3,4 секунды после получения последнего пакета. Этот процесс продолжается еще несколько раз, когда один хост посылает пакет с нулевым размером окна приема, а другой хост – пакет поддержания активным соединения (рис. 11.21).

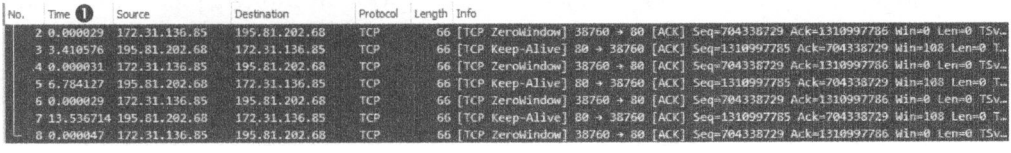


Рис. 11.21. Сервер и клиент продолжают посылать пакеты с нулевым размером окна приема и поддержания активным соединения соответственно

Передача пакетов поддержания активным соединения происходит через промежутки времени 3,4, 6,8 и 13,5 секунды ❶. Этот процесс может продолжаться довольно долго в зависимости от конкретной системы, под управлением которой работают устройства, обменивающиеся данными. Как следует из столбца `Time`, обмен данными прерывается приблизительно через 25 с. Если представить себе, что это происходит в процессе аутентификации с помощью контроллера домена или загрузки файла из Интернета, то подобная задержка в 25 с просто неприемлема!

Выводы из анализа пакетов для исправления ошибок и управления потоками данных по протоколу TCP

Попробуем ввести повторную передачу данных, дублирующие подтверждения и механизм скользящего окна в определенный контекст. Ниже приведены выводы из анализа упомянутых выше пакетов. Эти выводы следует иметь в виду, когда приходится разрешать затруднения, возникающие в связи с задержками в работе сети.

- **Повторно передаваемые пакеты.** Повторные передачи происходят потому, что клиент обнаружил, что сервер не принимает данные, которые он посылает. Следовательно, повторные передачи можно и не заметить, хотя это зависит от того, на какой именно стороне соединения анализируются пакеты. Так, если вы перехватываете данные от сервера и он действительно не принимает пакеты, посылаемые и повторно передаваемые клиентом, вы можете оказаться в неведении относительно повторно передаваемых пакетов. И если вы подозреваете, что стали жертвой потери пакетов на стороне сервера, попробуйте перехватить сетевой трафик от клиента, если это вообще возможно, чтобы убедиться в наличии повторно передаваемых пакетов.
- **Пакеты с дублирующими подтверждениями.** Я лично стремлюсь рассматривать дублирующее подтверждение как псевдопротивоположную повторную передачу, поскольку оно посылается в том случае, если сервер обнаруживает, что пакет от клиента, с которым он связан, был потерян при передаче. Зачастую дублирующие подтверждения можно наблюдать, перехватывая сетевой трафик на обеих сторонах установленного соединения. Напомним, что дублирующие подтверждения инициируются в том случае, если пакеты принимаются не по порядку. Так, если сервер получит лишь первый и третий из трех отправленных ему пакетов, он пошлет дублирующее подтверждение, чтобы затребовать от клиента повторную передачу второго пакета. А поскольку получены

лишь первый и третий пакеты, то второй пакет пропущен временно, независимо от конкретной причины, а следовательно, дублирующее подтверждение, вероятнее всего, будет отправлено и получено успешно. Безусловно, такой сценарий имеет место далеко не всегда, и если вы подозреваете потерю пакетов на стороне сервера и при этом не обнаруживаете никаких дублирующих подтверждений, попробуйте перехватить пакеты с клиентской стороны установленного соединения.

- **Пакеты с нулевым размером окна приема и поддержания активным соединения.** Механизм скользящего окна непосредственно связан с неспособностью сервера принимать и обрабатывать данные. Любое уменьшение размера окна приема или даже его установка в нулевое состояние является непосредственным результатом какого-то затруднения, возникшего на сервере. И если в сети происходит любое из этих событий, то уделите ему самое пристальное внимание. Как правило, вы обнаружите пакеты с обновленным размером окна приема на обеих сторонах установленного по сети соединения.

Выявление источника большой сетевой задержки

Иногда задержки в сети не приводят к потере пакетов. При этом можно обнаружить, что хотя сеть и работает медленно, но повторные передачи по протоколу TCP или дублирующие подтверждения, не возникают, несмотря на медленный обмен данными между двумя хостами. Следовательно, требуется какой-то другой способ, позволяющий обнаружить источник большой сетевой задержки.

К числу самых эффективных способов, позволяющих обнаружить источник большой сетевой задержки, относится исследование процедуры первоначальной установки связи и первой пары следующих после этого пакетов. Рассмотрим в качестве примера соединение клиента с веб-сервером с целью просмотра веб-сайта, размещаемого на этом сервере. В данном случае нас интересуют первые шесть пакетов в анализируемой последовательности обмена данными, состоящей из процедуры трехэтапной установки связи по протоколу TCP, первоначального HTTP-запроса по методу GET, подтверждения приема этого запроса и первого пакета данных, отправленного сервером клиенту.

ПРИМЕЧАНИЕ *Для проработки материала этого раздела установите соответствующий формат отображения времени в Wireshark, выбрав команду* View⇨Time Display Format⇨Seconds Since Previous Displayed Packet *(Вид⇨Формат отображения времени⇨Количество секунд, прошедших с момента отображения предыдущего пакета) из главного меню.*

Обычный обмен данными

Файл перехвата **`latency1.pcapng`**

Исходные характеристики сети мы обсудим позднее, а до тех пор достаточно знать, что эти характеристики обычного обмена данными требуются для сравнения с условиями большой сетевой задержки. Для анализа рассматриваемых здесь примеров мы воспользуемся файлом перехвата `latency1.pcapng`. Ранее уже пояснялись подробно особенности трехэтапной установки связи по протоколу TCP и обмена данными по протоколу HTTP, поэтому мы не будем рассматривать эти вопросы снова. В действительности мы вообще не будем обращаться к панели Packet Details, поскольку для наших целей достаточно проанализировать содержимое столбца **`Time`**, как показано рис. 11.22.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.030107	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.000075	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	0.000066	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.048778	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.022176	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

Рис. 11.22. Приведенный здесь сетевой трафик возникает очень быстро и может считаться обычным

Такая последовательность обмена данными происходит довольно быстро, причем весь этот процесс занимает меньше 0,1 секунды. Ряд анализируемых далее файлов перехвата содержат одни и те же образцы сетевого трафика, но главные отличия проявляются во временны́х характеристиках пакетов.

Медленный обмен данными из-за сетевой задержки

Файл перехвата **`latency2.pcapng`**

А теперь обратимся к анализу содержимого файла перехвата `latency2.pcapng`. Обратите внимание на то, что все пакеты в этом файле одинаковы, кроме временны́х характеристик двух из них, как показано на рис. 11.23.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.878530	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.016604	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	0.000335	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	1.155228	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.015866	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

Рис. 11.23. В пакетах 2 и 5 проявляется большая задержка

Постепенно анализируя эти шесть пакетов, в них можно сразу же обнаружить первый признак сетевой задержки. Так, первоначальный пакет `SYN` отправляется клиенту, находящемуся по адресу **`172.16.16.128`**, чтобы начать процедуру установки связи по протоколу TCP, и в ответ сервер, расположенный по адресу **`74.125.95.104`**, возвращает пакет `SYN/ACK` с задержкой 0,87 секунды.

И это первый признак сетевой задержки, обусловленной устройством, находящимся между клиентом и сервером.

Из этого анализа можно сделать вывод, что в данном случае имеет место сетевая задержка, исходя из характера типов передаваемых пакетов. Когда сервер получает пакет SYN, ему требуется минимальное количество времени на его обработку и отправку ответа, поскольку рабочая нагрузка на сервер не включает в себя никакой обработки на транспортном уровне. И даже если сервер испытывает весьма интенсивный сетевой трафик, он, как правило, довольно быстро посылает пакет SYN/ACK в ответ на принятый пакет SYN. Следовательно, сервер можно исключить из числа потенциальных источников большой сетевой задержки.

Клиента следует также исключить, поскольку в данный момент на стороне клиента не происходит никакой обработки, помимо получения пакета SYN/ACK. Исключение клиента и сервера из числа потенциальных источников большой сетевой задержки указывает на то, что эти источники скрываются в первых двух пакетах из анализируемого здесь перехваченного трафика.

Продолжая анализ пакетов, можно обнаружить, что передача пакета ACK, завершающего процедуру трехэтапной установки связи по протоколу TCP, выполняется очень быстро, как это происходит с HTTP-запросом по методу GET, который посылается клиентом. Вся обработка обоих пакетов выполняется локально на стороне клиента после приема пакета SYN/ACK, и поэтому они должны передаваться быстро, при условии, что клиент не испытывает большую вычислительную нагрузку.

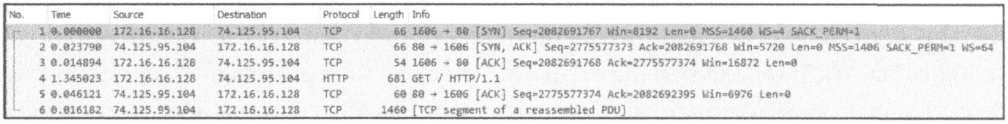
Анализ пакета 5 показывает, что это еще один пакет с очень большим временем задержки. В частности, после отправки HTTP-запроса по методу GET ответный пакет ACK был возвращен сервером через 1,15 с. После получения HTTP-запроса по методу GET, сервер должен сразу же отправить пакет ACK по протоколу TCP (т.е. еще до начала передачи данных), на что ему обычно требуется очень мало времени. И это еще один явный признак сетевой задержки.

Всякий раз, когда возникает задержка в сети, она почти всегда проявляется при обработке пакетов SYN/ACK во время первоначальной установки связи и других пакетов ACK при обмене данными. И хотя полученные сведения о сетевой задержке не сообщают о конкретном ее источнике, они свидетельствуют о том, что этим источником не является ни клиент, ни сервер. Им, скорее всего, является какое-то устройство, находящееся в промежутке между клиентом и сервером. И это служит основанием для дальнейшего анализа работы различных брандмауэров, маршрутизаторов и прокси-серверов, чтобы найти среди них виновника возникшей задержки.

Медленный обмен данными из-за задержки на стороне клиента

Файл перехвата **`latency3.pcapng`**

Следующий сценарий задержки мы исследуем на примере из файла перехвата `latency3.pcapng`, как показано на рис. 11.24.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.023790	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.014894	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	1.345023	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.046121	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.016182	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

Рис. 11.24. В этом перехваченном трафике медленным оказывается первоначальный HTTP-запрос, передаваемый по методу **`GET`**

Анализируемый здесь перехваченный трафик обычно начинается с трехэтапной установки связи по протоколу TCP, происходящей очень быстро и без всяких признаков сетевой задержки. Все выглядит правильно вплоть до пакета 4, который содержит HTTP-запрос по методу `GET`, посылаемый по окончании процедуры установки связи. В этом пакете наблюдается задержка в 1,34 с относительно предыдущего полученного пакета.

Чтобы определить источник такой задержки, необходимо выяснить, что же происходит в промежутке между обменом пакетами 3 и 4. В частности, пакет 3 является завершающим пакетом `ACK` в процедуре установки связи по протоколу TCP и посылается клиентом серверу. А пакет 4 содержит HTTP-запрос по методу `GET`, который посылается тем же самым клиентом серверу. Общим для этих пакетов служит то обстоятельство, что оба они посылаются клиентом и не зависят от сервера. HTTP-запрос по методу `GET` должен быть отправлен вскоре после отправки пакета `ACK`, поскольку все эти действия сосредоточены на стороне клиента.

К сожалению для конечного пользователя, переход от подтверждения к HTTP-запросу по методу `GET` не происходит быстро. Для составления и передачи пакета с HTTP-запросом по методу `GET` требуется обработка на уровне приложений, и задержка, возникающая при такой обработке, явно указывает на то, что клиент не сумел выполнить соответствующее действие своевременно. Таким образом мы выяснили, что в нашем случае клиент отвечает за большую задержку в обмене данными по сети.

Медленный обмен данными из-за задержки на стороне сервера

Файл перехвата **`latency4.pcapng`**

И последний сценарий задержки мы исследуем на примере из файла перехвата `latency4.pcapng`, как показано на рис. 11.25. И это пример задержки на стороне сервера.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.018583	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.016197	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	0.000172	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.047936	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.982983	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

Рис. 11.25. Большая задержка на стороне сервера не проявляется вплоть до последнего пакета в анализируемом здесь перехваченном трафике

В анализируемом здесь перехваченном трафике процесс установки связи по протоколу TCP завершается безошибочно и быстро, и поэтому обмен данными между двумя хостами начинается успешно. Следующая пара пакетов не вызывает никаких подозрений, поскольку первоначальные пакеты с HTTP-запросами по методу `GET` и подтверждающими ответами `ACK` доставляются очень быстро. И только в последнем пакете из рассматриваемого здесь файла перехвата проявляются признаки сетевой задержки.

Это шестой пакет, который посылается первым по протоколу HTTP с данными из сервера в ответ на запрос по методу `GET`, отправленный клиентом. Он поступает с задержкой 0,98 с после отправки сервером подтверждающего пакета `ACK` по протоколу TCP в ответ на полученный запрос по методу `GET`. Переход между пакетами 5 и 6 очень похож на переход между пакетами с запросом по методу `GET` и подтверждающим ответом `ACK`, как пояснялось в предыдущем сценарии. Но в данном случае в центре нашего внимания оказывается сервер, а не клиент.

Пакет 5 содержит подтверждение `ACK`, посылаемое сервером в ответ на запрос по методу `GET` от клиента. Как только этот пакет будет отправлен, сервер должен практически сразу же приступить к передаче запрашиваемых данных. Доступ к запрашиваемым данным, их упаковка и передача в этом пакете выполняется по протоколу HTTP. А поскольку это протокол уровня приложений, сервер должен выполнить некоторую обработку данных. Задержка в приеме этого пакета указывает на то, что сервер оказался не в состоянии обработать запрашиваемые данные в течение приемлемого промежутка времени, а следовательно, он и есть основной источник задержки, зафиксированной в анализируемом здесь файле перехвата.

Порядок обнаружения задержек в сети

Проанализировав шесть пакетов в ряде предыдущих сценариев, нам удалось обнаружить источник большой сетевой задержки между клиентом и сервером. Блок-схема, приведенная на рис. 11.26, должна помочь вам в диагностике затруднений, возникающих из-за сетевых задержек. Рассматриваемые здесь принципы можно применить к любому виду обмена данными по протоколу TCP.

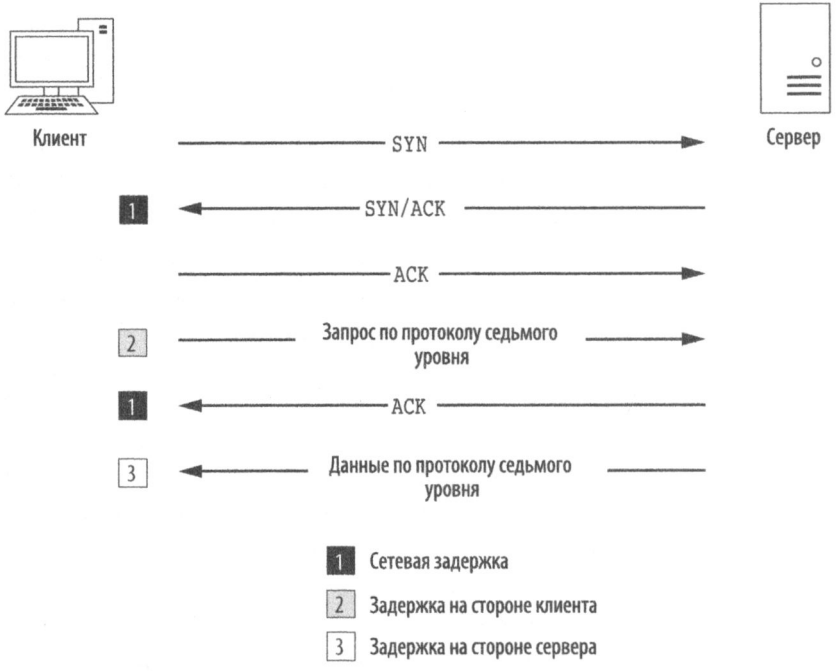


Рис. 11.26. Этой блок-схемой можно пользоваться при диагностике проблем, возникающих из-за сетевых задержек

ПРИМЕЧАНИЕ *Обратите внимание на то, что мы практически ничего не упоминали о сетевых задержках по протоколу UDP. Дело в том, что протокол UDP предназначен для быстрого, но ненадежного обмена данными, и поэтому в нем отсутствуют встроенные средства для обнаружения и устранения сетевых задержек. Вместо этого вся работа перекладывается на протоколы уровня приложений (и протокол ICMP), обеспечивающие надежную доставку данных по сети.*

Сравнение с исходными характеристиками сети

Если все описанные выше методики и способы не помогают, то остается только сравнение с исходными характеристиками сети, дающими едва ли не самые важные сведения о сети при диагностике ее медленной работы. *Исходные характеристики сети* состоят из образцового трафика, перехваченного в разных точках сети и включающего в себя немалую долю того, что принято считать "нормальным" сетевым трафиком. Назначение исходных характеристик сети — служить эталоном для сравнения в тех случаях, когда сеть или устройства в ней функционируют неверно.

В качестве примера рассмотрим сценарий, где несколько клиентов жалуются на медленную работу сети, когда они регистрируются на локальном сервере веб-приложений. Если перехватить этот сетевой трафик и сравнить его с исходными характеристиками сети, то можно обнаружить, что веб-сервер, как правило, отвечает, но внешние DNS-запросы, возникающие в результате встраивания внешнего содержимого в веб-приложение, выполняются в два раза медленнее, чем обычно.

Допустим, что медленно работающий внешний DNS-сервер удалось выявить, не прибегая к исходным характеристикам сети. Но совсем другое дело, когда приходится иметь дело с незначительными изменениями в работе сети. В частности, обработка каждого из десяти DNS-запросов может отнять на 0,1 с больше, чем в обычных условиях, что в конечном итоге приводит к такой же задержке, как и при обработке одного DNS-запроса в течение 1 с. Но обнаружить первую ситуацию намного труднее без сравнения с исходными характеристиками сети.

Двух одинаковых сетей не бывает, и поэтому отдельные составляющие исходных характеристик сети могут существенно отличаться. В последующих разделах представлены примеры отдельных видов составляющих исходных характеристик сети. Возможно, все эти составляющие или только некоторые из них вполне подойдут для инфраструктуры вашей сети. Но как бы там ни было, вы можете отнести каждую составляющую исходных характеристик своей сети к одной из следующих категорий: сайт, хост и приложение.

Исходные характеристики сети для сайта

Назначение исходных характеристики сети для сайта — получить общий моментальный снимок сетевого трафика на каждом сайте в сети. В идеальном случае это может быть каждый сегмент глобальной сети. К составляющим этой категории исходных характеристик сети относятся следующие.

- **Применяемые протоколы.** Чтобы проанализировать сетевой трафик от всех устройств, откройте окно Protocol Hierarchy Statistics (Статистика иерархии протоколов) по команде главного меню Statistics⇨Protocol Hierarchy в процессе перехвата сетевого трафика от всех устройств в сегменте сети, находящемся на ее краю (маршрутизаторов или брандмауэром). В дальнейшем его можно сравнить с выводом из иерархии протоколов, чтобы выяснить, отсутствуют ли обычно присутствующие или новые протоколы, самостоятельно внедренные в сети. Этим выводом можно также воспользоваться, чтобы выявить объемы сверх обычного определенных типов трафика в зависимости от конкретного протокола.
- **Широковещательный трафик.** К этой составляющей исходных характеристик относится весь широковещательный трафик в сегменте сети. Прослушивание сети в любой точке сайта позволяет перехватить весь широковещательный трафик и выяснить, кто или что обычно посылает в большом количестве широковещательные пакеты по сети. И тогда можно быстро определить, является широковещательная передача пакетов чрезмерной или недостаточной.
- **Последовательности аутентификации.** К этой составляющей исходных характеристик относится сетевой трафик, проходящий от произвольных клиентов, где выполняются процессы аутентификации, ко всем службам вроде Active Directory, веб-приложений и программного обеспечения, специально применяемого в организации. Аутентификация относится к одним из наиболее распространенных причин медленной работы служб. А исходные характеристики сети позволяют выяснить, является ли аутентификация виной медленного обмена данными по сети.
- **Скорость передачи данных.** Как правило, это мера, определяющая быстроту передачи крупного объема данных различным сайтам в сети. Скорость передачи данных и устойчивость соединения можно определить средствами Wireshark для вывода сводных статистических данных о перехваченном трафике и построения графиков, как пояснялось в главе 5, "Дополнительные возможности Wireshark". Это едва ли не самая важная исходная характеристика сети из всех доступных для сайта. Всякий раз, когда устанавливается или разрывается соединение с медленно работающим сегментом сети, можно передать тот же самый фрагмент данных, что и в исходных характеристиках сети, и сравнить полученные результаты. Такое сравнение позволит выяснить, является ли соединение медленным, а возможно, и выявить участок сети, на котором она начинает медленно работать.

Исходные характеристики сети для хоста

Возможно, определять исходные характеристики сети для каждого хоста, т.е. узла сети, и не нужно. Исходные характеристики сети для хоста следует определять только на серверах с интенсивным трафиком или ответственных серверах. По существу, если медленная работа сервера станет причиной гневных звонков со стороны начальства, то на такие случаи следует иметь в своем распоряжении исходные характеристики сети на данном хосте. К составляющим этой категории исходных характеристик сети относятся следующие.

- **Применяемые протоколы.** Эта составляющая исходных характеристик дает отличную возможность воспользоваться окном Protocol Hierarchy Statistics в процессе перехвата сетевого трафика от хоста. В дальнейшем его можно сравнить с данной составляющей исходных характеристик сети, чтобы выяснить, отсутствуют ли обычно присутствующие или новые протоколы, самостоятельно внедренные на хосте. Подобным способом можно также выявить необычные объемы определенных типов трафика в зависимости от конкретного протокола.
- **Холостой или интенсивный сетевой трафик.** Эта составляющая исходных характеристик сети состоит из общих перехватов обычного действующего трафика в периоды пиковой и непиковой нагрузки. Зная количество соединений и степень их пропускной способности в разное время суток, можно выяснить, является ли пользовательская нагрузка или иная проблема причиной медленной работы сети.
- **Последовательности запуска и остановки.** Чтобы получить эту составляющую исходных характеристик сети, придется перехватить сетевой трафик, формируемый во время последовательностей запуска и остановки хоста. Если же компьютер отказывается выполнить запуск, остановку или работает крайне медленно во время любой из этих последовательностей, то с помощью данной составляющей исходных характеристик можно выяснить, является ли тому причиной сама сеть.
- **Последовательности аутентификации.** Чтобы получить эту составляющую исходных характеристик сети, придется перехватить сетевой трафик, возникающий в процессах аутентификации во всех службах на данном хосте. Аутентификация — это одна из тех областей, где службы обычно действуют медленно. С помощью этой составляющей исходных характеристик сети можно выяснить, является ли аутентификация виной медленного объема данными.
- **Ассоциации и зависимости.** Эта составляющая исходных характеристик сети содержит длинный перехваченный трафик, чтобы выяснить,

от каких именно хостов зависит нормальная работа данного хоста, и наоборот, их работа от него. Чтобы выявить подобные ассоциации и зависимости, достаточно воспользоваться окном Conversations, доступным по команде главного меню Statistics⇨Conversations. Характерным тому примером служит хост SQL Server, от которого зависит нормальная работа веб-сервера. Нам не всегда известны базовые зависимости между хостами, и поэтому их можно выяснить с помощью этой составляющей исходных характеристик сети на хосте. И на этом основании можно сделать вывод, функционирует ли хост ненормально из-за неисправности или сильной зависимости от сетевой задержки.

Исходные характеристики сети для приложений

И к последней категории относятся исходные характеристики сети для приложений. Они должны быть определены во всех сетевых приложениях, критически важных для ведения коммерческой деятельности. К составляющим этой категории исходных характеристик сети относятся следующие.

- **Применяемые протоколы.** И эта составляющая исходных характеристик дает отличную возможность воспользоваться окном Protocol Hierarchy Statistics в процессе перехвата сетевого трафика от хоста, на котором работает приложение. В дальнейшем его можно сравнить с данной составляющей исходных характеристик сети, чтобы выяснить, нормально ли функционируют протоколы, от которых зависит данное приложение.
- **Последовательности запуска и остановки.** Эта составляющая исходных характеристик сети содержит перехваченный сетевой трафик, формируемый во время последовательностей запуска и остановки хоста. Если же приложение отказывается выполнить запуск или работает крайне медленно во время любой из этих последовательностей, то с помощью данной составляющей исходных характеристик можно выяснить, является ли тому причиной сама сеть.
- **Ассоциации и зависимости.** Эта составляющая исходных характеристик сети требует наличия длинного перехваченного трафика и окна Conversations, чтобы выяснить, от каких именно хостов и приложений зависит нормальная работа данного приложения. Нам не всегда известны базовые зависимости между приложениями, и поэтому их можно выяснить с помощью этой составляющей исходных характеристик сети на уровне приложения. И на этом основании можно сделать вывод, функционирует ли приложение ненормально из-за неисправности или сильной зависимости от сетевой задержки.

- **Скорость передачи данных.** Скорость передачи данных и устойчивость соединения можно определить средствами Wireshark для вывода сводных статистических данных о перехваченном трафике и построения графиков, как пояснялось в главе 5, "Дополнительные возможности Wireshark". Всякий раз, когда возникают жалобы на медленно работающее приложение, этой составляющей исходных характеристик сети на уровне приложения можно воспользоваться, чтобы выяснить, является ли причиной возникающих затруднений высокий коэффициент использования приложения и большая пользовательская нагрузка.

Дополнительные рекомендации относительно исходных характеристик сети

Ниже приведен ряд дополнительных рекомендаций, которые следует иметь в виду при получении исходных характеристик сети.

- При получении исходных характеристик сети перехватывайте сетевой трафик по каждой из категорий не меньше трех раз: один раз – в период малой нагрузки (рано утром), другой раз – в период большой нагрузки (в полдень), а в третий раз – в период холостой работы (в полночь).
- Старайтесь не перехватывать сетевой трафик непосредственно от тех хостов, где производится сравнение с исходными характеристиками сети. В периоды интенсивного трафика это может привести к увеличению нагрузки на устройство, понизить его производительность и нарушить достоверность исходных характеристик сети вследствие пропуска пакетов.
- Исходные характеристики сети будут содержать секретные данные о сети, поэтому их следует непременно защитить. Храните их в надежном месте, доступ к которому разрешен строго определенному кругу лиц. Но в то же время они должны быть доступны в любой удобный момент, когда они могут понадобиться. Выделите для этой цели отдельный накопитель USB-флеш или шифруемый раздел жесткого диска.
- Храните все полученные исходные характеристики сети в файлах перехвата с расширением `.pcap` и `.pcapng` и составьте памятку из наиболее употребительных составляющих исходных характеристик сети, например, ассоциаций или средних скоростей передачи данных.

Заключительные соображения

В этой главе основное внимание было уделено диагностике основных причин медленной работы сети. В ней был рассмотрен ряд полезных функциональных возможностей протокола TCP для повышения надежности и исправления ошибок при передаче данных, продемонстрированы методики обнаружения источников сетевых задержек при обмене данными, а также описаны категории составляющих исходных характеристик сети и их назначение. Применяя методики, представленные в этой главе, наряду со средствами построения графиков и анализа пакетов в Wireshark, вы должны быть вполне готовы к диагностике сети, когда кто-нибудь пожалуется на ее медленную работу.

12

АНАЛИЗ ПАКЕТОВ НА БЕЗОПАСНОСТЬ

Бо́льшая часть этой книги посвящена проведению анализа пакетов для диагностики сети. Тем не менее анализ пакетов проводится в немалой степени и в целях безопасности. Например, аналитик, исследующий незаконное вторжение в систему, может просмотреть сетевой трафик, исходящий от потенциальных нарушителей, а следователь-криминалист — попытаться выяснить, в какой степени был заражен зловредным программным обеспечением взломанный хост.

Проведение анализа пакетов в ходе расследования случаев нарушения безопасности всегда считалось непростым делом, поскольку оно включает в себя неизвестный элемент устройства, которое подвергается атаке злоумышленника. В самом деле, нельзя же добраться до рабочего места атакующего злоумышленника и подвергнуть его допросу или сравнить его обычный трафик с исходным. Вместо этого приходится иметь дело с перехваченным образцом взаимодействия своей и чужой системы. Правда, для удаленного вторжения в чужие системы злоумышленникам придется так или иначе взаимодействовать с сетью. И это им хорошо известно, поэтому у них нет недостатка в приемах маскирования своей вредоносной деятельности.

В этой главе рассматриваются различные практические аспекты нарушения безопасности системы на уровне сети, включая обследование сети,

переадресацию вредоносного трафика и самые распространенные методики внедрения вредоносного кода. В некоторых случаях будет проведен анализ незаконного вторжения в систему для интерпретации сетевого трафика на основании предупреждений от системы обнаружения вторжений. Проработав материал этой главы, вы получите ясное представление о сетевой безопасности, что может оказаться для вас очень важным, даже если вы и не занимаетесь в настоящий момент вопросами безопасности.

Обследование сети

Злоумышленник первым делом выполняет углубленное обследование целевой системы. Эта стадия обычно называется *получением отпечатка (footprinting)* и нередко осуществляется с помощью различных общедоступных ресурсов, включая веб-сайт целевой системы или компании Google. По окончании подобного обследования злоумышленник, как правило, приступает к сканированию целевой системы по ее IP-адресу (или DNS-имени) на наличие в ней открытых портов или действующих служб.

Сканирование дает злоумышленнику возможность выяснить, активна и доступна ли целевая система. Рассмотрим в качестве примера сценарий, где преступники собираются ограбить самый крупный банк в каком-нибудь городе. Потратив не одну неделю на тщательную проработку плана ограбления и прибыв на место преступления, они в конечном итоге обнаружили, что банк переехал на другую улицу. Представьте еще более скверную ситуацию, когда преступники собирались ограбить банк в обычные часы его работы, украв ценности из банковского хранилища, но, прибыв на место преступления, обнаружили, что банк в этот день закрыт. Таким образом, первое препятствие, которое придется преодолеть при ограблении банка или совершении атаки на сеть, — убедиться, что целевой хост активен и доступен.

Кроме того, сканирование сети позволяет злоумышленнику выяснить, через какие порты целевая система взаимодействует с сетью. Если снова обратиться к аналогии ограбления банка, то нетрудно представить, что произойдет, если преступники заявятся в банк, совершенно не зная планировку его помещения. Им не удастся добраться до банковского хранилища, не зная слабых мест в физической защите банка. В этом разделе будут рассмотрены самые распространенные методики сканирования сети, применяемые для выявления хостов, их открытых портов и уязвимостей в сети.

ПРИМЕЧАНИЕ *До сих пор в этой книге передающая и принимающая стороны сетевого соединения назывались клиентом и сервером. А в этой главе эти стороны называются злоумышленником и целевым хостом соответственно.*

Сканирование пакетами SYN

Файл перехвата `synscan.pcapng`

Зачастую первым видом обследования целевой системы является *сканирование пакетами SYN по протоколу TCP*, иначе называемое *скрытым сканированием* или же *сканированием полуоткрытых соединений*. Сканирование пакетами `SYN` считается самым распространенным по ряду следующих причин.

- Проводится очень быстро и надежно.
- Пригодно для всех платформ независимо от реализации стека протоколов TCP/IP.
- Вносит меньше помех, чем другие методики сканирования.

В основу сканирования пакетами `SYN` положен трехэтапный процесс установки связи по протоколу TCP, что позволяет выяснить, какие именно порты открыты на целевом хосте. В этом случае злоумышленник посылает пакет `SYN` по протоколу TCP на определенный ряд портов целевого хоста, как будто бы пытаясь установить с ним канал для обычного обмена данными через его порты. Как только этот пакет будет получен целевым хостом, может произойти одно из нескольких событий, как показано на рис. 12.1.

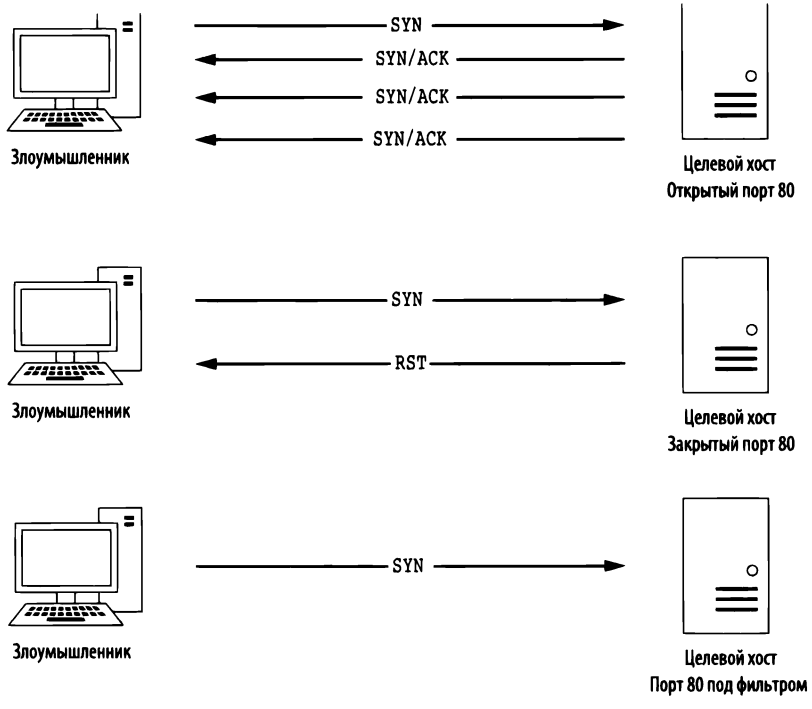


Рис. 12.1. Возможные результаты сканирования пакетами SYN по протоколу TCP

Если служба на целевой машине прослушивает сеть через порт, принимающий пакет SYN, она ответит злоумышленнику пакетом SYN/ACK по протоколу TCP, обычно отправляемом на втором этапе процесса установки связи по этому протоколу. В итоге злоумышленнику становится известно, что порт открыт и служба прослушивает сеть через него. При обычных обстоятельствах окончательный пакет ACK должен быть отправлен по протоколу TCP с целью завершить процесс установки связи. Но в данном случае злоумышленнику совсем не нужно, чтобы это произошло, поскольку ему не требуется больше связь с хостом в данный момент. Следовательно, злоумышленник не будет пытаться завершить процесс установки связи по протоколу TCP.

Если же ни одна из служб не прослушивает сеть через сканируемый порт, злоумышленник не получит пакет SYN/ACK. Но он может получить в ответ пакет RST, указывающий на то, что порт закрыт, хотя это зависит от конкретной конфигурации операционной системы целевого компьютера. С одной стороны, злоумышленник может вообще не получить ничего в ответ. Отсутствие ответа может означать, что сканируемый порт находится под фильтром на промежуточном устройстве, например, брандмауэре или на самом хосте. А с другой стороны, ответ может быть утерян при передаче. Таким образом, результат такого сканирования получается неоднозначным, несмотря на то, что он, как правило, указывает на то, что порт закрыт.

В файле перехвата synscan.pcapng предоставляется характерный пример сканирования пакетами SYN, выполняемого средствами Nmap — надежного приложения для сканирования сети, разработанного Гордоном Лайеном (Gordon Lyon) по прозвищу "Федор". Оно способно выполнять практически любые вообразимые виды сканирования сети. Свободно загрузить приложение Nmap можно по адресу http://www.nmap.com/download.html.

В данном примере перехваченный трафик состоит приблизительно из 2000 пакетов. Это означает, что анализируемое здесь сканирование сети произведено в приемлемых масштабах. Чтобы установить пределы такого сканирования, лучше всего просмотреть его результаты в окне Conversations, как показано на рис. 12.2. В этом окне должен быть показан единственный диалог по протоколу IPv4 ❶ между злоумышленником, находящимся по адресу **172.16.0.8**, и целевым хостом, расположенным по адресу **64.13.134.52**. Кроме того, в окне Conversations можно увидеть, что между указанными выше хостами проведено 1994 диалога по протоколу TCP ❷, а, по существу, для каждой пары портов, задействованных в обмене данными, организован новый диалог.

Сканирование сети происходит очень быстро, и поэтому прокрутка содержимого файла перехвата — далеко не самый лучший способ поиска ответа на каждый первоначальный пакет SYN. Ведь прежде получения ответа на первоначальный пакет может быть отправлен ряд дополнительных пакетов.

Правда, для обнаружения нужного сетевого трафика можно составить вспомогательные фильтры.

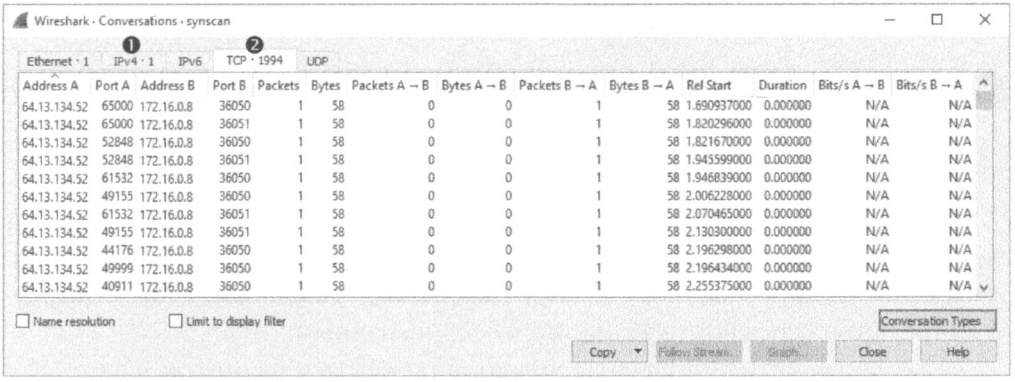


Рис. 12.2. В окне Conversations наглядно демонстрируется разнообразие обменов данными по протоколу TCP

Использование фильтров при сканировании пакетами SYN

В качестве примера фильтрации результатов сканирования сети рассмотрим первый пакет из упомянутого выше файла перехвата. Это пакет `SYN`, отправленный целевому хосту через порт **443** (по протоколу HTTPS). Чтобы выяснить, получен ли ответ на этот пакет, можно создать фильтр для показа всего исходящего трафика через порт **443**. Ниже поясняется, как это сделать очень быстро.

1. Выберите первый пакет из файла перехвата `synscan.pcapng`.
2. Разверните TCP-заголовок этого пакета в панели Packet Details.
3. Щелкните правой кнопкой мыши на поле Destination Port (Порт получателя), выберите команду Prepare as Filter⇨Selected (Подготовить как фильтр⇨Выбрать) из контекстного меню.
4. В итоге выбранный фильтр появится в диалоговом окне фильтров всех пакетов, проходящих через порт **443**. Щелкните в верхней части диалогового окна фильтров и удалите часть **dst** из выбранного фильтра, поскольку нам требуется получить список всех пакетов (как входящих, так и исходящих) по порту **443**.

Получающийся в итоге фильтр отсеет два пакета, которые являются пакетами `SYN`, передаваемыми по протоколу TCP от злоумышленника к целевому хосту, как показано на рис. 12.3.

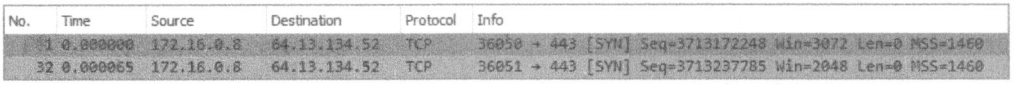


Рис. 12.3. Две попытки установки соединения с помощью пакетов SYN

ПРИМЕЧАНИЕ *В этом разделе при отображении пакетов используется формат времени* Seconds Since Previous Displayed Packet *(Количество секунд, прошедших с момента отображения предыдущего пакета).*

Отсутствие ответа на эти пакеты, вероятно, свидетельствует о том, что ответ был отфильтрован целевым хостом или промежуточным устройством, а, возможно, сканируемый порт оказался закрытым. В конечном счете результат сканирования через порт **443** оказался неоднозначным.

Ту же самую методику анализа можно опробовать и на другом пакете, чтобы попытаться получить иные результаты. С этой целью удалите предыдущий фильтр и выберите из списка пакет 9. Это пакет SYN, отправляемый в порт **53**, который обычно связан со службой DNS. Применяя описанную выше методику или внеся коррективы в предыдущий фильтр, составьте новый фильтр для отображения всего трафика, проходящего через порт **53** по протоколу TCP. Применив этот фильтр, вы должны в итоге обнаружить пять пакетов (рис. 12.4).

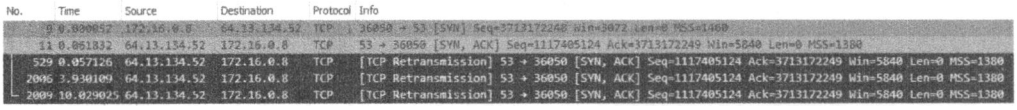


Рис. 12.4. Пять пакетов, указывающих на открытый порт

Первым из них является пакет 9 типа SYN, выбранный вначале перехваченного трафика. А вторым пакетом следует ответ SYN/ACK, посылаемый целевым хостом и вполне ожидаемый при трехэтапном процессе установки связи по протоколу TCP. При нормальных обстоятельствах следующим должен быть пакет ACK, отправляемый хостом, пославшим первоначальный пакет SYN. Но в данном случае злоумышленнику совсем не нужно завершать трехэтапный процесс установки связи, поэтому ответ не посылается. В итоге целевой хост повторно передаст пакет SYN/ACK еще три раза, прежде чем процесс установки связи будет аварийно завершен. А поскольку ответ SYN/ACK получен при попытки связаться с хостом через порт **53**, то благоразумнее предположить, что сканируемая служба прослушивает сеть через данный порт.

Очистите полученный результат и повторите данный процесс еще раз для пакета 13. Это пакет SYN, отправляемый в порт **113**, который обычно связан с протоколом Ident, зачастую применяемым для распознавания служб IRC и

аутентификации в них. Если применить фильтр того же самого типа к порту, перечисленному в данном пакете, то в итоге появятся четыре пакета, как показано на рис. 12.5.

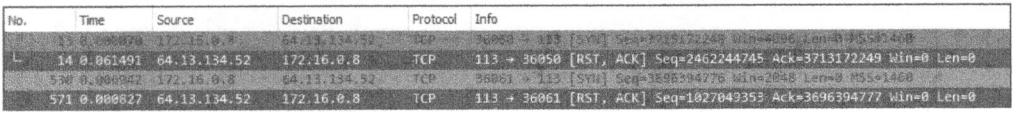


Рис. 12.5. Пакет SYN, вслед за которым посылается пакет RST, указывающий на то, что порт закрыт

Первым в данном случае следует первоначальный пакет SYN, вслед за которым целевой хост сразу же отправляет пакет RST. Это свидетельствует о том, что целевой хост не принимает подключения через свой порт и что служба на нем, вероятнее всего, не запущена.

Выявление открытых и закрытых портов

А теперь, когда стали понятны разные типы ответов, которые способно выявить сканирование пакетами SYN, требуется найти какой-нибудь быстрый способ выявления открытых и закрытых портов. И с этой целью можно снова обратиться к окну Conversations, где диалоги по протоколу TCP нетрудно отсортировать по номерам пакетов, чтобы первыми следовали наибольшие значения. Для этого достаточно щелкнуть на заголовке **Packets**, чтобы появилась стрелка, указывающая вниз, как показано на рис. 12.6.

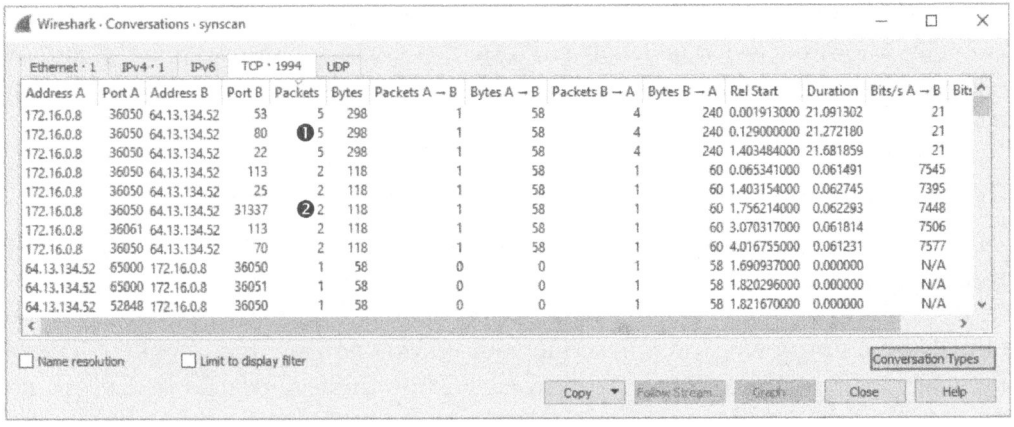


Рис. 12.6. Выявление открытых портов в окне Conversations

В каждом диалоге через три сканируемых порта проходят по пять пакетов ❶. Известно, что порты **53**, **80** и **22** открыты, поскольку эти пять пакетов представляют следующую последовательность установления связи:

первоначальный пакет SYN, соответствующий ему пакет SYN/ACK, а также три пакета SYN/ACK, повторно передаваемых целевым хостом.

Из пяти портов только два пакета были вовлечены в обмен данными ❷. Первым из них является первоначальный пакет SYN, а вторым — пакет RST, отправляемый целевым хостом. Эти пакеты указывают на то, что порты **113**, **25**, **31337** и **70** закрыты.

Остальные записи, отображаемые в окне Conversations, включают в себя только один пакет. Это означает, что целевой хост вообще не ответил на первоначальный пакет SYN. А оставшиеся порты, скорее всего, закрыты, но в этом нет никакой уверенности.

Рассматриваемая здесь методика подсчета пакетов вполне подошла для сканирования данного хоста, хотя она пригодна не для всех сканируемых хостов, а следовательно, полагаться только на них не стоит. Вместо этого особое внимание следует уделить изучению внешнего вида и возможного назначения обычных и необычных реакций на обычные стимулы.

Получение отпечатка операционной системы

Злоумышленник придает большое значение сведениям о целевой операционной системе. Эти сведения помогают ему правильно настроить все свои методы атаки на данную систему, а также позволяют выяснить местоположение определенных критически важных файлов и каталогов в целевой файловой системе, если, конечно, ему удастся получить доступ к ней.

Получение отпечатка операционной системы (operating system fingerprinting) обозначает группу методик, применяемых с целью определить операционную систему, работающую в целевой системе без физического доступа к данной системе. Имеются два способа получения отпечатка операционной системы: пассивный и активный.

Пассивное получение отпечатка операционной системы

Файл перехвата **passiveos-fingerprinting.pcapng**

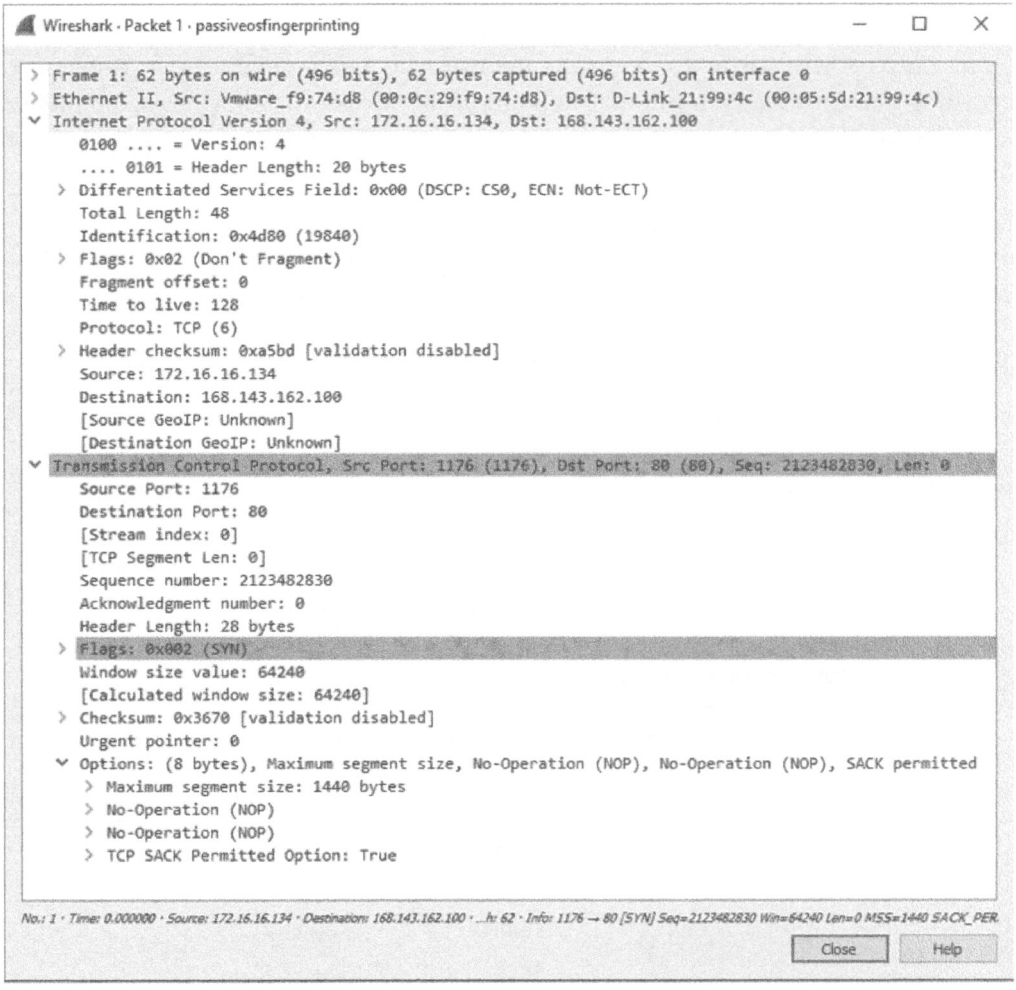
Используя методику *пассивного получения отпечатка*, можно исследовать определенные поля в пакетах, отправленных целевым хостом, чтобы определить применяемую в нем операционную систему. Такой способ считается пассивным, поскольку в этом случае вы принимаете только те пакеты, которые отправляет целевой хост, но сами активно не посылаете этому хосту никаких пакетов. Такой способ получения отпечатка операционной системы идеально подходит для злоумышленников, поскольку он дает им возможность действовать скрытно.

Так как же определить, какая именно операционная система работает на целевом хосте, исходя только из пакетов, которые он отправляет? Это возможно благодаря отсутствию стандартизированных значений в спецификациях, определенных в документах RFC на сетевые протоколы. Несмотря на то что различные поля в TCP-, UDP- и IP-заголовках имеют специальное назначение, устанавливаемые по умолчанию значения, как правило, определены не в каждом поле. Это означает, что реализация стека протоколов TCP/IP в каждой операционной системе требует определения собственных значений, устанавливаемых по умолчанию в этих полях. В табл. 12.1 перечислены наиболее употребительные поля и устанавливаемые в них значения, которые могут быть связаны с различными операционными системами. Следует, однако, иметь в виду, что эти значения могут быть изменены в новых выпусках операционных систем.

Таблица 12.1. Поля заголовков и их значения, используемые при пассивном получении отпечатка операционной системы

Заголовок протокола	Поле	Значение по умолчанию	Платформа
IP	`Initial time to live` (Первоначальное время жизни)	`64`	NMap, BSD, OS X, Linux
		`128`	Novell, Windows
		`255`	Cisco IOS, Palm OS, Solaris
IP	`Don't fragment` (Не фрагментировать)	Флаг установлен	BSD, OS X, Linux, Novell, Windows, Palm OS, Solaris
		Флаг сброшен	Nmap, Cisco IOS
TCP	`Maximum segment size` (Максимальный размер сегмента)	`0`	Nmap
		`1440-1460`	Windows, Novell
		`1460`	BSD, OS X, Linux, Solaris
TCP	`Window size` (Размер окна)	`1024-4096`	Nmap
		`65535`	BSD, OS X
		Переменное	Linux
		`16384`	Novell
		`4128`	Cisco IOS
		`24820`	Solaris
		Переменное	Windows
TCP	`SackOK` (Выборочное подтверждение разрешено)	Флаг установлен	Linux, Windows, OS X, OpenBSD
		Флаг сброшен	Nmap, FreeBSD, Novell, Cisco IOS, Solaris

Пакеты, содержащиеся в файле перехвата `passiveosfingerprinting.pcapng`, служат характерными примерами данной методики. В этом файле содержатся два пакета. Оба пакета относятся к типу `SYN` и отправляются по протоколу TCP в порт **80**, но они исходят из разных хостов. Используя только значения, содержащиеся в этих пакетах и обращаясь за справкой к табл. 12.1, можно определить архитектуру операционной системы, применяемой на каждом хосте. Подробные сведения о каждом пакете приведены на рис. 12.7.



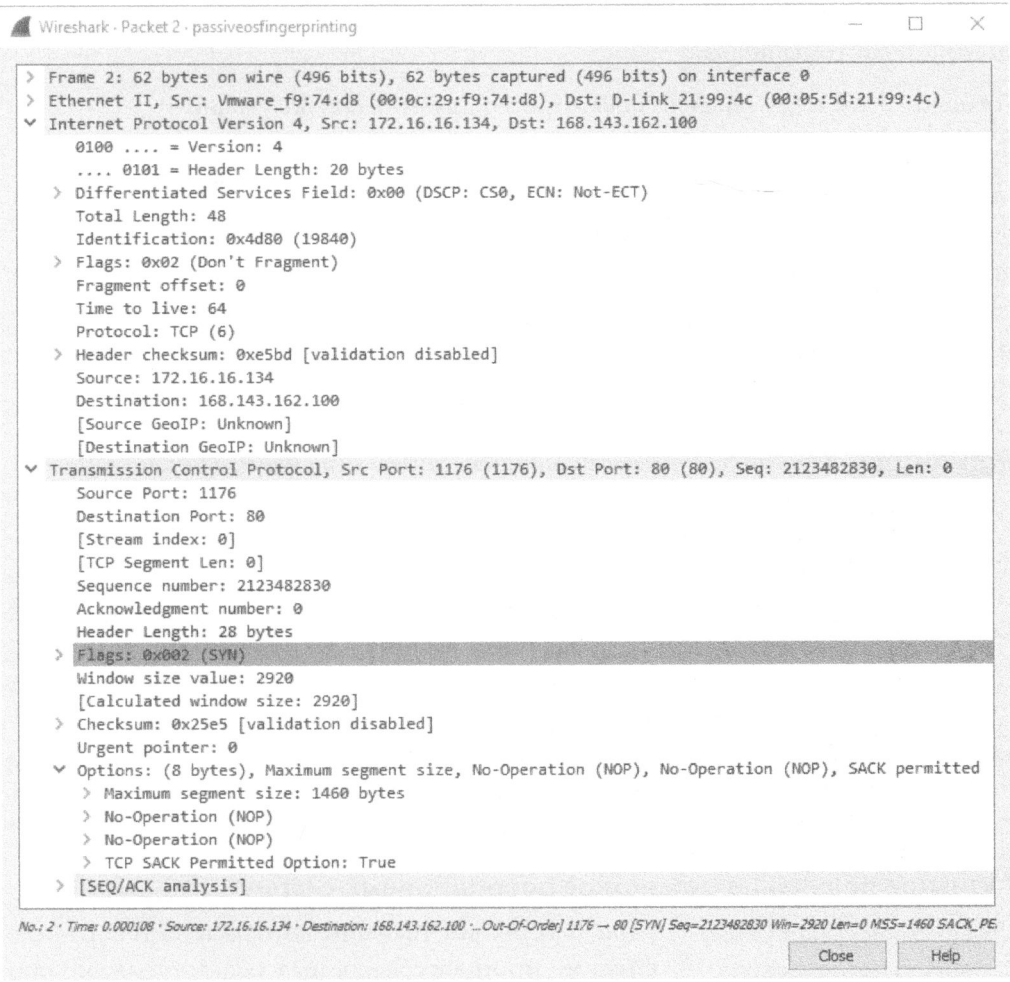


Рис. 12.7. По приведенным здесь пакетам можно узнать, из какой операционной системы они были присланы

Используя табл. 12.1 в качестве справки, можно составить табл. 12.2, где сведены соответствующие поля из этих пакетов. Основываясь на этих значениях, можно прийти к выводу, что пакет 1 был, вероятнее всего, отправлен устройством, работающим под управлением ОС Windows, а пакет 2 — устройством, работающим под управлением Linux.

Следует, однако, иметь в виду, что перечень полей, наиболее употребительных при получении отпечатка операционной системы и приведенных в табл. 12.2, нельзя считать исчерпывающим. Хотя методика применения этих полей может иметь немало неожиданных уверток, способных привести к заметным отклонениям от предполагаемых значений. Следовательно,

полностью полагаться на результаты получения пассивного отпечатка операционной системы нельзя.

Таблица 12.2. Сводка полей из пакетов, используемых при получении отпечатка операционной системы

Заголовок протокола	Поле	Значение из пакета 1	Значение из пакета 2
IP	`Initial time to live`	128	64
IP	`Don't fragment`	Флаг установлен	Флаг установлен
TCP	`Maximum segment size`	1440 байт	1460 байт
TCP	`Window size`	64240 байт	2920 байт
TCP	`SackOK`	Флаг установлен	Флаг установлен

ПРИМЕЧАНИЕ *Зачастую злоумышленники используют автоматизированные инструментальные средства для пассивного выявления типа операционной системы на целевом хосте. К числу таких средств, в которых применяются способы получения отпечатков операционных систем, относится p0f. Это инструментальное средство позволяет анализировать соответствующие поля в перехваченных пакетах и выводить сведения о предполагаемом типе операционной системы. Применяя инструментальные средства вроде p0f, можно получить сведения не только об архитектуре операционной системы, но иногда даже ее версию или уровень установки пакетов обновлений. Инструментальное средство p0f можно загрузить по адресу* **`http://lcamtuf.coredump.cx/p0f.shtml`**.

Активное получение отпечатков операционных систем

Файл перехвата `activeos-fingerprinting.pcapng`

Если пассивный текущий контроль сетевого трафика не приносит желаемых результатов, можно опробовать более прямой способ — *активное получение отпечатков операционных систем.* В этом случае злоумышленник активно посылает специально составленные пакеты целевому хосту, чтобы выявить ответы, позволяющие выяснить тип операционной системы на целевой машине. Безусловно, это не особенно скрытный, хотя и весьма эффективный способ, поскольку он подразумевает установление непосредственной связи с целевым хостом.

Пакеты, содержащиеся в файле перехвата `activeosfingerprinting.pcapng`, служат характерными примерами методики активного получения отпечатков операционных систем, инициируемых утилитой сканирования Nmap. В этом файле несколько пакетов получены в ответ на отправку утилитой Nmap ряда зондирующих пакетов с целью выявить в ответных пакетах характерную информацию, по которой можно будет идентифицировать тип операционной

системы целевого хоста. Утилита Nmap зафиксировала ответы на эти зондирующие пакеты и создала активный отпечаток операционной системы, который затем сравнила с информацией, хранящейся в базе данных для принятия решения.

ПРИМЕЧАНИЕ *Методики, применяемые в Nmap для активного получения отпечатка операционной системы, довольно сложные. Чтобы узнать больше, каким образом в Nmap выполняется активное получение отпечатка операционной системы, обратитесь за справкой к руководству по Nmap под названием Nmap Network Scanning, опубликованному в 2011 году его автором, Гордоном Лайеном по прозвищу "Федор".*

Манипулирование сетевым трафиком

Одна из самых главных идей, которые я попытался раскрыть в этой книге, заключается в том, что, анализируя подходящие пакеты, можно узнать немало о системе и ее пользователях. Следовательно, нет ничего удивительного в том, что злоумышленники нередко ищут возможность перехватить такие пакеты самостоятельно. Анализируя пакеты, сформированные системой, злоумышленник может узнать о типе операционной системы, применяемых в ней приложениях, учетных записях пользователей и много другой полезной для себя информации.

В этом разделе будут рассмотрены две методики, применяемые на уровне пакетов. С их помощью злоумышленник может воспользоваться заражением ARP-кеша для перехвата и фиксации целевого трафика или перехватом cookies-файлов протокола HTTP, с помощью которых можно проводить атаки по перехвату и подмене сеансов связи.

Заражение ARP-кеша

Файл перехвата `arppoison.pcapng`

Как обсуждалось в главе 7, "Протоколы сетевого уровня", протокол ARP служит для преобразования IP-адресов устройств в MAC-адреса в сети, а в главе 2, "Подключение к сети", пояснялось, как воспользоваться методикой заражения ARP-кеша для подключения к сети и перехвата сетевого трафика, исходящего из тех хостов, пакеты которых требуется проанализировать. Если заражение ARP-кеша применяется в благих целях, оно оказывает немалую помощь в диагностике сети. Но если эта методика употребляется со злым умыслом, то она превращается в летальную форму *атаки через посредника* (MITM), иначе *называемой атакой типа "человек посредине"*.

Совершая атаку через посредника, злоумышленник переадресует сетевой трафик между двумя хостами, чтобы перехватить или видоизменить пере-

даваемые данные. Имеется немало форм атак через посредника, включая подмену доменного имени и SSL-перехват. При заражении ARP-кеша составленные особым образом пакеты ARP вынуждают оба хоста действовать так, как будто они связаны друг с другом непосредственно, тогда как на самом деле они обмениваются данным через стороннего посредника, пересылающего им пакеты. Такое противозаконное применение обычных функциональных возможностей протокола ARP может делаться со злым умыслом.

Файл перехвата `arppoison.pcapng` содержит пример заражения ARP-кеша. Открыв этот файл, вы обнаружите, что перехваченный в нем сетевой трафик выглядит на первый взгляд нормально. Но если проанализировать пакеты из этого трафика, то можно заметить, что целевой хост, находящийся по адресу **`172.16.0.107`**, обращается к веб-ресурсу компании Google для выполнения поиска. В результате этого поиска появляется значительный трафик по протоколу HTTP, сочетающийся с некоторыми DNS-запросами.

Как известно, методика заражения ARP-кеша действует на втором уровне модели OSI, и если случайно бросить лишь беглый взгляд на пакеты в панели Packet List, то вряд ли удастся обнаружить какое-нибудь мошенничество. Поэтому в качестве вспомогательной меры необходимо добавить пару столбцов в панели Packet List, выполнив следующие действия.

1. Выберите команду Edit⇨Preferences из главного меню.
2. Щелкните на элементе Columns, который находится у левого края открывшегося окна Preferences.
3. Щелкните на кнопке со знаком "плюс" (+), чтобы добавить новый столбец.
4. Введите **`Source MAC`** (MAC-адрес источника) в области Title (Заголовок) и нажмите клавишу <Enter>.
5. Выберите вариант Hw src addr (resolved) (Аппаратный адрес отправителя (преобразованный)) из раскрывающегося списка Type.
6. Щелкните на вновь введенном столбце и перетащите его, расположив сразу после столбца **`Source`**.
7. Щелкните на кнопке со знаком "плюс" (+), чтобы добавить еще один новый столбец.
8. Введите **`Dest MAC`** (MAC-адрес места назначения) в области Title и нажмите клавишу <Enter>.
9. Выберите вариант Hw dest addr (resolved) (Аппаратный адрес получателя (преобразованный)) из раскрывающегося списка Type.
10. Щелкните на вновь введенном столбце и перетащите его, расположив сразу после столбца **`Destination`**.
11. Щелкните на кнопке OK, чтобы применить внесенные изменения.

Выполнив описанные выше действия, вы должны увидеть на экране то же самое, что и на рис. 12.8. Теперь в вашем распоряжении должны быть два дополнительных столбца, где отображаются MAC-адреса отправителя и получателя из анализируемых пакетов.

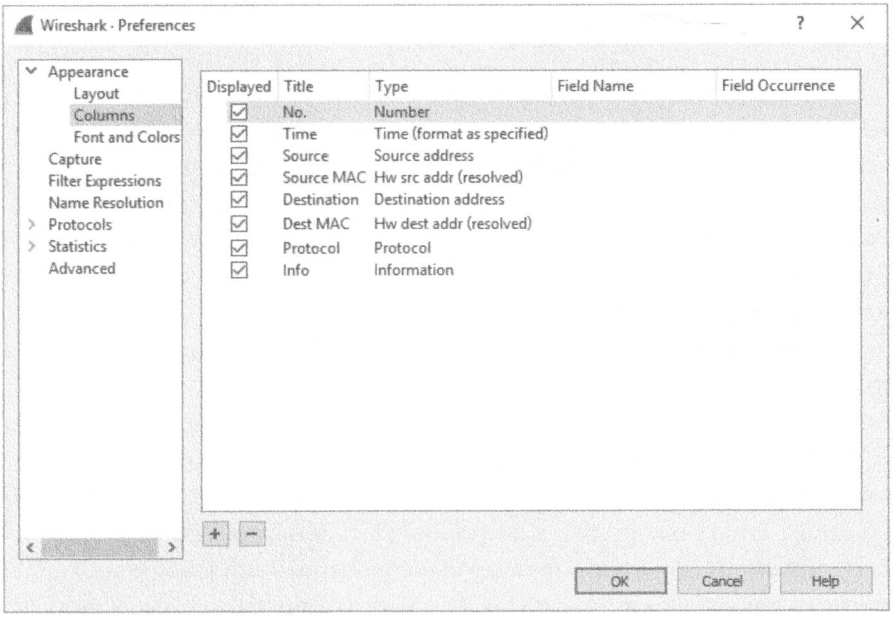


Рис. 12.8. Экран конфигурирования столбцов с двумя вновь введенными столбцами аппаратных адресов отправителя и получателя

Если у вас по-прежнему активизирован режим преобразования MAC-адресов, вы должны увидеть, что обменивающиеся данными устройства обладают MAC-адресами, обозначающими сетевое оборудование компаний Dell и Cisco. Эти сведения очень важно запомнить, поскольку, прокрутив дальше список перехваченных пакетов, можно заметить, что они изменяются в пакете 54, где происходит необычный трафик по протоколу ARP между хостом с оборудованием компании Dell (целевым хостом) и вновь внедренным хостом с оборудованием компании HP (злоумышленником), как показано на рис. 12.9.

No.	Time	Source	Source MAC	Destination	Dest MAC	Protocol	Info
54	4.171500	HewlettP_bf:91:ee	HewlettP_bf:91:ee	Dell_c0:56:f0	(1) Dell_c0:56:f0	ARP	Who has 172.16.0.107? Tell 172.16.0.1
(2) 55	0.000053	Dell_c0:56:f0	Dell_c0:56:f0	HewlettP_bf:91:ee	HewlettP_bf:91:ee	ARP	172.16.0.107 is at 00:21:70:c0:56:f0
56	0.000013	HewlettP_bf:91:ee	HewlettP_bf:91:ee	Dell_c0:56:f0	Dell_c0:56:f0	ARP	(3) 172.16.0.1 is at 00:25:b3:bf:91:ee

Рис. 12.9. Необычный трафик по протоколу ARP между сетевыми устройствами компаний Dell и HP

Прежде чем продолжить дальше, обратите внимание на конечные точки рассматриваемого здесь соединения, перечисленные в табл. 12.3.

Таблица 12.3. Конечные точки контролируемого соединения

Роль	Тип устройства	IP-адрес	MAC-адрес
Целевой хост	Dell	`172.16.0.107`	`00:21:70:c0:56:f0`
Маршрутизатор	Cisco	`172.16.0.1`	`00:26:0b:31:07:33`
Злоумышленник	HP	Неизвестен	`00:25:b3:bf:91:ee`

Но в чем необычность анализируемого здесь трафика? Как пояснялось при обсуждении протокола ARP в главе 7, "Протоколы сетевого уровня", имеются два основных типа пакетов ARP с запросами и ответами. В частности, пакет с запросом посылается как широковещательный всем хостам, чтобы обнаружить в сети машину, имеющую MAC-адрес, связанный с конкретным IP-адресом. На это искомая машина посылает запрашивающему устройству ответ в одноадресатном пакете. Принимая эти сведения во внимание, можно выяснить особенности данной последовательности обмена данными, как показано на рис. 12.9.

Прежде всего, пакет 54 содержит ARP-запрос, посылаемый злоумышленником (с MAC-адресом `00:25:b3:bf:91:ee`), в виде одноадресатного пакета непосредственно целевому хосту (с MAC-адресом `00:21:70:c0:56:f0`) ❶. Запрос данного типа должен быть широковещательным для всех хостов в сети, но анализируемый здесь запрос направляется только целевому хосту. Следует также иметь в виду, что в нем указан IP-адрес маршрутизатора, а не собственный адрес злоумышленника, несмотря на то, что данный пакет посылается этим злоумышленником с MAC-адресом его устройства, включенным в ARP-заголовок.

После данного пакета следует ответ от целевого хоста злоумышленнику, содержащему сведения о своем MAC-адресе ❷. Анализируемое здесь жульничество происходит в пакете 56, где злоумышленник посылает пакет целевому хосту, содержащий незатребованный ARP-ответ, в котором сообщается, что устройство с IP-адресом `172.16.0.1` находится по MAC-адресу `00:25:b3:bf:91:ee` ❸. Но дело в том, что IP-адресу `172.16.0.1` соответствует не MAC-адрес `00:25:b3:bf:91:ee`, а MAC-адрес `00:26:0b:31:07:33`. Об этом известно из проведенного ранее анализа пакетов, которыми маршрутизатор, расположенный по адресу `172.16.0.1`, обменивался с целевым хостом. А поскольку протокол ARP, по существу, не является защищенным и допускает незатребованные обновления ARP-таблицы, то целевой хост пошлет теперь свой трафик не маршрутизатору, как это должно быть, а злоумышленнику.

Как только целевой хост и маршрутизатор удастся ввести в заблуждение, обмен данными между ними будет происходить через хост злоумышленника, как показано на рис. 12.10.

ПРИМЕЧАНИЕ *Анализируемые здесь пакеты перехвачены из целевой машины и поэтому не дают полного представления о происходящем. Чтобы атака злоумышленника возымела действие, он должен послать ту же самую последовательность пакетов маршрутизатору с целью вынудить его принять компьютер злоумышленника за целевой хост. Но для того чтобы обнаружить эти пакеты, придется осуществить еще один перехват сетевого трафика, исходящего из маршрутизатора (или хоста злоумышленника).*

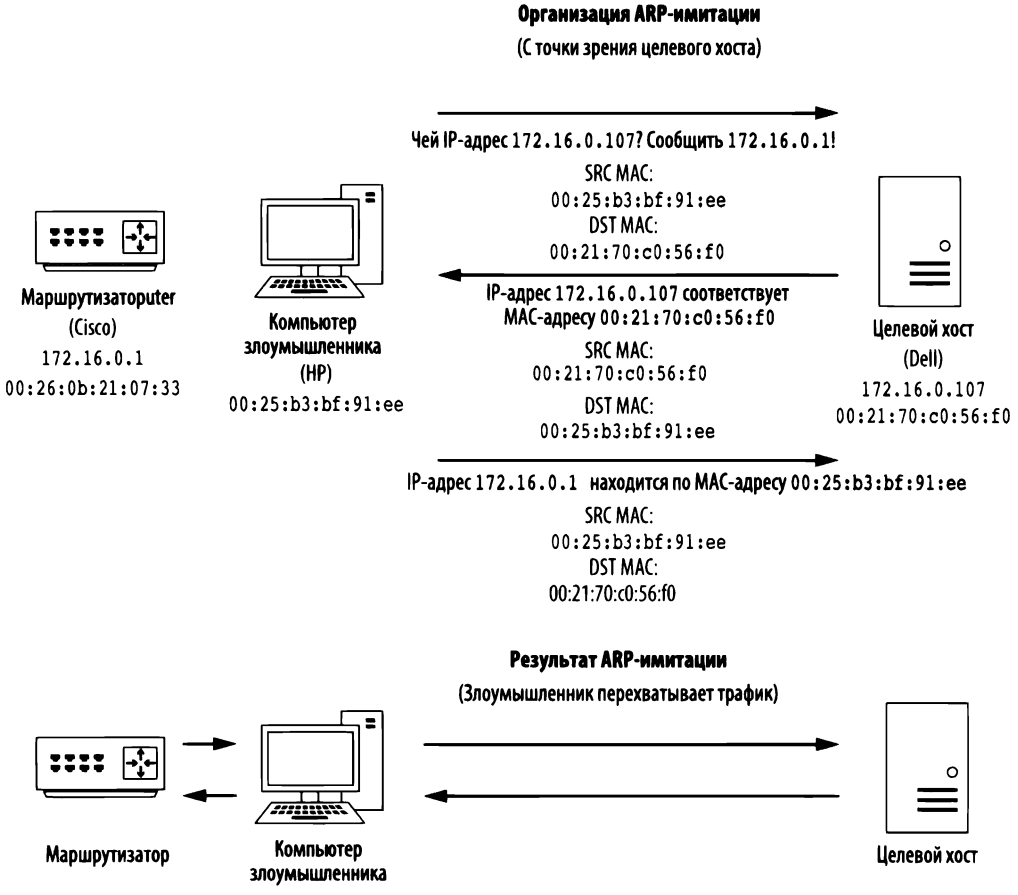


Рис. 12.10. Заражение ARP-кеша при атаке через посредника

Пакет 57 подтверждает удачный исход данной атаки. Если сравнить этот пакет с тем, что был послан прежде таинственного трафика по протоколу ARP (например, с пакетом 40; рис. 12.11), то можно заметить, что IP-адрес удаленного сервера (компании Google) остался тем же самым ❷ и ❹, тогда как MAC-адрес целевого хоста изменился ❶ и ❸. Такое изменение MAC-адреса

свидетельствует о том, что сетевой трафик теперь переадресовывается через атакующего злоумышленника, прежде чем достичь маршрутизатора.

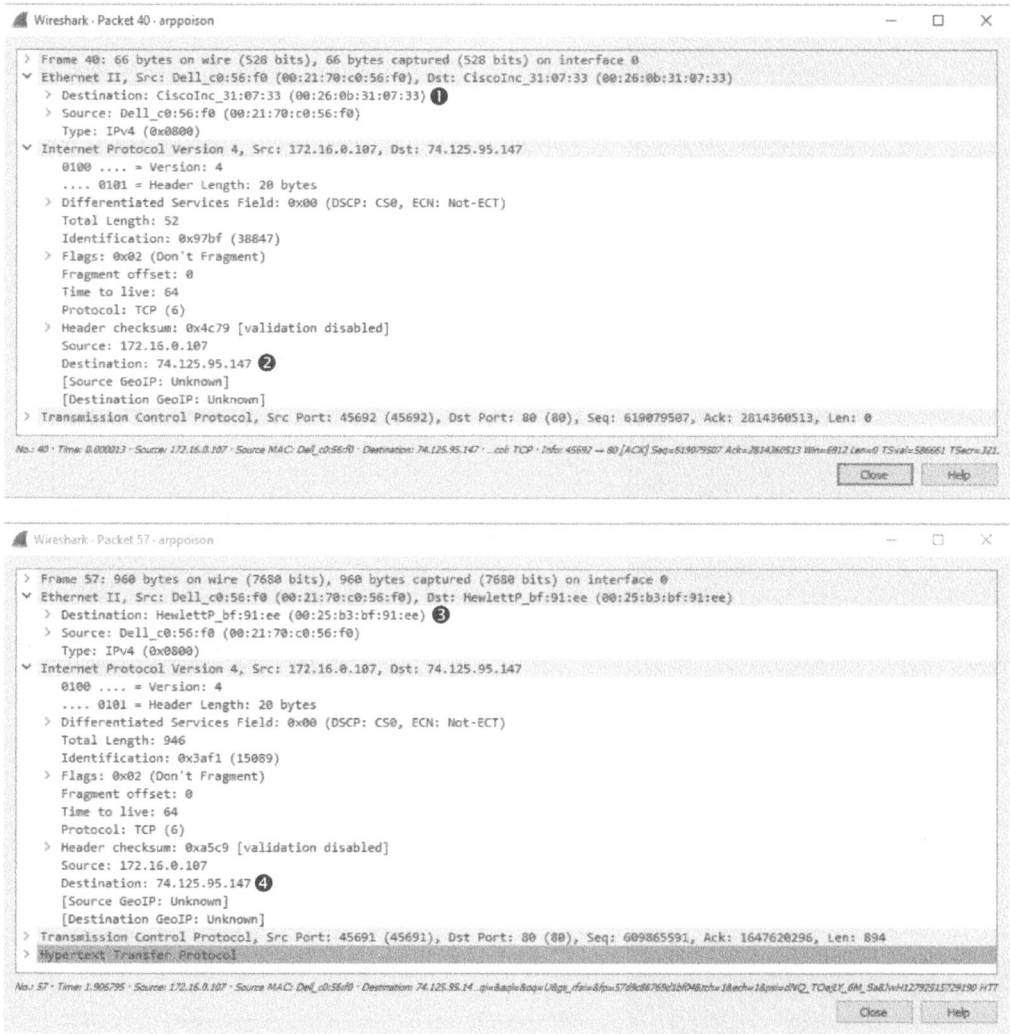


Рис. 12.11. Изменение MAC-адреса целевого хоста свидетельствует об успешном исходе данной атаки через посредника

Анализируемая здесь атака настолько незаметна, что обнаружить ее крайне сложно. Чтобы обнаружить ее, требуется помощь системы обнаружения вторжений, специально настроенной на решение подобной задачи, или программы, работающей на устройствах, предназначенных для обнаружения внезапных изменений в элементах ARP-таблицы. А поскольку вам, скорее всего, придется пользоваться методикой заражения ARP-кеша для перехвата

пакетов в анализируемых сетях, то вы должны знать, как эта методика может быть использована против вас.

Перехват сеансов связи

Файл перехвата **sessionhijacking.pcapng**

Рассмотрев порядок применения со злым умыслом заражения ARP-кеша, продемонстрируем методику *перехвата сеансов связи*, где оно выгодно применяется. При перехвате сеансов связи злоумышленник завладевает копией cookie-файла сеанса связи по протоколу HTTP, как будет показано ниже, чтобы с помощью этого файла выдать себя за другого пользователя. С этой целью злоумышленник может воспользоваться заражением ARP-кеша, чтобы перехватить сетевой трафик целевого хоста и извлечь соответствующую информацию из сеансового cookie-файла. Этой информацией злоумышленник может затем воспользоваться для доступа к веб-приложению целевого хоста как его авторизованный пользователь.

Начнем рассмотрение данного сценария с файла перехвата `sessionhijacking.pcapng`, содержащего сетевой трафик целевого хоста, находящегося по адресу **172.16.16.164** и обменивающегося данными с веб-приложением, действующим по адресу **172.16.16.181**. При этом целевой хост невольно стал жертвой злоумышленника, расположенного по адресу **172.16.16.154** и активно перехватывавшего обмен данными целевого хоста с веб-приложением. Эти пакеты были собраны с точки зрения веб-сервера, которая вероятнее всего будет точно такая же, как и у системы защиты инфраструктуры сервера от атак типа перехвата сеансов связи.

ПРИМЕЧАНИЕ *Упоминаемое здесь веб-приложение называется DVWA (Damn Vulnerable Web Application – Чертовски уязвимое веб-приложение). Оно намеренно сделано уязвимым ко многим видам атак и зачастую применяется в качестве учебного средства. Если у вас возникнет желание ознакомиться подробнее с веб-приложением DVWA, обратитесь по адресу* **http://www.dvwa.co.uk/**.

Анализируемый здесь сетевой трафик состоит, главным образом, из двух диалогов. Первый диалог представляет собой обмен данными между целевого хостом и веб-сервером и может быть селектирован фильтром **ip.addr == 172.16.16.164 && ip.addr == 172.16.16.181**. Этот обмен данными представлен обычным трафиком просмотра веб-содержимого, в котором нет ничего особенного. В анализируемых здесь запросах особый интерес вызывает cookie-значение. Так, если проанализировать запрос по методу `GET` (например, в пакете 14), то можно обнаружить, что cookie-значение

отображено на панели Packet Details (рис. 12.12). В данном случае cookie-значение **ncobrqrb7fj2a2sinddtk567q4** обозначает идентификатор сеанса связи PHPSESSID ❶.

Рис. 12.12. Просмотр сеансового cookie-значения целевого хоста

Веб-сайты пользуются cookie-файлами, чтобы поддерживать сеанс связи с отдельными хостами. Когда на веб-сайте появится новый посетитель, ему будет присвоен однозначно определяющий его идентификатор сеанса связи (PHPSESSID). Многие веб-приложения ожидают до тех пор, пока пользователь не пройдет аутентификацию со своим идентификатором сеанса связи, после чего в базе данных создается запись, где этот идентификатор распознается как обозначающий аутентифицированный сеанс связи. Любой пользователь с этим идентификатором сможет получить доступ к веб-приложению, пройдя такую аутентификацию. Безусловно, разработчикам необходимо обеспечить однозначность идентификатора, формируемого для каждого пользователя. Но такой способ обработки идентификаторов небезопасен, поскольку он дает злонамеренному пользователю возможность похитить идентификатор другого пользователя, чтобы выдать себя за него. И хотя имеются специальные методы, позволяющие предотвратить атаки типа перехвата сеанса связи, тем не менее, многие веб-сайты, включая и DVWA, остаются по-прежнему уязвимыми к подобным атакам.

Целевой хост даже не подозревает, что его сетевой трафик перехватывается злоумышленником или же у этого злоумышленника имеется доступ к сеансовому cookie-файлу (рис. 12.12). И в этом случае злоумышленнику остается лишь установить связь с веб-сервером, используя перехваченное

cookie-значение. Такую задачу можно решить с помощью определенного рода прокси-серверов, но это еще легче сделать, используя такие подключаемые к браузеру модули, как Cookie Manager для Chrome. С помощью этого подключаемого модуля злоумышленник может указать значение идентификатора `PHPSESSID`, полученное из сетевого трафика целевого хоста, как показано на рис. 12.13.

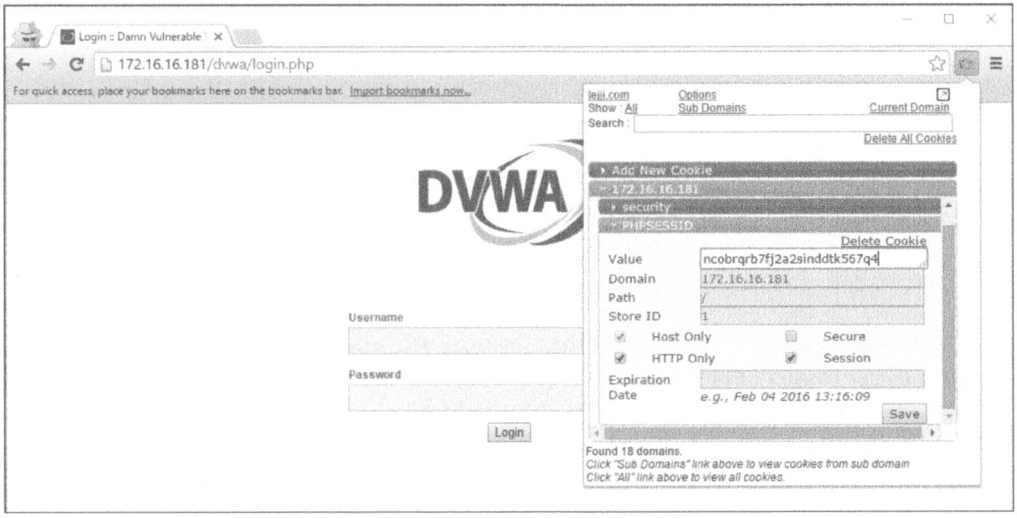


Рис. 12.13. Применение подключаемого модуля Cookie Manager злоумышленником с целью выдать себя за целевой хост

Если удалить фильтр, применявшийся ранее к файлу перехвата, и прокрутить полученный список пакетов вниз, то можно обнаружить IP-адрес компьютера злоумышленника, устанавливающегося связь с веб-сервером. Просмотр этой связи можно выявить с помощью следующего фильтра:

```
ip.addr == 172.16.16.154 && ip.addr == 172.16.16.181
```

Прежде чем углубляться в дальнейший анализ, добавьте еще один столбец на панели Packet List, чтобы отображать в нем cookie-значения. Если вы добавили столбцы в предыдущем разделе, посвященном заражению ARP-кеша, удалите их, прежде чем добавлять этот столбец. После этого выполните действия, описанные в предыдущем разделе, чтобы добавить новый столбец, исходя из имени поля **`http.cookie_pair`**, и разместите его справа от столбца **`Destination`**. В итоге содержимое экрана должно выглядеть так же, как и на рис. 12.14.

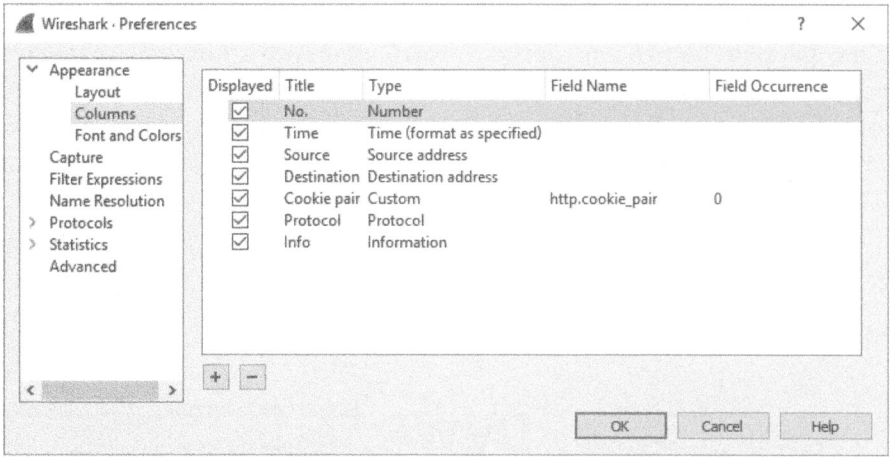


Рис. 12.14. Экран конфигурирования столбцов для исследования перехвата сеансов связи

Итак, сконфигурировав столбцы по-новому, видоизмените фильтр, чтобы отображать только HTTP-запросы, поскольку обмен данными по протоколу TCP для этой цели не подходит. Новый фильтр должен выглядеть так, как показано ниже. Пакеты, селектированные с помощью этого фильтра, приведены на рис. 12.15.

```
(ip.addr==172.16.16.154 && ip.addr==172.16.16.181)
        && (http.request.method || http.response.code)
```

No.	Time	Source	Destination	Cookie pair	Protocol	Info
77	16.563004	172.16.16.154	172.16.16.181	security=low,PHPSESSID=lup70ajeuodkrhrvbmsjtgrd71	HTTP	❶GET /dvwa/ HTTP/1.1
79	16.565584	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 302 Found❷
80	16.570187	172.16.16.154	172.16.16.181	security=low,PHPSESSID=lup70ajeuodkrhrvbmsjtgrd71	HTTP	❸GET /dvwa/login.php HTTP/1.1
81	16.575123	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html)❹
115	60.040166	172.16.16.154	172.16.16.181	security=low,PHPSESSID=ncobrqrb7fj2a2sinddtk567q4	HTTP	❺GET /dvwa/ HTTP/1.1
118	60.042241	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html❻
120	64.292056	172.16.16.154	172.16.16.181	security=low,PHPSESSID=ncobrqrb7fj2a2sinddtk567q4	HTTP	❼GET /dvwa/setup.php HTTP/1.1
122	64.293401	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html❽

Рис. 12.15. Анализ этих пакетов показывает, что злоумышленник выдает себя за пользователя целевого хоста

А теперь проанализируем обмен данными между компьютером злоумышленника и сервером. В первых четырех пакетах злоумышленник запрашивает каталог `/dvwa/` ❶ и получает в ответ код **302**, что считается обычным для веб-сервера способом переадресации посетителей по другим URL. В данном случае злоумышленник переадресовывается на страницу регистрации по адресу `/dvwa/login.php` ❷. Компьютер злоумышленника запрашивает страницу регистрации ❸, которая успешно возвращается ❹. В обоих запросах применяется идентификатор сеанса связи `lup70ajeuodkrhrvbmsjtgrd71`.

Далее следует новый запрос каталога `/dvwa/`, но на этот раз обратите внимание на другой идентификатор сеанса связи `ncobrqrb7fj2a2sinddtk567q4` ❺, который теперь совпадает с использовавшимся ранее идентификатором целевого хоста. Это означает, что злоумышленник манипулировал сетевым трафиком, чтобы воспользоваться украденным идентификатором. Вместо переадресации на страницу регистрации в ответ на отправленный запрос присылается код состояния **200** по протоколу HTTP, и страница доставляется так, как будто ее должен увидеть пользователь, аутентифицированный на целевом хосте ❻. Далее злоумышленник запрашивает другую страницу по адресу `dvwa/setup.php`, используя идентификатор целевого хоста ❼. И эта страница возвращается успешно ❽. Таким образом, злоумышленник просматривает вебсайт DVWA, как будто он действительно аутентифицирован как пользователь этого целевого хоста. И все это ему удается сделать, не зная даже ни имени пользователя, ни пароля.

Рассмотренный выше сценарий служит еще одним примером того, как злоумышленник может превратить анализ пакетов в наступательное средство. В общем, благоразумнее допустить, что если злоумышленник способен перехватывать и проанализировать пакеты, связанные с определенным обменом данными по сети, то это может привести к определенного рода зловредным действиям. И это одна из причин, по которым специалисты в области безопасности ратуют за то, чтобы защищать передаваемые данные путем шифрования.

Вредоносное программное обеспечение

Если совершенно законное программное обеспечение может быть использовано со злым умыслом, то термином *вредоносное программное обеспечение (malware)* принято обозначать код, специально написанный со злыми намерениями. Вредоносное программное обеспечение может принимать самые разные виды и формы, включая самораспространяющиеся "черви" и "троянские кони", маскирующиеся под вполне законные программы. С точки зрения защиты сетей бо́льшая часть вредоносного программного обеспечения неизвестна и не обнаруживается до тех пор, пока его зловредные действия не будут выявлены и проанализированы. Процесс такого анализа состоит из нескольких стадий, включая и ту, что специально предназначена для поведенческого анализа образцов обмена данными, совершаемого вредоносным программным обеспечением по сети. Иногда такой анализ проводится в специальной судебно-экспертной лаборатории по восстановлению алгоритма работы вредоносного программного обеспечения. Но чаще всего он проводится в естественных условиях, когда специалист по вопросам безопасности обнаруживает заражение вирусом в анализируемой им сети. В этом разделе разбирается ряд примеров настоящего вредоносного программного обеспечения и анализируется его поведение на уровне наблюдаемых пакетов.

Операция "Аврора"

Файл перехвата **aurora.pcapng**

В январе 2010 года был выявлен вредоносный код под названием операция "Аврора" (Operation Aurora), эксплуатировавший неизвестную ранее уязвимость в браузере Internet Explorer. Эта уязвимость позволила злоумышленникам получить удаленный контроль над целевыми серверами, среди прочего, компании Google.

Чтобы этот вредоносный код был выполнен, пользователю было достаточно посетить веб-сайт, используя уязвимую версию браузера Internet Explorer. И тогда злоумышленники получали немедленный доступ к машине пользователя с теми же правами, что и у зарегистрированного на ней пользователя[1]. Чтобы как-то обмануть или чем-нибудь прельстить жертву, злоумышленники применяли *целенаправленный фишинг*, состоявший в том, что они посылали по электронной почте сообщение с целью вынудить получателя щелкнуть на ссылке, ведущей к вредоносному веб-сайту.

Что же касается операции "Аврора", то мы выполнили перехват пакетов начиная с момента, когда целевой пользователь только щелкал на ссылке в сообщении целенаправленного фишинга, которое он получал по электронной почте. Полученные в итоге пакеты сохранялись в файле перехвата `aurora.pcapng`.

Анализируемый здесь перехваченный трафик начинается с трехэтапного процесса установки связи по протоколу TCP между целевого хостом, находящимся по адресу **192.168.100.206**, и компьютером злоумышленника, расположенным по адресу **192.168.100.202**. Первоначальное соединение устанавливается через порт **80**. Это может заставить поверить, что сетевой трафик осуществляется по протоколу HTTP. И такое предположение подтверждается в четвертом пакете, где содержится HTTP-запрос каталога `/info` ❶ по методу `GET`, как показано на рис. 12.16.

Как показано на рис. 12.17, компьютер злоумышленника подтверждает прием запроса по методу `GET` и посылает в ответ код состояния **302** (Moved Temporarily – Документ временно перемещен) в пакете 6 ❶. Этот код состояния обычно применяется для переадресации браузера на другую страницу, как в данном случае. Наряду с кодом состояния **302** в поле **Location** заголовка указано значение **/info?rFfWELUjLJHpP** ❷.

[1] Этот пример демонстрирует тот факт, что учетные записи обычных пользователей на целевом хосте не должны обладать правами администратора. Тогда заражение всей системы станет в принципе невозможным, поскольку для записи на системный диск и в реестр требуются права администратора. – *Примеч. ред.*

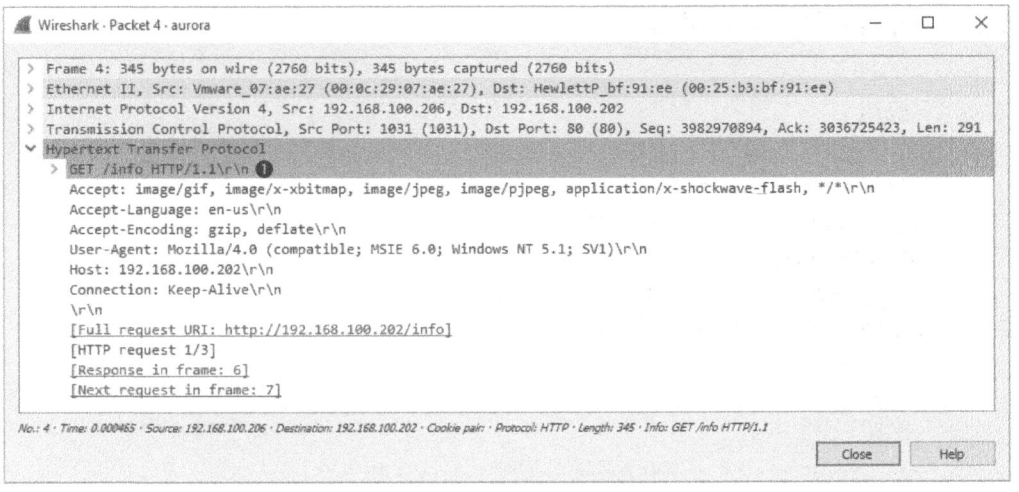


Рис. 12.16. Целевой хост делает запрос каталога **`/info`** *по методу* **`GET`**

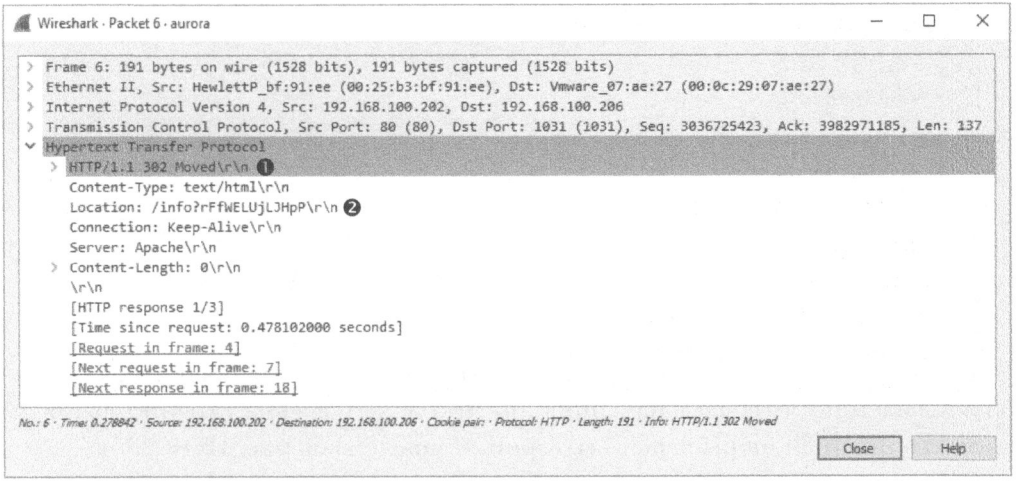


Рис. 12.17. В данном пакете браузер клиента переадресовывается на другую страницу

Получив пакет HTTP с кодом состояния **302**, клиент инициирует еще один запрос по методу `GET` по URL со значением **`/info?rFfWELUjLJHpP`** в пакете 7, на что получает подтверждение в пакете 8 типа `ACK`. После пакета `ACK` следует несколько пакетов, представляющих данные, передаваемые злоумышленником целевому хосту. Чтобы более тщательно проанализировать эти данные, щелкните правой кнопкой мыши на одном из пакетов в потоке данных (например, на пакете 9) и выберите команду Follow⇨TCP Stream (Отслеживать⇨Поток TCP) из контекстного меню. В этом потоке вывода данных можно обнаружить первоначальный запрос по методу `GET`, переадресацию по коду состояния **302**, а также второй запрос по методу `GET`, как показано на рис. 12.18.

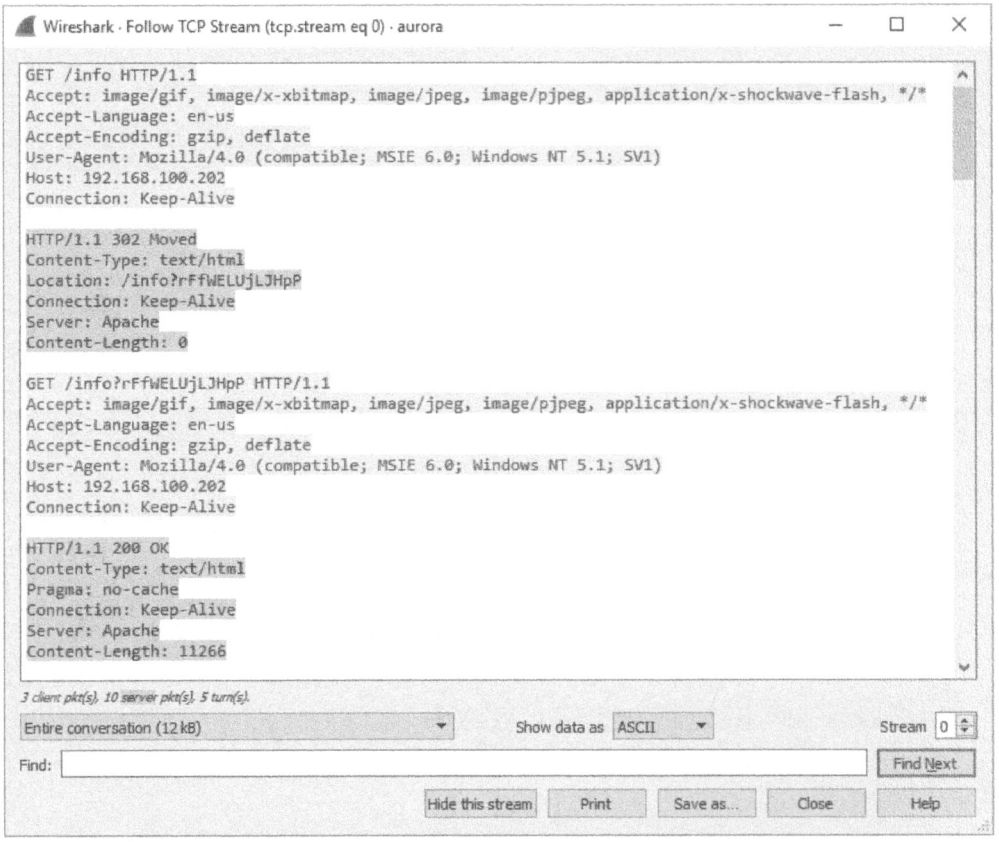


Рис. 12.18. Поток данных, передаваемый клиенту

После этого дело начинает принимать совершенно необычный оборот. Злоумышленник отвечает на запрос по методу `GET` весьма странно выглядящим содержимым, первая часть которого приведена на рис. 12.19.

Это содержимое состоит из целого ряда случайных чисел и букв в дескрипторе `<script>` ❶. Этот дескриптор применяется в HTML-документе для обозначения исходного кода на языке сценариев высокого уровня, предназначенного для выполнения на стороне HTTP-клиента. Как правило, в дескрипторе `<script>` указываются операторы, составляющие исполняемый код сценария. Но бессмысленное содержимое в данном случае зашифровано, чтобы скрыть сам исполняемый код от обнаружения. А поскольку нам известно, что анализируемый здесь трафик служит для эксплуатации уязвимостей в системе безопасности атакуемой службы, то можно предположить, что запутанный текст в дескрипторе `<script>` содержит шестнадцатеричное заполнение и код запуска оболочки со злым умыслом.

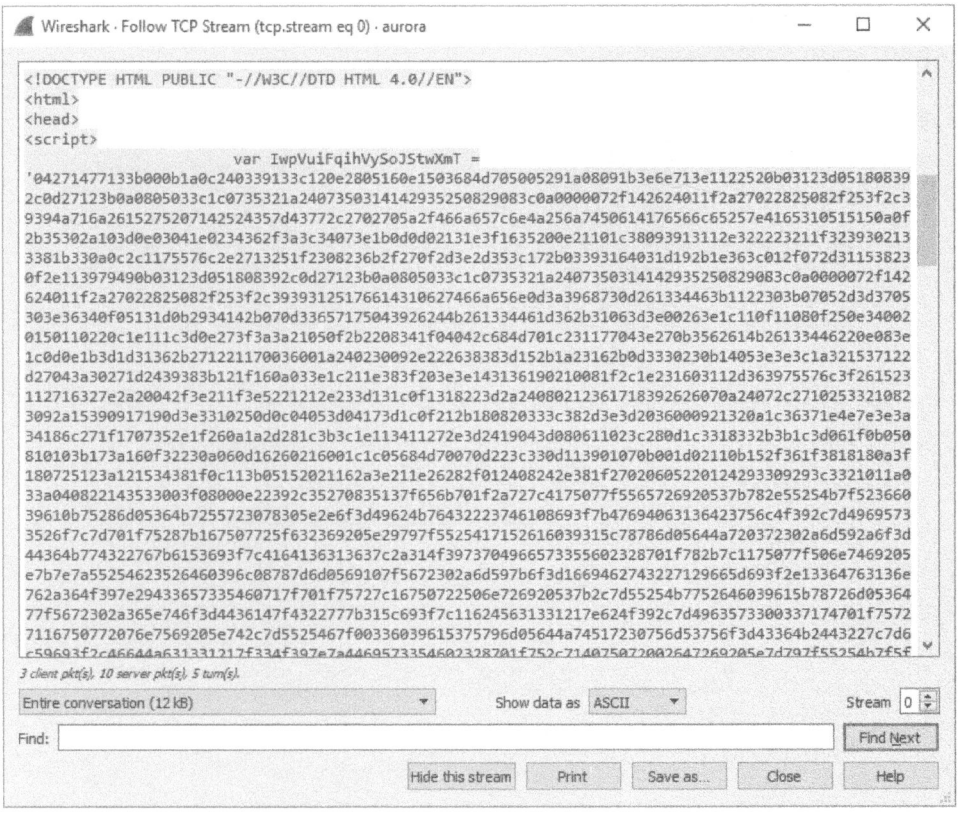


Рис. 12.19. Содержимое дескриптора ***<script>*** *оказывается зашифрованным*

ПРИМЕЧАНИЕ *Запутывание сценариев является весьма распространенной методикой, применяемой во вредоносном программном обеспечении с целью избежать обнаружения и скрыть зловредное содержимое. И хотя распутывание подобных сценариев выходит за рамки данной книги, умение делать это приобретается с опытом в ходе тщательного анализа обмена данными по сети со злым умыслом. Многие опытные специалисты по анализу вредоносного программного обеспечения способны мгновенно распознавать зловредные сценарии, бросив беглый взгляд на перехваченный сетевой трафик. Если вы желаете испытать себя, попробуйте распутать вредоносный сценарий, обнаруженный в данном примере.*

Во второй части содержимого, отправленного злоумышленником и приведенного на рис. 12.20, нам, наконец-то, удается обнаружить хотя бы немного удобочитаемого текста. Даже не имея особого опыта программирования, видно, что это исходный тест программы, где выполняется синтаксический анализ символьной строки на основании значений нескольких переменных.

И этот последний фрагмент текста расположен прямо перед закрывающим дескриптором </script>.

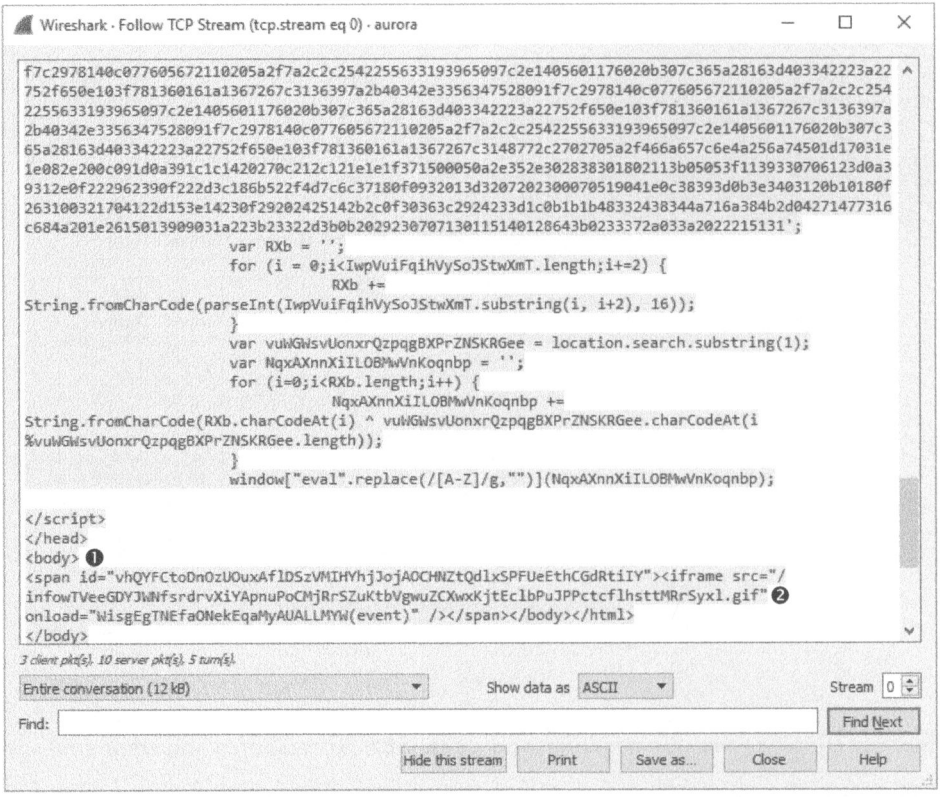


Рис. 12.20. Эта часть содержимого, отправленного сервером, содержит удобочитаемый текст и подозрительный встраиваемый фрейм `iframe`

Последний фрагмент данных, посылаемых злоумышленником целевому хосту, состоит из двух частей (см. рис. 12.20). Первая часть заключена в дескриптор `<span id="vhQYFCtoDnOzUOuxAflDSzVMIHYhjJojAOCHNZtQdlxSPFUeEthCGdRtiIY">` ❶, а вторая — в дескриптор `<iframe src="/infowTVeeGDYJWNfsrdrvXiYApnuPoCMjRrSZuKtbVgwuZCXwxKjtEclbPuJPPctcflhsttMRrSyxl.gif" onload="WisgEgTNEfaONekEqaMyAUALLMYW(event)" />` ❷. И это содержимое может свидетельствовать о злом умысле, если принять во внимание слишком длинные и произвольные строки неудобочитаемого и потенциально запутанного текста.

Часть содержимого, заключенная в дескриптор `<span>`, образует *встраиваемый фрейм*. Злоумышленники часто пользуются этим способом для встраивания дополнительного и не предполагаемого содержимого на HTML-страницу. А в дескрипторе `<iframe>` образуется *внутристрочный фрейм*, чтобы его не смог обнаружить пользователь. В данном случае в дескрипторе `<iframe>`

содержится ссылка на необычно именуемый файл формата GIF. Как показано на рис. 12.21, когда браузер целевого хоста обнаруживает ссылку на этот файл, он делает его запрос по методу `GET` в пакете 21 ❶, и в ответ немедленно посылается запрашиваемый файл формата GIF ❷.

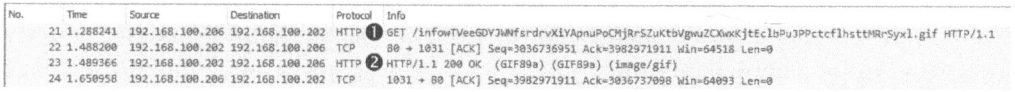


Рис. 12.21. Файл формата GIF, указанный в дескрипторе **<iframe>**, *запрашивается и загружается целевым хостом*

Самое необычное в анализируемом здесь перехваченном трафике происходит в пакете 25, когда целевой хост инициирует обратное соединение с компьютером злоумышленника через порт **4321**. Просмотр этого второго потока обмена данными на панели Packet Details мало что дает, и поэтому нам придется снова отследить поток TCP, чтобы получить более ясное представление об обмениваемых данных. Результат, выводимый в окне Follow TCP Stream, приведен на рис. 12.22.

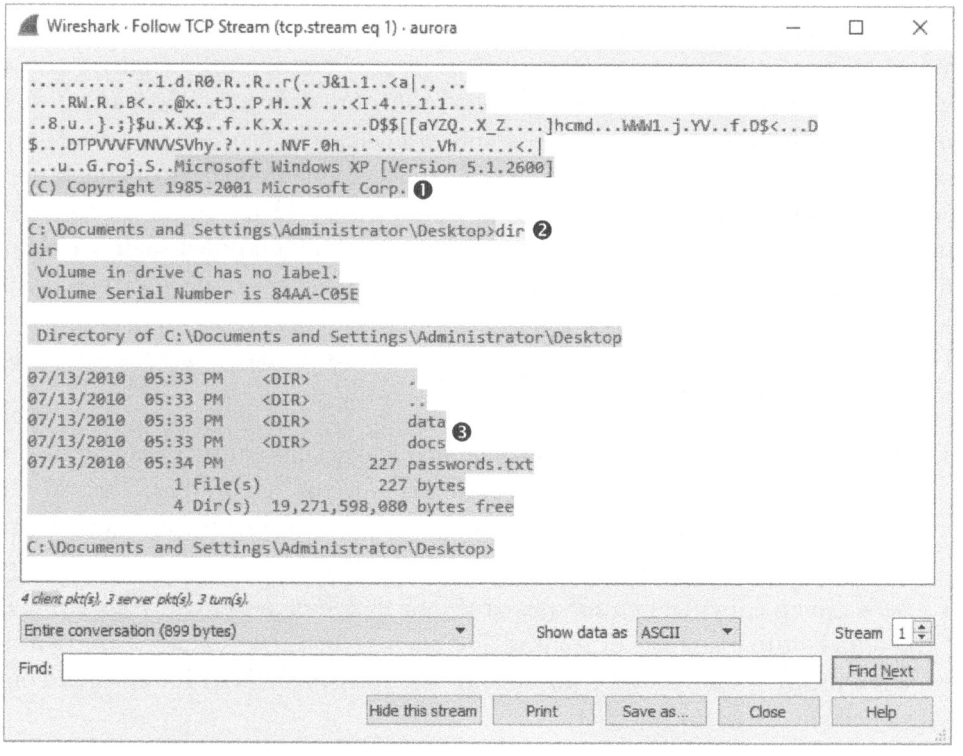


Рис. 12.22. Злоумышленник взаимодействует с командной оболочкой Windows через данное соединение

В данном окне наблюдается нечто такое, что сразу же должно вызвать тревогу: запуск команд из командной оболочки Windows ❶. Эта оболочка посылает данные выполнения команд на целевом хосте серверу, указывая на то, что попытка злоумышленника эксплуатировать уязвимость целевого хоста увенчалась успехом и полезная информация была получена злоумышленником. Целевой хост передал данные выполнения команд в командной оболочке обратно компьютеру злоумышленника сразу же после запуска вредоносного кода. В этом перехваченном трафике можно обнаружить взаимодействие злоумышленника с целевым хостом, которое заключается в выполнении команды **dir** ❷ для просмотра содержимого текущего каталога на машине целевого хоста ❸.

Если предположить, что в результате данной попытки воспользоваться уязвимостью целевого хоста злоумышленнику удалось подключиться к системному процессу, выполняющемуся с правами администратора, и запустить вредоносный код, то в результате злоумышленник может сделать с целевого хостом, все, что угодно. Достаточно один раз щелкнуть на ссылке кнопкой мыши, чтобы за доли секунды полностью отдать целевой хост во власть злоумышленника.

Подобные попытки воспользоваться уязвимостью целевого хоста обычно зашифрованы и их код трудно распознать. Поэтому при распространении по сети система обнаружения вторжений не всегда может их выявить. Следовательно, не зная заранее характер подобной попытки или образец ее кода, очень трудно без дополнительного анализа выяснить, что же на самом деле происходит в системе целевого хоста. Правда, в анализируемом здесь перехваченном трафике все же можно увидеть некоторые характерные признаки наличия вредоносного кода. К их числу относится зашифрованный и запутанный код в дескрипторах `<script>` и `<iframe>`, а также текстовый результат работы командной оболочки.

Ниже приведено краткое описание этапов при попытке воспользоваться уязвимостью в операции "Аврора".

- Целевой хост получает от атакующего злоумышленника сообщение по электронной почте, которое выглядит вполне законным, щелкает на ссылке в этом сообщении и посылает обратно запрос по методу `GET` протокола HTTP на вредоносный сайт злоумышленника.
- Веб-сервер злоумышленника в ответ выдает целевому хосту команду переадресации с кодом состояния **302**, на что браузер целевого хоста автоматически выдает запрос по методу `GET` протокола HTTP, переадресовывая его по указанному URL.
- Веб-сервер злоумышленника передает целевому хосту веб-страницу, содержащую запутанный код сценария на языке JavaScript, включающий в

себя попытку воспользоваться уязвимостью целевого хоста и встраиваемый фрейм, содержащий ссылку на запрашиваемый файл с изображением в формате GIF.

- Переданный ранее запутанный код сценария JavaScript распутывается при визуализации страницы в браузере целевого хоста, а затем выполняется на его машине с целью выявить уязвимость браузера Internet Explorer.
- Как только уязвимость будет выявлена, запустится на выполнение запутанный код, скрытый в переданной полезной информации. В результате будет открыт новый сеанс связи целевого хоста с компьютером злоумышленника через порт **4321**.
- В результате злоумышленник получает полный контроль над целевым хостом через команды, передаваемые по сети и выполняемые командной оболочкой, которая была запущена из вредоносного кода.

С точки зрения защиты сети анализируемый здесь файл перехвата может быть использован для создания сигнатуры, с помощью которой системе обнаружения вторжений удастся в дальнейшем выявлять случаи проявления подобного рода атак. С одной стороны, можно вычленить незапутанную часть перехваченного трафика, например, представленный в виде текста исходный код программы, расположенный в конце запутанного текста в дескрипторе `<script>`. А с другой стороны, можно написать сигнатуру для всего сетевого трафика по протоколу HTTP, в котором выполняется переадресация с кодом состояния **302** на веб-сайт, в URL которого присутствует слово **info**. Такая сигнатура потребует дополнительной настройки, чтобы стать полезной в условиях реальной эксплуатации, хотя ее написание и послужит неплохим началом. Не следует, однако, забывать, что сигнатуры можно побороть. Так, если злоумышленник внесет незначительные изменения в рассмотренные выше символьные строки или доставит вредоносный код через другой механизм, то созданные сигнатуры могут оказаться бесполезными. Это означает, что извечная борьба нападающих и защитников не прекратится никогда.

ПРИМЕЧАНИЕ *Умение создавать сигнатуры из образцов вредоносного трафика имеет решающее значение для тех, кто пытается защитить свою сеть от неизвестных угроз. Анализ перехваченного трафика, аналогичный описанному в этом разделе, дает отличную возможность выработать в себе навыки написания подобных сигнатур. Чтобы узнать больше об обнаружении незаконного вторжения и сигнатурах совершаемых атак, посетите веб-сайт проекта Snort по адресу* **http://www.snort.org/**.

Троянская программа удаленного доступа

Файл перехвата **ratinfected.pcapng**

До сих пор мы исследовали события, заранее зная хотя бы что-нибудь о происходящем. И хотя это отличный способ изучить атаки, на самом деле он далек от практики. Ведь в большинстве реальных сценариев те, кому поручено защищать сети, не будут анализировать каждый пакет, проходящий по сети. Вместо этого они воспользуются определенной формой системы обнаружения вторжений, которая должна предупредить их об аномалиях в сетевом трафике, который требует дальнейшего анализа на основании предопределенных сигнатур совершаемых атак.

В следующем примере мы начнем с простого предупреждения, взяв на себя роль настоящего аналитика. В данном случае система обнаружения вторжений формирует приведенное ниже предупреждение.

```
[**] [1:132456789:2] CyberEYE RAT Session Establishment [**]
[Classification: A Network Trojan was detected] [Priority: 1]
07/18-12:45:04.656854 172.16.0.111:4433 -> 172.16.0.114:6641
TCP TTL:128 TOS:0x0 ID:6526 IpLen:20 DgmLen:54 DF
***AP*** Seq: 0x53BAEB5E Ack: 0x18874922 Win: 0xFAF0 TcpLen: 20
```

На следующей стадии просматривается приведенное ниже правило сигнатуры, инициирующее данное предупреждение.

```
alert tcp any any -> $HOME_NET any
    (msg:"CyberEYE RAT Session Establishment";
content:"|41 4E 41 42 49 4C 47 49 7C|";
    classtype:trojan-activity; sid:132456789; rev:2;)
```

Это правило установлено для выдачи предупреждения всякий раз, когда обнаруживается пакет, поступающий из внутренней сети с шестнадцатеричным содержимым **41 4E 41 42 49 4C 47 49 7C**, которое преобразуется в удобочитаемую фразу **ANA BILGI**, если преобразовать эту последовательность в ASCII-код. Как только это содержимое обнаружится, инициируется предупреждение, указывающее на возможное наличие *троянской программы удаленного доступа* (Remote-Access Trojan – RAT), разработанной средствами CyberEYE. Программы RAT считаются вредоносными и незаметно выполняются на целевой машине, предоставляя злоумышленнику несанкционированный доступ к ней.

ПРИМЕЧАНИЕ *CyberEYE – это распространенное некогда инструментальное средство турецкого происхождения, предназначавшееся для создания исполняемого кода программ RAT и административного управления подвергавшихся угрозам хостов. По иронии судьбы, приведенное выше правило Snort*

инициирует предупреждение при обнаружении в пакете символьной строки **"ANA BILGI"**, *которая по-турецки означает "ОСНОВНАЯ ИНФОРМАЦИЯ".*

А теперь рассмотрим пример сетевого трафика, связанного с предупреждением из файла перехвата `ratinfected.pcapng`. Это предупреждение по правилу Snort обычно приводит к перехвату лишь одного пакета, инициировавшего данное предупреждение, но, к счастью, в нашем распоряжении имеется вся последовательность обмена данными между хостами. Чтобы выделить из анализируемого трафика самое главное, поищите шестнадцатеричную строку, упомянутую в правиле Snort, выполнив следующие действия.

1. Выберите команду Edit⇨Find Packet (Правка⇨Найти пакет) из главного меню или нажмите комбинацию клавиш <Ctrl+F>.
2. Выберите пункт Hex Value (Шестнадцатеричное значение) из раскрывающегося списка, расположенного слева от поля ввода.
3. Введите значение **41 4E 41 42 49 4C 47 49 7C** в текстовой области.
4. Щелкните на кнопке Find (Найти).

Как показано на рис. 12.23, вы должны теперь увидеть первое вхождение шестнадцатеричной строки в информационной части пакета 4 ❶.

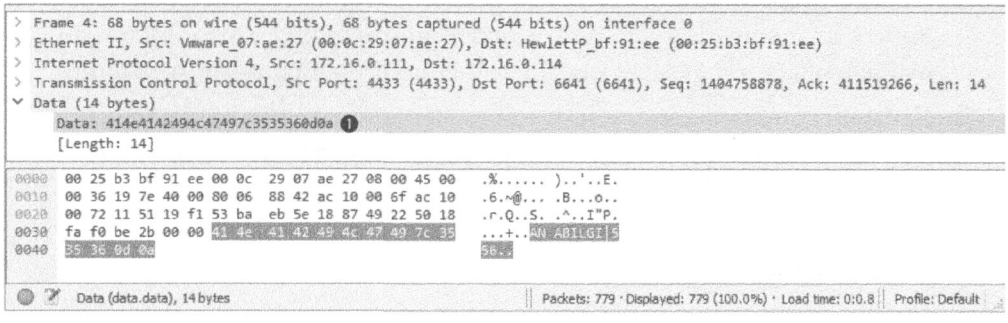


Рис. 12.23. Строка содержимого, на которую реагирует предупреждение по правилу Snort, впервые появляется в пакете 4

Если щелкнуть на кнопке Find еще несколько раз, то можно обнаружить, что данная строка присутствует также в пакетах 5, 10, 32, 156, 280, 405, 531 и 652. И хотя весь обмен данными в анализируемом здесь файле перехвата происходит между компьютером злоумышленника, находящимся по адресу **172.16.0.111**, и целевого хостом, расположенным по адресу **172.16.0.114**, вхождения этой строки оказываются и в других диалогах. В то время как обмен пакетами 4 и 5 происходит через порты **4433** и **6641**, большинство остальных вхождений рассматриваемой здесь строки обнаруживаются из обмена

пакетами между портом **4433** и произвольно выбранными временными портами. Заглянув на вкладку TCP в окне Conversations, можно убедиться в существовании нескольких диалогов, как показано на рис. 12.24.

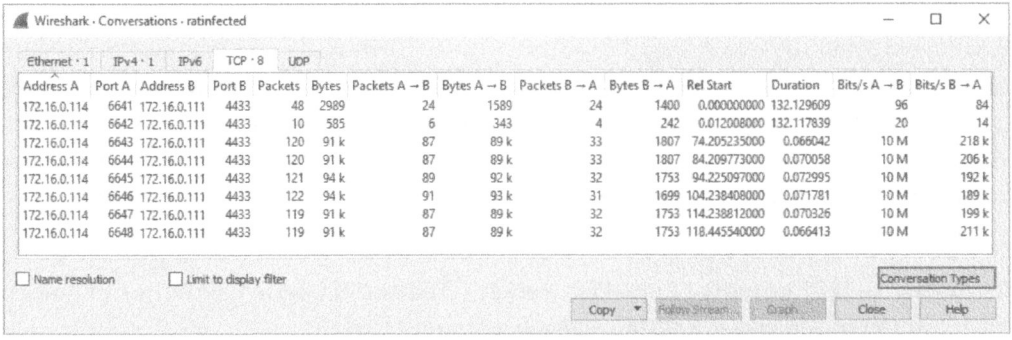


Рис. 12.24. Между компьютером злоумышленника и целевым хостом существуют три отдельных диалога

Различные диалоги можно разграничить визуально в рассматриваемом здесь файле перехвата, выделив их разным цветом. С этой целью выполните следующие действия.

1. Введите правило для фильтра **`(tcp.flags.syn == 1) && (tcp.flags.ack == 0)`** в диалоговом окне, расположенном выше панели Packet List, и нажмите клавишу <Enter>. В результате применения этого фильтра будут выбраны пакеты SYN для каждого диалога в анализируемом трафике.
2. Щелкните правой кнопкой мыши на первом пакете и выберите команду Colorize Conversation (Выделить диалог цветом) из контекстного меню.
3. Выберите сначала вариант TCP, а затем конкретный цвет.
4. Повторите данный процесс для остальных пакетов SYN, выбрав для каждого из них другой цвет.
5. По завершении щелкните на кнопке X, чтобы удалить применяемый фильтр.

Выделив диалоги разными цветами, можно удалить применяемый фильтр, чтобы посмотреть, каким образом они связаны друг с другом. Это поможет нам отследить процесс обмена данными между двумя хостами. Обмен данными (через порты **6641** и **4433**) между этими хостами начинается в первом диалоге, и поэтому начать анализ удобно именно с него. Итак, щелкните правой кнопкой мыши на любом пакете в первом диалоге и выберите команду Follow⇨TCP Stream (Отслеживать⇨Поток TCP) из контекстного меню. В итоге появится результат, приведенный на рис. 12.25.

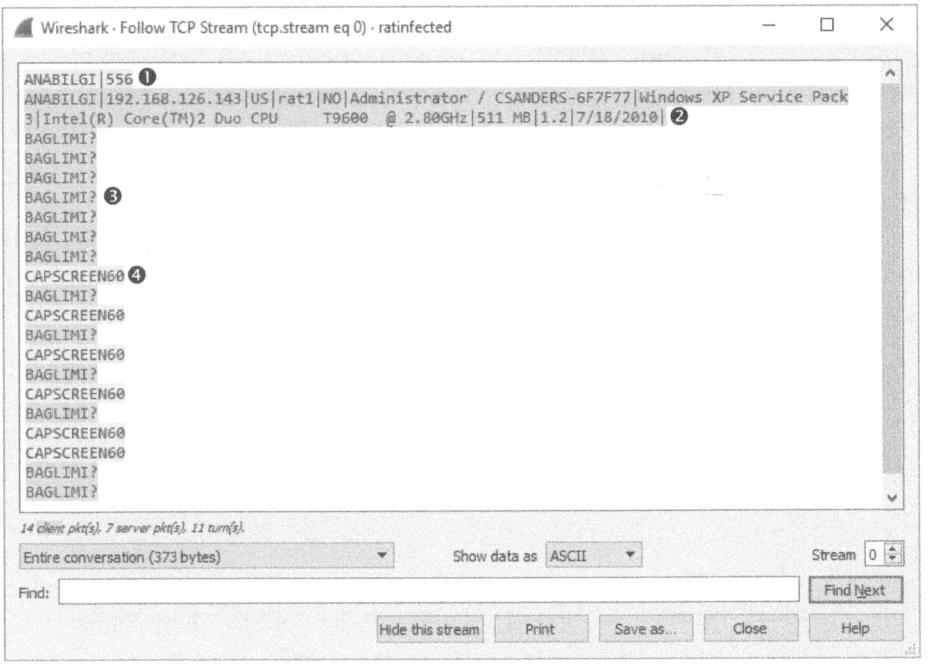


Рис. 12.25. Анализ первого диалога дает интересные результаты

И мы сразу же увидим, что текстовая строка `"ANABILGI|556"` посылается злоумышленником целевому хосту ❶. В итоге целевой хост ответит некоторым количеством основной информации о системе, включая имя компьютера (`CSANDERS-6F7F77`) и наименование применяемой операционной системы (`Windows XP Service Pack 3`) ❷, а затем начнет периодически передавать текстовую строку `"BAGLIMI?"` обратно злоумышленнику ❸. На это злоумышленник ответит лишь текстовой строкой `"CAPSCREEN60"` ❹, которая появляется шесть раз подряд.

Текстовая строка `"CAPSCREEN60"`, возвращаемая злоумышленником, представляет особый интерес, поэтому выясним, к чему она приведет. С этой целью непременно удалите любые фильтры отображения и найдите текстовую строку `"CAPSCREEN60"` в анализируемых здесь пакетах, выбрав опции String (Строка) и Packet bytes (Байты из пакета) из списков, расположенных на панели поиска, чтобы указать, где следует искать эту строку.

По окончании поиска перейдите к первому вхождению искомой строки в пакете 27. Обнаруженная здесь информация примечательна тем, что как только анализируемая здесь строка будет отправлена злоумышленником целевому хосту, последний подтвердит прием пакета, а затем начнется новый диалог в пакете 29. Теперь вам будет легче обнаружить начало нового диалога благодаря установленным ранее правилам выделения диалогов цветом.

Если теперь отследить поток вывода данных из этого нового диалога по протоколу TCP (рис. 12.26), то можно обнаружить уже знакомую текстовую строку "ANABILGI|12", следующую далее строку "SH|556" и строку "CAPSCREEN|C:\WINDOWS\jpgevhook.dat|84972" ❶. Обратите внимание на путь к файлу, указанный вслед за словом CAPSCREEN в последней строке, и следующий далее неудобочитаемый текст. Самое примечательное здесь состоит в том, что неудобочитаемый текст предваряется строкой "JFIF" ❷, которая обычно указывается в начале файлов формата JPG, как показывает быстрый поиск в Google.

Рис. 12.26. Оказывается, что злоумышленник инициирует запрос на загрузку файла изображения в формате JPG

В данный момент можно благоразумно заключить, что злоумышленник инициировал запрос на загрузку указанного файла изображения в формате JPG. Но еще важнее, что постепенно начинает проясняться структура команды, возникающая из сетевого трафика. Оказывается, что **CAPSCREEN** — это команда, посылаемая злоумышленником с целью начать передачу указанного файла изображения в формате JPG. В действительности всякий раз, когда посылается команда **CAPSCREEN**, получается одинаковый результат. Чтобы убедиться в этом, просмотрите поток TCP каждого диалога, где присутствует команда **CAPSCREEN**, или попытайтесь воспользоваться средством графического представления ввода-вывода в Wireshark, выполнив следующие действия.

1. Выберите команду Statistics⇨IO Graph из главного меню.
2. Щелкните на кнопке со знаком "плюс" (+), чтобы ввести пять строк.
3. Введите фильтры **tcp.stream eq 2**, **tcp.stream eq 3**, **tcp.stream eq 4**, **tcp.stream eq 5** и **tcp.stream eq 6** соответственно в пяти строках, вновь введенных в окне Display Filter. Присвойте каждому фильтру подходящее имя.
4. Измените на Bytes/s (Байт/с) единицы измерения, откладываемые по оси *y* при построении графика.
5. Щелкните на кнопках Graph 1, Graph 2, Graph 3, Graph 4 и Graph 5, чтобы активизировать точки данных для указанных фильтров.

Построенный в итоге график приведен на рис. 12.27.

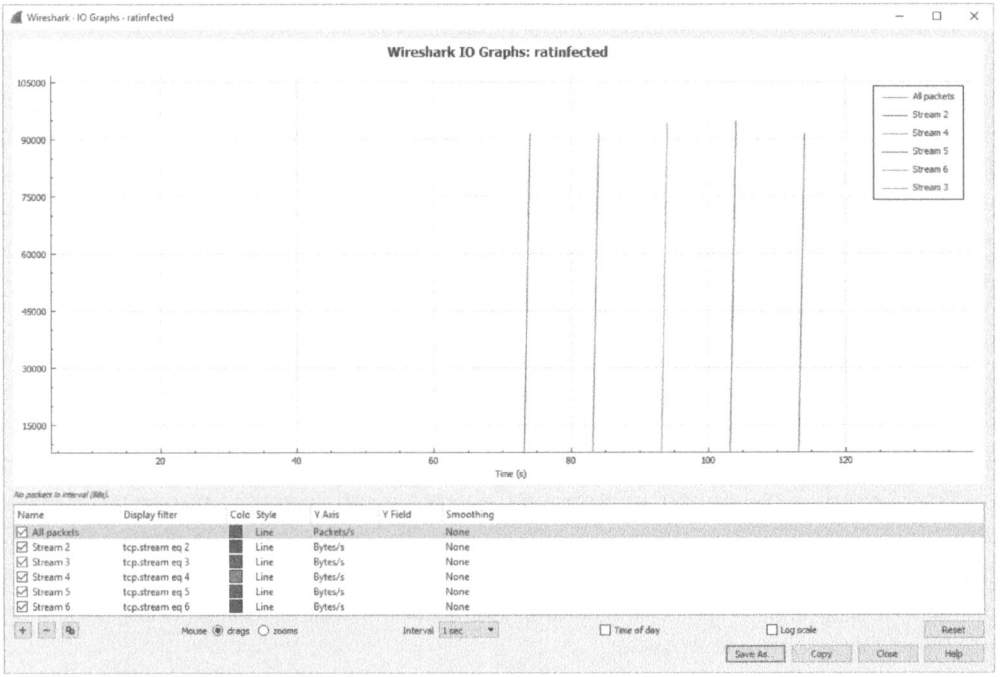


Рис. 12.27. На этом графике наглядно показано повторение одинаковых действий

Основываясь на этом графике, можно сделать вывод, что каждый диалог содержит приблизительно одинаковое количество данных и происходит в течение одного и того же периода времени. И эти действия повторяются несколько раз.

У вас, возможно, уже имеются некоторые догадки относительно содержимого передаваемого файла изображения в формате JPG, поэтому выясним, можно ли просмотреть такой файл. Чтобы извлечь данные из файла изображения в формате JPG в Wireshark, выполните следующие действия.

1. Прежде всего отследите поток TCP из подходящих пакетов, как это было сделано на рис. 12.25. Для это удобно выбрать пакет 29.
2. Обмен данными нужно изолировать, чтобы можно было увидеть только поток данных, отправленный целевым хостом злоумышленнику. С этой целью щелкните на кнопке со стрелкой раскрывающегося списка Entire Conversation (85033 bytes) (Весь диалог (85033 байта)). Выберите в нем нужное направление трафика 172.16.0.114:6643 --> 172.16.0.111:4433 (85 kB).
3. Выберите вариант RAW из раскрывающегося списка Show data as (Показать данные в формате).
4. Сохраните данные, щелкнув на кнопке Save As, чтобы сохранить файл с расширением `.jpg`.

Если теперь попытаться открыть файл изображения, то с удивлением можно обнаружить, что он не открывается. Дело в том, что для этого нужно сделать еще кое-что. В отличие от сценария, описанного в главе 10, "Основные реальные сценарии", где файл был извлечен непосредственно из трафика FTP, анализируемый здесь трафик дополнен, помимо данных, еще некоторым содержимым. В данном случае первые две строки, наблюдаемые в потоке TCP, фактически являются частью последовательности команд из вредоносной программы, а не данных, образующих изображение формата JPG (рис. 12.28). Это постороннее содержимое было сохранено вместе с выбранным потоком данных. В итоге средство просмотра файлов, искавшее заголовок файла изображения формата JPG, обнаружило содержимое, не соответствовавшее тому, что оно предполагало найти, и поэтому оно и не смогло открыть указанный файл изображения.

Устранение подобного недостатка — довольно простой процесс, требующий тем не менее умения работать шестнадцатеричном редакторе файлов. Все, что нужно сделать — вырезать нужную часть данных, составляющих изображение в формате JPG, из массива перехваченных данных. Для этого выполните следующие действия.

1. Просматривая поток TCP, приведенный на рис. 12.28, щелкните на кнопке Save as. Выберите удобное для запоминания имя файла и сохраните его в том месте, где его можно сразу же найти.
2. Загрузите шестнадцатеричный редактор файлов WinHex по адресу `https://www.x-ways.net/winhex/` и установите его.
3. Запустите редактор WinHex и откройте в нем файл, только что сохраненный в Wireshark.

Рис. 12.28. Постороннее содержимое, добавленное вредоносной программой, препятствует правильному открытию файла изображения

4. Выделите все постороннее содержимое в начале открытого файла. Это должны быть все байты, предшествующие байтам **FF D8 FF E0**, обозначающим начало нового файла изображения в формате JPG. Выделенные байты должны быть в итоге отмечены, как показано на рис. 12.29.

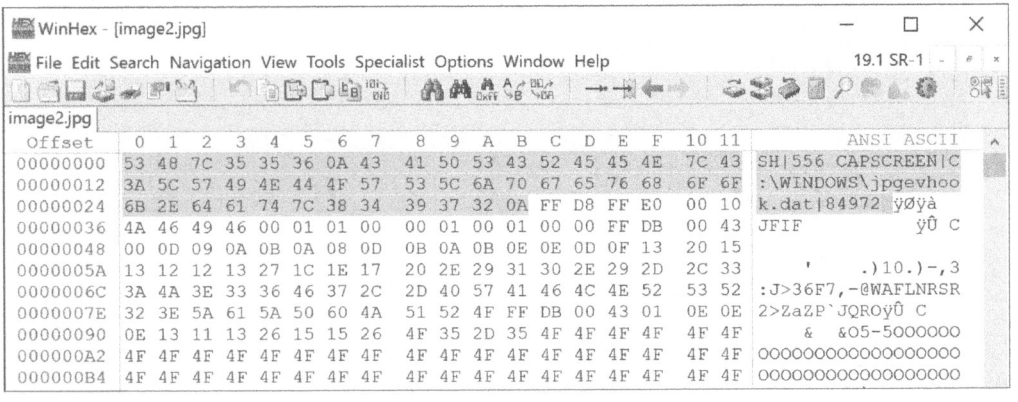


Рис. 12.29. Удаление посторонних байтов из файла изображения в формате JPG

5. Нажмите клавишу <Delete>, чтобы удалить выбранные данные.
6. Щелкните на кнопке Save главной панели инструментов редактора WinHex, чтобы сохранить внесенные изменения.

ПРИМЕЧАНИЕ *Я лично предпочитаю восстанавливать файл из перехваченного массива данных в редакторе WinHex под управлением Windows, но вы вольны воспользоваться для этой цели любым известным вам шестнадцатеричным редактором файлов.*

Итак, удалив ненужные байты данных, вы должны теперь суметь открыть рассматриваемый здесь файл. Теперь должно быть ясно, что троянская программа периодически делает копии экрана рабочего стола в операционной системе целевого хоста и передает их обратно злоумышленнику (рис. 12.30). Как только обмен данными будет завершен, далее последует процесс обычного разъединения соединения по протоколу TCP.

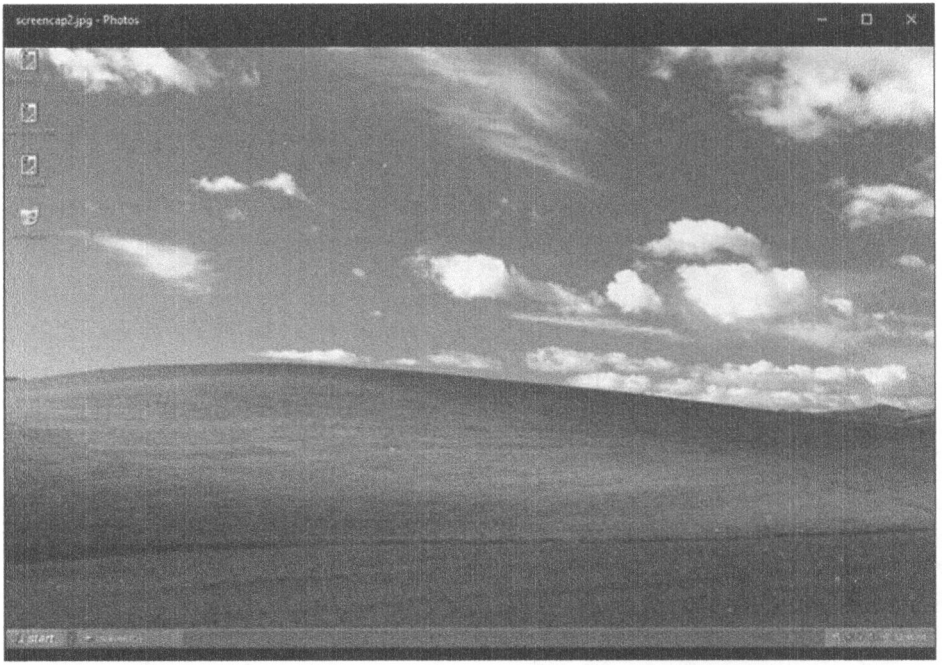


Рис. 12.30. В передаваемом файле в формате JPG содержится копия экрана рабочего стола операционной системы целевого компьютера

Рассматриваемый здесь сценарий служит характерным примером приведенного ниже хода мыслей специалиста, анализирующего сетевой трафик на основании предупреждения, полученного от системы обнаружения вторжений.

- Исследовать предупреждение и сформированную в нем сигнатуру.
- Убедиться, что сигнатура, совпавшая в анализируемом трафике, находится в надлежащем контексте.
- Исследовать трафик, чтобы выяснить, что сделал злоумышленник с поставленной под угрозу машиной.
- Начать процесс нейтрализации уязвимости, чтобы не допустить утечку секретной информации из поставленного под угрозу целевого хоста.

Набор эксплойтов и программы-вымогатели

Файлы перехвата **cryptowall4_c2.pcapng** *и* **ek_to_cryptowall4.pcapng**

И последний сценарий, рассматриваемый в этой главе, начинается с предупреждения от системы обнаружения вторжений. Сначала мы проанализируем пакеты, сформированные зараженной системой, а затем попытаемся отследить источник возникшей угрозы. В данном примере применяется настоящее вредоносное программное обеспечение, которое, вероятнее всего, является причиной заражения устройства в сети.

Рассматриваемый здесь сценарий начинается с предупреждения от системы обнаружения вторжений, которое было сгенерировано средствами Snort на консоли Sguil, как показано на рис. 12.31. Sguil — это инструментальное средство, применяемое для манипулирования, просмотра и исследования предупреждений от одного или нескольких датчиков системы обнаружения вторжений. И хотя это инструментальное средство предоставляет далеко не самый привлекательный пользовательский интерфейс, оно давно и широко применяется специалистами по анализу безопасности.

Об этом предупреждении в Sguil предоставляется немало сведений. Сводка этого предупреждения приведена на верхней панели ❶. Здесь можно увидеть момент времени, когда это предупреждение было сгенерировано, IP-адреса и порты отправителя и получателя, применяемый сетевой протокол, а также сообщение о событии, составленное на основании совпавшей сигнатуры из системы обнаружения вторжений. В данном случае дружественная локальная система, находящаяся по адресу **192.168.122.145**, связывается с неизвестной внешней системой, расположенной по адресу **184.170.149.44**, через порт **80**, который обычно связан с сетевым трафиком протокола HTTP. Внешнюю систему следует рассматривать как вредоносную, поскольку на обмен данными с ней среагировала система обнаружения вторжений. Одна из ее сигнатур свидетельствует о злонамеренном характере обмена данными, кроме того сам IP-адрес оказался никому не известен. Сигнатура, совпавшая с анализируемым здесь трафиком, представляет собой трафик регистрации (check-in),

исходящий из семейства вредоносных вирусных программ CryptoWall, если допустить, что штамм этой вирусной программы установлен на дружественной локальной системе.

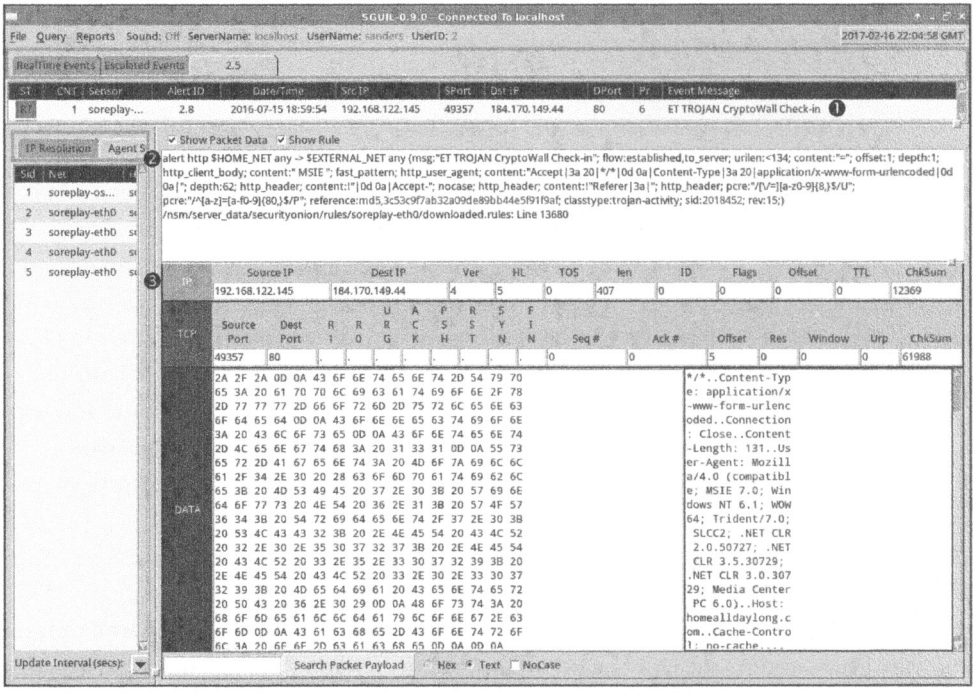


Рис. 12.31. Это предупреждение от системы обнаружения вторжений указывает на заражение вирусом CryptoWall 4

На консоли Sguil отображается синтаксис правила совпадения ❷ и данные из отдельных пакетов, где обнаружено совпадение с этим правилом ❸. Обратите внимание на то, что сведения из пакетов разделяются на заголовок протокола и информационные части аналогично представлению подобных сведений в Wireshark. Но, к сожалению, в Sguil предоставляются сведения только из одного совпавшего пакета, а нам потребуется более углубленный анализ пакетов. Поэтому исследуем далее сетевой трафик, связанный с рассматриваемым здесь предупреждением, непосредственно в Wireshark, чтобы попытаться проверить его на достоверность и выяснить, что же в нем происходит. Этот трафик содержится в файле перехвата `cryptowall4_c2.pcapng`.

Анализируемые здесь перехваченные пакеты содержат обмен данными, происходивший приблизительно в тот момент, когда возникло предупреждение, и он не особенно сложный. Первый диалог происходит в пакетах 1–16, и его можно без труда просмотреть, отследив поток TCP из этого диалога (рис. 12.32). В самом начале анализируемого здесь перехваченного трафика

локальная система устанавливает соединение по протоколу TCP с вредоносным хостом через порт **80** и выдает запрос по методу `POST`, содержащий URL адресата `http://homealldaylong.com/76N1Lm.php?x4tk7t4jo6` ❶ и небольшое количество буквенно-цифровых данных ❷. На это вредоносный хост отвечает буквенно-цифровой строкой ❹ и кодом состояния **`HTTP 200 OK`** ❸, прежде чем данное соединение будет корректно разорвано.

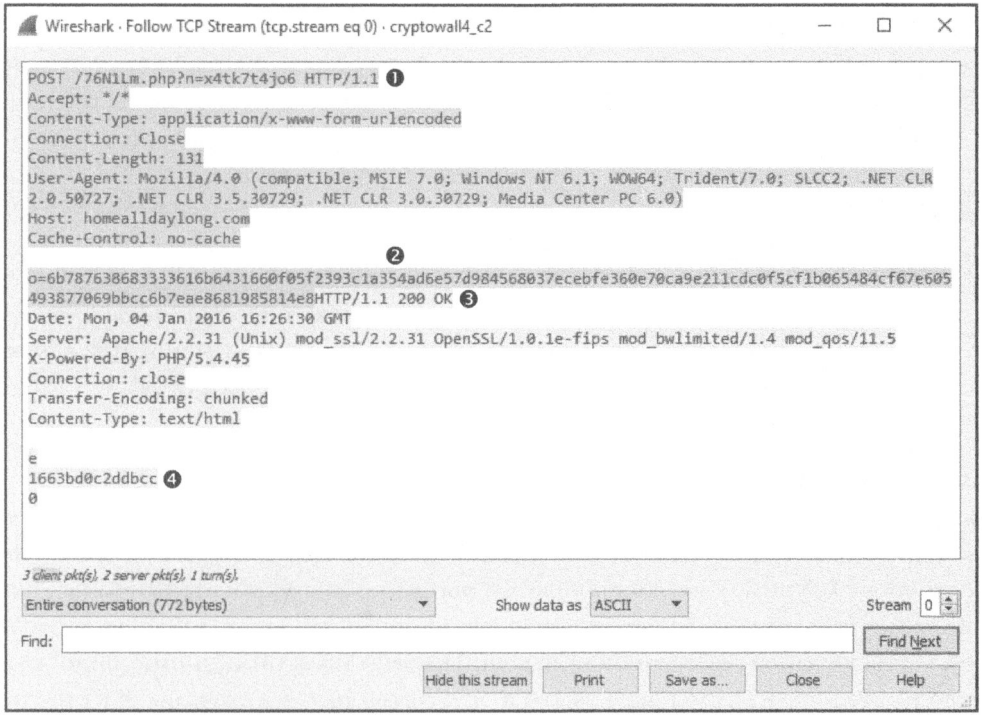


Рис. 12.32. Между этими хостами передается небольшое количество данных по протоколу HTTP

Если проанализировать остальную часть упоминаемого здесь файла перехвата, то можно заметить, что всякий раз повторяется практически одна и та же последовательность обмена данными между этими хостами, хотя и с незначительными изменениями в передаваемых данных. Чтобы посмотреть три разных соединения со сходной структурой URL, можно воспользоваться фильтром **`http.request.method == "POST"`** (рис. 12.33).

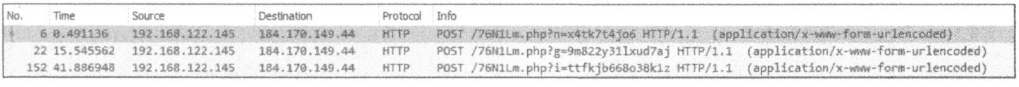


Рис. 12.33. Структура URL, демонстрирующая разные данные, передаваемые одной и той же странице

Часть **76N1Lm.php** структуры URL, относящаяся непосредственно к веб-странице, остается той же самой, тогда как остальная ее часть, относящаяся к параметрам и данным, передаваемым странице, изменяется. Повторяющаяся последовательность обмена данными в сочетании со структурой запросов согласуется с командно-управляющим поведением вредоносного программного обеспечения и сигнатурой, сгенерировавшей предупреждение. Следовательно, анализируемая здесь система, вероятнее всего, заражена вирусом CryptoWall, как и предполагается в самой сигнатуре. Можете убедиться в этом дополнительно, исследовав аналогичный пример, который можно найти на популярной странице CryptoWall Tracker, находящейся по адресу `https://www.cryptowalltracker.org/cryptowall-4.html#networktraffic`.

ПРИМЕЧАНИЕ *В этой книге недостаточно места, чтобы досконально разобрать данные, которыми обмениваются дружественная и вредоносная системы во время командно-управляющей последовательности в рассматриваемом здесь сценарии. Но если вас это заинтересует, дополнительные сведения о данном процессе вы можете получить по адресу* **`https://www.cryptowalltracker.org/communication-protocol.html`**.

Итак, убедившись, что в рассматриваемом здесь сценарии имеет место командно-управляющая последовательность обмена данными с вредоносной программой, было бы неплохо устранить этот недостаток, вылечив зараженную вирусом машину. Это особенно важно сделать, когда машина заражена вирусом вроде CryptoLocker, поскольку он пытается зашифровать пользовательские данные и предоставляет ключ расшифровки только в том случае, если пользователь заплатит солидный выкуп. Именно по этой причине подобные вредоносные программы называются *программами-вымогателями*. Рассмотрение способов излечения от подобного вируса выходит за рамки данной книги, но в реальном сценарии это следующая последовательность действий, которую должен предпринять специалист по анализу безопасности.

В связи с этим возникает следующий уточняющий вопрос: каким образом произошло заражение дружественной системы? Если источник заражения удастся определить, то могут быть обнаружены и другие устройства, зараженные аналогичным образом другими вирусами. А возможно, удастся выработать механизмы защиты или обнаружения, чтобы предупредить заражение.

В проанализированных выше пакетах командно-управляющую последовательность обмена данными удалось выявить лишь после заражения. В тех сетях, где проводится текущий контроль безопасности и постоянный перехват пакетов, многие сетевые датчики настроены на хранение данных из пакетов в течение нескольких часов или дней для судебной экспертизы. Ведь далеко не каждая организация оснащена средствами оперативного реагирования на

предупреждения. Временное хранение пакетов дает возможность проанализировать их данные со стороны дружественного хоста, прежде чем началась командно-управляющую последовательность обмена данными. И такие пакеты зафиксированы в файле перехвата `ek_to_cryptowall4.pcapng`.

Прокрутив содержимое этого файла перехвата, можно обнаружить намного больше пакетов для анализа, хотя все они относятся к протоколу HTTP. Зная принцип действия этого протокола, попробуем перейти сразу к делу, ограничив отображаемые пакеты только запросами с помощью фильтра **`http.request`**. В итоге останется лишь одиннадцать HTTP-запросов, исходящих из дружественного хоста (рис. 12.34).

No.	Time	Source	Destination	Protocol	Info
4	0.534405	192.168.122.145	113.20.11.49	HTTP	GET /index.php/services HTTP/1.1
35	5.265859	192.168.122.145	45.32.238.202	HTTP	GET /contrary/1653873/quite-someone-visitor-nonsense-tonight-sweet-await-gigantic-dance-third HTTP/1.1
39	6.109508	192.168.122.145	45.32.238.202	HTTP	GET /occasional/bXJkeHFlYXhmaA HTTP/1.1
123	9.126714	192.168.122.145	45.32.238.202	HTTP	GET /goodness/1854096/earnest-fantastic-thorough-weave-grotesque-forth-awaken-fountain HTTP/1.1
130	14.020289	192.168.122.145	45.32.238.202	HTTP	GET /observation/enVjZ2dtcnpz HTTP/1.1
441	30.245463	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?v=x4tk7t4jo6 HTTP/1.1 (application/x-www-form-urlencoded)
456	41.772768	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?n=x4tk7t4jo6 HTTP/1.1 (application/x-www-form-urlencoded)
472	45.628284	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?w=9m822y31lxud7aj HTTP/1.1 (application/x-www-form-urlencoded)
487	56.827194	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?g=9m822y31lxud7aj HTTP/1.1 (application/x-www-form-urlencoded)
619	71.971402	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?h=ttfkjb668o38k1z HTTP/1.1 (application/x-www-form-urlencoded)
634	83.168580	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?i=ttfkjb668o38k1z HTTP/1.1 (application/x-www-form-urlencoded)

Рис. 12.34. В результате фильтрации осталось лишь одиннадцать HTTP-запросов, исходящих из дружественного хоста

Первый запрос направляется дружественным хостом, находящимся по адресу **`192.168.122.145`**, неизвестному внешнему хосту, расположенному по адресу **`113.20.11.49`**. Анализ HTTP-заголовка из пакета с этим запросом показывает (рис. 12.35), что пользователь запросил страницу по адресу `http://www.sydneygroup.com.au/index.php/services/` ❶, которую он нашел в поисковой системе Bing по запросу `sydneygroup.com.au` ❷. До сих пор все вроде бы выглядит нормально.

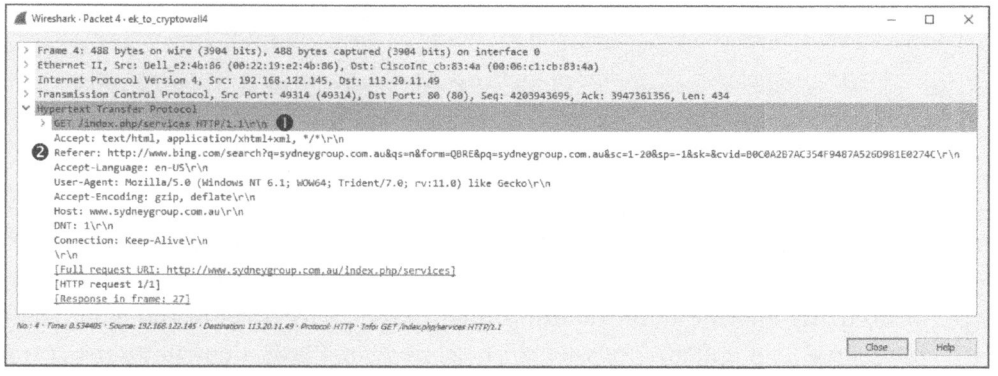


Рис. 12.35. HTTP-запрос, отправленный неизвестному внешнему хосту

Далее дружественный хост посылает в пакетах 35, 39, 123 и 130 четыре запроса на другой неизвестный внешний хост, расположенный по адресу

45.32.238.202. Как демонстрировалось в предыдущих примерах, браузеры обычно извлекают содержимое из дополнительных хостов при просмотре веб-страницы, на которой хранится встраиваемое содержимое или рекламные объявления из сторонних серверов. Само по себе это не вызывает тревогу, хотя домен в этих запросах выглядит так, словно он составлен случайным образом, что вызывает подозрение.

Любопытнее дело обстоит в запросе по методу `GET` в пакете 39. Отследив поток TCP в этом обмене данными (рис. 12.36), можно обнаружить, что в нем запрашивается файл `bXJkeHFlYXhmaA` ❶. Имя этого файла выглядит не совсем обычно и вообще не включает в себя расширение.

Рис. 12.36. Загрузка Flash-файла с не совсем обычным именем

Более тщательный анализ показывает, что веб-сервер обозначает содержимое этого файла как **`x-shockwave-flash`** ❷. Flash является распространенным подключаемым модулем для воспроизведения мультимедийных потоков в браузере, и поэтому вполне естественно обнаружить, что устройство загружает Flash-содержимое. Однако подключаемый модуль Flash печально известен своими уязвимостями, которые зачастую не устраняются. Таким образом, упомянутый выше Flash-файл успешно загружается по запросу.

После загрузки Flash-файла в пакете 130 выдается запрос на другой аналогично именуемый файл. Отследив этот поток TCP (рис. 12.37), можно

обнаружить, что в нем запрашивается файл по имени `enVjZ2dtcnpz` ❶. Тип этого файла здесь не обозначается расширением или сервером. После этого запроса целевой хост загружает фрагмент неудобочитаемых данных объемом 358400 байт ❷.

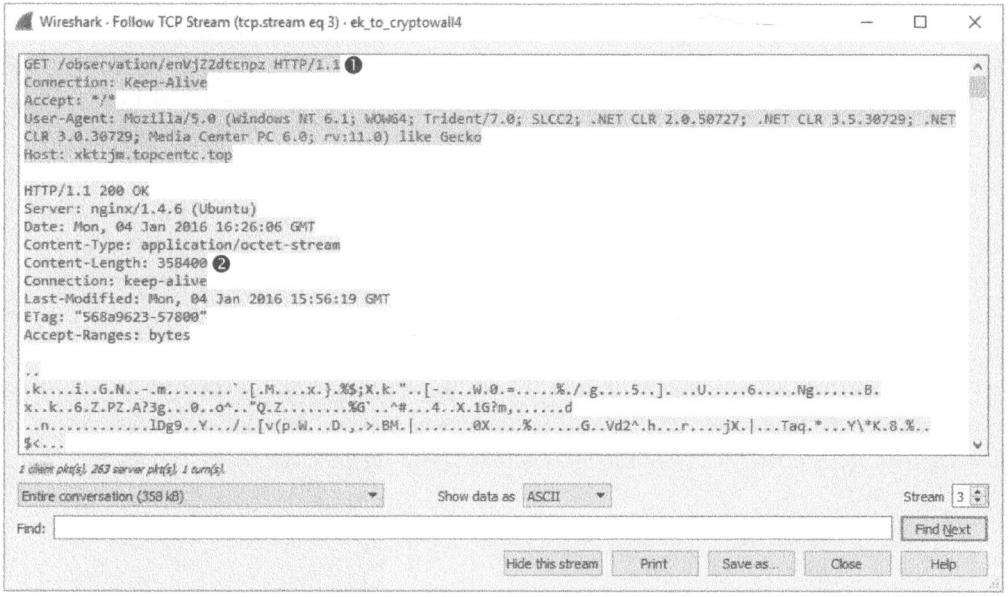


Рис. 12.37. Загрузка еще одного файла с не совсем обычным именем, но на этот раз тип этого файла не определен

Менее чем за 20 секунд после загрузки упомянутого выше файла должно появиться нечто знакомое в списке HTTP-запросов, приведенных на рис. 12.34. Начиная с пакета 441, дружественный хост начнет посылать HTTP-запросы по методу `POST` двум серверам, используя ту же самую командно-управляющую последовательность, что и рассмотренная выше. Очень похоже на то, что мы выявили источник заражения, ответственность на которое возлагается на оба загруженных файла. Первый файл, запрошенный в пакете 39, доставил средство эксплуатации уязвимостей (эксплойт), а второй файл, запрошенный в пакте 130, — вредоносную программу.

ПРИМЕЧАНИЕ *Для декодирования и анализа файлов, указанных в перехваченных пакетах, можно воспользоваться методиками анализа вредоносного программного обеспечения. Если вам интересно узнать больше о методиках восстановления алгоритмов работы вредоносного программного обеспечения, прочитаете книгу Practical Malware Analysis Майкла Сикорски (Michael Sikorski) и Эндрю Хонига (Andrew Honig), которая вышла в издательстве No Starch Press в 2012 году и которой я отдаю предпочтение.*

В рассматриваемом здесь сценарии демонстрируется одна из самых распространенных методик заражения вирусами. Она состоит в том, что, блуждая по Интернету, пользователь попадает на сайт, зараженный вредоносным переадресовывающим кодом из набора эксплойтов. Такие наборы служат для заражения вполне законных и надежных серверов и снятия отпечатков с клиентов для выявления их уязвимостей. Зараженная страница называется *посадочной страницей* эксплойта и служит для переадресации клиента на другой сайт, содержащий средство эксплуатации уязвимостей, определенное в этом наборе как наиболее эффективное для нанесения вреда целевой системе.

Проанализированные здесь пакеты перехвачены из набора эксплойтов Angler, который наиболее часто употреблялся атакующими злоумышленниками в 2015 и 2016 гг. Как только пользователь попадал на сайт, зараженный эксплойтом Angler, этот набор сразу же определял имеет ли подключаемый модуль Flash в браузере пользователя уязвимость. И если да, то тогда в эту систему сначала доставлялся Flash-файл, чтобы воспользоваться ее уязвимостью, а затем в качестве вторичной полезной информации был загружен и установлен вредоносный вирус CryptoWall на целевом хосте. Вся последовательность заражения целевой системы наглядно показана на рис. 12.38.

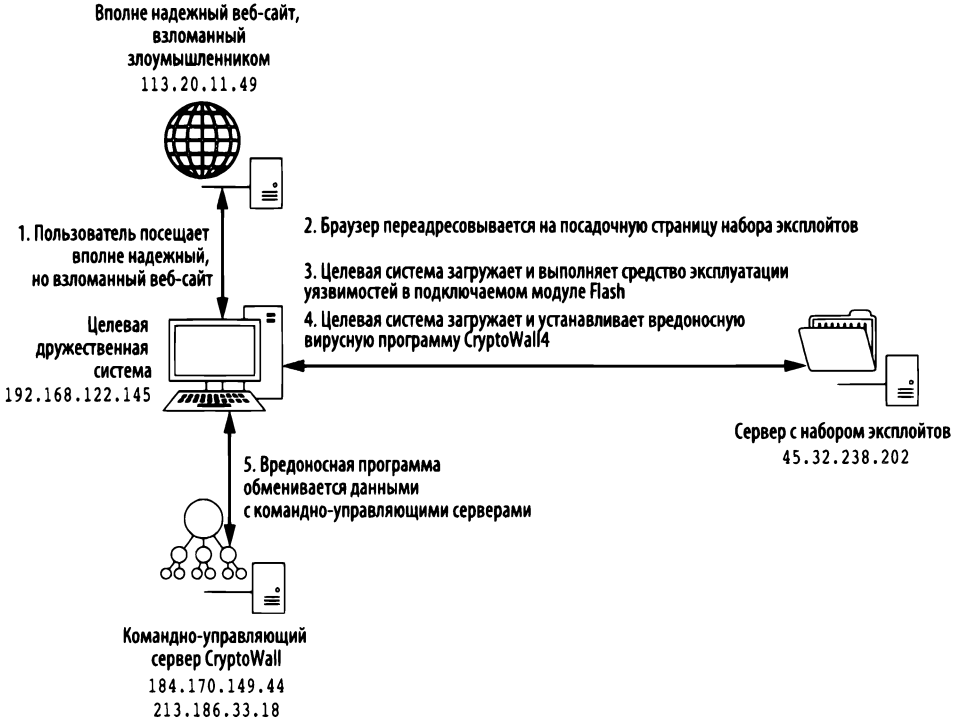


Рис. 12.38. Последовательность заражения целевой системы с помощью набора эксплойтов

Заключительные соображения

На тему анализа пакетов в сценариях, связанных с нарушением безопасности систем, распознавания распространенных видов атак и реагирования на предупреждения системы обнаружения вторжений можно написать целые книги. В этой главе были рассмотрены лишь некоторые общеупотребительные методики сканирования сетей и получения отпечатков операционных систем, особенности атаки через посредника, а также ряд примеров того, как можно эксплуатировать уязвимости системы и к каким последствиям это может привести.

13

АНАЛИЗ ПАКЕТОВ В БЕСПРОВОДНЫХ СЕТЯХ

Организация беспроводных сетей несколько отличается от организации традиционных проводных сетей. И хотя в беспроводных сетях применяются обычные сетевые протоколы, такие как TCP и IP, порядок действий несколько изменяется при переходе от самых нижних уровней модели OSI.

В данном случае особое значение приобретает канальный уровень в силу особого характера беспроводных сетей и физического уровня. Вместо простых проводных протоколов вроде Ethernet, которые не особенно изменились со временем, в данном случае приходится принимать во внимание особенности беспроводных протоколов вроде 802.11, которые развиваются очень быстро. А это накладывает новые ограничения на доступ к данным и их перехват.

Если принять во внимание упомянутые выше дополнительные соображения, то и не удивительно, что перехвату и анализу пакетов в беспроводных сетях посвящена отдельная глава этой книги. В данной главе поясняются характерные особенности анализа пакетов в беспроводных сетях и способы преодоления любых трудностей, возникающих на этом пути. Безусловно, все это будет продемонстрировано на конкретных практических примерах перехвата и анализа пакетов в беспроводных сетях.

Физические особенности беспроводных сетей

Прежде чем перейти непосредственно к перехвату и анализу данных, передаваемых по беспроводной сети, необходимо рассмотреть особенности физической среды передачи данных в подобных сетях. До сих пор мы не переходили на физический уровень потому, что обмен данными происходил по физически осязаемым проводным каналам. А в данном случае он происходит по физически невидимым каналам в эфире, где пакеты буквально пролетают мимо нас.

Анализ пакетов по отдельным каналам

Особенности перехвата сетевого трафика в беспроводной локальной вычислительной сети (WLAN — wireless local area network) состоит в том, что частотный спектр беспроводной связи распределен в среде передачи информации по отдельным физическим радиоканалам. В отличие от проводных сетей, где каждый клиент подключается к коммутатору с помощью отдельного кабеля, среда передачи данных по беспроводным сетям является общей для всех клиентов, хотя радиус ее действия ограничен. В данном случае одна WLAN занимает только часть частотного спектра по протоколу 802.11 (т.е. один радиоканал). Это дает возможность нескольким системам действовать в одной и той же физической среде передачи информации, но в разных частях частотного спектра.

ПРИМЕЧАНИЕ *Беспроводные сети организуются по стандарту 802.11, разработанному Институтом инженеров по электротехнике и электронике (Institute of Electrical and Electronics Engineers – IEEE). В этой главе терминами беспроводная сеть и WLAN обозначаются сети, отвечающие стандарту 802.11. К наиболее распространенным версиям этого стандарта относятся 802.11a, b, g и n. В каждой из этих версий предоставляется особый ряд свойств и характеристик, причем чем новее версия стандарта, тем выше скорость передачи данных. Но во всех этих версиях стандарта 802.11 используется один и тот же частотный спектр.*

Благодаря такому разделению физического пространства общий частотный спектр удалось разбить на отдельные каналы связи, где *канал* — это часть частотного спектра беспроводной связи. Так, в США для беспроводной связи выделено 11 каналов, тогда как в ряде других стран допускается большее количество. И это важно, поскольку WLAN может одновременно работать только на одном канале, а следовательно, анализировать пакеты можно только по очереди в отдельных каналах, как показано на рис. 13.1. Так, если проводится диагностика WLAN, работающей на канале 6, то систему необходимо настроить на перехват сетевого трафика именно на этом канале связи.

Рис. 13.1. Анализ пакетов в беспроводной сети затруднен тем, что он может быть проведен только по очереди в отдельных каналах

Традиционный анализ пакетов в беспроводной сети может быть одновременно проведен только в одном канале связи, если только в некоторых приложениях для сканирования беспроводных сетей не применяется методика, называемая *переключением каналов* и предназначенная для быстрой смены каналов в целях сбора данных. К числу наиболее распространенных приложений такого типа относится инструментальное средство Kismet (`http://www.kismetwireless.net/`), способное переключать до 10 каналов в секунду, и благодаря этому оно становится весьма эффективным для одновременного анализа пакетов в нескольких каналах беспроводной связи.

Перекрестные помехи в беспроводных сетях

При беспроводной связи иногда нельзя полагаться на целостность данных, передаваемых в эфире, поскольку сигналы, распространяемые по радиоканалам, могут накладываться друг на друга, создавая помехи. И хотя в беспроводных сетях приняты некоторые меры для борьбы с перекрестными помехами, они не всегда оказываются действенными. Поэтому, перехватывая пакеты в беспроводной сети, необходимо уделить особое внимание окружающей среде, чтобы в ней отсутствовали такие источники помех, как большие отражающие поверхности, крупные металлические предметы, микроволновые электроприборы, мобильные телефоны, работающие на частоте 2,4 ГГц, толстые стены или очень плотные поверхности. Они могут стать причиной потери, дублирования и неверного формирования пакетов.

Возможны также перекрестные помехи и в каналах связи. И хотя анализ пакетов допускается одновременно проводить только в одном канале, тем

не менее, это можно делать со следующей небольшой оговоркой: вследствие ограниченного диапазона частот, выделенного каналам связи, частотные спектры отдельных каналов могут незначительно перекрываться, как показано на рис. 13.2. Так, если в каналах 4 и 5 присутствует сетевой трафик и пакеты анализируются в одном из них, то вполне возможно, что часть пакетов будет невольно перехвачена из другого канала. Впрочем, подобная проблема возникает редко, поскольку при развертывании беспроводных сетей в одном районе, им, как правило, выделяются неперекрывающиеся каналы связи 1, 6 и 11. Но на всякий случай о подобном перекрытии каналов беспроводной связи необходимо знать.

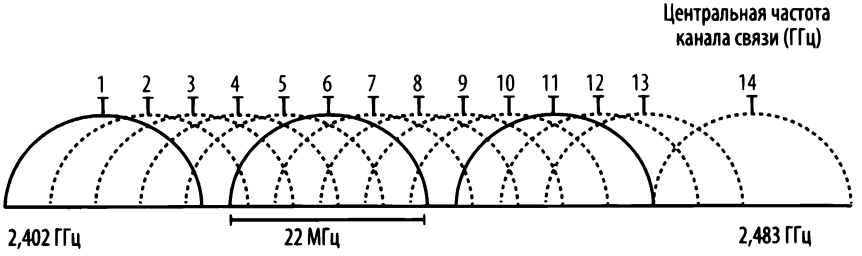


Рис. 13.2. Вследствие ограниченного диапазона частот, выделенного для работы каналов беспроводной связи, частотные спектры отдельных каналов могут незначительно перекрываться

Обнаружение и анализ наложения сигналов

Диагностику наложения сигналов в беспроводной сети нельзя провести, анализируя только пакеты в Wireshark. Если вы собираетесь специально заниматься диагностикой WLAN, вам придется регулярно выявлять факт наложения сигналов. И эта задача решается с помощью *анализатора спектра* — специального прибора для проверки и отображения взаимных помех в определенном частотном спектре.

Профессиональные анализаторы спектра могут стоить очень дорого, но для повседневной практики имеется отличное решение в виде устройства Wi-Spy, выпускаемого компанией MetaGeek. Оно подключается к компьютеру через порт USB и способно контролировать весь спектр сигналов по стандарту 802.11. А вместе с программным обеспечением inSSIDer или Chanalyzer той же самой компании MetaGeek это устройство позволяет выводить анализируемый частотный спектр в графическом виде, чтобы упростить процесс диагностики. Пример такого вывода из Chanalyzer на экран приведен на рис. 13.3.

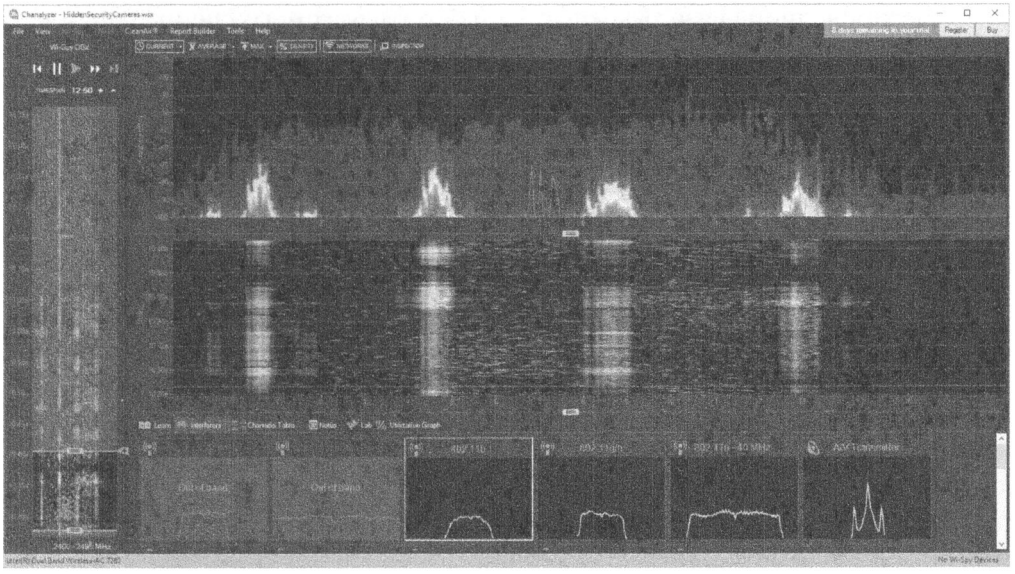


Рис. 13.3. В этом примере графического вывода наглядно показано, что спектры четырех каналов связи равномерно распределены по всему диапазону частот, выделенному для беспроводной связи

Режимы работы адаптера беспроводной связи

Прежде чем перейти, собственно, к анализу пакетов в беспроводной сети, необходимо рассмотреть различные режимы работы адаптера беспроводной связи, имеющие непосредственное отношение к перехвату пакетов. Ниже вкратце описаны четыре режима, имеющиеся для этой цели.

- **Управляемый режим (Managed mode).** Применяется в том случае, если клиент беспроводной сети подключается непосредственной к точке беспроводного доступа. В подобных случаях программный драйвер, связанный с адаптером беспроводной связи, использует точку беспроводного доступа для управления всем процессом обмена данными по беспроводной сети.
- **Режим прямого подключения (Ad hoc mode).** Применяется в том случае, если организована беспроводная сеть, в которой устройства подключаются непосредственно друг к другу. В этом режиме два клиента беспроводной сети, которым требуется обмениваться данными друг с другом, разделяют обязанности, которые обычно возлагаются на точку беспроводного доступа.
- **Ведущий режим (Master mode).** Некоторые адаптеры беспроводной связи поддерживают также ведущий режим. В этом режиме адаптеру

беспроводной связи разрешается работать вместе со специальным программным драйвером, чтобы компьютер, на котором установлен этот адаптер, действовал в качестве точки беспроводного доступа для других устройств.

- **Режим текущего контроля (Monitor mode).** Это наиболее важный режим для перехвата и анализа пакетов. Режим текущего контроля применяется в том случае, если клиенту беспроводной сети требуется остановить передачу и прием данных и вместо этого прослушивать пакеты, распространяемые в эфире. Для перехвата и анализа пакетов в Wireshark адаптер беспроводной связи вместе с программным драйвером должен поддерживать режим текущего контроля, называемый также режимом RFMON, т.е. режимом радиочастотного контроля.

Большинство пользователей применяют адаптеры беспроводной связи только в режиме прямого подключения или в управляемом режиме. Графическое представление принципа действия каждого из описанных ранее режимов работы адаптера беспроводной сети показано на рис. 13.4.

ПРИМЕЧАНИЕ *Меня часто спрашивают, какой адаптер беспроводной связи я бы рекомендовал для анализа пакетов в беспроводной сети. Я сам пользуюсь и настоятельно рекомендую адаптеры, производимые компанией ALFA Network. Их сетевое оборудование считается одним из самых лучших на рынке для перехвата всевозможных пакетов, рентабельным и переносимым. Сетевое оборудование компании ALFA Network предлагается большинством торговцев вычислительной техникой в Интернете.*

Анализ пакетов в беспроводной сети в системе Windows

Даже если в вашем распоряжении имеется адаптер беспроводной связи, поддерживающий режим текущего контроля, большинство драйверов подобных сетевых устройств в системе Windows не позволяют перейти в этот режим. Это означает, что вы сможете перехватывать только пакеты, входящие и исходящие из беспроводного интерфейса на устройстве, через которое вы подключаетесь к сети. А для перехвата пакетов между всеми устройствами в канале связи вам потребуется дополнительное оборудование.

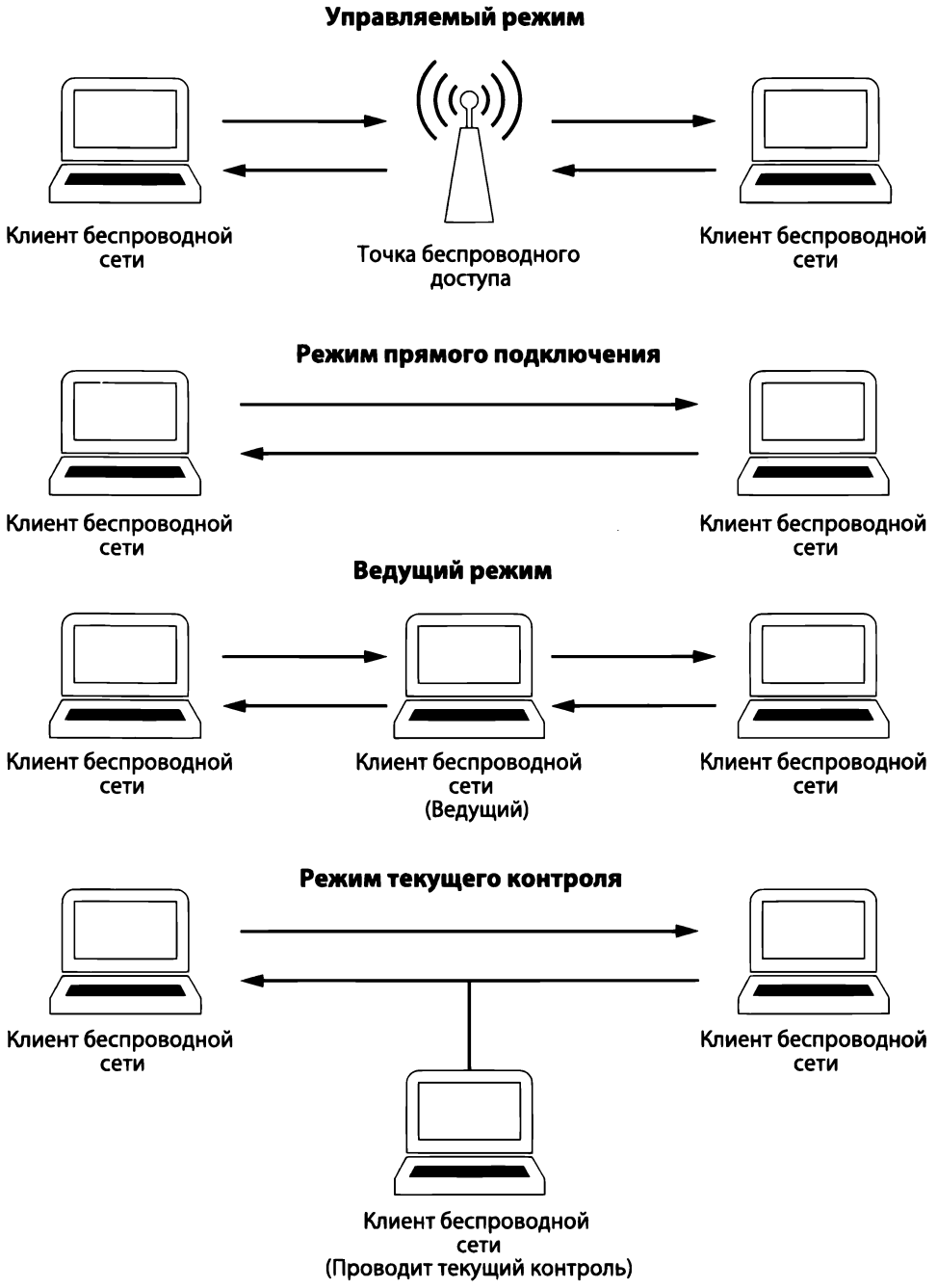


Рис. 13.4. Разные режимы работы адаптера беспроводной сети

Настройка устройства AirPcap

Устройство AirPcap от компании Riverbed Technologies (`http://www.riverbed.com/`) предназначено для преодоления ограничений, которые Windows накладывает на анализ пакетов в беспроводной сети. Оно представляет собой небольшое USB-устройство, аналогичное флеш-накопителю (рис. 13.5). Устройство AirPcap служит для перехвата беспроводного сетевого трафика из одного или нескольких выбранных каналов связи. В устройстве AirPcap применяются программный драйвер WinPcap и специальная утилита конфигурирования клиента.

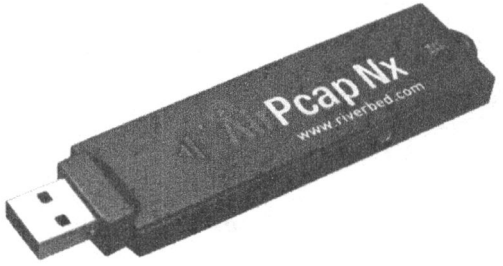


Рис. 13.5. Устройство AirPcap очень компактно и легко стыкуется с настольным компьютером

Воспользоваться утилитой конфигурирования AirPcap, приведенной на рис. 13.6, совсем не трудно. Она содержит нескольких перечисленных ниже настраиваемых параметров.

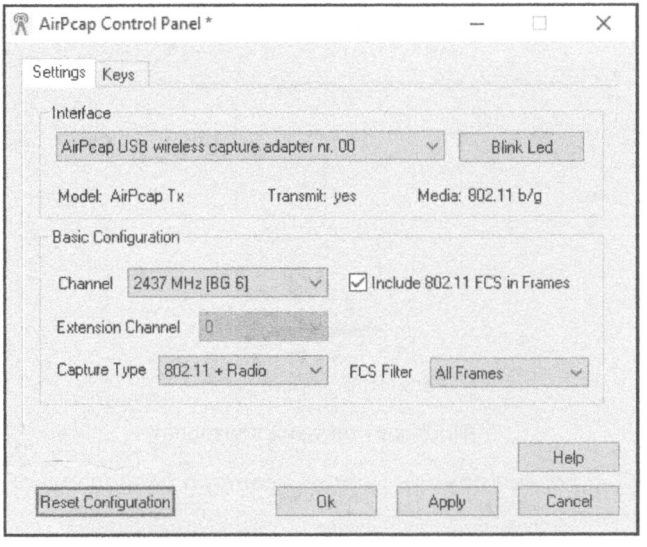


Рис. 13.6. Утилита конфигурирования AirPcap

- **Список Interface (Интерфейс).** Из этого раскрывающегося списка можно выбрать устройство для перехвата пакетов. Для некоторых сценариев сложного анализа, одновременно проводимого в нескольких каналах, может потребоваться несколько устройств AirPcap.
- **Кнопка Blink Led (Мигающий светодиод).** Если щелкнуть на этой кнопке, светодиод, светящийся на корпусе устройства AirPcap, начнет мигать. Такой режим применяется, главным образом, для идентификации конкретного сетевого адаптера, если одновременно применяется несколько устройств AirPcap.
- **Список Channel (Канал).** Из этого раскрывающегося списка можно выбрать канал связи, который требуется прослушивать через устройство AirPcap.
- **Список Extension Channel (Канал расширения).** Из этого раскрывающегося списка можно выбрать канал расширения связи, позволяющий создавать более широкополосные каналы в соответствии со стандартом 802.11n.
- **Флажок Include 802.11 FCS in Frames (Включить последовательность проверки фреймов по стандарту 802.11).** По умолчанию в некоторых системах отбрасываются последние четыре бита контрольной суммы из полученных беспроводных пакетов. Эта контрольная сумма называется *последовательностью проверки фреймов* (frame check sequence, FCS) и гарантирует целостность пакетов при передаче. Установите этот флажок, если только у вас нет веских оснований не включать контрольные суммы FCS в беспроводные пакеты.
- **Список Capture Type (Тип перехвата).** Из этого раскрывающегося списка можно выбрать следующие три варианта: 802.11 Only, 802.11 + Radio или 802.11 + PPI. Вариант 802.11 Only позволяет включать заголовок по стандарту 802.11 во все перехваченные пакеты, а вариант 802.11 + Radio — дополнительно предварять этот заголовок еще одним заголовком по стандарту Radiotap (Ответвитель радиоканала) с дополнительными сведениями о пакете, включая скорость передачи данных, уровень сигнала и уровень шума. И, наконец, вариант 802.11 + PPI позволяет добавлять заголовок PPI (Per-Packet Information — информация по каждому пакету) с дополнительными сведениями об отдельных пакетах по стандарту 802.11n.
- **Список FCS Filter (Фильтр последовательности проверки фреймов).** Из этого раскрывающегося списка можно выбрать вариант для фильтрации пакетов с нарушенной контрольной суммой FCS, даже если сброшен флажок Include 802.11 FCS in Frames. Чтобы отобразить только те

пакеты, которые получены успешно, судя по контрольной сумме FCS, выберите вариант Valid Frames (Достоверные фреймы).

- **Панель WEP Configuration (Настройка алгоритма WEP).** Эта панель доступна на вкладке Keys (Ключи) панели управления устройством AirPcap. В ней можно ввести ключи расшифровки по алгоритму WEP для анализируемых беспроводных сетей. Чтобы иметь возможность правильно интерпретировать данные, зашифрованные по алгоритму WEP, необходимо ввести подходящие ключи в данной области. Подробнее о ключах шифрования по алгоритму WEP речь пойдет далее, в разделе "Успешная аутентификация по алгоритму WEP".

Перехват сетевого трафика с помощью устройства AirPcap

Итак, установив и сконфигурировав устройство AirPcap, можно приступить к уже знакомому процессу перехвата пакетов. Чтобы начать сбор пакетов, достаточно запустить Wireshark на выполнение и выбрать интерфейс AirPcap (рис. 13.7).

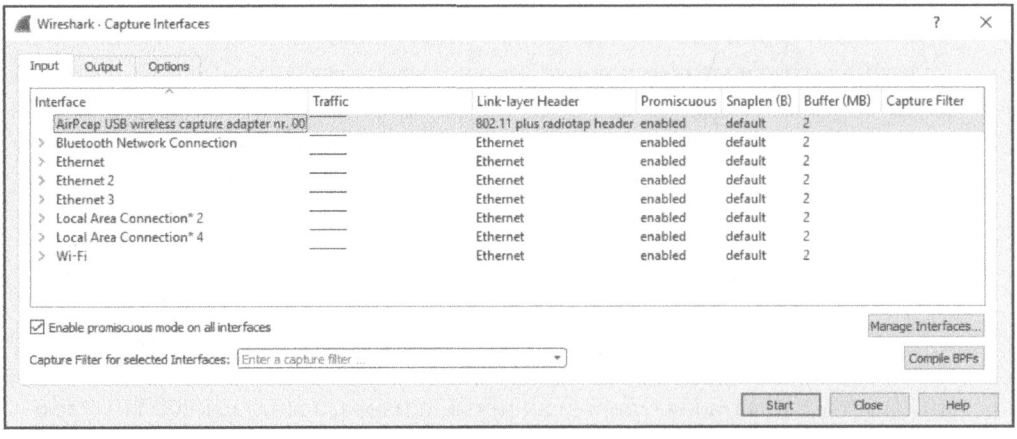


Рис. 13.7. Выбор интерфейса AirPcap для перехвата пакетов

Напомним, что пакеты будут перехватываться из того канала связи, который выбран в утилите конфигурирования устройства AirPcap. Если вы не обнаружите искомые пакеты, то, скорее всего, вы выбрали не тот канал связи. В таком случае смените канал связи, остановив активный перехват, выбрав новый канал в утилите конфигурирования устройства AirPcap и начав перехват снова. В процессе смены канала связи нельзя активно перехватывать пакеты.

Если требуется проверить, в каком именно канале происходит перехват из Wireshark, это нетрудно сделать, просмотрев статистические данные о перехвате беспроводных пакетов. С этой целью выберите команду Wireless⇨WLAN

Traffic (Беспроводная сеть⇨Трафик WLAN) из главного меню. В открывшемся окне появятся наблюдаемые устройства, а также сведения о них, включая канал связи по стандарту 802.11, как показано на рис. 13.8.

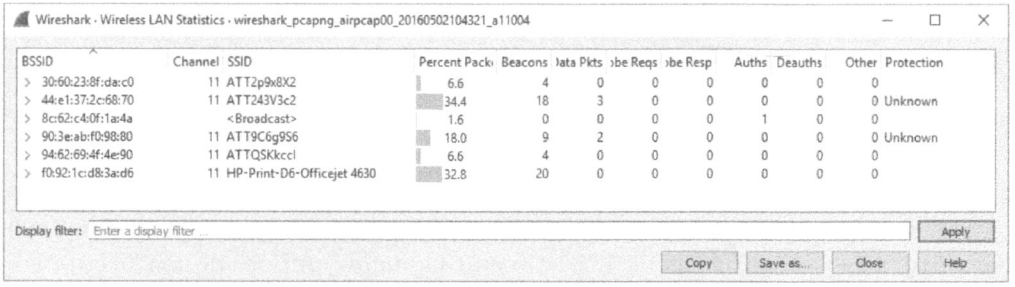


Рис. 13.8. Окно со статистическими данными о WLAN, где указано, что эти данные были перехвачены путем прослушивания канала 11

Анализ пакетов в беспроводной сети в системе Linux

Чтобы проанализировать пакеты в беспроводной сети в системе Linux, достаточно активизировать режим текущего контроля (monitor mode) в адаптере беспроводной сети и запустить Wireshark на выполнение. К сожалению, сама процедура активизации режима текущего контроля отличается в зависимости от конкретной модели адаптера беспроводной сети, и поэтому здесь нельзя дать какой-нибудь определенной рекомендации, как это следует сделать. В действительности некоторые модели адаптеров беспроводной связи не требуют активизации режима текущего контроля. В этой связи лучше всего найти нужную модель, произведя быстрый поиск в Google, и выяснить, действительно ли следует активизировать режим текущего контроля, а если требуется, то как это сделать.

К числу наиболее распространенных способов активизации режима текущего контроля в Linux относится применение встроенных беспроводных расширений, для доступа к которым служит команда **iwconfig**. Так, если ввести команду **iwconfig** на консоли, то в конечном итоге будет получен следующий результат:

```
$ iwconfig
❶ Eth0 no wireless extensions
  Lo0 no wireless extensions
❷ Eth1 IEEE 802.11g ESSID: "Tesla Wireless Network"
       Mode: Managed Frequency: 2.462 GHz
       Access Point: 00:02:2D:8B:70:2E
```

```
Bit Rate: 54 Mb/s Tx-Power-20 dBm Sensitivity=8/0
Retry Limit: 7 RTS thr: off Fragment thr: off
Power Management: off
Link Quality=75/100 Signal level=-71
dBm Noise level=-86 dBm
Rx invalid nwid: 0 Rx invalid crypt:
0 Rx invalid frag: 0
Tx excessive retries: 0 Invalid misc:
0 Missed beacon: 2
```

Как следует из приведенного выше результата выполнения команды **iwconfig**, сетевой интерфейс `Eth1` можно сконфигурировать для беспроводной связи. И это вполне очевидно, поскольку в данном случае отображаются данные для протокола 802.11g ❷, тогда как сетевые интерфейсы `Eth0` и `Lo0` возвращают сообщение "`no wireless extensions`" (беспроводные расширения отсутствуют) ❶.

Наряду со всеми сведениями о беспроводной сети, предоставляемыми по команде **iwconfig**, включая установленный идентификатор расширенной беспроводной службы (Extended Service Set ID, ESSID) и несущую частоту, обратите внимание и на то, что во второй строке под заголовком `Eth1` в качестве текущего указан управляемый режим (`Mode: Managed`). Именно этот режим и требуется изменить.

Чтобы установить в сетевом интерфейсе `Eth1` режим текущего контроля, необходимо войти в систему с правами доступа суперпользователя (`root`) непосредственно или по команде смены пользователя (**su**), как показано ниже.

```
$ su
Password: <enter root password here>
```

Войдя в систему с правами суперпользователя, можно ввести команды для настройки параметров беспроводного сетевого интерфейса. В частности, чтобы настроить сетевой интерфейс `Eth1` на режим текущего контроля, достаточно ввести следующую команду:

```
# iwconfig eth1 mode monitor
```

Если снова запустить команду **iwconfig** после установки режима текущего контроля в адаптере беспроводной связи, то в выводимых по этой команде результатах будут отражены изменения, внесенные в сетевой интерфейс `Eth1`. Чтобы гарантировать работоспособность сетевого интерфейса `Eth1`, достаточно ввести следующую команду:

```
# iwconfig eth1 up
```

По команде **iwconfig** можно также сменить канал связи, выбранный для прослушивания. В частности, чтобы выбрать канал 3 в сетевом интерфейсе `Eth1`, следует ввести команду

```
# iwconfig eth1 channel 3
```

ПРИМЕЧАНИЕ *Каналы связи можно сменять по ходу перехвата пакетов, поэтому делайте это не колеблясь по своему усмотрению. Чтобы упростить этот процесс, можно создать отдельный сценарий на основе команды* **iwconfig**.

Завершив все описанные выше настройки, запустите Wireshark на выполнение, чтобы приступить к перехвату пакетов.

Структура пакета по стандарту 802.11

Файл перехвата `80211beacon.pcapng`

Главное отличие пакетов в беспроводной сети от пакетов в проводной сети заключается в наличии дополнительного заголовка по стандарту 802.11. Этот заголовок второго уровня содержит дополнительные сведения о пакете и среде, в которой он передается. Имеются следующие типы пакетов.

- **Пакеты управления (Management).** Такие пакеты служат для установки связи между хостами на втором уровне. К наиболее важным подтипам пакетов управления относятся пакеты аутентификации, установки связи и сигнальные пакеты.
- **Пакеты контроля (Control).** Такие пакеты обеспечивают гарантированную доставку пакетов управления и данных и предназначены для управления процессом устранения перегрузки. К наиболее распространенным подтипам пакетов контроля относятся пакеты с запросами на передачу и подтверждением готовности к приему.
- **Пакеты данных (Data).** Такие пакеты содержат конкретные данные и относятся к тем типам пакетов, которые могут быть направлены из беспроводной сети в проводную сеть.

Тип или подтип беспроводного пакета определяет его структуру, поэтому имеется большое количество возможных структур. Рассмотрим одну из таких структур на примере анализа единственного пакета из файла перехвата `80211beacon.pcapng`. Этот файл содержит пример пакета управления, называемого *сигнальным*, как показано на рис. 13.9.

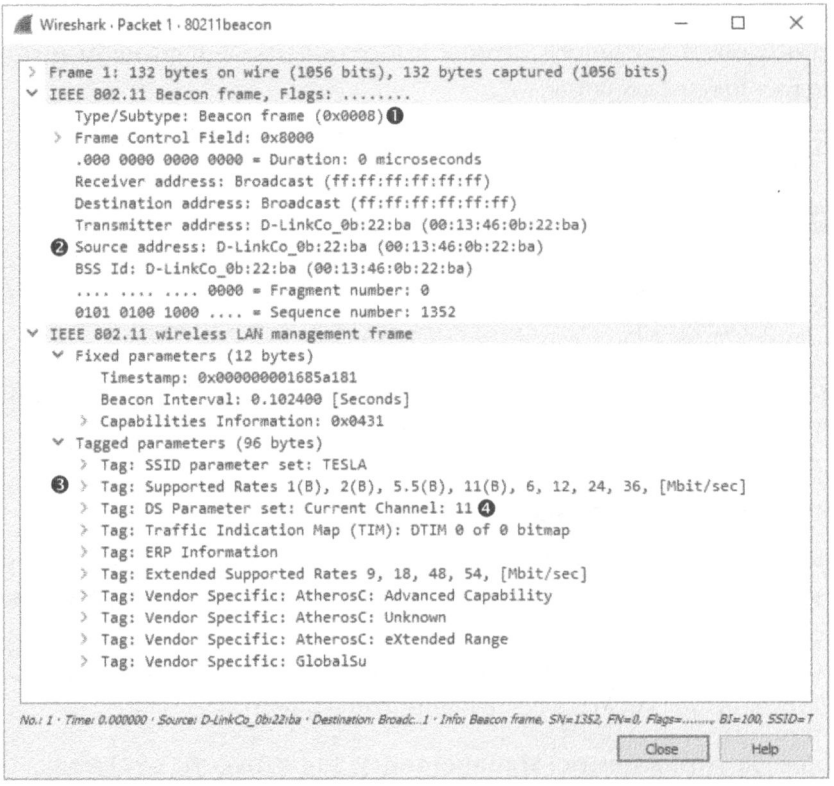


Рис. 13.9. Это сигнальный пакет по стандарту 802.11

Сигнальные пакеты относятся к самым информативным беспроводным пакетам, которые только можно обнаружить. Они посылаются в широковещательном режиме по беспроводному каналу из точки беспроводного доступа, чтобы уведомить всех подключенных клиентов о некотором событии, а также определить параметры, которые должны быть заданы для подключения к точке доступа. Как следует из данного файла перехвата, анализируемый пакет определяется как сигнальный в поле **`Type/Subtype`** в заголовке по стандарту 802.11 ❶.

Значительную часть дополнительной информации можно обнаружить в заголовке фрейма управления по стандарту 802.11, включая следующее.

- **`Timestamp` (Отметка времени).** Обозначает момент времени, когда пакет был передан.
- **`Beacon Interval` (Сигнальный интервал).** Обозначает интервал повторной передачи сигнального пакета.
- **`Capabilities Information` (Сведения о возможностях).** Обозначает сведения об аппаратных возможностях точки беспроводного доступа.

- **`SSID parameter set` (Набор параметров SSID).** Обозначает идентификатор беспроводной службы (SSID), используемый в точке беспроводного доступа для передачи пакетов.
- **`Supported Rates` (Поддерживаемые скорости передачи).** Обозначает скорости передачи данных, которые поддерживает точка беспроводного доступа.
- **`DS Parameter set` (Набор параметров DS).** Обозначает номер канала связи, на котором работает точка беспроводного доступа.

В рассматриваемый здесь заголовок включаются также адреса отправителя и получателя, а также сведения о производителе оборудования. Исходя из этого, можно выяснить немало любопытного о точке беспроводного доступа, передающей сигнальный пакет в рассматриваемом здесь файле перехвата. Очевидно, что в данном случае стандарт 802.11b (B) ❸ применяется в сетевом устройстве D-Link ❷ для передачи данных по каналу 11 ❹. И хотя конкретное содержимое и назначение пакетов управления по стандарту 802.11 изменяется, общая структура остается той же самой и соответствует рассмотренной в данном примере.

Добавление столбцов, характерных для беспроводной сети, на панель Packet List

В примерах анализа пакетов из предыдущих глав выгодно использовался гибкий пользовательский интерфейс Wireshark для добавления столбцов, подходящих для конкретной ситуации. Прежде чем продолжить дальше анализ пакетов в беспроводной сети, полезно добавить следующие три столбца на панель Packet List.

- Столбец **`Channel`** для отображения канала, в котором перехвачен пакет.
- Столбец **`Signal Strength`** для отображения мощности сигнала в канале в момент перехвата пакета, измеряемый в децибелах по отношению к 1 мВт (dBm).
- Столбец **`Data Rate`** для отображения скорости передачи данных в момент перехвата пакета.

Сведения в этих столбцах могут оказать немалую помощь при диагностике беспроводных соединений. Например, несмотря на то, что программное обеспечение беспроводного клиента сообщает, что мощность сигнала находится в норме, обратившись за справкой к этим столбцам при анализе перехваченных пакетов, можно выявить числовое значение, не отвечающее установленным требованиям к мощности сигнала.

Чтобы добавить эти столбцы на панель Packet List, выполните следующие действия.

1. Выберите команду Edit⇨Preferences из главного меню.
2. Перейдите к разделу Appearance⇨Columns и щелкните на кнопке со знаком "плюс" (+).
3. Введите **Channel** в поле Title, выберите вариант Custom (Специальный) из раскрывающегося списка Type и выберите фильтр **wlan_radio.channel** из списка Field Name.
4. Повторите этот процесс для добавления столбцов **Signal Strength** и **Data Rate**. Присвойте им подходящие заголовки и выберите, соответственно, фильтры **wlan_radio.signal_dbm** и **wlan_radio.data_rate** из раскрывающегося списка Field Name. На рис. 13.10 показано, как должно выглядеть окно Preferences после добавления трех упомянутых выше столбцов.
5. Щелкните на кнопке OK, чтобы сохранить внесенные изменения.

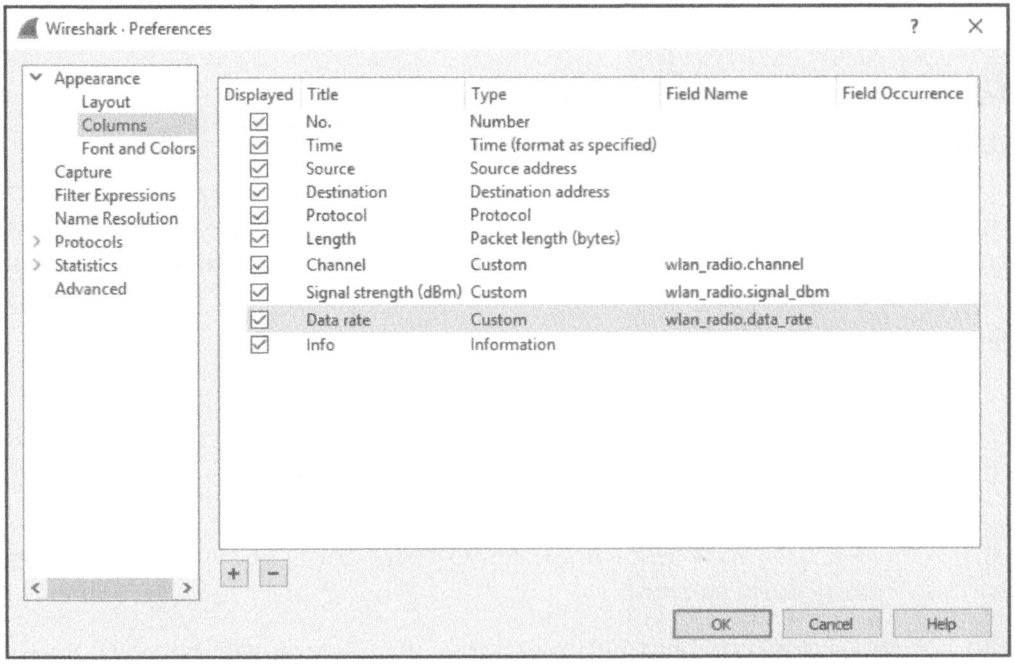


Рис. 13.10. Добавление столбцов на панель Packet List специально для анализа пакетов в беспроводной сети по стандарту IEEE

Специальные фильтры для беспроводных сетей

Преимущества фильтров перехвата и отображения уже обсуждались в главе 4, "Обработка перехваченных пакетов". Отфильтровать сетевой трафик в инфраструктуре проводной сети намного проще, поскольку каждое устройство подключено к сети по отдельному кабелю. Но в беспроводной сети весь сетевой трафик, сформированный беспроводными клиентами, сосуществует в общих каналах. Это означает, что перехваченные пакеты в каком-нибудь одном канале могут содержать сетевой трафик от сотен клиентов. Этот раздел посвящен обсуждению некоторых фильтров пакетов, которыми можно пользоваться для поиска интересующего вас трафика.

Фильтрация сетевого трафика по конкретному идентификатору BSSID

Каждой точке беспроводного доступа в сети присваивается однозначно определяющий ее *идентификатор базового набора услуг* (Basic Service Set Identifier, BSSID). Этот идентификатор посылается в каждом беспроводном пакете управления и пакете данных из передающей точки доступа.

Зная идентификатор BSSID точки беспроводного доступа, для ее исследования достаточно найти пакет, отправленный из этой конкретной точки. В приложении Wireshark передающая точка доступа отображается в столбце **Info** на панели Packet List, и поэтому найти сведения об этой точке, как правило, нетрудно.

Получив пакет из представляющей интерес точки доступа, ее идентификатор BSSID можно найти в соответствующем поле из заголовка по стандарту 802.11, где, по существу, указан MAC-адрес. На основании MAC-адреса можно создать приведенный ниже фильтр. После применения этого фильтра останется только тот сетевой трафик, который проходит через указанную точку беспроводного доступа.

```
wlan.bssid == 00:11:22:33:44:55
```

Фильтрация пакетов по конкретным типам

Ранее в этой главе обсуждались разные типы пакетов, которые можно наблюдать в беспроводной сети. И зачастую возникает потребность выявить пакеты этих типов или их подтипов. Это можно сделать с помощью фильтра `wlan.fc.type` для отдельных типов пакетов или фильтра `wc.fc.type_subtype` для отдельных подтипов пакетов или их сочетаниям. Например, чтобы выявить пакеты, не содержащие данных (NULL data packet), достаточно применить фильтр **`wlan.fc.type_subtype == 0x24`**, где шестнадцатеричная цифра **2**

обозначает тип 2, а шестнадцатеричная цифра **4** – подтип 4 искомого пакета. В табл. 13.1 перечислены для краткой справки наиболее употребительные фильтры, которые могут потребоваться для фильтрации типов и подтипов пакетов по стандарту 802.11.

Таблица 13.1. Типы и подтипы беспроводных пакетов и синтаксис соответствующих фильтров

Тип или подтип пакета	Синтаксис фильтра
Фрейм управления	`wlan.fc.type == 0`
Фрейм контроля	`wlan.fc.type == 1`
Фрейм данных	`wlan.fc.type == 2`
Запрос на установку связи	`wlan.fc.type_subtype == 0x00`
Ответ на установку связи	`wlan.fc.type_subtype == 0x01`
Запрос на повторную установку связи	`wlan.fc.type_subtype == 0x02`
Ответ на повторную установку связи	`wlan.fc.type_subtype == 0x03`
Запрос на зондирование	`wlan.fc.type_subtype == 0x04`
Ответ на зондирование	`wlan.fc.type_subtype == 0x05`
Сигнальный пакет	`wlan.fc.type_subtype == 0x08`
Разрыв связи	`wlan.fc.type_subtype == 0x0A`
Аутентификация	`wlan.fc.type_subtype == 0x0B`
Отказ в аутентификации	`wlan.fc.type_subtype == 0x0C`
Фрейм действия	`wlan.fc.type_subtype == 0x0D`
Запросы на подтверждение блокировки	`wlan.fc.type_subtype == 0x18`
Подтверждение блокировки	`wlan.fc.type_subtype == 0x19`
Опрос энергосбережения	`wlan.fc.type_subtype == 0x1A`
Запрос на передачу	`wlan.fc.type_subtype == 0x1B`
Готовность к приему	`wlan.fc.type_subtype == 0x1C`
Подтверждение (ACK)	`wlan.fc.type_subtype == 0x1D`
Окончание бесконфликтного периода	`wlan.fc.type_subtype == 0x1E`
Пустые данные (NULL data)	`wlan.fc.type_subtype == 0x24`
Данные о качестве обслуживания (QoS)	`wlan.fc.type_subtype == 0x28`
Пустые данные о качестве обслуживания (Null QoS)	`wlan.fc.type_subtype == 0x2C`

Фильтрация пакетов по отдельным каналам

Для анализа сетевого трафика, который составляют пакеты из нескольких каналов, очень полезной оказывается фильтрация по отдельным каналам. Так, если предполагается, что сетевой трафик должен присутствовать только в каналах 1 и 6, можно применить фильтр для отображения всего трафика в канале 11. Если при этом обнаружится какой-нибудь посторонний трафик, это будет означать, что в беспроводной сети что-то не так, например, нарушена

конфигурация или неисправно какое-то устройство. Чтобы выявить трафик в отдельном канале, достаточно воспользоваться следующим синтаксисом фильтров:

```
wlan_radio.channel == 11
```

В данном примере отображается сетевой трафик в канале 11, а для фильтрации трафика в другом канале достаточно подставить его номер вместо значения **11**. Имеются сотни других фильтров, которыми можно воспользоваться для анализа трафика, перехваченного в беспроводной сети. Подробнее об этих фильтрах можно узнать, перейдя по ссылке, соответствующей типу фильтров (перехвата или отображения), на вики-странице Wireshark по адресу `http://wiki.wireshark.org/`.

Сохранение профиля беспроводной сети

Для анализа пакетов в беспроводной сети приходится прилагать немало усилий, добавляя отдельные столбцы и сохраняя специальные фильтры. Но вместо того чтобы изменять всякий раз конфигурацию программы и удалять столбцы, можно создать специальный профиль для быстрой смены конфигурации при переходе от беспроводной сети к проводной и обратно с целью анализа пакетов.

Чтобы сохранить профиль, сконфигурируйте сначала столбцы и фильтры для подключения к беспроводной сети. Затем щелкните правой кнопкой мыши на списке активных профилей в правой нижней части экрана и далее щелкните на кнопке New. Присвойте новому профилю имя **`Wireless`** (Беспроводный) и щелкните на кнопке OK.

Безопасность в беспроводной сети

При развертывании и администрировании беспроводной сети больше всего хлопот вызывает безопасность данных, передаваемых по такой сети. В частности, данные, передаваемые в эфире, очень важно зашифровать, иначе их может перехватить всякий посторонний средствами Wireshark и AirPcap.

ПРИМЕЧАНИЕ *Если применяется шифрование на другом уровне модели OSI (например, по протоколу SSL или SSH), сетевой трафик уже будет зашифрован при переходе на более низкие уровни, поэтому обмен данными пользователя невозможно будет прочитать с помощью анализатора пакетов.*

Первоначально наиболее предпочтительным методом защиты данных, передаваемых по беспроводным сетям, считалось соблюдение стандарта WEP

(Wired Equivalent Privacy — эквивалент конфиденциальности проводных сетей). Стандарт WEP успешно применялся многие годы до тех пор, пока не было обнаружено несколько слабых мест в его алгоритме управления ключами шифрования. Поэтому для повышения безопасности были разработаны новые стандарты, в том числе WPA (Wi-Fi Protected Access — защищенный беспроводный доступ к сети) и WPA2 (с более безопасным алгоритмом). И хотя стандарты WPA и WPA2 не вполне надежны и подвержены ошибкам, они все же считаются более безопасными, чем стандарт WEP. В этом разделе будут рассмотрены некоторые примеры анализа сетевого трафика по стандартам WEP и WPA с удачными и неудачными попытками аутентификации.

Успешная аутентификация по алгоритму WEP

Файл перехвата **3e80211_WEPauth.pcapng**

Рассматриваемый здесь файл перехвата `3e80211_WEPauth.pcapng` содержит пример успешного подключения к беспроводной сети по стандарту WEP. Безопасность в этой сети обеспечивается с помощью WEP-ключа шифрования, который требуется указать при настройке точки беспроводного доступа для аутентификации и расшифровки посылаемых из нее данных. В этом отношении WEP-ключ можно рассматривать как своего рода пароль для доступа к беспроводной сети.

Как показано на рис. 13.11, сетевой трафик из рассматриваемого здесь файла перехвата начинается с запроса, направляемого в пакете 3 ❶ из точки беспроводного доступа, находящейся по адресу **28:c6:8e:ab:96:16**, беспроводному клиенту, расположенному по адресу **ac:cf:5c:78:6c:9c**. Назначение этого запроса — выяснить, имеется ли WEP-ключ у беспроводного клиента. Этот запрос можно увидеть, развернув заголовок стандарта 802.11 и проанализировав промаркированные (`Tag`) в нем параметры.

В ответ на этот запрос беспроводный клиент расшифровывает текст запроса ❶ с помощью WEP-ключа и возвращает его в открытом виде в пакете 4, как показано на рис. 13.12. WEP-ключ вводится пользователем при попытке подключения к беспроводной сети.

В свою очередь, точка беспроводного доступа отвечает беспроводному клиенту пакетом 5, как показано на рис. 13.13. В этом ответе содержится уведомление об удачном завершении процесса аутентификации ❶.

И, наконец, после удачной аутентификации клиент может передать запрос на установку связи, получить подтверждение и завершить процесс подключения, как показано на рис. 13.14.

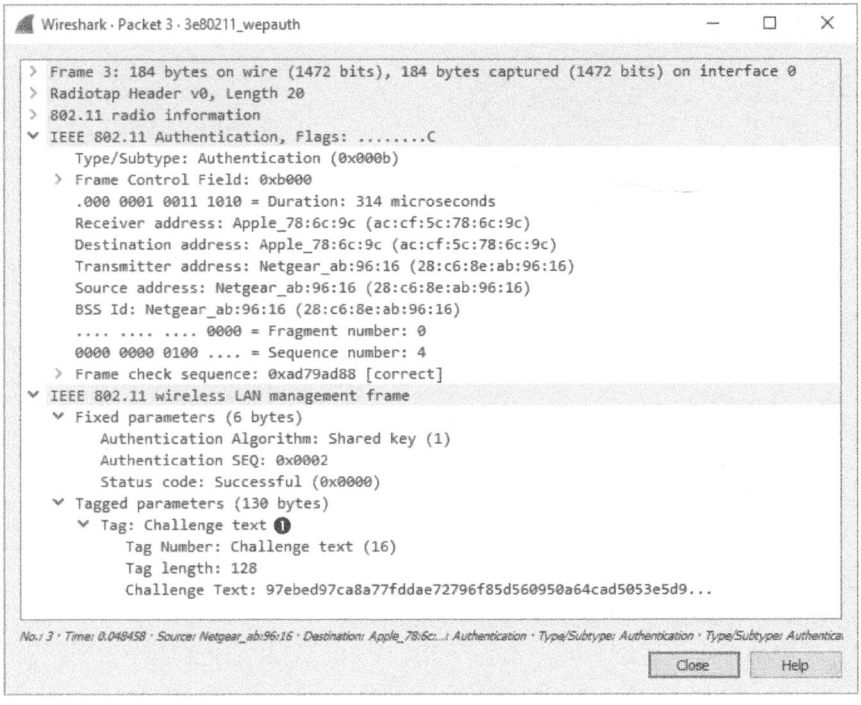


Рис. 13.11. Из точки беспроводного доступа делается запрос беспроводного клиента

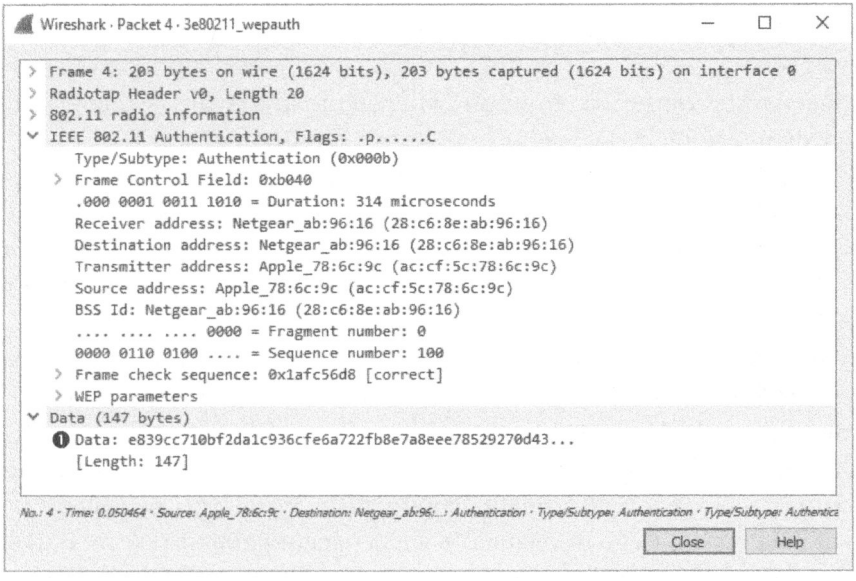


Рис. 13.12. Беспроводный клиент посылает незашифрованный текст запроса обратно в точку доступа

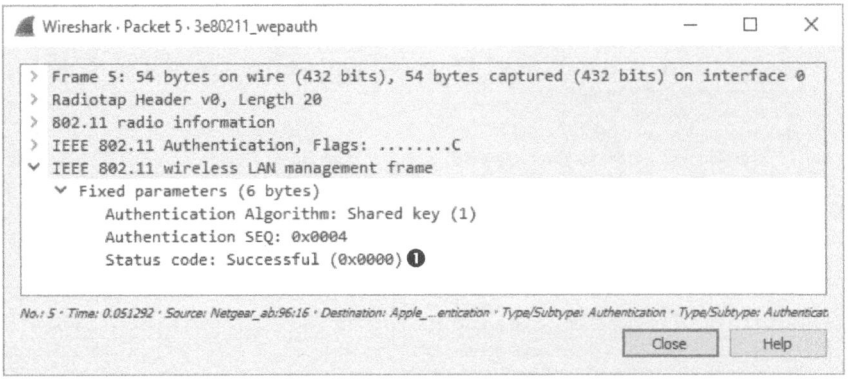
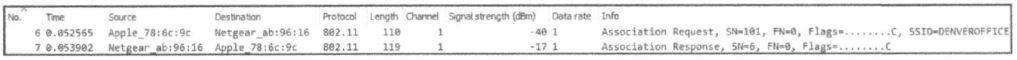


Рис. 13.13. Точка беспроводного доступа уведомляет беспроводного клиента об удачном завершении процесса аутентификации



No.	Time	Source	Destination	Protocol	Length	Channel	Signal strength (dBm)	Data rate	Info
6	0.052565	Apple_78:6c:9c	Netgear_ab:96:16	802.11	110	1	-40	1	Association Request, SN=101, FN=0, Flags=........C, SSID=DENVEROFFICE
7	0.053902	Netgear_ab:96:16	Apple_78:6c:9c	802.11	119	1	-17	1	Association Response, SN=6, FN=0, Flags=........C

Рис. 13.14. По окончании аутентификации следует простой процесс запроса и ответа на установку связи, состоящий из двух пакетов

Неудачная аутентификация по алгоритму WEP

Файл перехвата **`3e80211_WEPauthfail.pcapng`**

В следующем примере пользователь вводит WEP-ключ для подключения к точке беспроводного доступа. Но через несколько секунд беспроводный клиент отвечает, что ему не удалось подключиться к беспроводной сети, хотя и не объясняет конкретную причину. Полученный в итоге сетевой трафик сохранен в файле перехвата `3e80211_WEPauthfail.pcapng`.

Как и при удачной попытке аутентификации, рассматриваемый здесь обмен данными начинается с отправки текста запроса из точки беспроводного доступа беспроводному клиенту в пакете 3. В свою очередь, беспроводный клиент отправляет ответ в пакете 4, используя WEP-ключ, введенный пользователем.

В этот момент можно было бы ожидать уведомления об удачном завершении аутентификации, но в пакете 5 мы наблюдаем нечто совсем иное ❶, как показано на рис. 13.15.

В данном сообщении уведомляется, что ответ беспроводного клиента на текст запроса оказался неверным, вероятно потому, что был введен неправильный WEP-ключ, использовавшийся для расшифровки текста. В итоге процесс подключения к беспроводной сети завершился неудачно. Поэтому его придется повторить снова, но уже с правильным WEP-ключом.

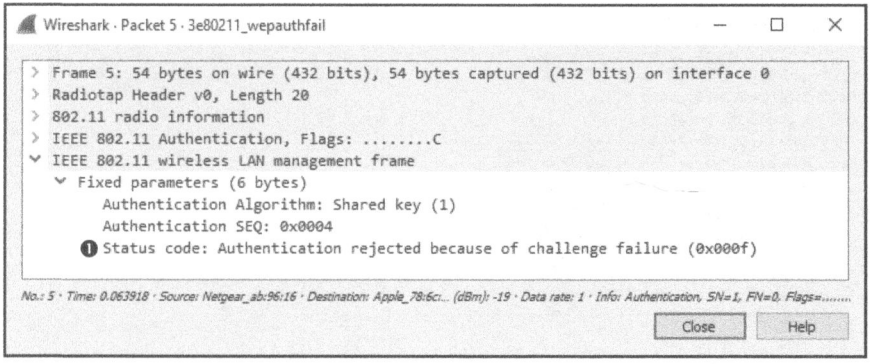


Рис. 13.15. В этом сообщении уведомляется, что аутентификация завершилась неудачно

Удачная аутентификация по алгоритму WPA

Файл перехвата **3e80211_WPAauth.pcapng**

В стандарте WPA применяется совершенно другой механизм аутентификации, чем в стандарте WEP, но и в нем по-прежнему используется ключ шифрования, введенный пользователем на стороне беспроводного клиента при подключения к сети. Пример удачной аутентификации по алгоритму WPA приведен в файле перехвата `3e80211_WPAauth.pcapng`.

Первым в сетевом трафике из этого файла перехвата следует сигнальный пакет, переданный в широковещательном режиме из точки беспроводного доступа. Разверните заголовок стандарта 802.11 из этого пакета, перейдите к промаркированным параметрам и разверните заголовок **Vendor Specific** (Параметры производителя), как показано на рис. 13.16. В итоге вы должны обнаружить, что данная часть пакета посвящена атрибутам WPA точки беспроводного доступа ❶. Анализируя ее, можно выяснить версию и реализацию стандарта WPA, если она вообще поддерживается в точке беспроводного доступа.

Как только сигнальный пакет будет получен, беспроводный клиент, находящийся по адресу **ac:cf:5c:78:6c:9c**, передает в широковещательном режиме зондирующий запрос (probe request) в пакете 2, который принимается точкой беспроводного доступа, расположенной по адресу **28:c6:8e:ab:96:16** и отправляющей в ответ пакет 3. После этого происходит обмен пакетами с запросами на аутентификацию и установку связи между беспроводным клиентом и точкой беспроводного доступа в пакетах 4–7. Эти пакеты подобны пакетам с запросами на аутентификацию и установку связи, рассмотренные ранее в примере применения алгоритма WEP, но в данном случае никакого запроса и ответа на него не происходит, поскольку такой обмен выполняется далее.

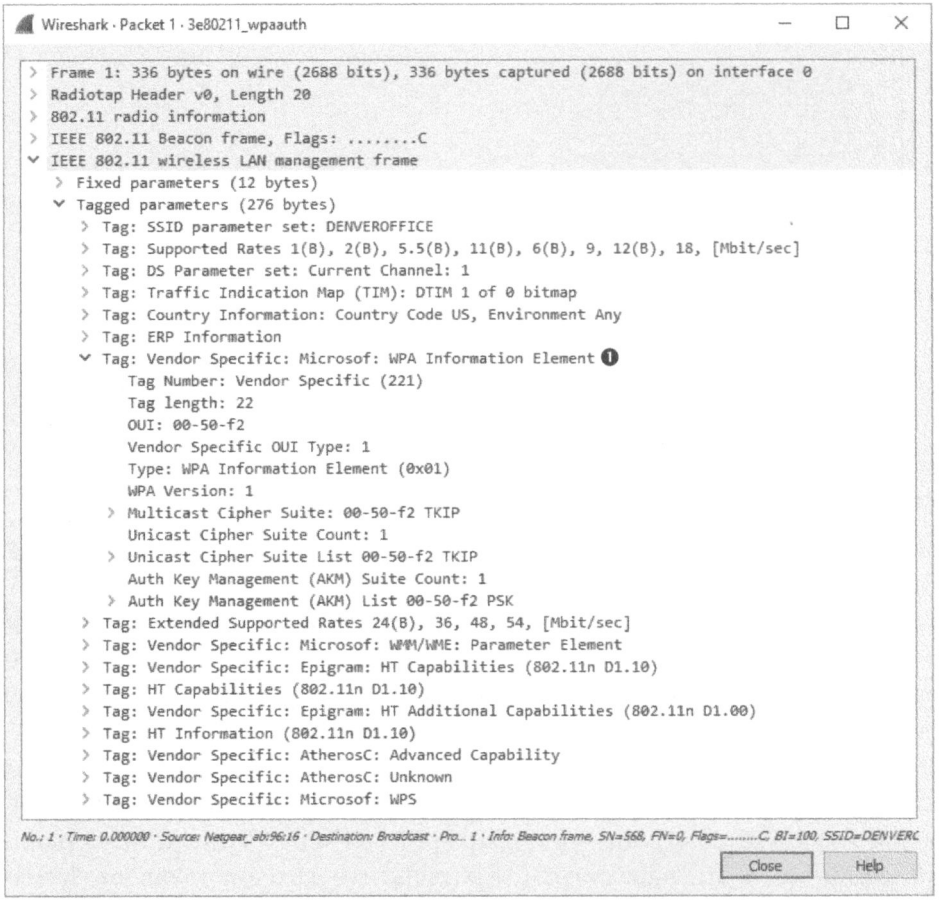


Рис. 13.16. Из этого сигнального пакета можно выяснить, что в точке беспроводного доступа поддерживается аутентификация по алгоритму WPA

Дело начинает принимать любопытный оборот в пакете 8. Именно здесь и начинается процесс установки связи по алгоритму WPA, который продолжается вплоть до пакета 11. В ходе этого процесса отравляется запрос и получается ответ на него, как показано на рис. 13.17.

No.	Time	Source	Destination	Protocol	Length	Channel	Signal strength (dBm)	Data rate	Info
8	0.…	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-18	24	Key (Message 1 of 4)
9	0.…	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-42	1	Key (Message 2 of 4)
10	0.…	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	181	1	-18	36	Key (Message 3 of 4)
11	0.…	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	157	1	-42	1	Key (Message 4 of 4)

Рис. 13.17. Эти пакеты являются частью процесса установки связи по алгоритму WPA

В данном случае отправляются два запроса и принимаются два ответа. Их соответствие можно определить, исходя из содержимого поля **Replay Counter** (Счетчик ответов) в заголовке **Authentication** по стандарту 802.1x, как пока-

зано на рис. 13.18. В частности, обратите внимание на то, что значение в поле **Replay Counter** из первого пакета установки связи равно **1** ❶, а во втором пакете — 2 ❷. При удачном завершении процесса установки связи и аутентификации начинается обмен данными между беспроводным клиентом и точкой доступа.

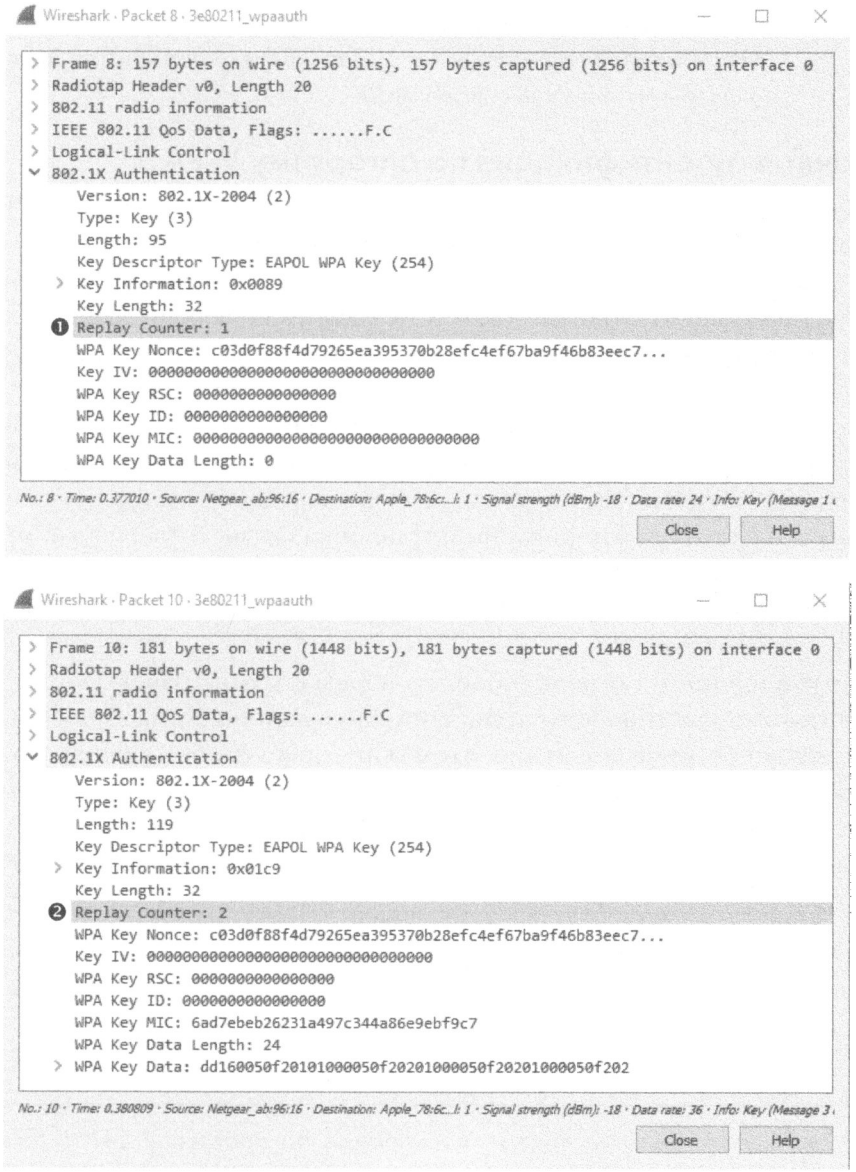


Рис. 13.18. Анализ значения в поле **Replay Counter** *помогает связать вместе пары запросов и ответов на них*

ПРИМЕЧАНИЕ *Сетевой трафик из данного примера взят из точки беспроводного доступа, где применяется алгоритм WPA с TKIP для шифрования данных в WLAN. В различных точках беспроводного доступа применяются разные алгоритмы шифрования данных. Так, если в точке беспроводного доступа применяется другой алгоритм шифрования данных или версия стандарта WPA, то ее характеристики на уровне пакетов будут заметно отличаться. Чтобы узнать подробнее о том, как должна выглядеть последовательность установки беспроводного соединения, обращайтесь к соответствующим документам RFC.*

Неудачная аутентификация по алгоритму WPA

Файл перехвата `3e80211_WPAauthfail.pcapng`

Как и в примере применения алгоритма WEP, мы проанализируем в данном случае, что произойдет, если пользователь введет WPA-ключ, на который беспроводный клиент ответит, что ему не удалось подключиться к беспроводной сети, хотя и без указания конкретной причины. Полученный в итоге результат сохранен в файле перехвата `3e80211_WPAauthfail.pcapng`.

Сетевой трафик в этом файле перехвата начинается таким же образом, как и в предыдущем примере, где рассматривалась удачная аутентификация по алгоритму WPA. Это означает, что вначале передаются зондирующие пакеты, а затем пакеты аутентификации и установки связи. В частности, процесс установки связи по алгоритму WPA начинается с пакета 8, но в данном случае вместо четырех пакетов, как при предыдущей попытке аутентификации, передаются восемь пакетов.

Пакеты 8 и 9 представляют собой два первых пакета, наблюдаемых в процессе установки связи по алгоритму WPA. Но в данном случае текст запроса, отправляемый беспроводным клиентом обратно точке беспроводного доступа, оказывается неверным. В итоге последовательность обмена данными повторяется в пакетах 10–15, как показано на рис. 13.19. Каждый запрос и ответ могут быть объединены в пары с помощью значения в поле **`Replay Counter`**.

No.	Time	Source	Destination	Protocol	Length	Channel	Signal strength (dBm)	Data rate	Info
8	0.073773	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-18	24	Key (Message 1 of 4)
9	0.076510	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-30	1	Key (Message 2 of 4)
10	1.074290	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-19	24	Key (Message 1 of 4)
11	1.076573	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-32	1	Key (Message 2 of 4)
12	2.075292	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-18	36	Key (Message 1 of 4)
13	2.077610	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-29	1	Key (Message 2 of 4)
14	3.077211	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-18	48	Key (Message 1 of 4)
15	3.079537	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-32	1	Key (Message 2 of 4)

Рис. 13.19. Дополнительные пакеты, посылаемые по протоколу EAPoL (Extensible Authentication Protocol over LAN — расширяемый протокол аутентификации в локальной сети), указывают на неудачный исход аутентификации по алгоритму WPA

Как только попытка установки соединения завершится неудачно четыре раза подряд, связь будет прервана. Как показано на рис. 13.20, точка беспроводного доступа отказывает в аутентификации клиенту в пакете 16 ❶.

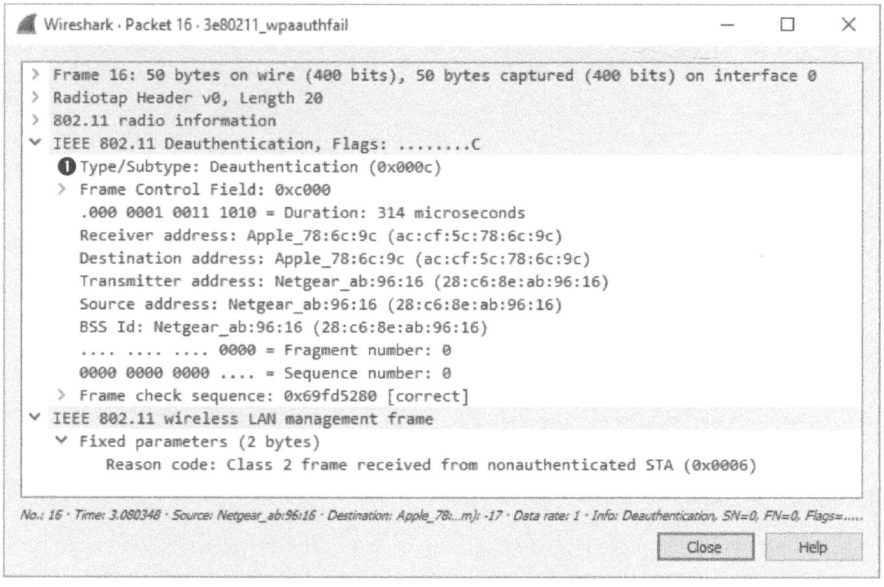


Рис. 13.20. После четырех неудачных попыток установки связи по алгоритму WPA точка доступа отказывает клиенту в аутентификации

Заключительные соображения

Несмотря на то что беспроводные сети до сих пор считаются не вполне безопасными, если только не снабдить их целым рядом дополнительных механизмов защиты, это обстоятельство не помешало их развертыванию в различных организационных средах. По мере того как беспроводная связь все больше становится нормой, очень важно уметь перехватывать и анализировать данные в беспроводных сетях так же, как и в проводных. Принципы и методики, рассмотренные в этой главе, нельзя считать исчерпывающими, хотя они могут послужить неплохой отправной точкой для дальнейшего углубленного изучения особенностей диагностики беспроводных сетей посредством анализа пакетов.

A

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

В примерах из этой книги для анализа пакетов применялось в основном приложение Wireshark. Хотя для анализа пакетов, включая диагностику проводных и беспроводных сетей, борьбу с медленно работающими сетями и обеспечение сетевой безопасности, имеется немало других удобных инструментальных средств. В этом приложении кратко описан ряд других инструментальных средств и перечислены дополнительные ресурсы для дальнейшего изучения анализа пакетов.

Инструментальные средства для анализа пакетов

Рассмотрим ряд других инструментальных средств, которые могут оказаться полезными для анализа пакетов.

CloudShark

Инструментальное средство CloudShark, разработанное компанией QA Café, относится, на мой взгляд, к числу самых лучших для хранения, индексирования и сортировки перехваченных пакетов. Это коммерчески доступное веб-приложение, предназначенное для организации хранилища перехваченных пакетов. Оно позволяет помечать перехваченные пакеты для быстрого нахождения и снабжать их комментариями и даже предоставляет

некоторые средства для анализа пакетов, аналогичные тем, что имеются в Wireshark (рис. А.1).

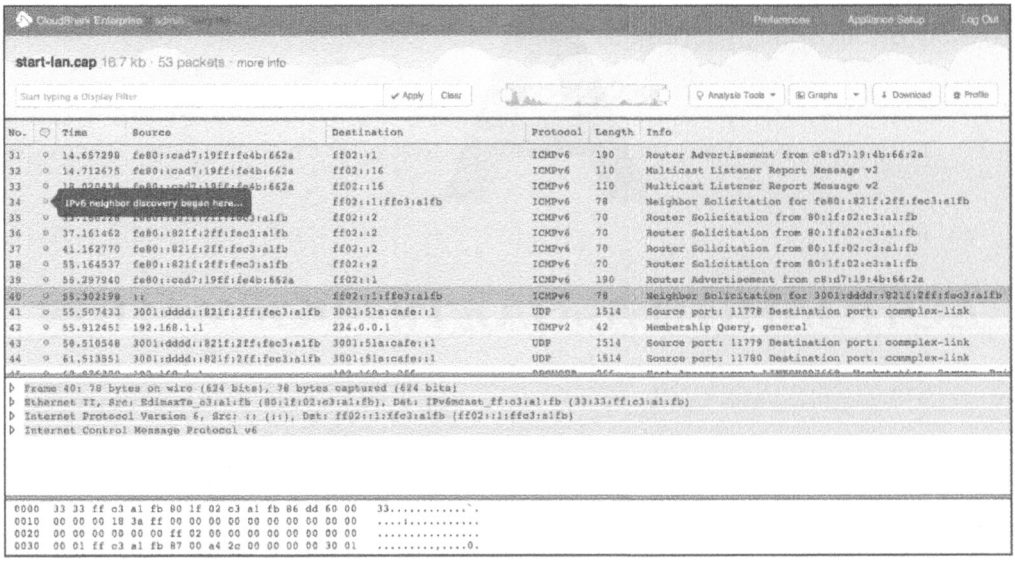


Рис. А.1. Пример просмотра файла перехвата в CloudShark

Если в вашей организации ведется крупная библиотека образцов перехвата пакетов или же вы, как и я, часто теряете файлы перехвата, CloudShark поможет вам организовать их надежное хранение. Я, например, развернул CloudShark в своей сети и воспользовался этим инструментальным средством для хранения и организации всех файлов перехвата, употреблявшихся в примерах из этой книги. Подробнее о CloudShark можно узнать по адресу `https://www.cloudshark.org/`.

WireEdit

Чтобы протестировать применяемую систему обнаружения вторжений или разработку сетевого программного обеспечения, возможно, придется создавать специально отформатированные пакеты. В частности, можно воссоздать сценарий, в котором нужные пакеты будут формироваться в лабораторных условиях, хотя для того, чтобы сделать это, потребуется немало времени. Еще один способ состоит в том, чтобы найти аналогичный пакет и подправить его вручную под свои потребности. Для решения подобных задач рекомендуется графическое инструментальное средство WireEdit, позволяющее редактировать значения в отдельных полях пакета. Его удобный пользовательский интерфейс очень похож на аналогичный интерфейс в Wireshark.

Более того, WireEdit позволяет пересчитывать контрольные суммы пакетов, чтобы они не оказались недостоверными при открытии в Wireshark. Подробнее о WireEdit можно узнать по адресу `https://wireedit.com/`.

Cain & Abel

Как обсуждалось в главе 2, Cain & Abel относится к числу самых лучших инструментальных средств Windows для заражения ARP-кеша. По существу, Cain & Abel представляет собой весьма надежный набор инструментальных средств, которым вы, вероятнее всего, найдете и другое применение. Инструментальное средство Cain & Abel доступно по адресу `http://www.oxid.it/cain.html`.

Scapy

Scapy — это очень эффективная библиотека на языке Python, с помощью которой можно создавать пакеты и манипулировать ими по сценариям, выполняемым из командной строки в своей среде. Проще говоря, библиотека Scapy оказывается наиболее эффективной и удобной для обработки пакетов. Подробнее узнать о Scapy, загрузить и просмотреть примеры сценариев Scapy можно по адресу `http://www.secdev.org/projects/scapy/`.

TraceWrangler

Файлы перехвата пакетов содержат немало сведений о сети. Если требуется поделиться файлом перехвата пакетов из своей сети с провайдером или коллегой, то вряд ли стоит сообщать им все сведения о своей сети. И здесь на помощь придет инструментальное средство TraceWrangler, позволяющее производит "саночистку" перехваченных пакетов, чтобы сделать анонимными различные типы присутствующих в них адресов. TraceWrangler обладает рядом других возможностей, в том числе редактировать и объединять файлы перехвата, но я пользуюсь этим инструментальным средством только для "саночистки". Загрузить TraceWrangler можно по адресу `https://www.tracewrangler.com/`.

Tcpreplay

После сбора пакетов в файле перехвата часто требуется выполнить их повторную передачу, чтобы посмотреть, как на них будут реагировать сетевые устройства. Для этого используется утилита Tcpreplay. Саму программу можно загрузить по адресу `http://tcpreplay.synfin.net/`.

NetworkMiner

NetworkMiner — это инструментальное средство, применяемое для судебной экспертизы сетей, но я нашел его полезным во многих других случаях. И хотя NetworkMiner можно использовать для перехвата пакетов, истинный потенциал этого инструментального средства состоит в способности выполнять синтаксический анализ пакетов из файлов перехвата. В этом случае NetworkMiner выбирает файл перехвата формата PCAP и разбирает его содержимое по типам операционных систем, обнаруживаемых в сеансах связи между хостами. NetworkMiner позволяет даже извлекать переданные файлы непосредственно из перехваченного трафика (рис. А.2). Все эти возможности доступны в свободно распространяемой версии NetworkMiner. А в коммерчески доступной версии предоставляется ряд других полезных возможностей, в том числе получение отпечатков операционных систем, сравнение полученных результатов с "белым" списком и повышение скорости обработки перехваченных пакетов. Свободно загружаемая версия NetworkMiner доступна по адресу `http://www.netresec.com/?page=NetworkMiner`.

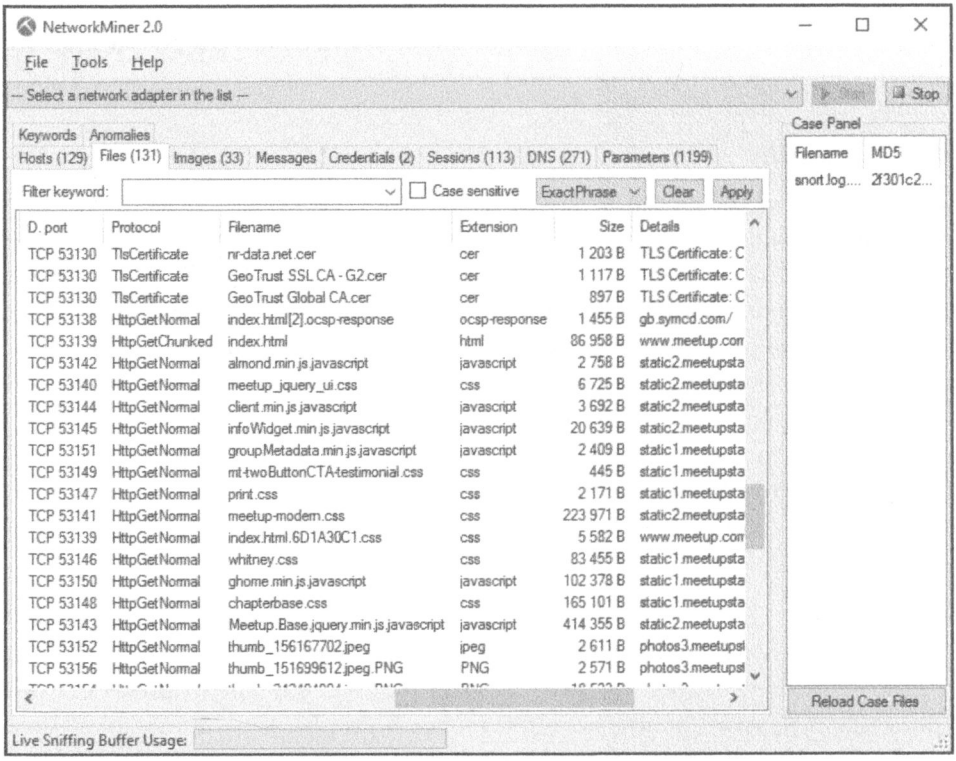


Рис. А.2. Применение NetworkMiner для синтаксического анализа содержимого файлов перехвата пакетов

CapTipper

Надеюсь, что, прочитав эту книгу, вы научитесь анализировать данные из перехваченных пакетов под разными углами зрения, чтобы найти ответы на насущные вопросы диагностики сетей. Инструментальное средство CapTipper специально предназначено для специалистов по безопасности сетей, занимающихся анализом сетевого трафика по протоколу HTTP (рис. A.3). CapTipper предоставляет богато оснащенную среду командной оболочки, которая дает пользователю возможность исследовать отдельные диалоги в интерактивном режиме в поисках переадресации, файловых объектов и зловредного содержимого. В CapTipper предоставляются также удобные средства для взаимодействия с раскрываемыми данными, включая возможность извлекать данные из архивов типа gzip и передавать хеш-суммы файлов в службу VirusTotal. Загрузить CapTipper можно по адресу `https://www.github.com/omriher/CapTipper/`.

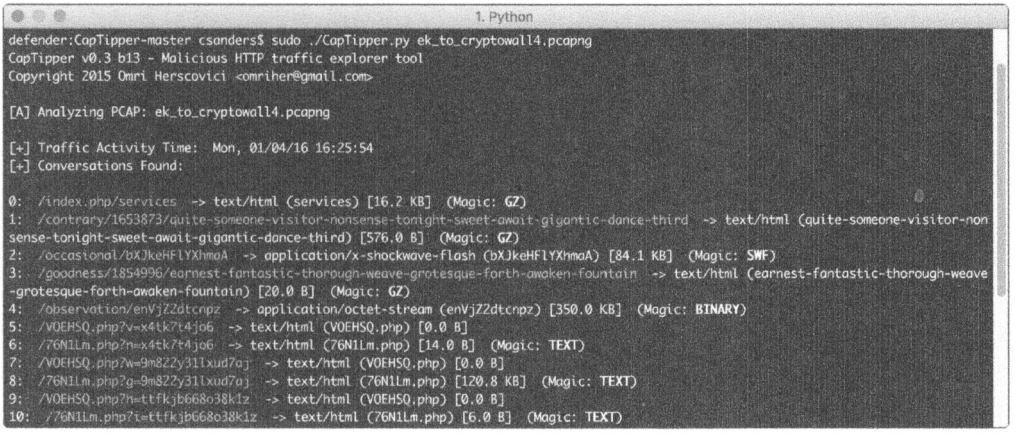


Рис. A.3. Анализ доставки вредоносного содержимого по протоколу HTTP в CapTipper

ngrep

Если вы знакомы с Linux, то, несомненно, пользовались утилитой **`grep`** для поиска данных. Аналогичным образом утилита **`ngrep`** позволяет находить конкретные данные в перехваченных пакетах. Я обычно пользуюсь **`ngrep`** в тех случаях, когда фильтры перехвата и отображения оказываются непригодными или слишком сложными для выявления нужных данных. Подробнее об утилите **`ngrep`** можно узнать по адресу `http://ngrep.sourceforge.net/`.

libpcap

Если вы собираетесь проводить расширенный синтаксический анализ пакетов или разрабатывать приложения для обработки пакетов, непременно

ознакомьтесь с библиотекой libpcap. Проще говоря, libpcap – это переносимая библиотека C/C++ для перехвата сетевого трафика. Wireshark, `tcpdump` и большинство других приложений для анализа пакетов в какой-то степени используют библиотеку libpcap. Подробнее о библиотеке libpcap можно узнать по адресу `http://www.tcpdump.org/`.

Npcap

Npcap – это библиотека из проекта под названием Nmap Project, основанная на библиотеке WinPcap/libpcap и предназначенная для анализа пакетов в Windows. Как сообщается, Npcap служит для повышения производительности при перехвате пакетов и предоставляет дополнительные возможности для обеспечения безопасности, позволяющие ограничить перехват пакетов пользователя, обладающими правами сетевых администраторов и выгодно пользоваться управлением учетными записями пользователей в Windows. Npcap можно установить в качестве альтернативы WinPCap и применять вместе с Wireshark. Подробнее узнать о Npcap можно по адресу `https://www.github.com/nmap/npcap/`.

hping

Инструментальное средство hping относится к числу самых универсальных в арсенале тех, кто занимается анализом пакетов. Оно служит для обработки, правки и передачи пакетов в режиме командной строки. Инструментальное средство hping поддерживает различные сетевые протоколы, очень просто и интуитивно в употреблении. Загрузить hping можно по адресу `http://www.hping.org/`.

Python

Python – это не инструментальное средство, а интерпретируемый язык программирования, заслуживающий здесь особого упоминания. По мере совершенствования в анализе пакетов вы непременно столкнетесь с такими случаями, когда для удовлетворения ваших потребностей у вас не окажется подходящих автоматизированных средств. В подобных случаях приходит на помощь язык Python, позволяющий создавать инструментальные средства, способные выполнять немало интересных действий над пакетами. Кроме того, для взаимодействия с библиотекой Scapy требуется хотя бы немного знать язык Python. И здесь вам, как нельзя кстати, придется книга *Автоматизация рутинных задач с помощью Python: практическое руководство для начинающих*, перевод которой на русский язык вышел в издательстве "Диалектика" в 2016 году (ISBN 978-5-6040724-2-4).

Ресурсы по анализу пакетов

По анализу пакетов имеется немало ресурсов: от начальной страницы веб-сайта, посвященного Wireshark, до различных учебных курсов, форумов и блогов. Ниже перечислены лишь некоторые из них.

Начальная страница веб-сайта, посвященного Wireshark

Самым главным и первоочередным ресурсом по Wireshark является начальная страница веб-сайта этого приложения, доступная по адресу `http://www.wireshark.org/`. На этой странице содержатся ссылки на документацию по Wireshark, очень полезные вики-страницы с образцами файлов перехвата и сведения о подписке на список рассылки по Wireshark. Чтобы задать вопросы по применению Wireshark или отдельным возможностям этого приложения, достаточно перейти на форум по адресу `https://ask.wireshark.org/`, где активно действует сообщество пользователей Wireshark, готовое оказать всяческую помощь коллегам.

Онлайновые практические курсы по анализу пакетов

Если вам понравится эта книга, вы можете также пройти дополнительные онлайновые курсы по практическому анализу пакетов. Эти курсы состоят из учебных видеоматериалов для проработки примеров перехвата пакетов, упомянутые не только в этой книги, но и в ряде других источников. На этих курсах проводятся также лабораторные занятия по перехвату пакетов, где вы можете проверить свои навыки, а также ведется дискуссионный форум, где можно поучиться у других учащихся в ходе обучения. Узнать подробнее об условиях обучения можно по адресу `http://www.chrissanders.org/training/`, а подписаться на список рассылки для извещения о возможностях обучения — по адресу `http://www.chrissanders.org/list/`.

Углубленные курсы в институте SANS по обнаружению вторжений

Углубленные курсы SANS SEC503 по обнаружению вторжений посвящены вопросам безопасности, решаемым при анализе пакетов. Даже если вы не занимаетесь непосредственно вопросами сетевой безопасности, в течение первых двух дней на этих курсах вы сможете получить ясное и полное представление об анализе пакетов и утилите `tcpdump`. Эти курсы проводятся несколько раз в год по всему миру. Подробнее узнать о курсах SEC503 и самом институте SANS можно по адресу `http://www.sans.org/`.

Блог Криса Сандерса

Время от времени я пишу статьи по анализу пакетов и публикую их в своем блоге по адресу `http://www.chrissanders.org/`. Мой блог служит также порталом для ссылок на другие написанные мной статьи и книги, а также мою контактную информацию. Здесь можно найти и ссылки на примеры перехвата пакетов, включенные в эту книгу, а также ряд других примеров.

Веб-сайт Бреда Дункана, посвященный анализу вредоносного трафика

Моим излюбленным ресурсом по перехвату пакетов, связанных с нарушением сетевой безопасности, является веб-сайт MTA (Malware Traffic Analysis — анализ вредоносного трафика) Бреда Дункана. На своем сайте Бред публикует несколько раз в неделю примеры перехвата пакетов, содержащие цепочки заражения вредоносным кодом. Эти примеры дополняются соответствующим двоичным представлением вредоносного кода и описанием того, что происходит при заражении им. Если вы стремитесь приобрести опыт и навыки в выявлении фактов заражения вредоносным кодом, а также изучить современные методики применения вредоносного кода, начните с загрузки этих примеров перехвата пакетов и попытайтесь разобраться в них. Посетить веб-сайт MTA можно по адресу `http://www.malware-traffic-analysis.net/` или же следить за сообщениями Бреда о последних обновлениях примеров перехвата пакетов в Twitter по адресу `@malware_traffic`.

Веб-сайт IANA

Администрация адресного пространства Интернета (IANA — Internet Assigned Numbers Authority), доступная по адресу `http://www.iana.org/`, осуществляет надзор над распределением IP-адресов и присваиванием номеров портов в Северной Америке. На веб-сайте этой организации предоставляется ряд ценных справочных систем, позволяющих находить номера портов, просматривать сведения о доменных именах верхнего уровня и переходить по ссылкам на сопутствующие веб-сайты для поиска и просмотра документов RFC.

Иллюстрированная серия по протоколам TCP/IP Ричарда У. Стивенса

Иллюстрированная серия *TCP/IP Illustrated* Ричарда У. Стивенса (W. Richard Stevens), вышедшая в издательстве Addison-Wesley в 1994–1996 гг., считается настольной книгой по стеку протоколов TCP/IP многими из тех, кто

постоянно занимается диагностикой сетей на уровне пакетов. Второе издание первого тома из этой серии, написанное Стивенсом в соавторстве с доктором Кивеном (Dr. Keven) и Р. Фоллом (R. Fall), увидело свет в 2012 году.

Руководство по стеку протоколов TCP/IP

Книга *The TCP/IP Guide* Чарльза Козьерока (Charles Kozierok, издательство No Starch Press, 2005 г.) служит еще одним справочным пособием и полезным источником информации по стеку протоколов TCP/IP. В этом руководстве объемом более 1600 страниц очень подробно и с многочисленными наглядными иллюстрациями описаны особенности работы протоколов TCP/IP.

Б

ИНТЕРПРЕТАЦИЯ ПАКЕТОВ

В этом приложении рассматриваются способы представления пакетов. В нем описываются полностью интерпретируемые и шестнадцатеричные представления пакетов, а также порядок толкования и обращения к значениям из пакетов с помощью схем пакетов.

Имеется немало программных средств для автоматической интерпретации данных из пакетов, что дает возможность проводить анализ пакетов, даже не зная материала этого приложения. Но если уделить время изучению данных из пакетов и их структуры, то можно лучше понять, каким образом пакеты автоматически анализируются в таких инструментальных средствах, как Wireshark. Чем меньше уровень абстракции между вами и данными, которые вы анализируете, тем лучше.

Представление пакетов

Пакеты могут быть представлены самыми разными способами для их последующей интерпретации. Исходные данные из пакета могут быть представлены в двоичном виде, т.е. в виде определенного сочетания единиц и нулей по основанию 2, аналогично приведенному ниже.

```
01100000010100110101110000001010110000010000000000
01000000000000001000110000010
11010101101110000000000000000000000000000000000000
00000100000010000000000001011
01101000000000000010000001100000000000000000000000
10000000100000000010000000010
```

Двоичные числа представляют цифровую информацию на самом низком из возможных уровней, где **1** обозначает наличие электрического сигнала, а **0** – его отсутствие. Каждая цифра обозначает отдельный бит, а восемь бит – отдельный байт. Но двоичные данные крайне неудобны для чтения и интерпретации, и поэтому они обычно преобразуются в шестнадцатеричные, т.е. в определенное сочетание букв и цифр по основанию 16. Приведенный выше пакет в шестнадцатеричном виде выглядит следующим образом:

```
4500 0034 40f2 4000 8006 535c ac10 1080
4a7d 5f68 0646 0050 7c23 5ab7 0000 0000
8002 2000 0b30 0000 0204 05b4 0103 0302
0101 0402
```

В *шестнадцатеричной* системе счисления для представления отдельных значений применяются числа от **0** до **9** и буквы от **A** до **F**. Это один из самых распространенных способов представления пакетов, поскольку он краток и легко преобразуется в более элементарную двоичную форму. В шестнадцатеричном виде отдельный байт, состоящий из восьми бит, представлен двумя символами. Каждый символ в байте обозначает *полубайт* (т.е. четыре бита), причем значение слева представляет *старший полубайт*, а значение справа – *младший полубайт*. В рассматриваемом здесь примере пакета это означает, что первый байт представлен шестнадцатеричным числом **45**, где старший полубайт обозначен цифрой **4**, а младший – цифрой **5**.

Положение байтов в пакете обозначается в виде смещения относительно его начала, или нулевого смещения. Следовательно, первый байт в пакете (**45**) находится на позиции **0x00**, второй байт (**00**) – на позиции **0x01**, третий байт (**00**) – на позиции **0x02** и т.д., где **0x** обозначает используемое шестнадцатеричное представление. Чтобы обозначить положение, охватывающее больше одного байта, после двоеточия указывается количество дополнительных байтов в числовом виде. Например, положение первых четырех байтов (**4500 0034**) из рассматриваемого здесь примера пакета обозначается как **0x00:4**. Приведенное здесь пояснение понадобится далее в разделе "Интерпретация неизвестного пакета", где описывается порядок интерпретации неизвестных протоколов по схемам пакетов.

ПРИМЕЧАНИЕ *Когда люди пытаются анализировать фрагменты пакетов, то нередко совершают ошибку, забывая начинать отсчет с нуля. Привыкнуть к этому действительно нелегко, поскольку большинство людей приучены начинать отсчет с единицы. И хотя я уже давно занимаюсь разбором пакетов, тем не менее, совершаю подобную ошибку до сих пор. И самый лучший совет, который я мог бы дать в этом отношении, – считать на пальцах. На первый взгляд такой совет кажется глупым, но на самом деле стыдиться считать на пальцах не стоит, особенно если это помогает найти правильный ответ.*

На более высоком уровне инструментальное средство вроде Wireshark может представить пакет в полностью интерпретированном виде, используя рассматриваемый далее интерпретатор протоколов. Так, рассмотренный ранее пакет может быть представлен в Wireshark полностью интерпретированным, как показано на рис. Б.1.

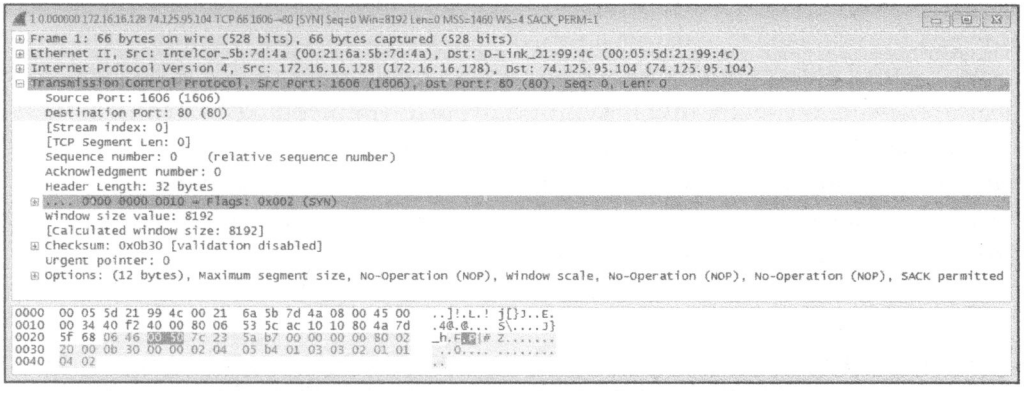


Рис. Б.1. Пакет, полностью интерпретированный в Wireshark

Сведения о пакете отображаются в Wireshark с помощью описательных меток. Пакеты не содержат подобные метки, но их данные преобразуются в конкретный формат, указанный в стандарте на сетевой протокол. Полная интерпретация пакета означает чтение данных по стандарту на протокол и их разделение на размечаемый описательными метками, удобочитаемый текст.

Wireshark и аналогичные инструментальные средства способны полностью интерпретировать данные из пакетов, поскольку в них встроены интерпретаторы протоколов, определяющие положение, длину и значения в отдельных полях заголовков сетевых протоколов. Например, пакет, приведенный на рис. Б.1, может быть разделен на четыре части согласно протоколу TCP, где отдельные поля и их значения обозначены описательными метками. Так, меткой **`Source Port`** отмечено поле, где находится десятичное значение **`1606`**,

обозначающее порт отправителя. Благодаря этому упрощается поиск информации, которая требуется для проведения анализа. И всякий раз, когда появляется такая возможность, она используется как самый эффективный способ достижения цели, поставленной при анализе пакетов.

И хотя в состав Wireshark входят буквально тысячи интерпретаторов протоколов, в ходе анализа перехваченных пакетов могут встретиться такие протоколы, интерпретировать которые в Wireshark невозможно, поскольку они неизвестны. Такое нередко случается с протоколами, используемыми отдельными производителями оборудования, которые не находят широкого применения, а также со специальными протоколами, используемые во вредоносном программном обеспечении. В подобных случаях пакеты оказываются интерпретированными частично. Именно поэтому в нижней части экрана Wireshark исходные данные из пакета по умолчанию отображаются в шестнадцатеричном виде (см. рис. Б.1).

Чаще всего программы, работающие в режиме командной строки (например, утилита `tcpdump`), не обладают столь большим разнообразием интерпретаторов протоколов и поэтому в них исходные данные из пакетов отображаются в шестнадцатеричном виде. Это особенно справедливо для более сложных протоколов уровня приложений, которые труднее поддаются синтаксическому анализу. Следовательно, частично интерпретированные пакеты считаются нормой, когда применяются подобные инструментальные средства. Характерный пример применения утилиты `tcpdump` для интерпретации данных из пакета приведен на рис. Б.2.

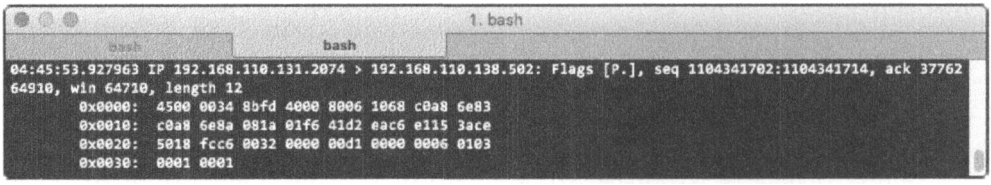


Рис. Б.2. Пакеты, частично интерпретированные в `tcpdump`

Работая с частично интерпретированными пакетами, приходится прибегать к изучению структуры заголовков пакета на более фундаментальном уровне. Wireshark, `tcpdump` и большинство других инструментальных средств допускают такую возможность, отображая исходные данные из пакета в шестнадцатеричной форме.

Применение схем пакетов

Как пояснялось в главе 1, "Анализ пакетов и основы организации сетей", в пакете представлены данные, отформатированные согласно стандартам

сетевых протоколов. Но поскольку в наиболее употребительных протоколах данные форматируются в пакетах особым образом, чтобы аппаратные и программные средства могли интерпретировать эти данные, в пакетах должны точно соблюдаться правила форматирования. Такое форматирование можно обозначить и пользоваться им для интерпретации данных из пакетов с помощью схем пакетов. *Схема пакета* – это графическое представление пакета, которое позволяет аналитику сопоставлять байты из пакета с полями, применяемыми в любом заданном протоколе. В схеме, полученной из документа RFC, в котором приведена спецификация протокола, отображаются поля, присутствующие в протоколе, их длина и порядок следования.

Рассмотрим в качестве примера схему пакета протокола IPv4, упоминавшуюся в главе 7, "Протоколы сетевого уровня", и представленную в качестве напоминания на рис. Б.3.

Межсетевой протокол версии 4 (IPv4)							
Смещение	Октет	0		1	2		3
Октет	Бит	0–3	4–7	8–15	16–18	19–23	24–31
0	0	Версия	Длина заголовка	Тип службы	Общая длина		
4	32	Идентификация			Флаги	Смещение фрагмента	
8	64	Время жизни		Протокол	Контрольная сумма заголовка		
12	96	IP-адрес отправителя					
16	128	IP-адрес получателя					
20	160	Параметры					
24+	192+	Данные					

Рис. Б.3. Схема пакета протокола IPv4

На этой схеме по горизонтали обозначены отдельные двоичные разряды, или биты, пронумерованные от **0** до **31**. В свою очередь, биты сгруппированы в 8-разрядные байты, пронумерованные от **0** до **3**. А по вертикали отдельные биты и байты обозначены рядами с описательными метками, причем каждый ряд соответствует одной из 32-разрядных (или 4-байтовых) частей пакета. Для определения положения конкретного поля, нужно сначала определить смещение его 4-байтовой части по вертикали, в которой находится это поле, а затем прибавить к этому значению горизонтальное смещение. В первом ряду расположены первые четыре байта пакета (смещения **0–3**, обозначенные соответствующими метками по горизонтали). Во втором ряду расположены следующие четыре байта (смещения **4–7**); их смещения также могут быть отсчитаны по горизонтали. В данном случае отсчет начинается с байта 4, который соответствует нулевому смещению по горизонтали. Затем следует байт 5, т.е. первый байт по горизонтали, и т.д.

Например, по схеме пакета можно определить, что для протокола IPv4 в байте со смещением **0x01** расположено поле **Type of Service** (Тип службы). Для его поиска нам нужно взять первый ряд (ему соответствует нулевое смещение в пакете) и перейти от нулевого к первому байту этого ряда. В качестве еще одного примера рассмотрим байт со смещением **0x08**, которому соответствует поле **Time to Live** (Время жизни). По вертикали можно определить, что байт со смещением 8 оказывается в третьем ряду сверху (там же находятся байты со смещением 8–11). А по горизонтали восьмому байту пакета, соответствует нулевое смещение в третьем ряду. Поскольку байт 8 расположен первым в третьем ряду, то он оказывается в нулевом столбце по горизонтали и ему соответствует поле **Time to Live**.

Некоторые поля, в том числе поле **Source IP Address** (IP-адрес отправителя), занимают несколько байт, как следует из обозначения **0x0C:4** (в десятичной форме **12:4**), а часть полей могут быть разделены на полубайты. Примером тому служит байт со смещением **0x00**, в старшем полубайте которого расположено поле **Version**, а в младшем полубайте — поле **Header Length** (Длина заголовка). В байте со смещением **0x06** содержится еще больше информации, поскольку отдельные его биты служат для представления конкретных полей. Если в поле находится единственное двоичное значение, то оно зачастую называется *флагом*. Примерами тому служат поля **Reserved** (Зарезервировано), **Don't fragment** (Не фрагментировать) и **More Fragments** (Дополнительные фрагменты) в заголовке протокола IPv4. Флаг может принимать только двоичное значение **1** (истина) или **0** (ложь), а следовательно, он считается "установленным", если его значение равно **1**. Конкретное значение, соответствующее установленному флагу, зависит от спецификации сетевого протокола и полей в его заголовке.

Рассмотрим еще один пример схемы, приведенной на рис. Б.4. Мы ее уже рассматривали в главе 8, "Протоколы транспортного уровня".

<table>
<tr><th colspan="7">Протокол управления передачей (TCP)</th></tr>
<tr><th>Смещение</th><th>Октет</th><th colspan="2">0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><th>Октет</th><th>Бит</th><th>0–3</th><th>4–7</th><th>8–15</th><th>16–23</th><th>24–31</th></tr>
<tr><td>0</td><td>0</td><td colspan="3">Порт отправителя</td><td colspan="2">Порт получателя</td></tr>
<tr><td>4</td><td>32</td><td colspan="5">Порядковый номер</td></tr>
<tr><td>8</td><td>64</td><td colspan="5">Номер подтверждения</td></tr>
<tr><td>12</td><td>96</td><td>Смещение данных</td><td>Зарезервировано</td><td>Флаги</td><td colspan="2">Размер окна</td></tr>
<tr><td>16</td><td>128</td><td colspan="3">Контрольная сумма</td><td colspan="2">Указатель срочности</td></tr>
<tr><td>20+</td><td>160+</td><td colspan="5">Параметры</td></tr>
</table>

Рис. Б.4. Схема пакета протокола TCP

На этой схеме показана структура заголовка пакета протокола TCP. Глядя на нее, можно найти ответы на многие вопросы о пакете TCP, даже не зная назначения протокола TCP. Рассмотрим в качестве примера заголовок пакета TCP, приведенный ниже в шестнадцатеричной форме.

```
0646 0050 7c23 5ab7 0000 0000 8002 2000
0b30 0000 0204 05b4 0103 0302 0101 0402
```

Используя схему пакета можно идентифицировать и интерпретировать отдельные поля из этого заголовка. В частности, можно определить следующее.

- Номер порта отправителя указан в позиции **0x00:2** и имеет шестнадцатеричное значение **0x0646** (десятичное значение **1606**).
- Номер порта получателя указан в позиции **0x02:2** и имеет шестнадцатеричное значение **0x0050** (десятичное значение **80**).
- Длина заголовка указана в поле **Data Offset** (Смещение данных), а точнее – в старшем полубайте со смещением **12** в десятичной форме и имеет шестнадцатеричное значение **80**.

А теперь применим все описанное выше для интерпретации неизвестного пакета.

Интерпретация неизвестного пакета

Ранее на рис. Б.2 был приведен частично интерпретированный пакет. Проанализировав его проинтерпретированную часть, можно убедиться, что это пакет протокола TCP/IP, переданный из одного устройства к другому в одной и той же сети. Но, помимо этого, нельзя ничего больше узнать о переданных в нем данных. Ниже приведена полная шестнадцатеричная форма вывода этого пакета.

```
4500 0034 8bfd 4000 8006 1068 c0a8 6e83
c0a8 6e8a 081a 01f6 41d2 eac6 e115 3ace
5018 fcc6 0032 0000 00d1 0000 0006 0103
0001 0001
```

Как показывает быстрый подсчет, этот пакет состоит из 52 байт. По схеме данного пакета для IP-заголовка можно определить, что обычная длина этого заголовка составляет 20 байт. В этом можно убедиться, проанализировав значение длины заголовка в младшем полубайте, расположенном со смещением **0x00**. Из схемы пакета для TCP-заголовка следует, что его длина также составляет 20 байт, если отсутствуют дополнительные параметры, а они

действительно отсутствуют (подробнее дополнительные параметры протокола TCP рассматривались в главе 8, "Протоколы транспортного уровня"). Это означает, что первые 40 байт в приведенной выше шестнадцатеричной форме анализируемого здесь пакета относятся к данным, уже интерпретированным согласно протоколов TCP и IP. Следовательно, интерпретировать осталось лишь 12 байт, как показано ниже.

```
00d1 0000 0006 0103 0001 0001
```

Это затруднение может поставить в тупик, если не знать, как интерпретировать пакеты. Но вы уже знаете, как применять схемы пакетов к неинтерпретированным байтам. В данном случае из данных, интерпретированных по протоколу TCP, следует, что местом их назначения является порт **502**. И хотя просмотр портов, употребляемых в сетевом трафике, считается ненадежным методом "расшифровки" неинтерпретированных данных, тем не менее, он может стать неплохой отправной точкой для дальнейшей интерпретации неизвестного пакета. Достаточно произвести быстрый поиск в Google, чтобы выяснить, что порт **502** чаще всего употребляется в протоколе Modbus over TCP, который обычно применяется в сетях ICS (Industrial Control System – система управления производственными процессами). В этом можно убедиться, сравнив шестнадцатеричную форму вывода анализируемого здесь пакета со схемой пакета по протоколу Modbus over TCP, приведенной на рис. Б.5.

Протокол Modbus over TCP					
Смещение	**Октет**	0	1	2	3
Октет	**Бит**	0–7	8–15	16–23	24–31
0	0	**Идентификатор транзакции**		**Идентификатор протокола**	
4	32	**Длина**		**Идентификатор блока**	**Код функции**
8+	64+	**Переменная**			

Рис. Б.5. Схема пакета по протоколу Modbus over TCP

Эта схема пакета была составлена на основании сведений, полученных из руководства по реализации протокола Modbus, доступного по адресу `http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf`. Как следует из этой схемы пакета, заголовок пакета должен быть длиной 7 байт, включая поле **Length** (Длина), расположенное на позиции **0x04:2** относительно начала заголовка. Отсчитывая от этой позиции, мы дойдем до шестнадцатеричного значения **0006** (т.е. десятичного значения **6**), которое обозначает, что после поля **Length** должно быть 6 байт. И это именно так и есть.

Следовательно, интерпретируемые здесь данные действительно относятся к протоколу Modbus over TCP.

Сравнивая эту схему пакета с полной шестнадцатеричной формой его вывода, можно получить следующие сведения.

- Поле **Transaction Identifier** (Идентификатор транзакции) находится на позиции **0x00:2** и содержит шестнадцатеричное значение **00d1**. Это поле служит для объединения запроса в одну пару с ответом.
- Поле **Protocol Identifier** (Идентификатор протокола) находится на позиции **0x02:2** и содержит шестнадцатеричное значение **0000**. Это поле служит для обозначения протокола и его значение соответствует протоколу Modbus.
- Поле **Length** находится на позиции **0x04:2** и содержит шестнадцатеричное значение **0006**. Это поле служит для обозначения длины данных в пакете.
- Поле **Unit Identifier** (Идентификатор блока) находится на позиции **0x06** и содержит шестнадцатеричное значение **01**. Это поле служит для внутрисистемной маршрутизации.
- Поле **Function Code** (Код функции) находится на позиции **0x07** и содержит шестнадцатеричное значение **03**. Это поле служит для обозначения функции Read Holding Registers (Чтение регистров временного хранения), предназначенной для чтения значений данных из системы.
- Принимая во внимание код функции **3**, в анализируемом здесь пакете следует ожидать еще два поля данных. В частности, поля **Reference Number** (Номер ссылки) и **Word Count** (Количество слов) находятся на позиции **0x08:4**, и каждое из них содержит шестнадцатеричное значение **0001**.

Таким образом, рассматриваемый здесь неизвестный пакет может быть полностью интерпретирован в контексте протокола Modbus. Полученных в итоге сведений должно быть достаточно для дальнейшей диагностики системы, отвечающей за данный пакет. Рассмотренный здесь пример наглядно показывает, как следует подходить к интерпретации неизвестного протокола и анализу неинтерпретированного пакета, используя схемы протоколов даже в том случае, если раньше вообще не приходилось сталкиваться с такими протоколами, как Modbus.

На практике лучше всего учитывать уровень абстракции, существующий между вами и анализируемыми данными. Это помогает принимать более взвешенные и грамотные решения и позволяет работать с пакетами в самых разных ситуациях. Я сам не раз оказывался в таких ситуациях, когда для

анализа пакетов мог пользоваться только инструментальными средствами вроде tcpdump, работающими в режиме командной строки. А поскольку большинство подобных средств испытывает недостаток в анализаторах многих протоколов седьмого уровня, то умение разбирать вручную отдельные байты в интерпретируемых пакетах имеет решающее значения для успешного анализа пакетов.

ПРИМЕЧАНИЕ *Однажды мой коллега по работе помогал в разрешении чрезвычайной ситуации в среде со строгими мерами защиты. Он был допущен к просмотру данных, которые требовалось проанализировать, но не к самой системе, где они хранились. За время, которое ему было предоставлено, он мог вывести пакеты только из отдельных диалогов. И благодаря тому, что он знал структуру пакетов и порядок их интерпретации, ему удалось найти нужную информацию в выведенных данных. Безусловно, это был длительный процесс подобно студеной тягучей массе, медленно стекающей по замерзлой ветке. И хотя это крайний случай, тем не менее, он наглядно показывает, насколько важно знать и уметь пользоваться нестандартными методами анализа пакетов независимо от универсальных средств.*

По всем перечисленным выше причинам полезно уделять время разбору пакетов вручную, чтобы приобрести опыт разносторонней их интерпретации. Приобретя достаточный опыт интерпретации пакетов вручную, я распечатал несколько наиболее употребительных схем пакетов на отдельных листах, скрепил из вместе и всегда держу под рукой на своем рабочем месте. Кроме того, я храню их в цифровом виде на своем переносном и планшетном компьютерах для быстрой справки в поездке. Для большего удобства я включил несколько наиболее употребительных схем пакетов в архивный файл формата ZIP, содержащий файлы перехвата из прилагаемых к этой книге и доступный по адресу `https://www.nostarch.com/packetanalysis3/`.

Заключительные соображения

В этом приложении было показано, как интерпретировать данные из пакетов в самых разных форматах и как пользоваться схемами пакетов для разбора вручную данных, не интерпретированных в пакете. Приобретя основные знания и навыки интерпретации данных из пакетов вручную, вы не будете больше испытывать трудности при анализе пакетов, какими бы инструментальными средствами вы ни пользовались для просмотра данных из пакетов.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Б

В

Г

Д

З

И

К

М

Н

О

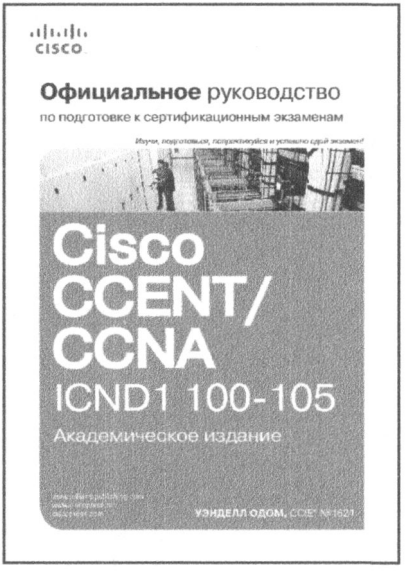
П

P

Э

ОФИЦИАЛЬНОЕ РУКОВОДСТВО CISCO
по подготовке к сертификационным экзаменам
CISCO CCENT/CCNA ICND1 100-105
АКАДЕМИЧЕСКОЕ ИЗДАНИЕ

Уэнделл Одом, CCIE® №1624



Нынешнее академическое издание — исчерпывающий справочник и учебное пособие по фундаментальным концепциям работы с сетями и вспомогательным приложениям. Автор бестселлеров и опытный преподаватель Уэнделл Одом делится советами по подготовке к экзамену, помогая выявить слабые стороны, улучшить концептуальные знания и практические навыки.

Книги этой серии являются официальным первоисточником для подготовки к экзамену. Они предоставляют теоретические и практические материалы, помогут кандидатам на сертификат Cisco Career Certification сконцентрировать усилия по изучению и повысить уверенность в себе по мере приближения дня экзамена.

Хорошо выверенный по уровню детализации, оснащенный средствами оценки, вопросами и упражнениями, этот официальный учебник поможет вам преуспеть на экзамене.

www.williamspublishing.com

ISBN 978-5-9909446-4-0 **в продаже**

НЕ СМОТРИТЕ ИЗУМЛЕННО НА КОД В ПЕРЕХВАЧЕННЫХ ПАКЕТАХ, А АНАЛИЗИРУЙТЕ ИХ!

Загрузите файлы перехвата, используемые в примерах из этой книги, по адресу https://www.nostarch.com/packetanalysis3/ или http://www.williamspublishing.com/Books/978-5-6040723-0-1.html

Проанализировать пакеты как в проводных, так и в беспроводных сетях с помощью Wireshark — самого популярного в мире сетевого анализатора — совсем не трудно. Но как на основании анализа этих пакетов выяснить, что же на самом деле происходит в сети?

Прочитав это третье издание книги, обновленное по версии Wireshark 2.x, вы научитесь разбираться в перехваченных пакетах и лучше понимать стоящие перед вами задачи диагностики сетей. В нем вы найдете дополнительное изложение сетевых протоколов IPv6 и SMTP, новую главу, посвященную эффективным анализаторам пакетов tcpdump и TShark, работающим в режиме командной строки, а также приложение, в котором поясняется, как вручную интерпретировать данные из пакетов, используя схемы пакетов.

Основные темы книги

- Проведение текущего анализа сетевого трафика в реальном времени и его активный перехват
- Составление специальных фильтров для перехвата и отображения пакетов
- Анализ пакетов для выявления и разрешения типичных проблем, возникающих в сети, включая потерю связи, медленную работу сети и решение вопросов, связанных со службой DNS
- Исследование современных наборов эксплойтов (средств эксплуатации уязвимостей) и вредоносных программ на уровне пакетов
- Извлечение файлов, пересылаемых по сети, из перехваченных пакетов
- Построение графиков из перехваченного сетевого трафика для наглядного представления потоков данных, проходящих по сети
- Использование дополнительных средств Wireshark, позволяющих разобраться в непонятных образцах перехвата сетевого трафика
- Составление статистических и прочих отчетов, помогающих лучше объяснить технические данные неспециалистам

Независимо от уровня вашей квалификации эта книга поможет научиться пользоваться популярными сетевыми анализаторами и с их помощью разбираться в состоянии любых сетей и оперативно разрешать возникающие в них проблемы.

Об авторе

Крис Сандерс — консультант, исследователь и инструктор по сетевой безопасности. Помимо этой книги, он написал книгу *Applied Network Security Monitoring* и регулярно ведет блог по адресу *http://chrissanders.org/*. Крис регулярно проводит анализ пакетов для выявления злоумышленников и вредоносного кода в сети.

Категория: Безопасность в сети
Предмет рассмотрения: Анализатор пакетов Wireshark версии 2.x
Уровень: Начальный — промежуточный

Гонорар от издания этой книги автор собирается пожертвовать в "Фонд поддержки технологий в сельской местности" (Rural Technology Fund – RTF; http://ruraltechfund.org/).

ISBN 978-5-6040723-0-1

no starch press
www.nostarch.com
Диалектика
www.williamspublishing.com