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).

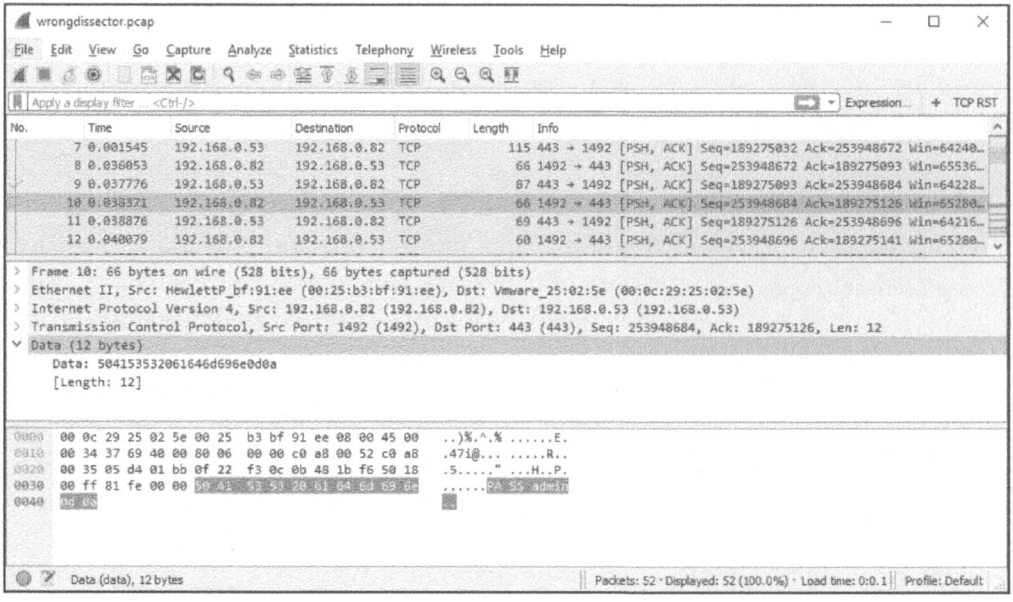

Рис. 5.11. Если имена пользователей и пароли представлены в виде открытого текста, это похоже скорее на протокол FTP, а не SSL!

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.

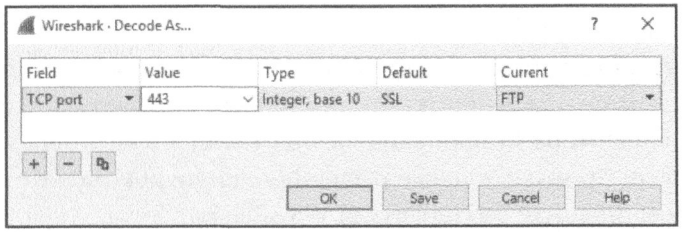

Рис. 5.12. В диалоговом окне Decode As... можно задать условия для принудительной расшифровки анализируемых пакетов

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).

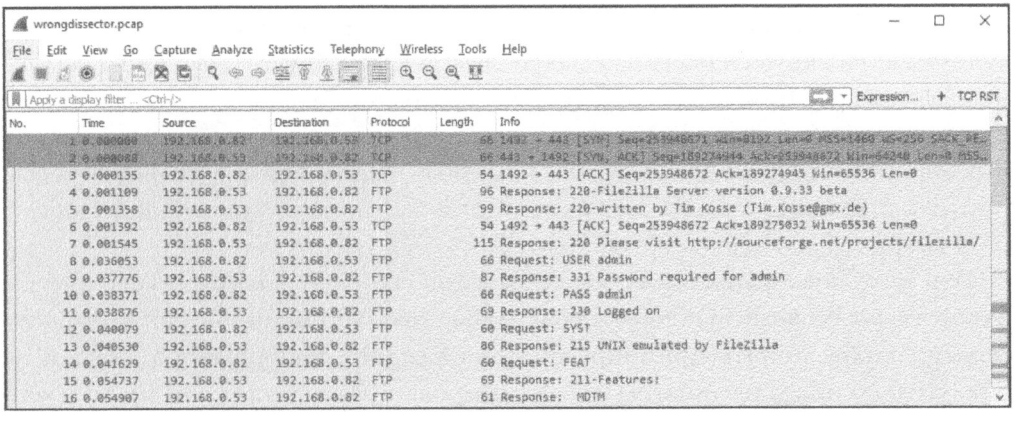

Рис. 5.13. Просмотр правильно расшифрованного трафика FTP

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.

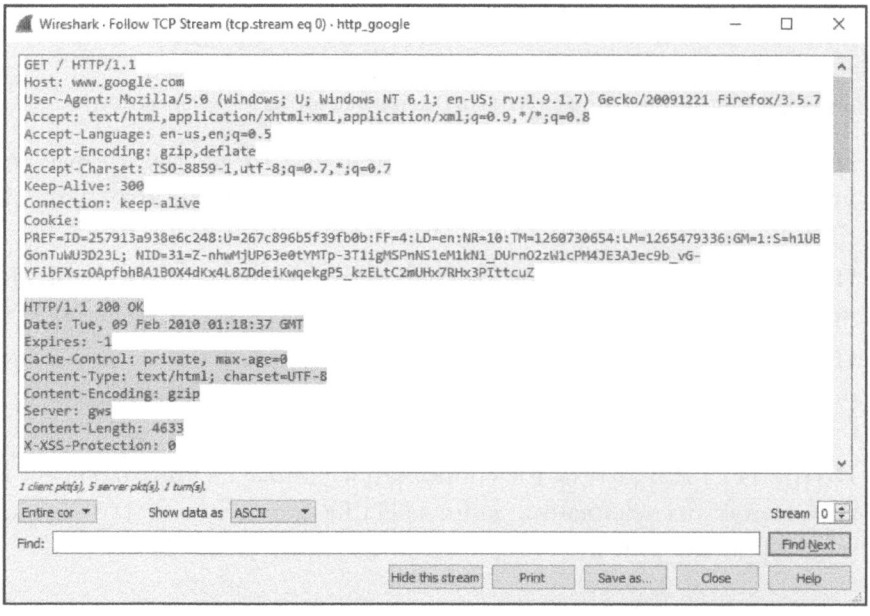

Рис. 5.14. В окне Follow TCP Stream повторно собранные передаваемые данные представлены в удобочитаемом формате

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).

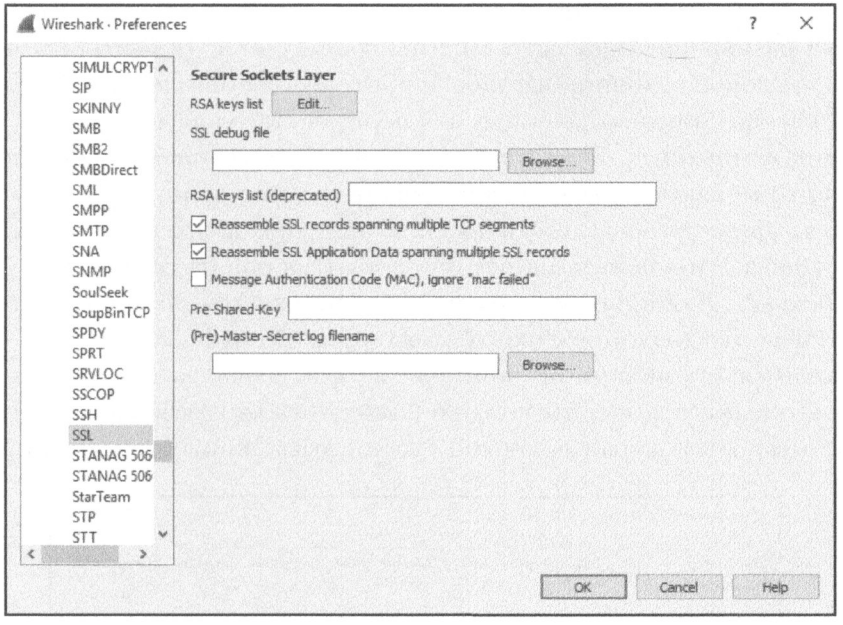

Рис. 5.15. Ввод сведений о дешифровке по протоколу SSL

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.