Частное учреждение образования

Колледж бизнеса и права

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_\_Е.В.Фалей  «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_2016 |

|  |  |
| --- | --- |
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Защита компьютерной информации» |
| Составлена на основании учебной программы, утвержденной директором Колледжа бизнеса и права 02.02.2016 | |

1. Лабораторная работа №16

**Инструкционно-технологическая карта**

Тема:Изучение функциональных возможностей программы-анализатора сетевого трафика Wireshark

Цель: Научиться пользоваться основными функциональными возможностями программы Wireshark

Время выполнения: 2 часа

СОДЕРЖАНИЕ РАБОТЫ

1. Теоретические сведения для выполнения работы
2. Порядок выполнения работы
3. Контрольные вопросы (вопросы для проверки изученного материала)
4. Литература
5. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ

Wireshark — это достаточно известный инструмент для захвата и анализа сетевого трафика, фактически стандарт как для образования, так и для траблшутинга.

Wireshark работает с подавляющим большинством известных протоколов, имеет понятный и логичный графический интерфейс на основе GTK+ и мощнейшую систему фильтров.

Кроссплатформенный, работает в таких ОС как Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Mac OS X, и, естественно, Windows. Распространяется под лицензией GNU GPL v2. Доступен бесплатно на сайте wireshark.org.

Зачем вообще нужны анализаторы пакетов?

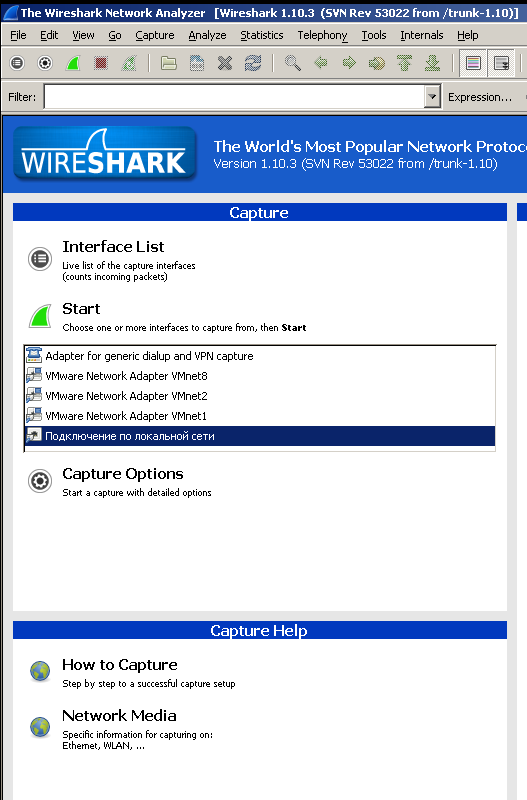
Для того чтобы проводить исследования сетевых приложений и протоколов, а также, чтобы находить проблемы в работе сети, и, что важно, выяснять причины этих проблем.

Вполне очевидно, что для того чтобы максимально эффективно использовать снифферы или анализаторы трафика, необходимы хотя бы общие знания и понимания работы сетей и сетевых протоколов.

Во многих странах использование сниффера без явного на то разрешения приравнивается к преступлению.

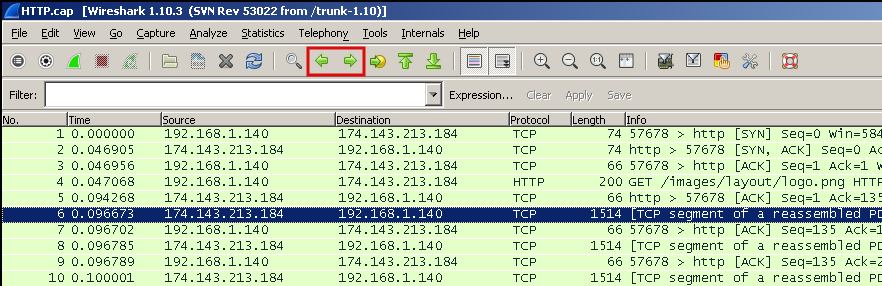
**Начинаем плаванье**

Для начала захвата достаточно выбрать свой сетевой интерфейс и нажать Start.



После чего и начнется процесс захвата, причем прилетевшие пакеты будут появляться в реальном времени.

В процессе рассмотрения и изучения пакетов бывают ситуации, когда нужно вернуться предыдущему пакету. Для этого есть две кнопки (см скриншот).



А следующая за ними кнопка позволяет сделать быстрый переход к пакету, указав его номер.

В случае если колонки перекрываются и наползают друг на друга, можно кликнуть по такой колонке правой кнопкой мыши и выбрать “Resize Column”.

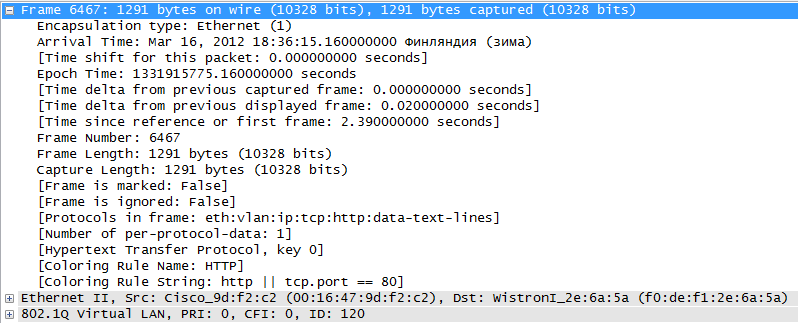
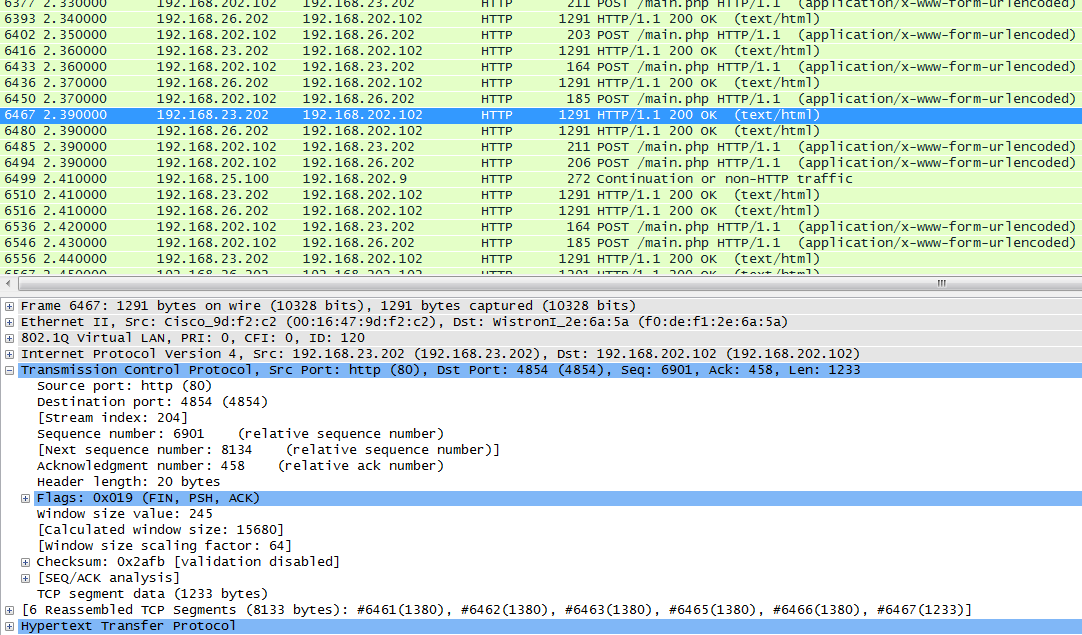
Произойдет автоматическая подгонка размеров под текущую ситуацию.

И кроме того, есть кнопка “Resize all Columns”, которая приведет в порядок все колонки.

Используя меню View – Time Display Format, можно, например, настроить, чтобы отсчет времени шел не с начала захвата, а с момента получения предыдущего пакета (Since Previous Captured Packet).

Самое важное в каждой программе (Help – About Wireshark) покажет не только версию и список авторов, но и содержит закладку Folders, которая покажет пути размещения каталогов с конфигурациями.

Изучая интерфейс, можно выбрать, например, пакет http, и увидеть, что HTTP инкапсулируется в TCP (транспортный уровень), TCP инкапсулируется в IP (сетевой уровень), а IP в свою очередь инкапсулируется в Ethernet (перед этим даже мелькает 802.1Q).

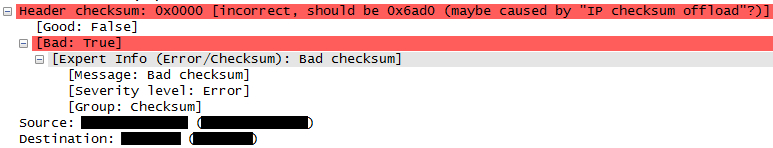
И на самом верху идет нечто вроде небольшого обзора собранной информации о кадре.

Про фильтры мы поговорим дальше, а на данном этапе, если нужно быстро отфильтровать лишние пакеты, достаточно сделать правый клик на пакете, выбрать меню Apply as Filter – Not selected и изменения сразу же вступят в силу.

Если нужно еще что-то убрать, то в следующий раз выбирать “and not Selected”, и новое правило просто добавится к фильтру.

**Убираем заусенцы**

Довольно часто при работе с Wireshark возникает ошибка IP checksum offload – ошибка контрольной суммы заголовка IP пакета.

Современные сетевые карты насколько умные, что сами считают контрольную сумму, зачем это делать на уровне стека TCP/IP программно, если можно делать хардварно.

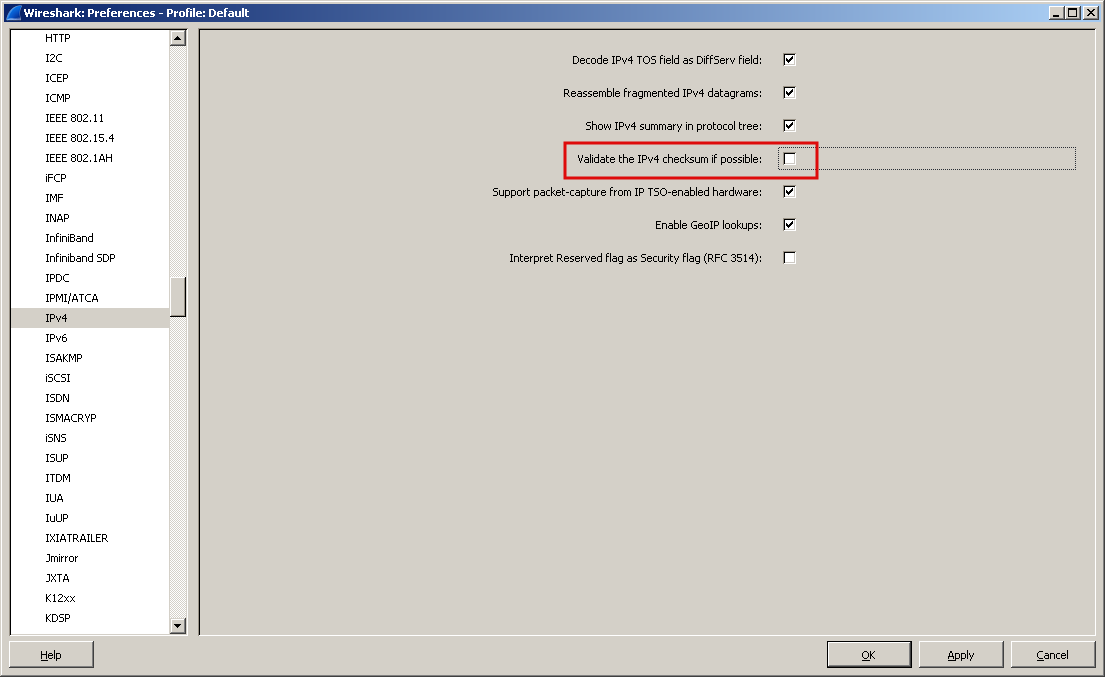
А Wireshark натурально перехватывает пакеты, до того как они попадают в сеть.

И до того как эта сумма была просчитана и была добавлена в заголовок пакета.

Соответственно есть два пути решения этой проблемы — выключать функцию offload в настройках сетевой карты или в настройках сниффера указать, чтобы он не обращал внимание на это значение.

Хардваные функции зачастую лучше софтварных, в основном из-за скорости обработки (в железе обычно выше) поэтому лучше изменить настройки самого сниффера.

Для этого нужно зайти в настройки (Edit — Preferences), затем Protocols – IPv4 – и снять флаг с “Validate IPv4 checksum if possible”.

Перед тем как захватывать трафик нужно определиться с тем, что, собственно, нужно захватывать.

Разместить анализатор трафика можно в нескольких местах:

Локально на своем хосте;

Организовать зеркалирование трафика на коммутаторе;

Подключаться непосредственно в интересующие места;

или же отравление протокола ARP (еще более незаконно, чем пассивное прослушивание трафика)

**Фильтруем поток**

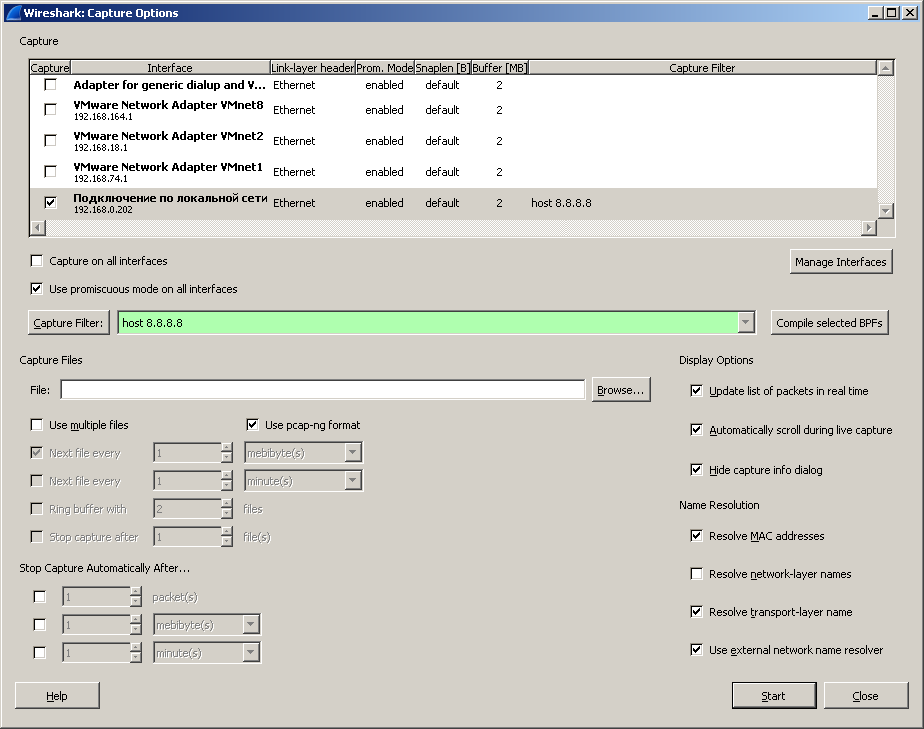
Wireshark содержит два вида фильтров – захвата (Capture Filters) и отображения (Display Filters).

Вначале рассмотрим Capture Filters.

Как можно догадаться по названию, они служат для фильтрации еще на этапе захвата трафика.

Но в таком случае, безусловно, можно безвозвратно потерять часть нужного трафика.

Фильтр представляет собой выражение, состоящее из встроенных значений, которые при необходимости могут объединяться логическими функциями (and, or, not).

Для того, чтобы его задействовать, нужно зайти в меню Сapture, затем Options, и в поле Capture Filter набрать, например, host 8.8.8.8 (или, например, net 192.168.0.0./24)

Так же, конечно, можно выбрать и заранее созданный фильтр (за это отвечает кнопка Capture Filter).

В любом из вариантов фильтр появится возле интерфейса, можно жать Start.

Теперь перейдем к Display Filters.

Они фильтруют исключительно уже захваченный трафик.

Что можно фильтровать?

— Практически все — протоколы, адреса, специфические поля в протоколах.

Операции, которые можно использовать при построении фильтров:

|  |  |  |
| --- | --- | --- |
| **Команда** | **Значение** | **Пример использования** |
| == | Равенство | ip.dst == 193.168.3.10 |
| != | Не равно | udp.dst != 53 |
| < | Меньше чем | ip.ttl < 24 |
| > | Больше чем | frame.len > 10 |
| <= | Меньше или равно | frame.len <= 0x20 |
| >= | Больше или равно | tcp.analysis.bytes\_in\_flight >= 1000 |
| Matches | Регулярное выражение | frame matches "[Pp][Aa][Ss][Ss]" |
| Contains | Содержит | dns.resp.name contains google |

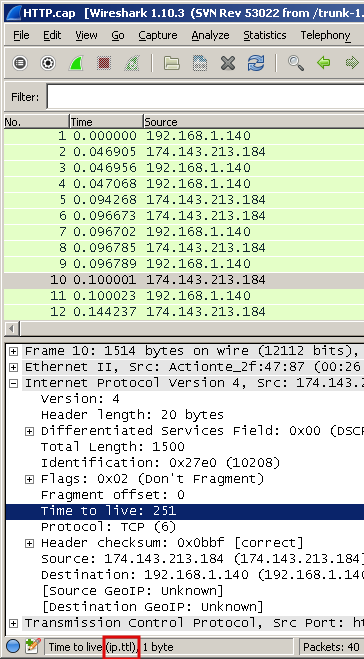
Как вы, наверное, заметили, в таблице в качестве примеров были разнообразные выражения, достаточно понятные и зачастую говорящие сами за себя.

Например, ip.dst – это поле протокола IP.

Чтобы увидеть это поле, можно просто посмотреть на пакет, и в нижней части окна можно увидеть его значение, которое потом можно применять в любом фильтре.

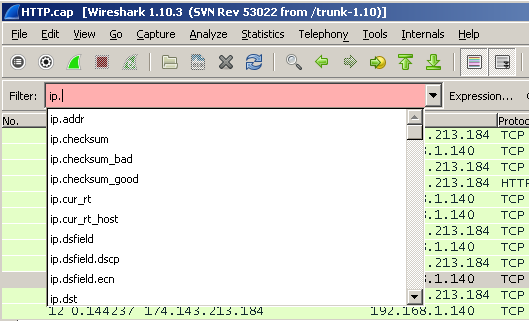
Например, нас интересует, как создать фильтр, где будет проверяться значение TTL.

Для этого раскрываем L3 часть и становимся на соответствующее поле:



И видим, что для построения фильтра, нужно использовать выражение ip.ttl.

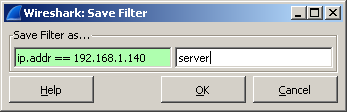
Если начать набирать фильтр, то после точки автоматически появится список возможных значений:

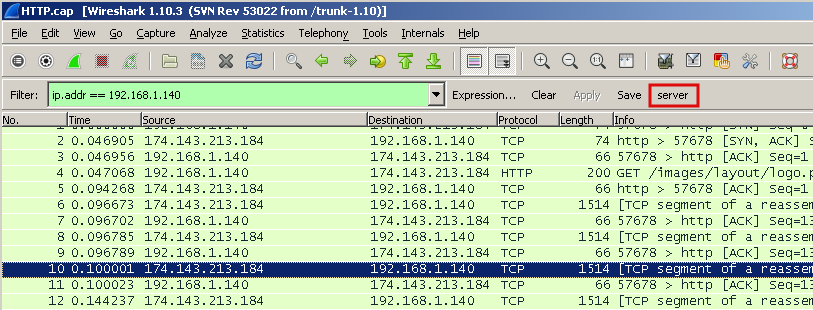
Чтобы применить фильтр, достаточно нажать enter или кнопку Apply.

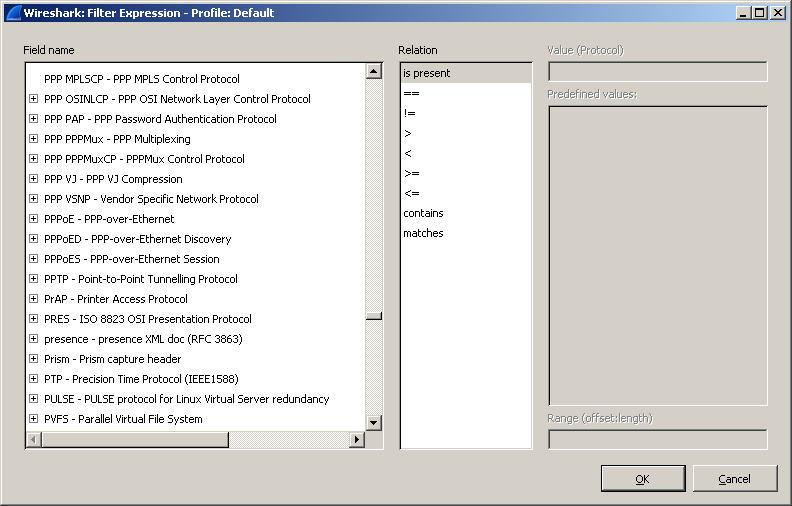
Само поле для ввода фильтра может менять цвет в зависимости от того, что было набрано.

Зеленый цвет означает, что все в порядке. Красный — допущена ошибка, желтый — получен неожиданный результат, потому что существуют другие варианты написания фильтра (например можно написать ip.dst != 8.8.8.8 или же !ip.dst == 8.8.8.8, именно второй вариант более предпочтительный).

Фильтры можно сохранять для дальнейшего использования, нажав кнопку Save, затем ввести произвольное название

и после нажатия на кнопку ОК фильтр появится как кнопка на панели.

А если кликнуть на расположенную неподалеку кнопку «Expression…», то откроется достаточно мощный конструктор выражений, по которому можно чуть ли не изучать сетевые протоколы. Количество поддерживаемых протоколов постоянно увеличивается.

Как уже упоминалось ранее, можно выделить любой пакет и в контекстном меню выбрать Apply as Filter и в подменю выбрать режим — selected или not selected и соответственно сразу же появится фильтр, который будет показывать только выбранное или наоборот уберет выбранное с экрана.

Таким образом можно гибко выбирать, что видеть на экране, а что — нет.

Это может быть определенный ip-адрес, ttl, порт, dns ответ и многое другое.

Кроме того, есть два варианта для таких быстрых фильтров — Prepare as Filter и Apply as Filter.

Как можно догадаться по названию — разница заключается в том, что в первом случае только появится в поле для ввода Display Filter, но не применится (удобно, если например, добавлять таким способом несколько фильтров, а затем сразу применить готовый результат), а во втором — сразу же и применится.

Фильтры можно объединять, используя знакомые по булевой алгебре логические операции:

(dns) && (http) логическое и

(dns) || (http) это логическое или

Таким образом можно строить большие и сложные фильтры вроде:

(tcp.flags.syn==1) && (ip.src == 172.16.10.2) && (ip.dst == 172.16.10.1)

Здесь видим, что выбираются только TCP SYN сегменты, только с определенным адресом отправителя и получателя. При составлении больших фильтров нужно помнить, что фильтр по сути — логическое выражение, и если оно истинно, то пакет отобразится на экране, если ложно — нет.

**Ныряем глубже**

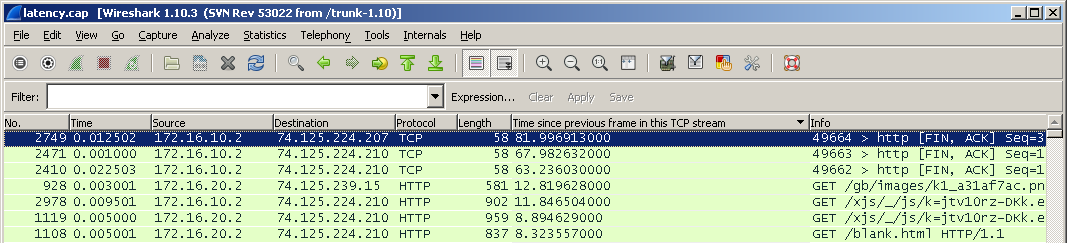
Достаточно частая ситуация, когда возникают жалобы на медленную работу сети, причин этого может быть множество.

Попробуем разобраться, в чем может быть причина, и рассмотрим два способа.

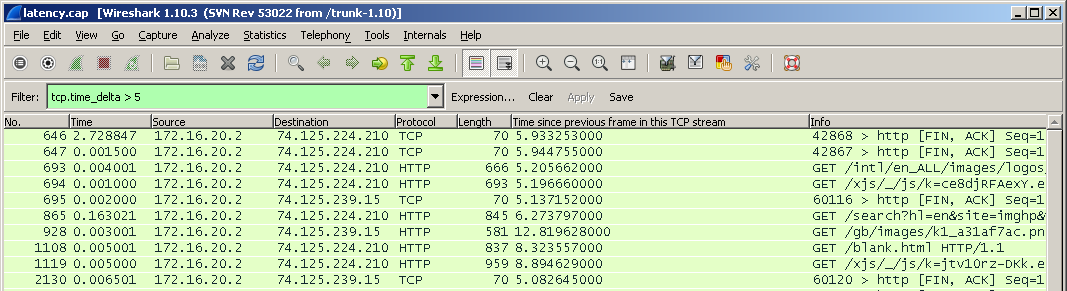
Первый состоит в добавлении колонки TCP delta.

Открываем пакет, находим поле Time since previous frame in this TCP frame, правый клик и выбираем Apply as Column. Появится новая колонка.

На ней можно кликнуть правой кнопкой мыши и выбрать режим сортировки, например, Sort Descending.

И сразу же рассмотрим второй способ.

Относительно недавно (в версии 1.10.0) появился фильтр tcp.time\_delta, который, собственно, учитывает время с момента последнего запроса.

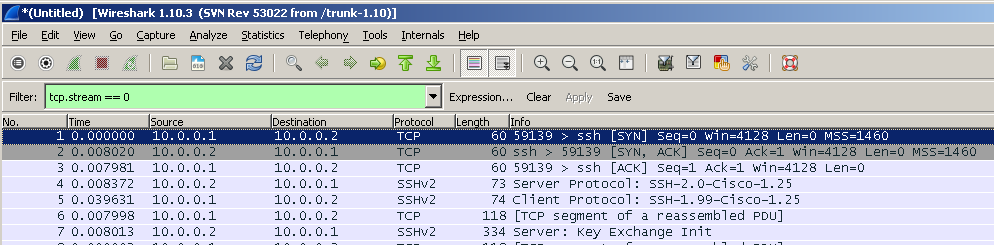
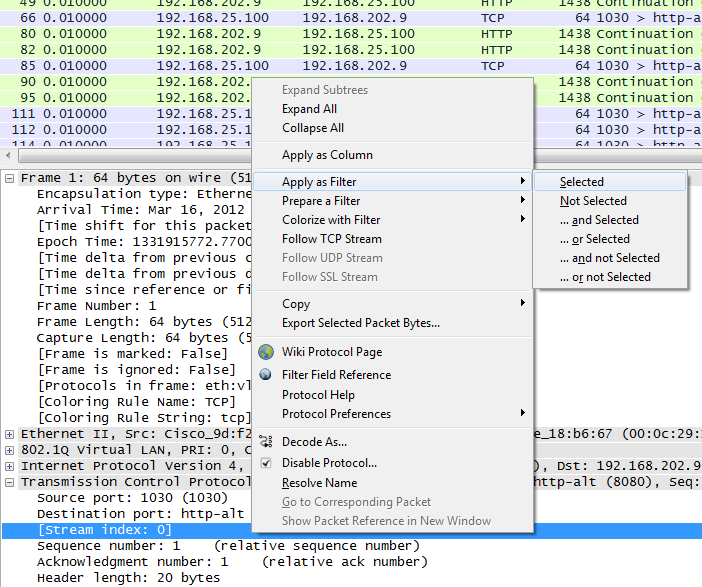
Если клиент делает запрос и получает ответ через 10 миллисекунд, и клиент говорит, что у него все медленно работает, то, возможно, проблема у самого клиента.

Если же клиент делает запрос и получает ответ через 2-3 секунды, тут уже, возможно, проблема кроется в сети.

**Еще глубже**

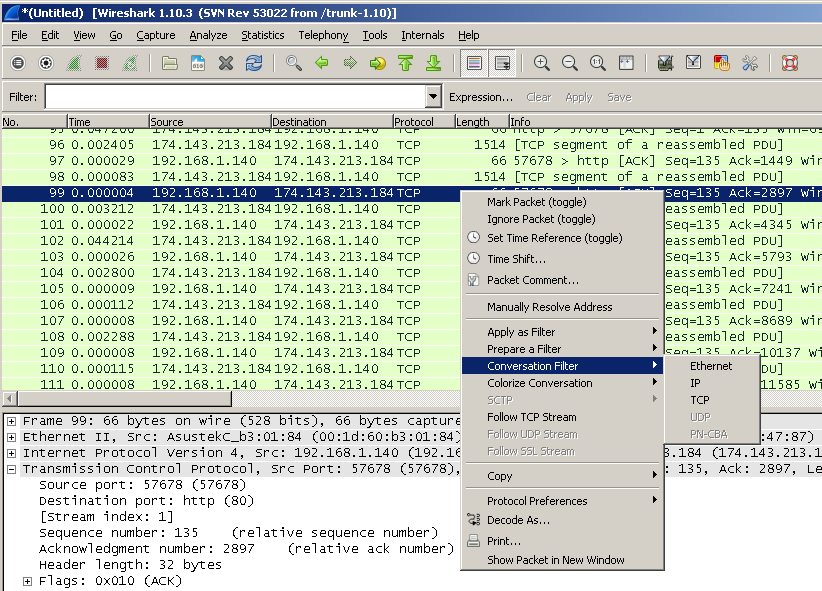
Если посмотреть в TCP пакет (или сегмент если быть точным), то можно увидеть там Stream index, который начинается обычно с нуля.

Само поле будет называться tcp.stream.

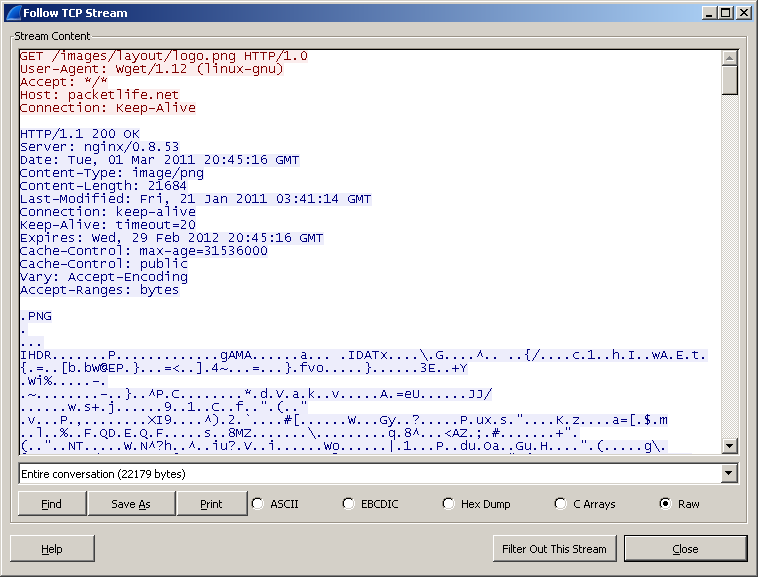
По нему можно сделать правый клик и создать фильтр.

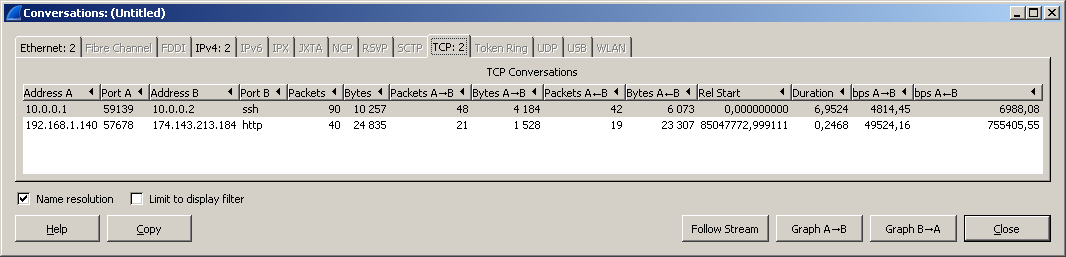
В итоге мы опять увидим взаимодействие двух хостов.

Еще один способ – сделать правый клик на самом пакете, выбрать Conversation Filter и создать фильтр для l2 l3 l4 уровня соответственно.

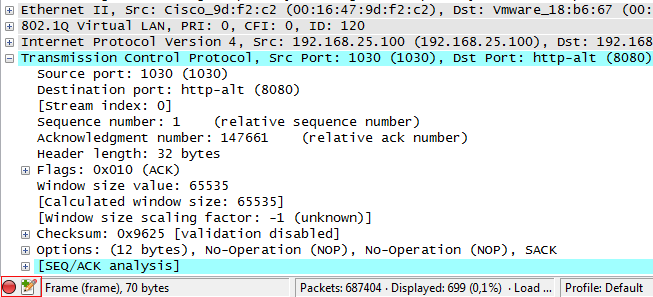
И третий вариант — это одна из самых интересных фич — Follow TCP Stream.

Для того чтобы его задействовать, нужно опять таки кликнуть правой кнопкой мыши на пакете и выбрать “Follow TCP Stream”. Появится окно, где будет наглядно продемонстрирован весь обмен между двумя узлами.

Если же зайти в меню Statistics – Conversations, то, выбирая закладки, можно увидеть статистику по таким “разговорам” и различные сессии, при этом можно отсортировать их по различным колонкам, например, по количеству переданных данных.

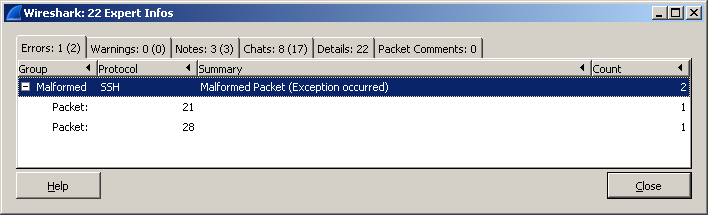
И прямо в этом окне можно правой кнопкой взывать контекстное меню и опять же применить как фильтр.

Со временем приходит опыт

После некоторого времени, проведенного за захватом разнообразного трафика, можно заметить какую-то шарообразную кнопку в нижнем левом углу, которая еще иногда меняет цвет.

Нажатие на эту кнопку приведет к открытию окна Expert Infos.

Того же результата можно добиться, пройдя в меню Analyze – Expert Info.

В этом окне будет содержаться информация по найденным пакетам, разбитая на группы Errors, Warnings, Notes и Chats.

Цветовая раскраска для этих групп выглядит следующим образом:

Ошибки — красный цвет

Предупреждения — желтый

Примечания — сине-зелёный (cyan)

Чат — серый

Wireshark содержит в себе мощный анализатор и умеет автоматически обнаруживать большое количество проблем, возникающих в сети.

Как вы уже могли заметить, буквально везде можно использовать фильтры и Expert Info не является исключением.

Для того чтобы создать такой фильтр, нужно использовать конструкцию expert.severity.

Например, expert.severity==error.

**Грабим трафик**

Можно ли с помощью Wireshark узнать, что было скачано?

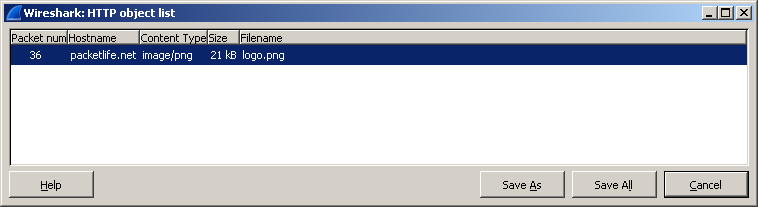
Да, можно. И сейчас это увидим.

Вначале возьмем HTTP трафик.

Сделаем правый клик по HTTP пакету — Protocol Preferences – и видим тут массу опций, которые непосредственно влияют на извлечение файлов из веб трафика.

Для того чтобы увидеть, что можно извлечь из текущего дампа нужно перейти в меню File – Export Objects – HTTP.

Появится окно, которое покажет все захваченные http объекты — текстовые файлы, картинки и т.д. Для того чтобы вытащить любой файл из этого списка, достаточно просто выделить его и нажать Save As.

Таким же способом, можно извлекать и потоковое видео/аудио.

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ
2. Сделайте несколько http-запросов и найдите их в wireshark.
3. Воспользуйтесь утилитой telnet, чтобы посылать пакеты со своим содержанием (telnet localhost 80 {Enter} «Свой текст» {Enter}). Покажите, что wireshark перехватила эти пакеты, покажите их содержание.
4. Вставьте в отчет экранные снимки, демонстрирующие основные функции программы Wireshark (сохранение и восстановление перехваченных сетевых пакетов, фильтрация сетевых пакетов, выделение цветом по правилам сетевых пакетов, сводная статистика, предназначение модулей, расположенных в меню Statistics и др).
5. Извлеките из траффика несколько файлов (например картинок).
6. КОНТРОЛЬНЫЕ ВОПРОСЫ
7. Предназначение анализаторов сетевого трафика.
8. Что такое сниффер?
9. Способы осуществления перехвата сетевого трафика.
10. Что может быть обнаружено в результате анализа сетевого трафика?
11. Наименование и функциональные возможности программ-анализаторов сетевого трафика.
12. Основные функциональные возможности программы Wireshark.
13. ЛИТЕРАТУРА

Тетерукова Н. А., Апанасевич С. А. Защита компьютерной информации (лабораторный практикум) - МГВРК, 2013.

Преподаватель А.Г.Тараренко

Рассмотрено на заседании цикловой комиссии

программного обеспечения информационных технологий № 10

Протокол № \_\_\_\_от «\_\_\_» \_\_\_\_\_\_2016

Председатель ЦК \_\_\_\_\_\_ Т.Г.Багласова