Лабораторная работа №2
Тема: Анализ работы протокола TCP.

Студента группы МФ-21
Чистякова Артема

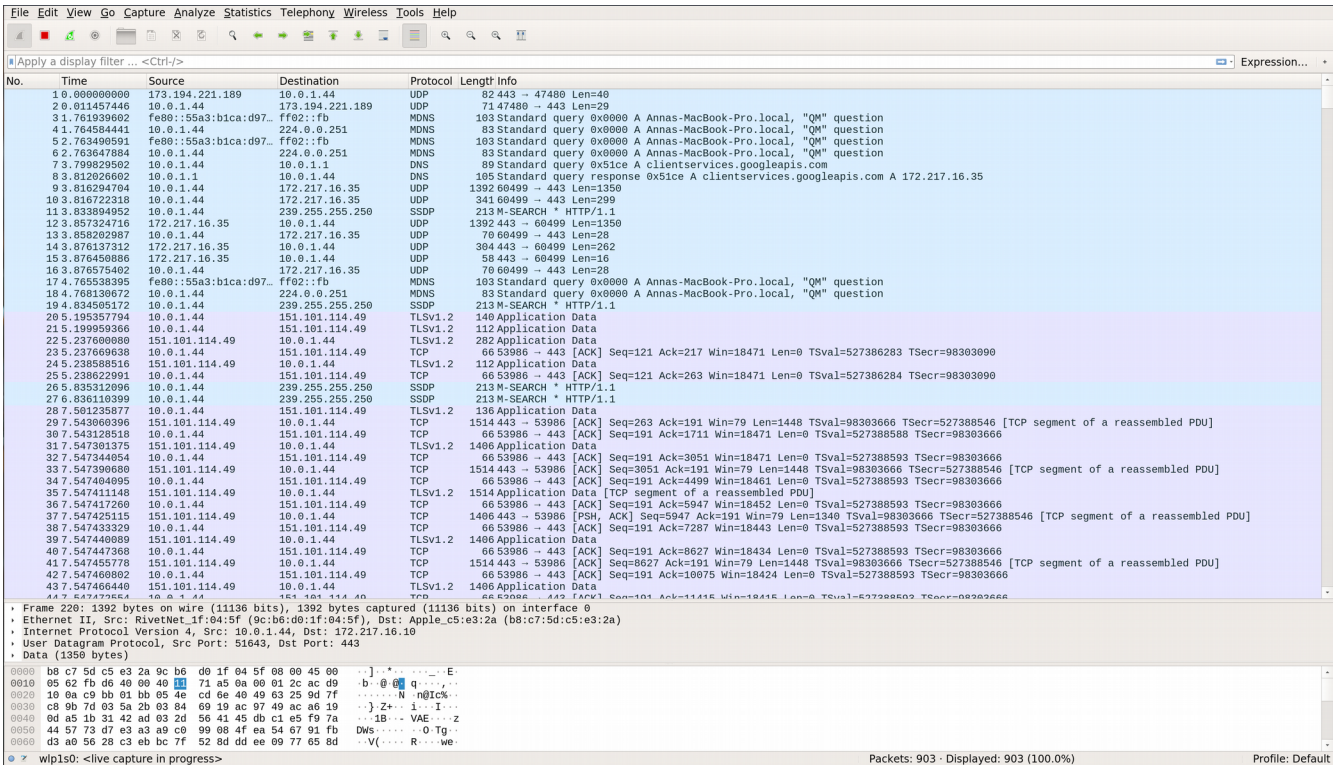
1.

Определить хосты (IP-адреса и протоколы, порты), взаимодействующие с рабочим местом.

Решение

Запустим **Wireshark**, настроим на отслеживание трафика сетевой карты: на моем компьютере – интерфейс **wlp1s0**.

Увидим следующую таблицу:



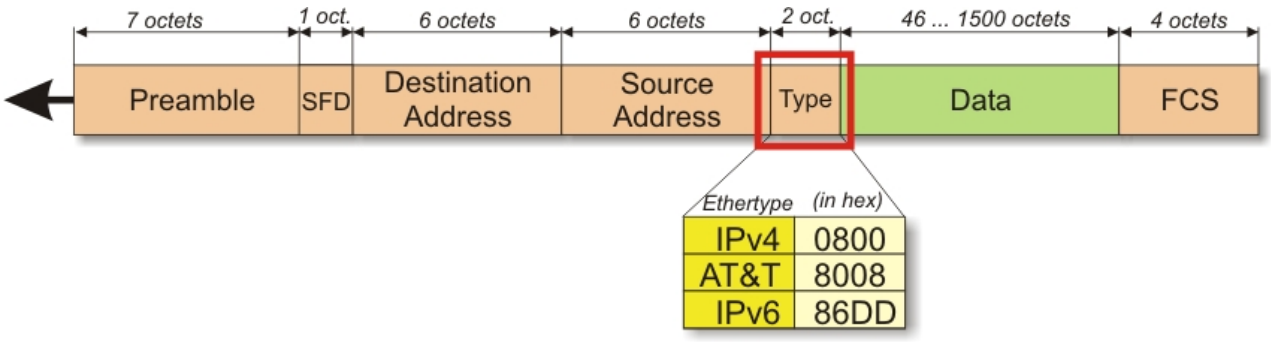
На картинке можно заметить как входящие, так и исходящие пакеты: IP компьютера – **10.0.1.44**.

2.

Определить формат Ethernet кадра.

Решение

Рассмотрим следующую иллюстрацию:



- **Поле преамбулы** состоит из семи байтов синхронизирующих данных. Каждый байт содержит одну и ту же последовательность битов - 10101010.
- **Начальный ограничитель кадра** состоит из одного байта с набором битов 10101011. Появление этой комбинации является указанием на предстоящий прием кадра.
- **Адрес получателя** – может быть длиной 2 или 6 байтов (MAC-адрес получателя). Первый бит адреса получателя - это признак того, является адрес индивидуальным или групповым: если 0, то адрес указывает на определенную станцию, если 1, то это групповой адрес нескольких (возможно всех) станций сети. При широковещательной адресации все биты поля адреса устанавливаются в 1. Общепринятым является использование 6-байтовых адресов.
- **Адрес отправителя** – 2-х или 6-ти байтовое поле, содержащее адрес станции отправителя. Первый бит - всегда имеет значение 0.
- **Поле типа протокола**. Это поле предназначено для указания типа протокола верхнего уровня, *вложившего свой пакет в поле данных этого кадра*.

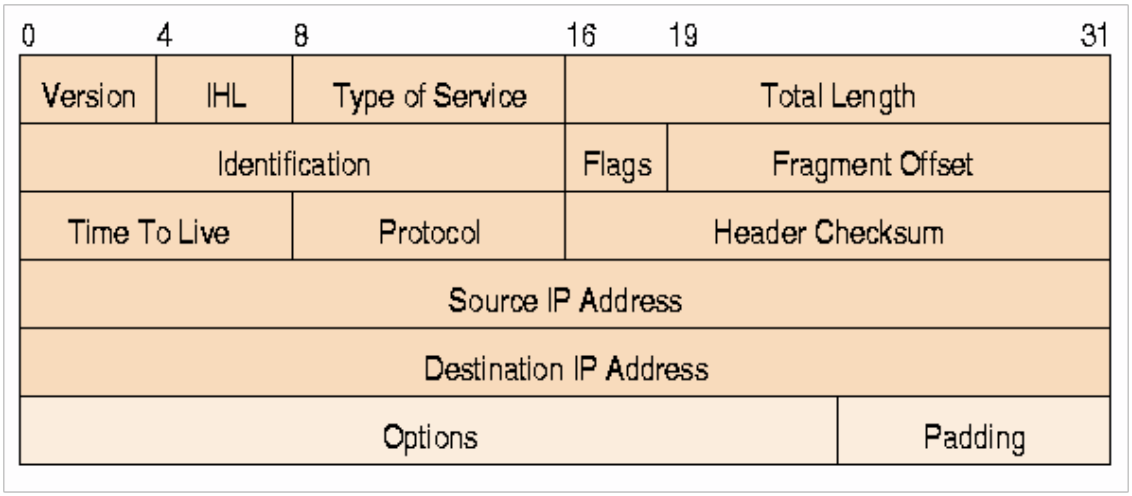
- **Поле данных** может содержать от 0 до 1500 байт. Но если длина поля меньше 46 байт, то используется следующее поле - поле заполнения, чтобы дополнить кадр до минимально допустимой длины.
- **Поле контрольной суммы** – 4 байта, содержащие значение, которое вычисляется по определенному алгоритму (полиному CRC-32). После получения кадра рабочая станция выполняет собственное вычисление контрольной суммы для этого кадра, сравнивает полученное значение со значением поля контрольной суммы и, таким образом, определяет, не искажен ли полученный кадр.

3.

Определить формат IP пакета.

Решение

Рассмотрим следующую иллюстрацию:



Протокол IP является надстройкой над протоколом Ethernet – его header вложен в поле данных протокола Ethernet.

- **Поле номера версии** занимает 4 бита и идентифицирует версию протокола IP. Сейчас повсеместно используется версия 4 (IPv4), хотя все чаще встречается и новая версия (IPv6).
- **Значение длины заголовка** IP-пакета занимает 4 бита и измеряется в 32-битных словах. Обычно заголовок имеет длину в 20 байт (пять 32-битных слов), но при добавлении некоторой служебной информации это значение может быть увеличено. Наибольшая длина заголовка составляет 60 байт.
- **Поле типа сервиса** (Type of Service, ToS) служит одной цели – хранению признаков, которые отражают требования к качеству обслуживания пакета. Первые три бита содержат значение приоритета пакета: от самого низкого – 0 до самого высокого – 7. Следующие три бита поля ToS определяют критерий выбора маршрута. Если бит D установлен в 1, то маршрут должен выбираться для минимизации задержки доставки данного пакета, установленный бит T – для максимизации пропускной способности, а бит R – для максимизации надежности доставки. Оставшиеся два бита имеют нулевое значение.
- **Поле общей длины** занимает 2 байта и характеризует общую длину пакета с учетом заголовка и поля данных. Если это кадры Ethernet, то выбираются пакеты с максимальной длиной 1500 байт, умещающиеся в поле данных кадра Ethernet. В стандартах TCP/IP предусматривается, что все хосты должны быть готовы принимать пакеты длиной вплоть до 576 байт.
- **Идентификатор пакета** занимает 2 байта и используется для распознавания пакетов, образовавшихся путем деления на части исходного пакета. Все части одного пакета должны иметь одинаковое значение этого поля.
- **Флаги** занимают 3 бита и содержат признаки, связанные с фрагментацией. Установленный в 1 бит DF запрещает маршрутизатору фрагментировать данный пакет, а установленный в 1 бит MF говорит о том, что данный пакет является промежуточным (не последним) фрагментом. Оставшийся бит зарезервирован.
- **Поле смещения фрагмента** занимает 13 бит и задает смещение в байтах поля данных этого фрагмента относительно начала поля данных исходного

пакета. Используется при сборке/разборке фрагментов пакетов. Смещение должно быть кратно 8 байт.

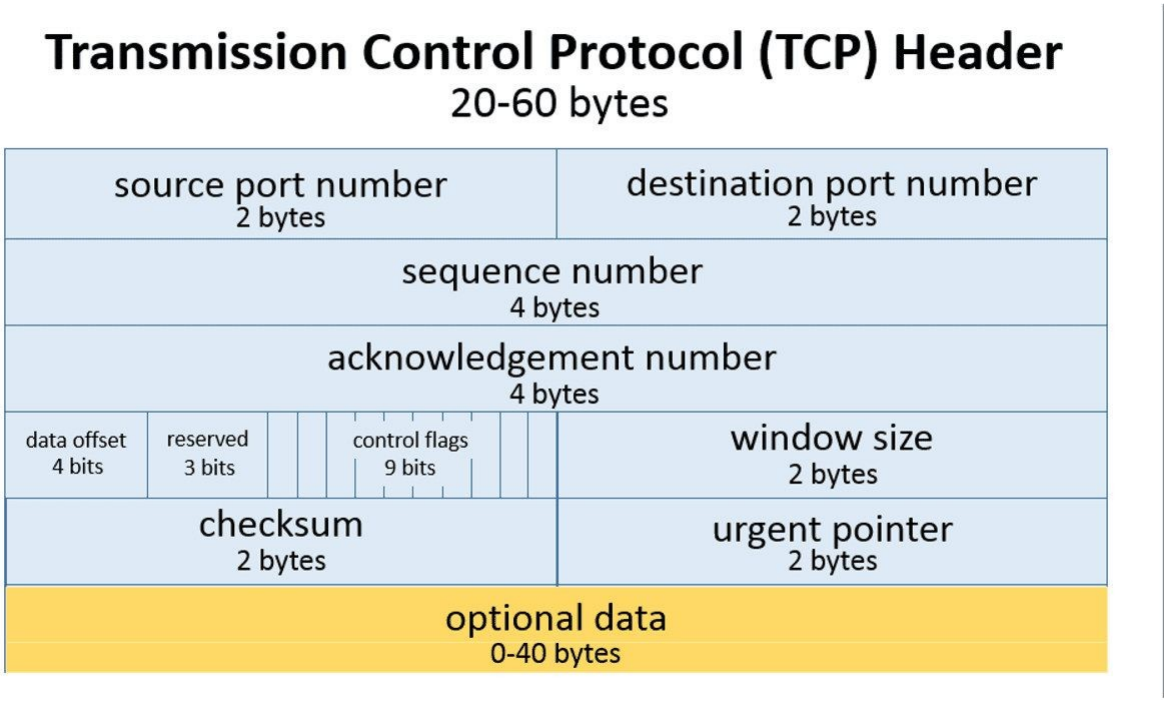
- **Поле времени жизни** (Time To Live, TTL) занимает один байт и используется для задания предельного срока, в течение которого пакет может перемещаться по сети. Время жизни можно интерпретировать как максимальное число транзитных узлов, которые разрешено пройти пакету.
- **Поле протокола верхнего уровня** занимает один байт и содержит идентификатор, указывающий, какому протоколу верхнего уровня принадлежит информация, размещенная в поле данных пакета. Например, 6 означает, что в пакете находится сообщение протокола TCP, 17 – протокола UDP.
- **Контрольная сумма** заголовка занимает 2 байта (16 бит) и рассчитывается только по заголовку. Контрольная сумма проверяется и повторно рассчитывается на каждом маршрутизаторе и конечном узле. Если контрольная сумма неверна, то пакет отбрасывается.
- **Поля IP-адресов** источника и приемника имеют одинаковую длину — 32 бита.
- **Поле параметров** является необязательным и используется обычно только при отладке сети. В этом поле можно указывать точный маршрут, регистрировать проходимые пакетом маршрутизаторы, помещать данные системы безопасности или временные отметки. В конце заголовка должно быть добавлено несколько нулевых байтов для выравнивания заголовка пакета по 32-битной границе.

4.

Определить формат TCP пакета.

Решение

Рассмотрим следующую иллюстрацию:



Протокол TCP является надстройкой над протоколов IP, его header располагается сразу после header`a IP, в том же поле данных протокола Ethernet. Он гарантирует порядок и доставку отправленных данных.

- **Порт источника и порт приемника** – 16-битовые поля, содержащие номера портов, соответственно, источника и адресата TCP-пакета.
- **Номер в последовательности** – 32-битовое поле, содержимое которого определяет «номер TCP пакета данных» внутри исходящего потока данных, существующего в рамках текущего логического соединения.

- **Номер подтверждения** – 32-битовое поле, содержимое которого определяет количество принятых TCP пакетов данных из входящего потока.
- **Смещение данных –** четырехбитовое поле, содержащее длину заголовка TCP-пакета в 32-битовых словах и используемое для определения начала расположения данных в TCP-пакете.
- **Флаг URG** – бит, установленное в 1 значение которого означает, что TCP-пакет содержит важные (urgent) данные.
- **Флаг ACK –** бит, значение которого означает, что TCP-пакет содержит в поле "номер подтверждения" верные данные.
- **Флаг PSH –** бит, значение которого означает, что данные содержащиеся в TCP-пакете должны быть немедленно переданы прикладной программе, для которой они адресованы. Подтверждение для TCP-пакета, содержащего единичное значение во флаге PSH, означает, что и все предыдущие TCP-пакеты достигли адресата.
- **Флаг RST** – бит в TCP-пакете, отправляемом в ответ на получение неверного TCP-пакета. Также может означать запрос на переустановление логического соединения.
- **Флаг SYN** – бит, означающий, что TCP-пакет представляет собой запрос на установление логического соединения. Получение пакета с установленым флагом SYN должно быть подтверждено принимающей стороной.
- **Флаг FIN** – бит, означающий, что TCP-пакет представляет собой запрос на закрытие логического соединения и является признаком конца потока данных, передаваемых в этом направлении. Получение пакета с установленым флагом FIN должно быть подтверждено принимающей стороной.
- **Размер окна** – 16-битовое поле, содержащее количество байт информации, которое может принять в свои внутренние буфера TCP-модуль. Определение оптимального размера окна - одна из наиболее сложных задач реализации протокола TCP.
- **Контрольная сумма –** 16-битовое поле, содержащее Internet-контрольную сумму, подсчитанную для TCP-заголовка, данных пакета и псевдозаголовка.

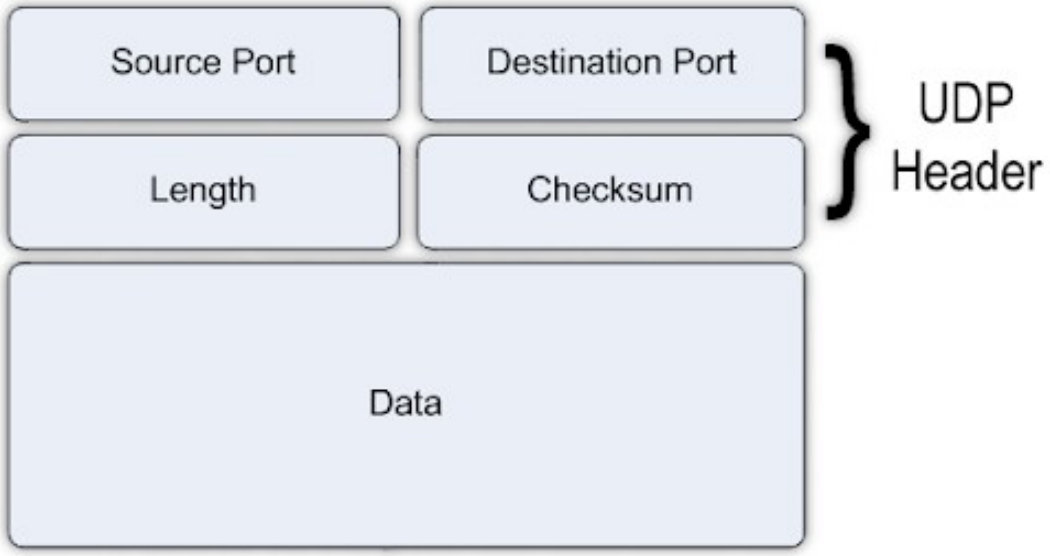
- **Псевдозаголовок** состоит из IP-адреса отправителя, IP-адреса получателя , типа протокола и длины TCP-пакета. Он предназначен для «страховки» неправильной маршрутизации TCP-пакета.

5.

Определить формат UDP пакета.

Решение

Рассмотрим следующую иллюстрацию:



Протокол UPD является надстройкой над протоколов IP, его header располагается сразу после header`а IP, в том же поле данных протокола Ethernet. В отличии от протокола TCP, он не гарантирует доставку и порядок отправляемых данных.

- **Порт отправителя** – поле может содержать номер порта, с которого был отправлен пакет, когда это имеет значение (например отправитель ожидает ответа). Если это поле не используется, оно заполняется нулями.
- **Порт назначения** – это порт компьютера, на который пакет будет доставлен.

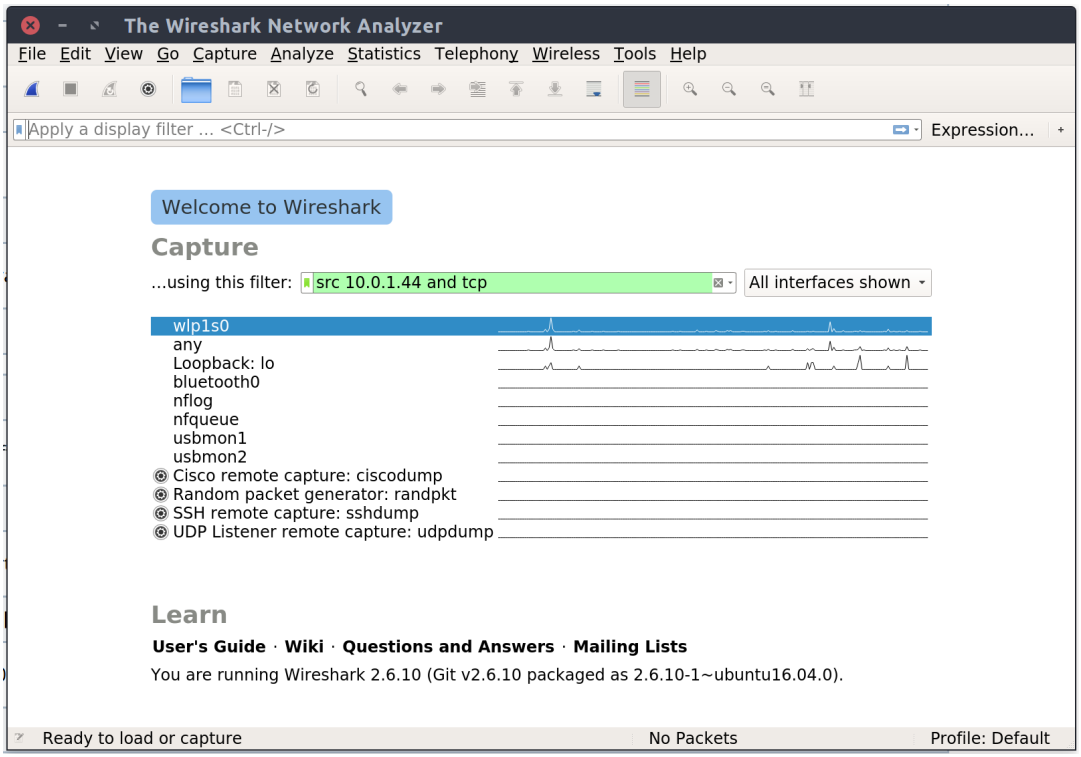
- **Поле длины** – длина (в байтах) этой дейтаграммы, включая заголовок и данные. (Минимальное значение этого поля равно 8).
- **Поле контрольной суммы –** контрольная сумма UDP-пакета, представляющая собой побитное дополнение 16-битной суммы 16-битных слов (аналогично TCP). В вычислении участвуют: данные пакета, заголовок UDP-пакета, псевдозаголовок, поля выравнивания по 16-битной границе.

6.

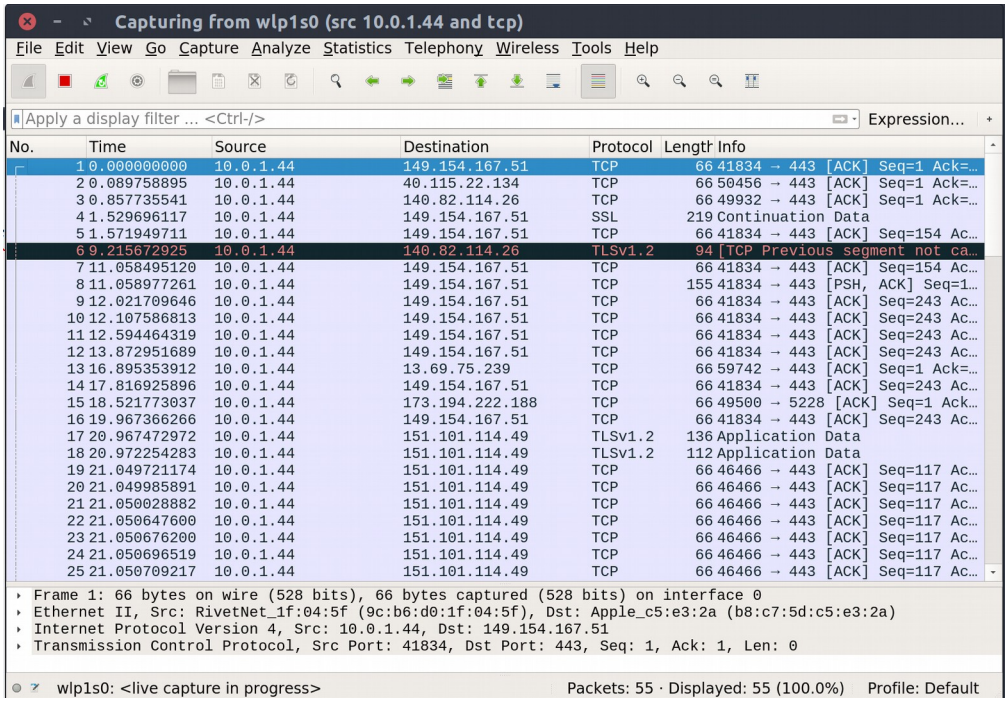
Настроить анализатор протоколов на перехват только исходящего TCP трафика на рабочем месте.

Решение

Настроим фильтр пакетов следующим образом:



Запустим sniffer.



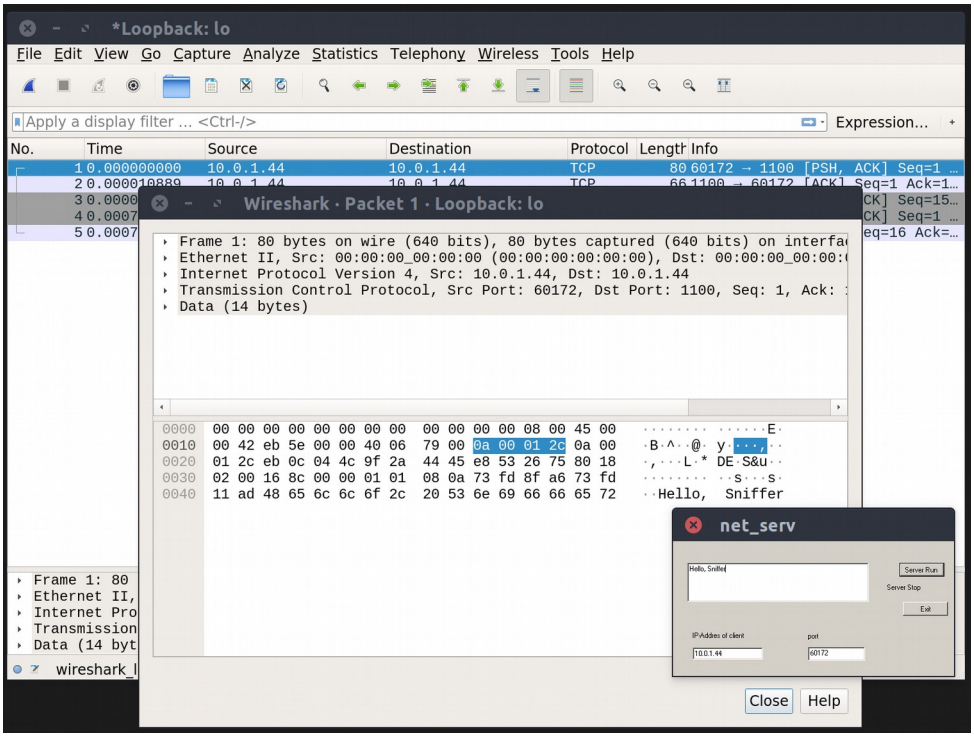
7.

Настроить анализатор протоколов на перехват всего входящего трафика на сервер.

Решение

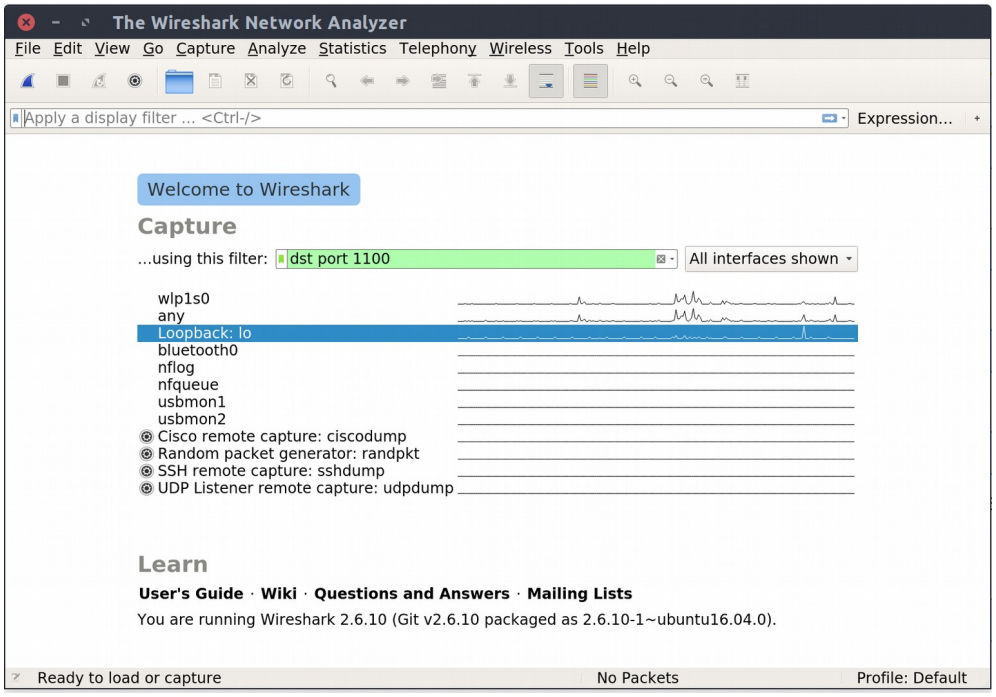
Для решения данной задачи нужно запустить sniffer для отслеживания всех входящих пакетов на рабочую машину, найти в них пакет, который отправил клиент и принял сервер и извлечь из него номер порта, на котором запущен сервер.

Запустим sniffer и найдем соответствующий пакет:

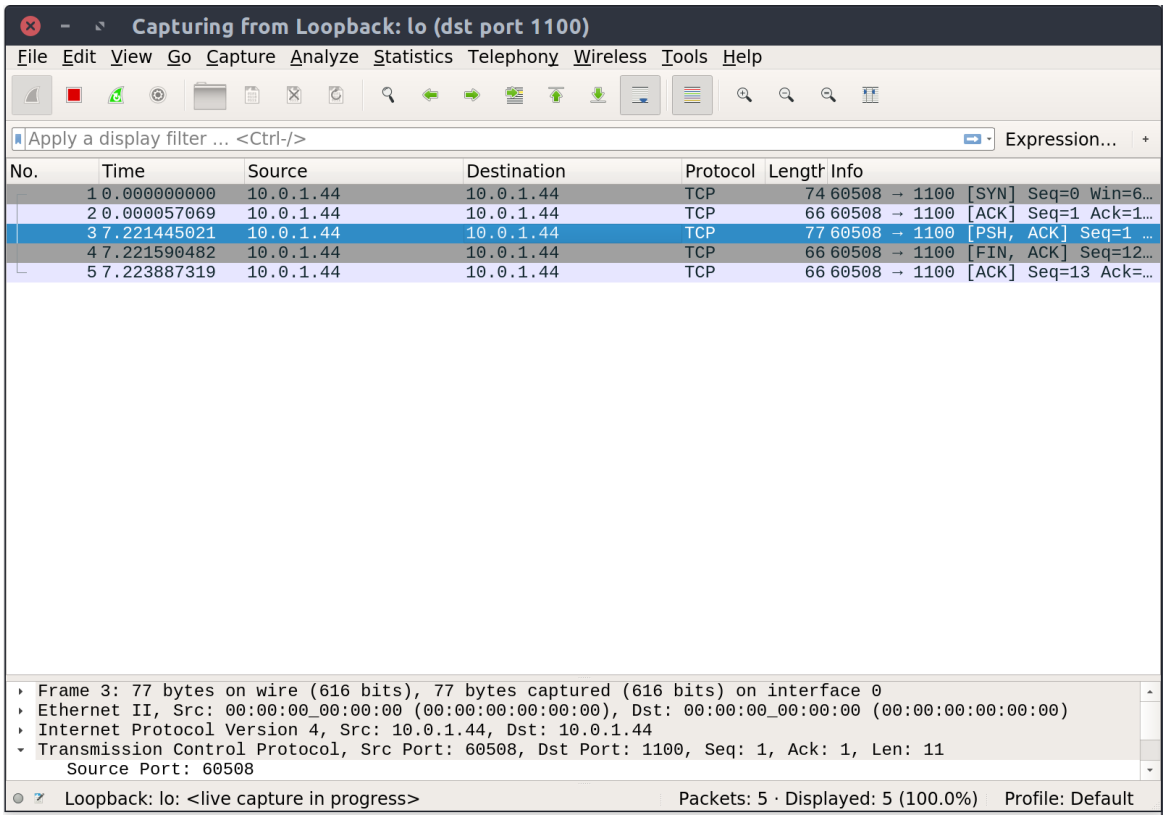


Убедимся, что данные соответствуют передаваемому сообщению. Найдем поле **Dst port = 1100**. Этот порт соответствует порту, на котором запущен сервер.

Настроим Wireshark на прослушивание порта **1100**:



Результат прослушивания порта **1100** при отправке клиентом сообщения:



8.

Алгоритм установления соединения.

Решение

Когда мы хотим установить соединение, мы отправляем удаленной системе пакет следующей структуры:

Client --- SYN (856779) --- Host

Где Client- это мы, а Host - это удаленная система. Мы посылаем пакет лишь с указанием SYN - это значит, что этот пакет первый.

SYN образуется от первоначального номера очереди (ISN) - это 32-битный номер от 1 до 4294967295 (2 в 32-ой степени). ISN при перезагрузке системы равен 1, затем каждую секунду он увеличивается на 128000 (строго говоря изменение происходит каждые 4 микросекунды) + при каждом установленном соединении он увеличивается на 64000. Получается, что цикл уникальности ISN, при условии того, что никакие соединения не устанавливались, составляет примерно 4,55 часа. Поскольку ни один пакет так долго по сети не путешествует, мы можем полагать, что SYN будет абсолютно уникальным.

Получив наш пакет, удаленная система отвечает, что получила и готова установить соединение. Данные пакета выглядит так:

Host --- SYN (758684758) и ACK (856780) --- Client

Удаленная система посылает нам ACK с номером "наш SYN+1". В добавок к этому свой SYN (мы же тоже будем отвечать). А ответ наш будет такой:

Client --- SYN (856780) и ACK (758684759) --- Host

Пакет означает следующее: ваш пакет с SYN (758684758) получен, соединение установлено, наш SYN равен 856780.

Эту процедуру называют "трехкратным рукопожатием". Первые два этапа необходимы для синхронизации SYN систем, а третий - подтверждение того, что синхронизация произошла.

9.

Алгоритм передачи данных.

Решение

Модуль TCP обеспечивает защиту от повреждения, потери, дублирования и нарушения очередности получения данных.

Для выполнения этих задач все октеты в потоке данных сквозным образом пронумерованы в возрастающем порядке. Заголовок каждого сегмента содержит число октетов данных в сегменте и порядковый номер первого октета той части потока данных, которая пересылается в данном сегменте. Например, если в сегменте пересылаются октеты с номерами от 2001 до 3000, то номер первого октета в данном сегменте равен 2001, а число октетов равно 1000.

Также для каждого сегмента вычисляется контрольная сумма, позволяющая обнаружить повреждение данных.

При удачном приеме октета данных принимающий модуль посылает отправителю подтверждение о приеме - номер удачно принятого октета. Если в течение некоторого времени отправитель не получит подтверждения, считается, что октет не дошел или был поврежден, и он посылается снова. В действительности подтверждение посылается не для одного октета, а для некоторого числа последовательных октетов.

Нумерация октетов используется также для упорядочения данных в порядке очередности и обнаружения дубликатов (которые могут быть посланы из-за большой задержки при передаче подтверждения или потери подтверждения).

10.

Алгоритм завершения соединения.

Решение

Алгоритм завершение соединения аналогичен алгоритму установки соединения, но вместо флага SYN устанавливается флаг FIN, разрыв соединения можно рассмотреть в три этапа:

- Посылка серверу от клиента флага FIN на завершение соединения.
- Сервер посылает клиенту флаги ответа ACK , FIN, что соединение закрыто.
- После получения этих флагов клиент закрывает соединение и в подтверждение отправляет серверу ACK , что соединение закрыто.