# Лекция № 10. Средства и методы анализа сетевого трафика.

Обзор средств анализа сетевого трафика. Реализация средств перехвата сетевого трафика. Основные функции. Способы отбора и анализа сетевых пакетов. Работа с файлами перехвата трафика с целью выявления типовых атак и вредоносного ПО. Автоматизация при работе с большими объемами данных.

Введение

Очень часто для поиска проблем в работе сети используются анализаторы сетевого трафика или сниферы. Wireshark и Tcpdump является одними из наиболее часто используемых данного класса программ, они позволяет прослушать (отобразить/сохранить) и проанализировать работу сети на уровне передаваемых сетевых пакетов, фреймов и др. единиц передачи сетевого трафика. В зависимости от конфигурации сети, снифер может прослушивать не только пакеты, предназначенные данному MAC-адресу, но и широковещательные пакеты. Прослушивание перехват сетевых пакетов основан на "беспорядочном" (promiscuous) режиме работы сетевого адаптера. В зависимости от используемого сетевого оборудования для соединения компьютеров в сети Ethernet существуют следующие возможности прослушивания трафика:

– в сети на основе концентраторов весь трафик с концентратора (хаба) доступен любому сетевому хосту;

– в сетях на основе коммутаторов (свичей) сетевому хосту доступен только ее трафик, а также весь широковещательный трафик данного сегмента;

– некоторые управляемые коммутаторы имеют функцию копирования трафика данного порта на порт мониторинга ("зеркалирование",мониторинг порта);

– при использовании специальных средств (ответвителей), включаемых в разрыв сетевого подключения и передающих трафик подключения на отдельный порт возможно прослушивание соответствующего порта;

– "трюк" с концентратором — порт коммутатора, трафик которого необходимо прослушать, включают через концентратор, подключив к концентратору также узел-монитор (при этом в большинстве случаев уменьшается производительность сетевого подключения).

## Место установки снифера

Эффективность работы снифера в первую очередь зависит от места его установки в сети. Далее разберем особенности некоторых типов сетей и, связанных с эти правил установки средства перехвата трафика.

### Перехват трафика в коммутируемых сетях

Несмотря на то, что использование коммутаторов позволяет лучше контролировать сеть, более эффективно использовать пропускную способность, осуществлять изоляцию сетевого трафика, с точки зрения сетевого аналитика это значительно усложняет порядок использования в такой сети средств анализа трафика с точки зрения информационной безопасности. Если подключить снифер (например, ПК с установленным Wireshark) к порту коммутатора то мы будем видеть только следующие 4 категории траффика:

широковещательный трафик;

мультикастовый трафик (если он передается коммутатором);

трафик от и к нашему аппаратному адресу (адресу MAC);

трафик к неизвестным аппаратным адресам.

Это связана с тем, что по своему функционалу коммутатор осуществляет передачу трафика между подключенными ПК к различным портам напрямую, минуя рассылки на остальные порты (как делает концентратор или хаб).

Поэтому существуют следующие способы осуществления перехвата трафика в коммутируемых сетях:

использование устройств типа хаб (в настоящее время может использоваться только для крайних случаев, т.к. данная технология крайне устарела);

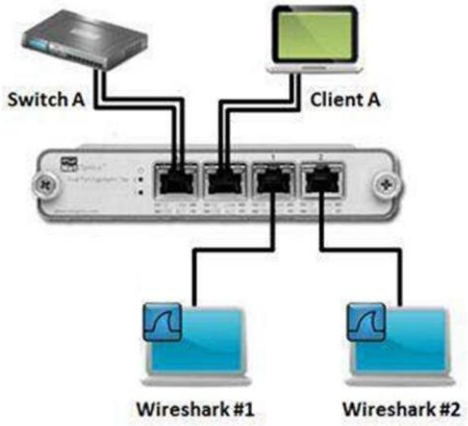
использование TAP ответвителей;

использование SPAN порта;

установка снифера не целевую систему.

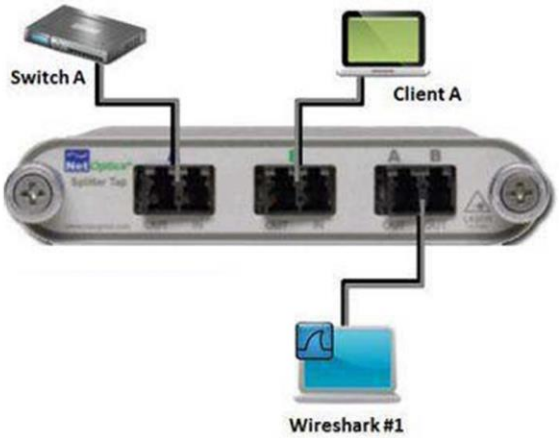
Пропуская рассмотрения первого метода, необходимо сказать, что TAP (или Test Access Point) могут использоваться в полнодуплексных и полудуплексных сетях. TAP являются пассивным устройством, которое включается в разрез линии между устройствами. В отличии от SPAN портов коммутатора, TAP могут переадресовывать пакеты, которые содержат ошибки на физическом уровне (такие как ошибки CRC) на порты мониторинга. Также особенностями TAP ответвителей является то, что они не вносят задержек, не позволяют модифицировать передаваемое содержимое и являются так сказать «fail open», т.е. при пропадании питания не прерываю передачу трафика между устройствами.

Установка TAP ответвителя зависит от реализованных в нем функций. Например, если TAP не производит агрегацию трафика, т.е. передает трафик передачи и приема на отдельные порты, то он может быть установлен по следующей схеме



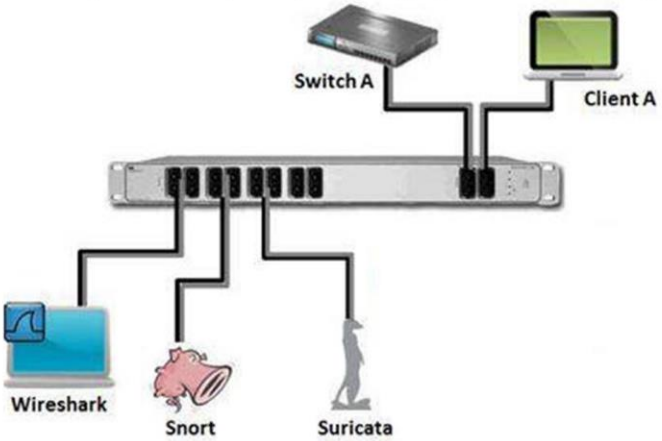

Таким образом можно выделить следующий тип TAP: TAP без агрегации. Для работы с перехваченным трафиком необходимо использовать два ПК с установленными сниферами. Для корректного анализа полученного трафика можно использовать опцию снифера File\Merge для слияния трафика в Wireshark, или программу Mergecap. Также можно использовать и один ПК с двумя сетевыми интерфейсами. Однако при таких схемах необходимо учитывать возможную временную задержку приема пакетов, возникающую, например, ели одна сетевая карта подключена через USB. Это может повлиять на порядок прихода пакетов в результирующей сборке.

Соответственно противоположностью TAP без агрегации является TAP с функцией агрегации.

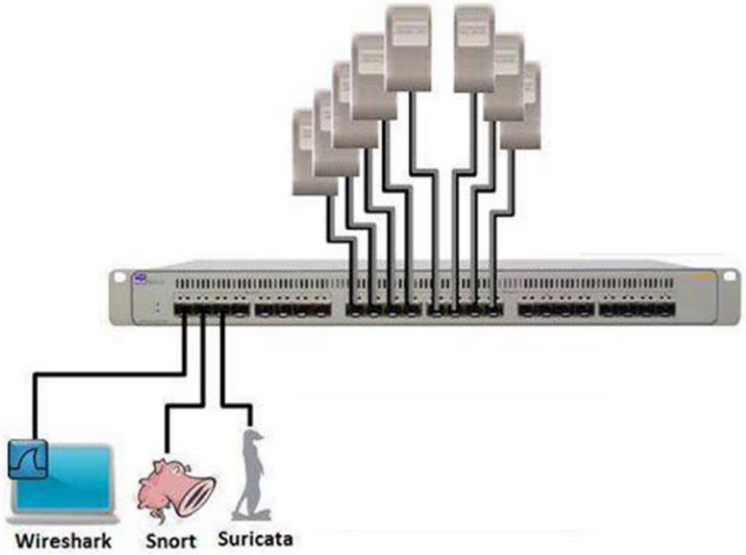



Данный тип TAP позволяет агрегировать двунаправленный трафик в один исходящий порт. Соответственно для такой схемы требуется только один ПК с установленным снифером. Однако проблему со скоростями передачи это не снимает: если скорость двунаправленного трафика превышает скорость подключения порта ответвителя (например порта по 100Мб/с и один TAP тоже на 100Мб/с) при максимальной загрузке не сможет отдать все пакеты на порт TAP.

Развитием функциональности TAP является с регенерацией. Такой ответвитель позволяет передавать информацию не на один, а сразу на несколько портов для использования различных средств, например, средства записи трафика, IDS и пр.




TAP с агрегацией связей. Такой TAP позволяет передавать информацию о взаимодействии не двух а сразу нескольких узлов на один или как правило несколько TAP портов.

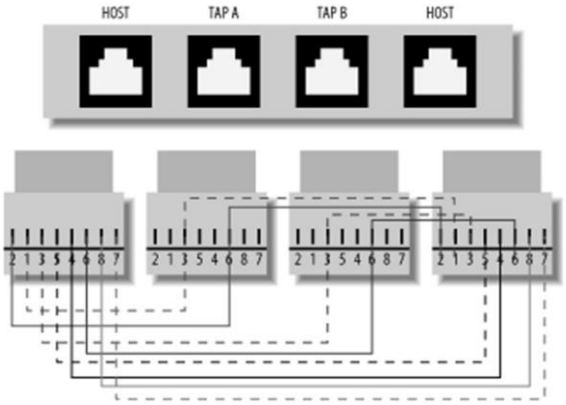



Следующий тип это интеллектуальные TAP. Данный тип TAP позволяет учитывать содержимое трафика, устанавливать временные метки для каждого полученного пакета, осуществлять фильтрацию пакетов. Конкретный набор функций зависит от реализации интеллектуального TAP.

Пассивный TAP

Invisibly Sniffing Between Two Network Points
Problem: You want to insert a tap between two particular points on your network.
Solution: Construct a passive tap.

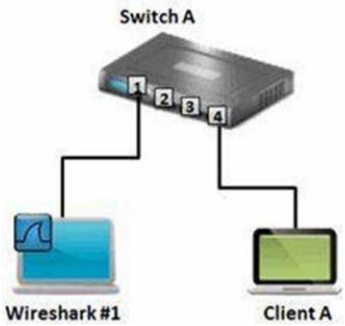
Discussion
A passive tap is slightly more complex than the receive-only Ethernet cable. You require a four-port Ethernet housing, four category 5e modular snap-in jacks, and bit of category 5e cabling.




Наряду с использованием локальных устройств возможно применение программного обеспечения удалённого мониторинга или перехвата пакетов (например rspan), или агентов,

которые устанавливаются непосредственно в анализируемую систему. Данное ПО может устанавливаться также непосредственно на коммутаторы и позволяет работать с перехваченным трафиком в удаленной консоли путем его передачи через заданный интерфейс. Для выполнения данных функций можно также использовать не специализированное ПО, а утилиты уделенного управления типа ssh, VNC, RAdmin и др.

Использование SPAN порта реализуется за счет применения различными разработчиками оборудования технологии зеркалирования портов коммутаторов.




В рамках этого используется следующая терминология:

порт-источник SPAN (Source SPAN Port): Порт источника, который мониторится с использованием опции SPAN (здесь порт №4 это SPAN порт);

VLAN-источник SPAN (Source SPAN VLAN): VLAN, чей трафик мониторится с использованием опции SPAN;

порт-назначения SPAN (Destination Span Port или Monitor Port) (здесь это порт №1);

входящий трафик (Ingress Traffic): трафик входящий в коммутатор (при настройке SPAN иногда требуются указать интересующий трафик);

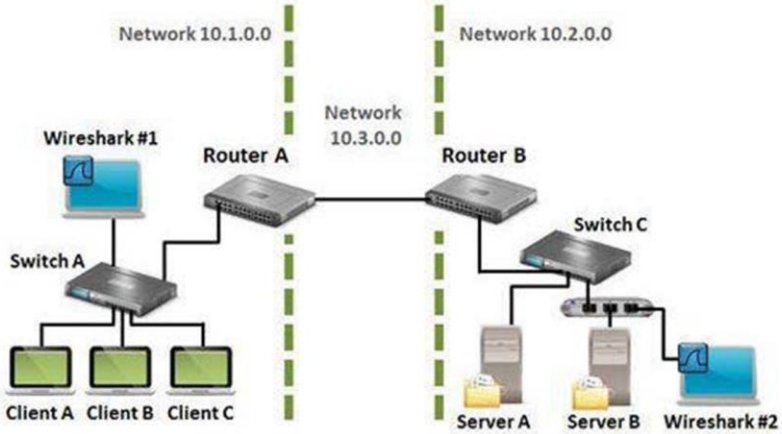
исходящий трафик (Egress Traffic): трафик исходящий из коммутатора (при настройке SPAN иногда требуются указать интересующий трафик);

Пример настройки для Cisco IOS:

```
monitor session 1 source interface fa0/4
monitor session 1 destination interface fa0/1
show monitor session all
```

### Перехват трафика в маршрутизируемых сетях

Маршрутизаторы изолирую трафик используя IP адреса (для IP-сетей). Поэтому при установке снифера с внешней (по отношению к нашей сети) стороны маршрутизатора мы сможем определить только трафик идущий в нашу сеть или исходящий из нее, но не трафик внутри сети.




На данной схеме представлены сети 10.1.0.0, 10.2.0.0, 10.3.0.0 с маской 16. Трафик между клиентами и серверами сети 10.1.0.0 не виден для Wireshark #2 в сети 10.2.0.0. Wireshark #1 настроен на перехват трафика через SPAN порты для прослушивания интерфейса маршрутизатора Router A. Поэтому он может видеть трафик исходящий (входящий) от клиентов A, B, C и идущий в сеть 10.2.0.0. Wireshark #2 подключен к агрегирующему TAP, который подключает сервер B к коммутатору C. Он может слушать трафик идущий к и от сервера И к локальным и удаленным станциям.

## Isolating Network Sensors[1]

In this section, we briefly cover common methods for redirecting copies of Ethernet traffic from constituency networks to the SOC's sensors. As these topics are covered extensively in existing literature, we summarize some important points related to reliability, cost, and protection of the sensor in question.

Figure 28 illustrates popular approaches to making copies of network traffic.

Starting at the top left, we see a network hub that is being used to copy network traffic to a passive network sensor. This is the most straightforward approach. By inserting a layer 1 Ethernet device between Alice's and Bob's networks, the sensor will see a copy of the traffic passed. However, there are a number of reasons this is less than desirable. Packet collisions and packet latency can become a serious problem. By using a hub, packets will be dropped and the sensor will miss traffic. In addition, most modern networks operate at 1 Gb/s or 10 Gb/s speed; hubs essentially do not exist in speeds faster than 100 Megabits/s. Finally, hubs are generally not very fault tolerant. Thus, network owners are unlikely to approve the placement of a flimsy device between two networks.

At the top right, we replace the hub with a layer 2 or layer 3 network switch. This switch is configured with a switched port analyzer (SPAN) to copy or "span" traffic from one or more source ports or virtual LANs (VLANs) to the port hosting a network sensor.

This approach offers most of the benefits of the hub approach, by using enterprise-grade network equipment that probably is already in operation across the constituency. The major caveat to this approach is that the SPAN must be set properly, and that its configuration must continue to match the intended source ports and VLANs down the road. As network topology changes, respective SPAN configurations must be altered to match.

In the middle of the diagram, we see two different scenarios that leverage a network tap. A network tap is essentially a device inserted between two network nodes that makes a copy of all network traffic flowing between them. On the left, we see a passive network tap (in the case of

[1] MITRE. "Ten Strategies of a World-Class Cybersecurity Operations Center"

copper) that simply makes an electrical copy of the traffic flowing, or, in the case of optical taps, uses a mirror to actually split the transmit (TX) and receive (RX) light beams. With "dumb" or passive network taps, we must use a sensor that has built-in

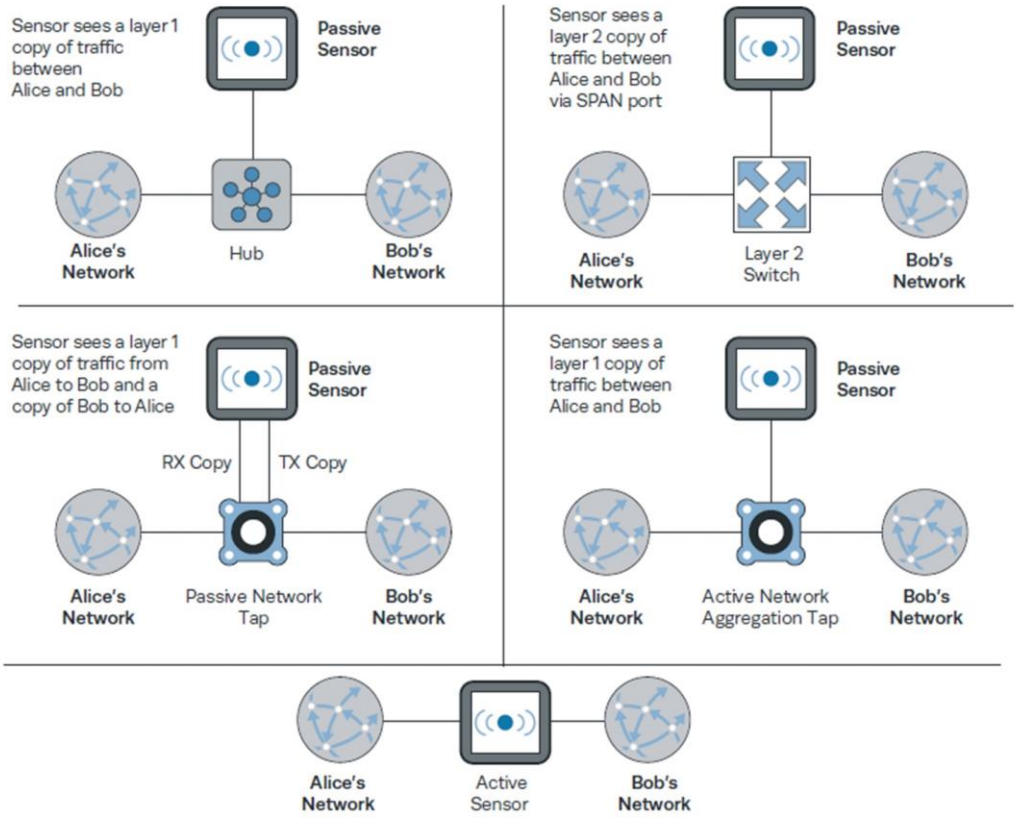



Figure 28. Copying Network Traffic

logic that recombines the RX and TX lines into one logical stream of traffic suitable for decoding. On the right, an active network aggregation tap does this work for us but, at the same time, has the disadvantage of saturating its monitor port if both Alice's and Bob's aggregate bandwidth exceeds that of the monitor port. Popular manufacturers of network taps include NetOptics [236], Fluke [237], Network Critical [238], and Datacom Systems [239]. Network taps aren't generally subject to the same range of misconfigurations that switch SPANs are; that said, network sensors certainly get disconnected from time to time.

Keeping SPAN configurations up-to-date so network sensors don't go "dark" is a major pain point for many SOCs with large sensor fleets. Keeping tabs on sensor traffic statistics and "health and welfare" is a daily job, even when using network taps.

Finally, at the bottom, we see a sensor placed directly in-line between Alice's and Bob's networks (e.g., a NIPS). Most modern COTS NIDS/NIPS appliances offer a monitoring mode where the sensor can sit in-line and passively listen to traffic without any intentional interference. Less robust sensors may "fail closed" (e.g., if an error occurs or the device loses power) and disconnect Alice and Bob. However, most best-of-breed products take great care to avoid this problem.

In every case discussed, the network sensor must be physically near the network devices it monitors. This usually means that the network sensor is in the same rack, server room, or building as the monitored network segment. While it is certainly possible to send a copy of network traffic to a distant sensor using a remote SPAN, long-range optical connection, or even a WAN link, doing so can become very expensive. In essence, this usually means doubling the network throughput from the source network equipment to the location of the sensor. This almost never makes architectural or financial sense. Physically placing sensors adjacent to their monitored network segment is almost always the cheapest option; as a result, effective remote management is essential.

We summarize the pros and cons of approaches to traffic redirection in Table 19.

All these traffic redirection options have implications for how we prevent the network sensor from compromise or discovery. First, the monitoring port or ports should not have an IP address assigned to them. This will minimize the likelihood that it will talk back out on the network or bind services to the port. It follows, of course, that the monitoring port(s) should be used exclusively by the monitoring program(s) on the sensor and not be used for management or other services.

| What Pros Cons | What Pros Cons | What Pros Cons |
|---|---|---|
| Hub | Cheap<br>Easy to install<br>Can attach as many monitoring devices as there are free ports | Sensor only sees what links at its same speed, not what links on other speeds, leading to gaps in visibility.<br>Network management personnel may not be okay with non-enterprise-grade devices in critical routing points in the network.<br>Sensors will miss packets due to collisions. Inherent inefficiency in hub's handling of traffic may lead to network bottlenecks.<br>It is nearly impossible to find hubs that operate at gigabit (or above) speeds, limiting use to links that are throttled at 100 Mb/s. |
| SPAN | Free to use if monitoring points already have managed switches in place, which is very likely<br>RX and TX are combined; one network cable off a SPAN port can plug right into a sensor.<br>Straightforward for monitoring traffic from any device hanging off a switch (such as a firewall, WAN link, or cluster of servers)<br>Can attach as many monitoring devices as there are free ports<br>Can be used to monitor network core, such as spanning multiple ports off a core switch or router | An adversary with access to the enterprise network management platform (e.g., a terminal access controller access control system [TACACS] server) can disable monitoring feeds to the sensor.<br>Some older or cheaper switches support only one SPAN port per switch, meaning additional switches may be needed if more than one sensor is desired.<br>When spanning traffic from multiple source ports, the destination SPAN port may become oversaturated if the source ports' traffic aggregate bandwidth exceeds the SPAN port's speed.<br>Changes to VLAN or port configurations after initial SPAN configuration can partially or completely blind the network sensor without the SOC necessarily realizing it. |
| Tap | Essentially invisible from a logical perspective.<br>They only operate at layer 1, meaning the adversary does not have an obvious target to exploit or circumvent.<br>Should not alter packets in any way<br>Active network regen taps support multiple monitoring devices. | An additional device (albeit usually well built and simple) that can fail is introduced into critical network links.<br>Only appropriate when observing conversation between two networked devices (as opposed to many with a network switch SPAN), as is often the case in perimeter network monitoring.<br>Every monitoring point requires the purchase of an additional tap device.<br>With a passive tap, RX and TX lines need to be recombined; some sensor technologies do have |

| | | |
|---|---|---|
| | | the internal logic to do so. Passive network taps only support one monitoring device. |
| In-line | Sensor can actively block traffic, depending on rule set. | If sensor goes down, it may cut off communication unless resiliency features are built in (e.g., "fail open").<br>Some sensor technologies may introduce packet latency or packet reordering, which in turn can sometimes degrade network quality of service or make the sensor detectable.<br>More than one monitoring device means serial attachment of devices in-line, each being a separate point of failure. |

In more extreme cases, this isn't enough. There are approaches we can take at a hardware level to ensure that the IDS's monitor ports cannot interact with the monitored network. In the case of a network tap, this has already been done. These devices, by their very nature, make copies of traffic that are destined for the sensor's monitoring port and do not accept response traffic.

In the case of COTS in-line devices, the opposite is the case. We must trust the implementation of the sensor technology to ensure the device will not interact with the network it is monitoring (except for blocking traffic, when appropriate). In the case of FOSS in-line devices, this duty falls upon both the authors of the FOSS monitoring software and the integrators of the FOSS sensor platform.

If we are using a switch to span traffic to a sensor, we have some interesting options at our disposal. We must prevent the sensor from transmitting packets back out on the network, thereby mitigating the effects of most attacks against the sensor (with the exception of DoS). However, the network switch that is sending the sensor packets must also establish a layer 1 link before it will transmit packets.

As a result, we can't just clip the transmit leads onto the Ethernet cable running from the sensor to the switch—the switch will think there is no device attached and drop the spanned traffic. In this case, we have two options. The first is to fabricate a receive-only Ethernet cable, as is specified in [240]. Compared to other options, this approach is quick, cheap, and easy.

If the passive network sensors are to reside on a network with a significantly different trust level than the network being monitored, a more robust solution such as a data diode may be necessary. Many commercial data diode solutions exist (e.g., those by Owl [241] or HP/Tenix [242]). However, these tend to be expensive and don't really meet our needs. We need an unaltered stream of network traffic data directly from the monitored network (something most commercial data diodes don't do). Also, in large enterprises, we may have many dozens or hundreds of sensors, necessitating an economical solution. It is possible to use three copper to fibre network transceivers to do this, as described in [243] and shown in Figure 29.

In Figure 29, we see the three fibre transceivers in light green. Each transceiver has three connections: (1) fibre transmit (TX), (2) fibre receive (RX), and (3) a copper RJ45 combined receive/transmit connection (Cu). They are connected in such a way that data can only flow from the monitored network to the sensor. In this arrangement, we use the third data diode at the bottom of the diagram to fool the data diode on the top right that it has a valid Ethernet link. What is really happening is that only the RX link from the transceiver on the monitored network side is connected to the TX link on the transceiver connected to the network sensor, at the top left.

For further information on network sensor isolation techniques, see Chapter 3 of [9] and [240].

## Wireshark

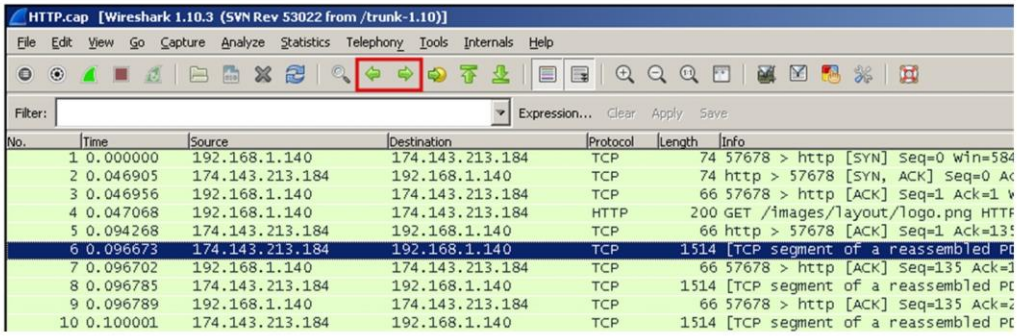
Wireshark[2] — это достаточно известный инструмент для захвата и анализа сетевого трафика, фактически стандарт как для образования, так и для траблшутинга. Wireshark работает с подавляющим большинством известных протоколов, имеет понятный и логичный

[2] Wireshark на русском. Знакомство с акулой. («ws-ALL+Ex!.htm»)

графический интерфейс на основе GTK+ и мощнейшую систему фильтров. Wireshark кроссплатформенный и 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.

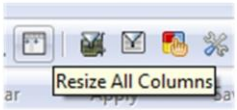
### Порядок работы

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.

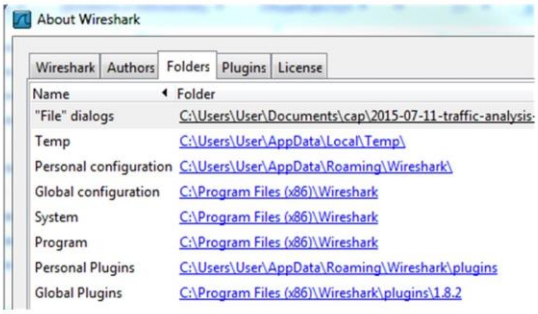



А следующая за ними кнопка позволяет сделать быстрый переход к пакету, указав его номер.

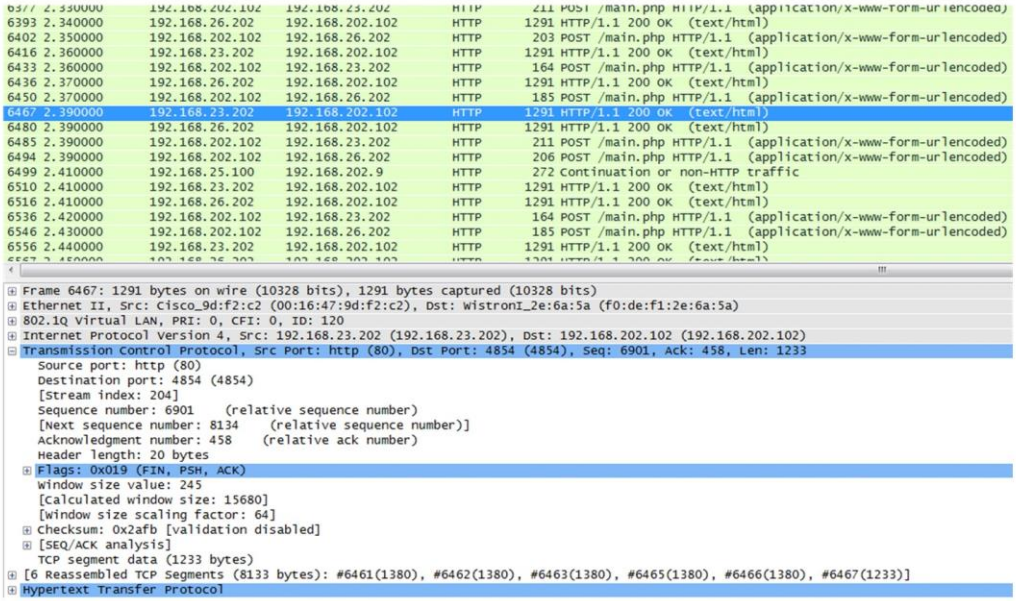
В случае если колонки перекрываются и наползают друг на друга, можно кликнуть по такой колонке правой кнопкой мыши и выбрать “Resize Column”. Произойдет автоматическая подгонка размеров под текущую ситуацию. И кроме того, есть кнопка “Resize all Columns”, которая приведет в порядок все колонки.




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). Справка (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, это VLAN).




И на самом верху идет небольшой обзор собранной информации о кадре.

```
⊟ Frame 6467: 1291 bytes on wire (10328 bits), 1291 bytes captured (10328 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Mar 16, 2012 18:36:15.160000000 Финляндия (зима)
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1331915775.160000000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.020000000 seconds]
    [Time since reference or first frame: 2.390000000 seconds]
    Frame Number: 6467
    Frame Length: 1291 bytes (10328 bits)
    Capture Length: 1291 bytes (10328 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:vlan:ip:tcp:http:data-text-lines]
    [Number of per-protocol-data: 1]
    [Hypertext Transfer Protocol, key 0]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80]
⊞ Ethernet II, Src: Cisco_9d:f2:c2 (00:16:47:9d:f2:c2), Dst: WistronI_2e:6a:5a (f0:de:f1:2e:6a:5a)
⊞ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 120
```

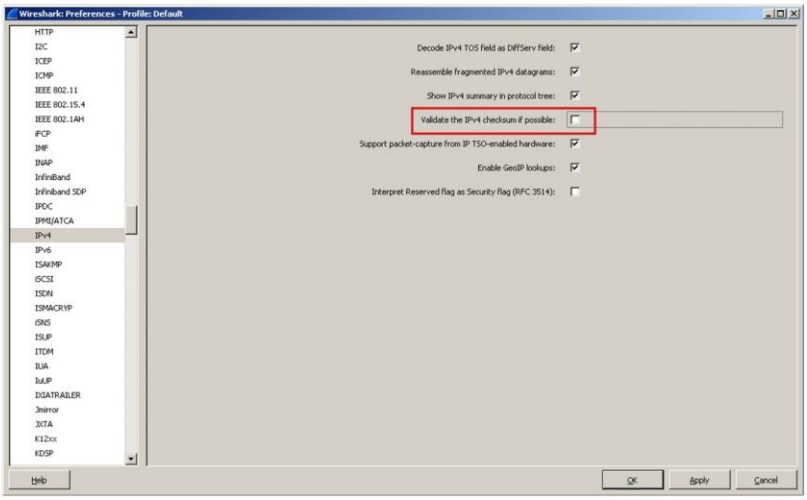
Если нужно быстро отфильтровать лишние пакеты, достаточно сделать правый клик на пакете, выбрать меню Apply as Filter – Not selected и изменения сразу же вступят в силу. Если нужно еще что-то убрать, то в следующий раз выбирать “and not Selected”, и новое правило просто добавится к фильтру.

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

```
⊟ Header checksum: 0x0000 [incorrect, should be 0x6ad0 (maybe caused by "IP checksum offload"?)]
    [Good: False]
  ⊟ [Bad: True]
    ⊟ [Expert Info (Error/Checksum): Bad checksum]
        [Message: Bad checksum]
        [Severity level: Error]
        [Group: Checksum]
  Source: [redacted] ([redacted])
  Destination: [redacted] ([redacted])
```

Современные сетевые карты насколько умные, что сами считают контрольную сумму, зачем это делать на уровне стека TCP/IP программно, если можно делать аппаратно. А Wireshark выполняет так сказать native-перехват пакетов, 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 снифера указать, чтобы он не обращал внимание на это значение.

Аппаратные функции зачастую лучше программных, в основном из-за скорости обработки поэтому лучше изменить настройки самого снифера. Для этого нужно зайти в настройки (Edit — Preferences), затем Protocols – IPv4 – и снять флаг с “Validate IPv4 checksum if possible”.



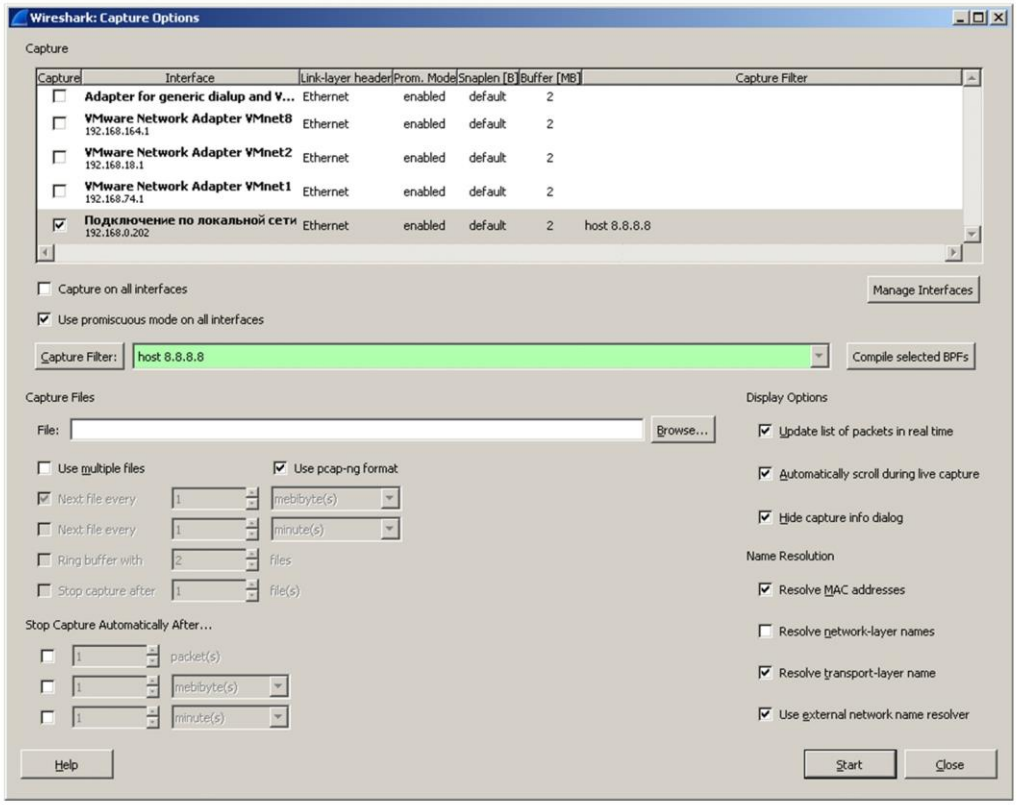

Перед тем как захватывать трафик нужно определиться с тем, что, собственно, нужно захватывать. Разместить анализатор трафика можно в нескольких местах, например:

- локально на своем хосте;
- организовать зеркалирование трафика на коммутаторе;
- подключаться непосредственно в интересующие места;
- или же использовать «отравление» протокола ARP (Arp-poison или Arp-spoofing).

### Фильтрация потока

Wireshark содержит два вида фильтров – захвата (Capture Filters) и отображения (Display Filters). Вначале рассмотрим Capture Filters. Как ясно 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 Capture, затем Options, и в поле Capture Filter набрать, например, host 8.8.8.8 (или, например, net 192.168.0.0/24)

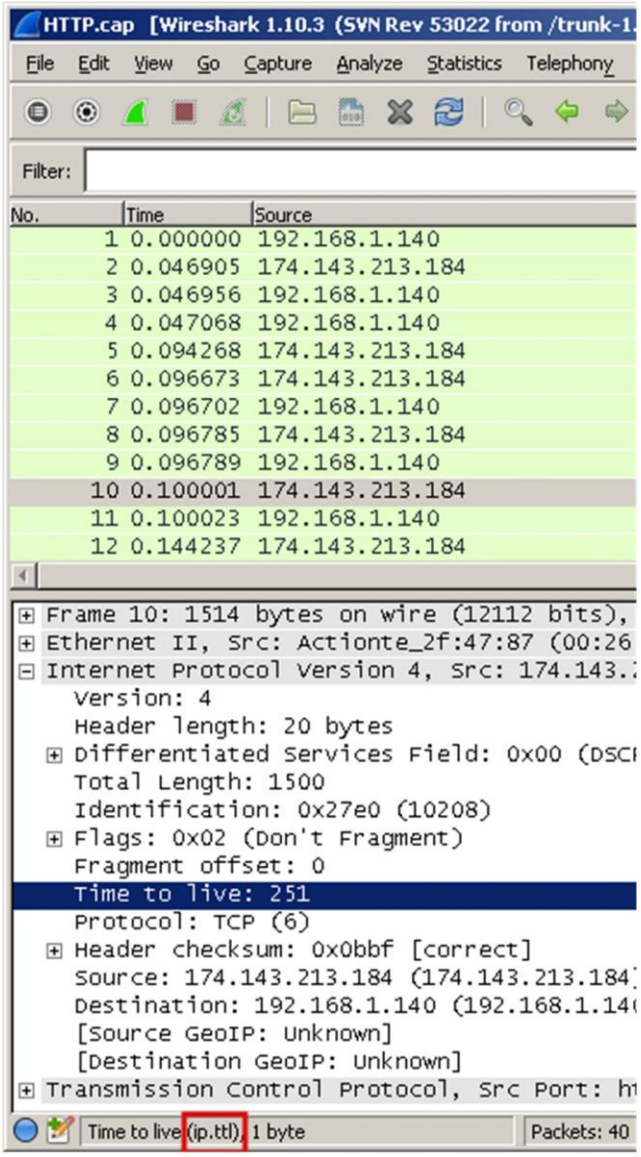



Так же, конечно, можно выбрать и заранее созданный фильтр (за это отвечает кнопка Capture Filter). В любом из вариантов фильтр появится возле интерфейса, можно жать Start. Теперь перейдем к Display Filters. Они фильтруют исключительно уже захваченный трафик.

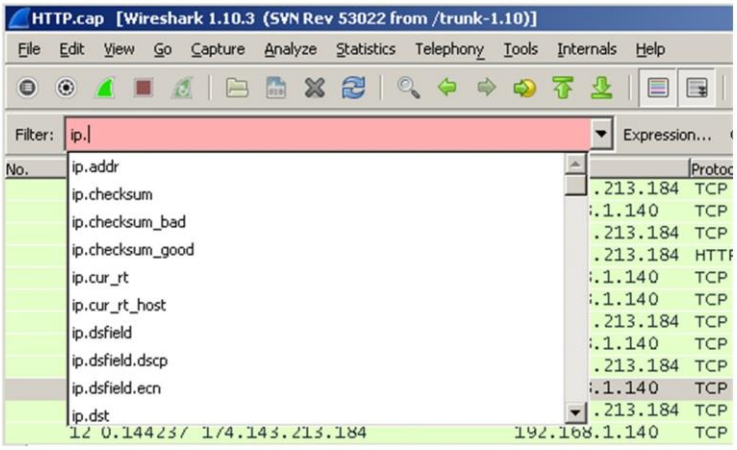
Можно фильтровать протоколы, адреса, специфические поля в протоколах. Операции, которые можно использовать при построении фильтров:

| 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 | регулярные выражения | frame matches “[Pp][Aa][Ss][Ss]” |
| contains | содержит | dns.resp.name contains google |

Чтобы увидеть поле по которому осуществляется фильтрация, 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:

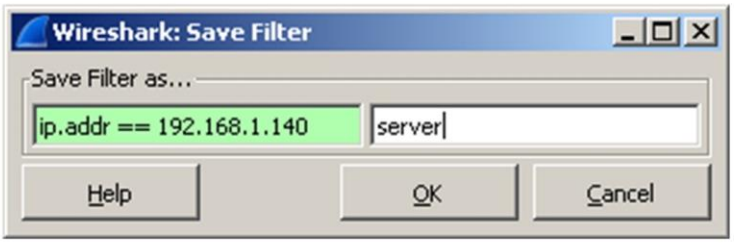



И видим, что для построения фильтра, нужно использовать выражение ip.ttl. Если начать набирать фильтр, то после точки автоматически появится список возможных значений:

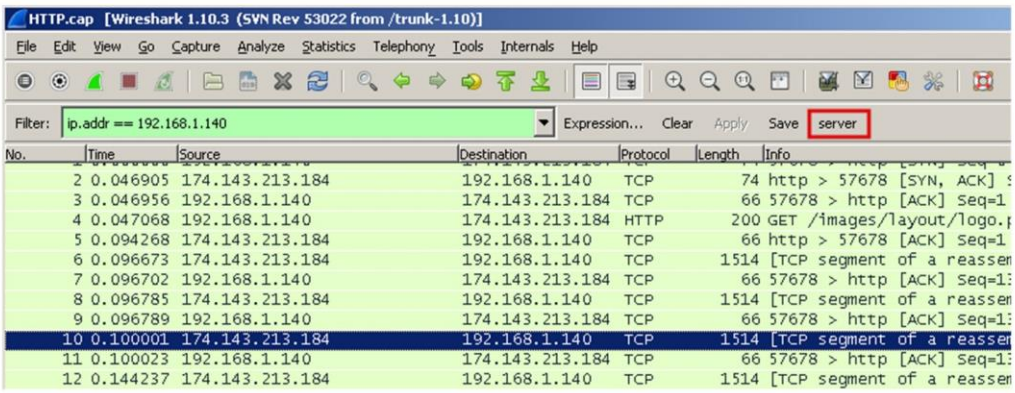



Чтобы применить фильтр, достаточно нажать enter или кнопку Apply. Само поле для ввода фильтра может менять цвет в зависимости от того, что было набрано.

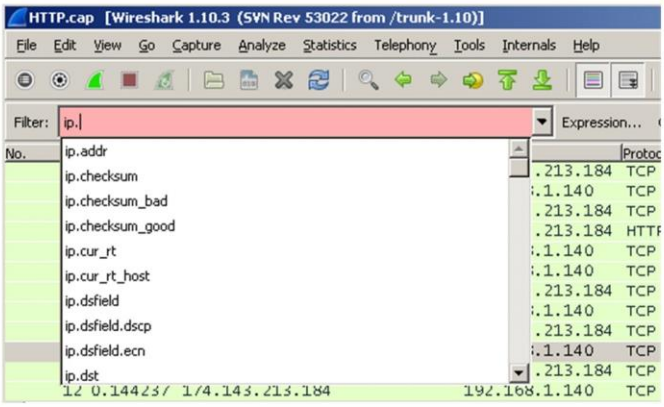
Зеленый цвет означает, что все в порядке. Красный — допущена ошибка, желтый — получен неожиданный результат, потому что существуют другие варианты написания фильтра (например можно написать ip.dst != 8.8.8.8 или же !ip.dst == 8.8.8.8, именно второй вариант более предпочтительный). Фильтры можно сохранять для дальнейшего использования, нажав кнопку Save, затем ввести произвольное название




и после нажатия на кнопку OK 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.

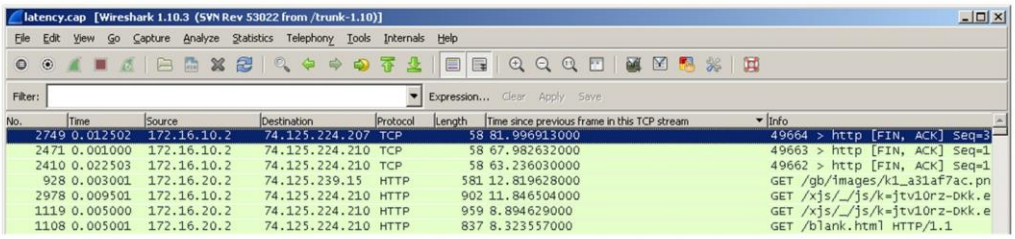
Как можно догадаться по названию — разница заключается в том, что в первом случае только появится в поле для ввода Display Filter, но не применится (удобно, если например, добавлять таким способом несколько фильтров, а затем сразу применить готовый результат), а во втором — сразу же и применится.

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.

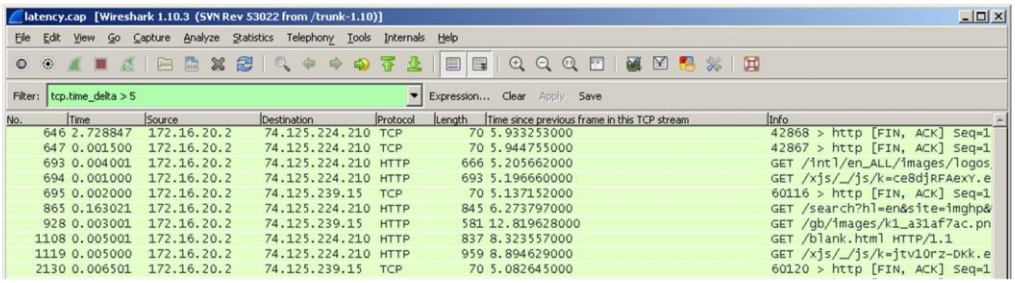
Достаточно частая ситуация, когда возникают жалобы на медленную работу сети, причин этого может быть множество. Попробуем разобраться, в чем может быть причина, и рассмотрим два способа. Первый состоит в добавлении колонки TCP delta.

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.



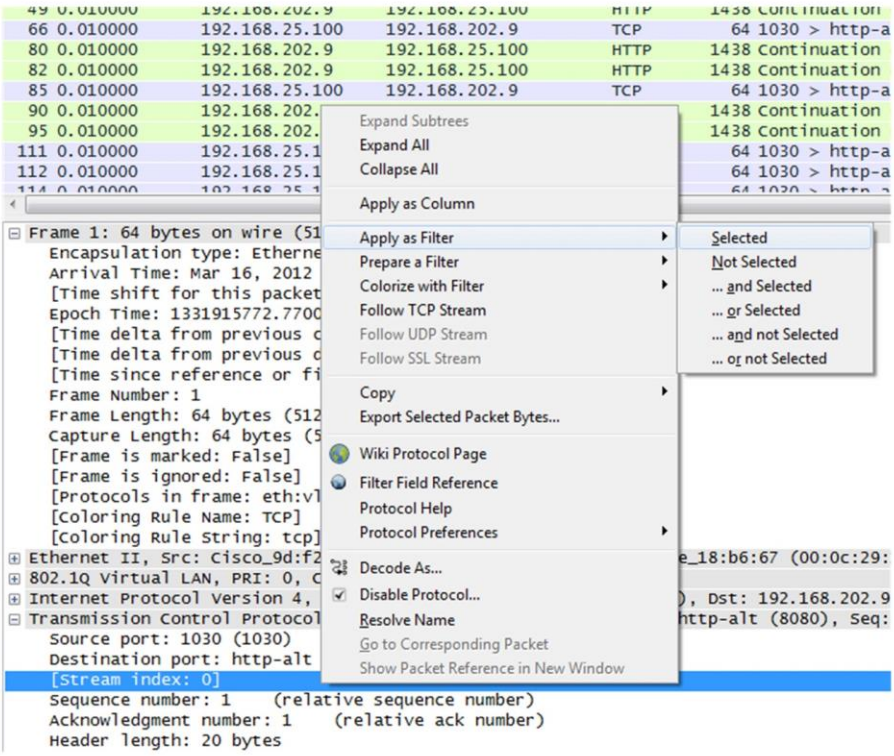
И сразу же рассмотрим второй способ.

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

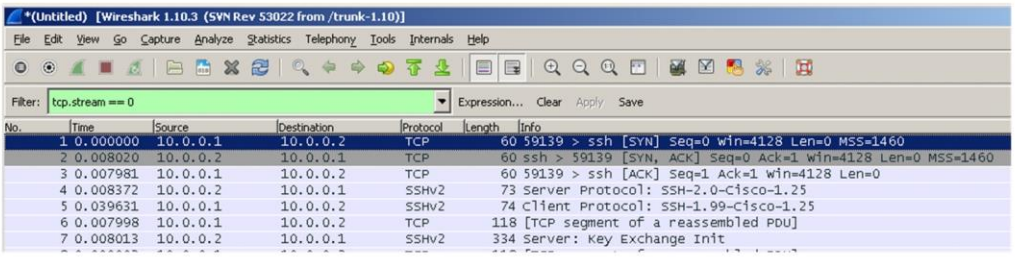



Если клиент делает запрос и получает ответ через 10 миллисекунд, и клиент говорит, что у него все медленно работает, то, возможно, проблема у самого клиента. Если же клиент делает запрос и получает ответ через 2-3 секунды, тут уже, возможно, проблема кроется в сети.

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.

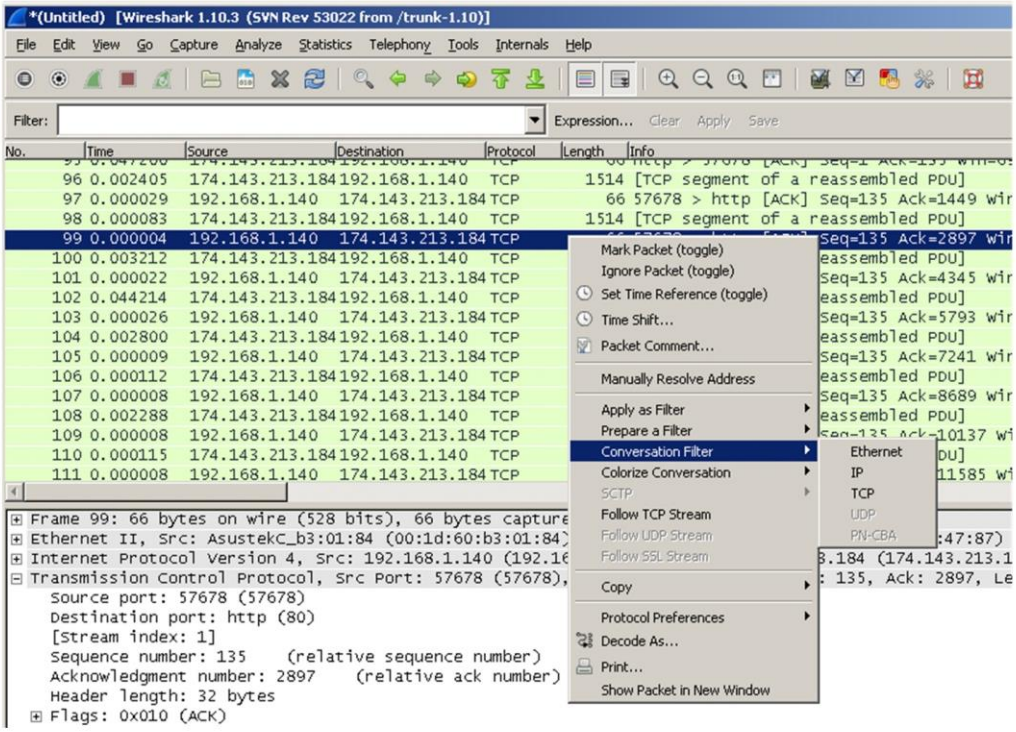


По нему можно сделать правый клик и создать фильтр.



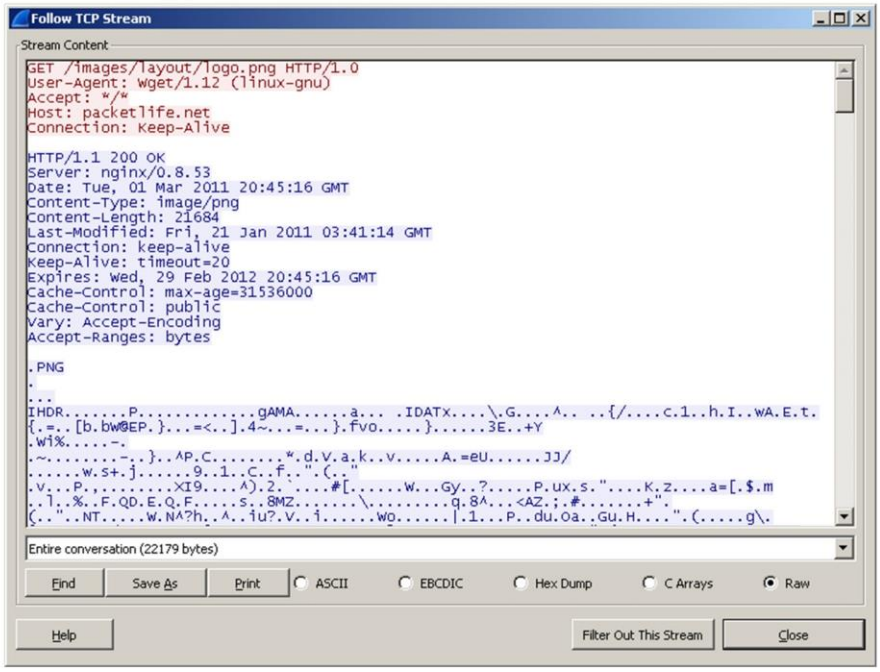

Таким образом можно фильтровать нужные соединения.

Еще один способ – сделать правый клик на самом пакете, выбрать Conversation Filter и создать фильтр для L2 L3 L4 уровня соответственно.

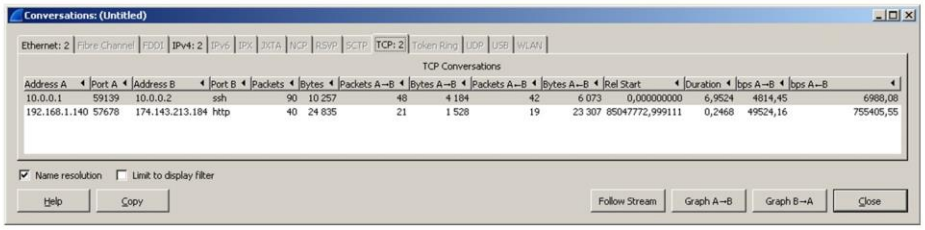



В итоге мы опять увидим взаимодействие двух хостов.

И третий вариант — 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.

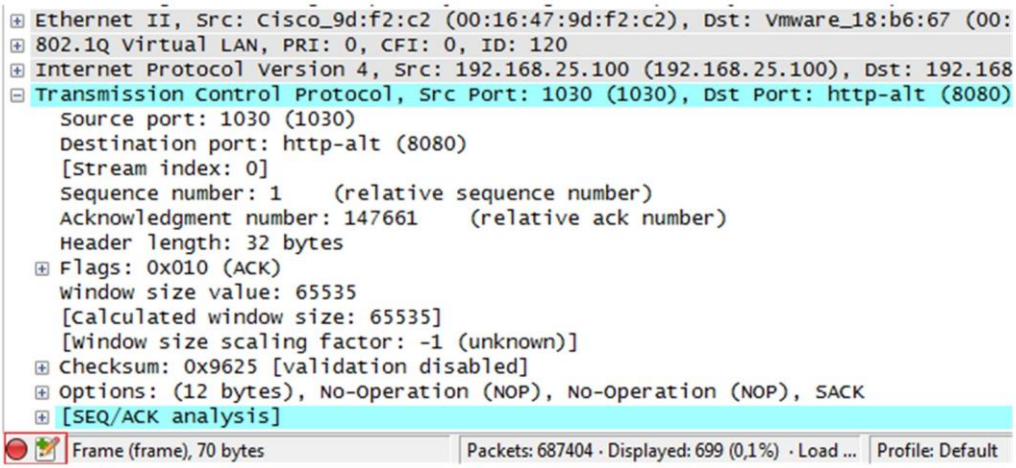



Если же зайти в меню Statistics – Conversations, то, выбирая закладки, можно увидеть статистику по таким “разговорам” и различные сессии, при этом можно отсортировать их по различным колонкам, например, по количеству переданных данных.

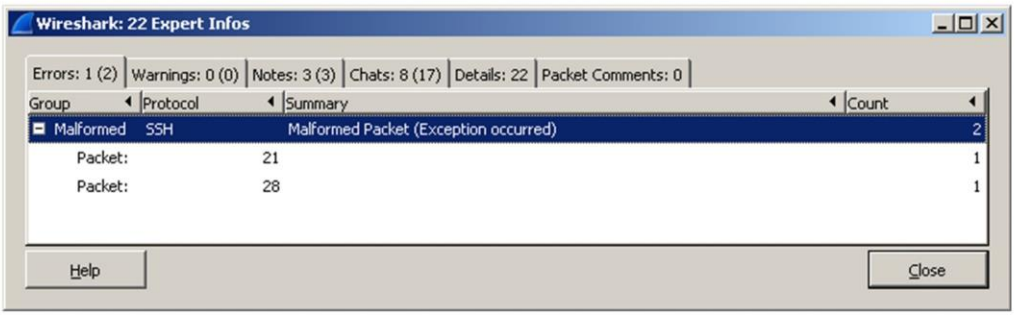



И прямо в этом окне можно правой кнопкой взывать контекстное меню и опять же применить как фильтр.

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:

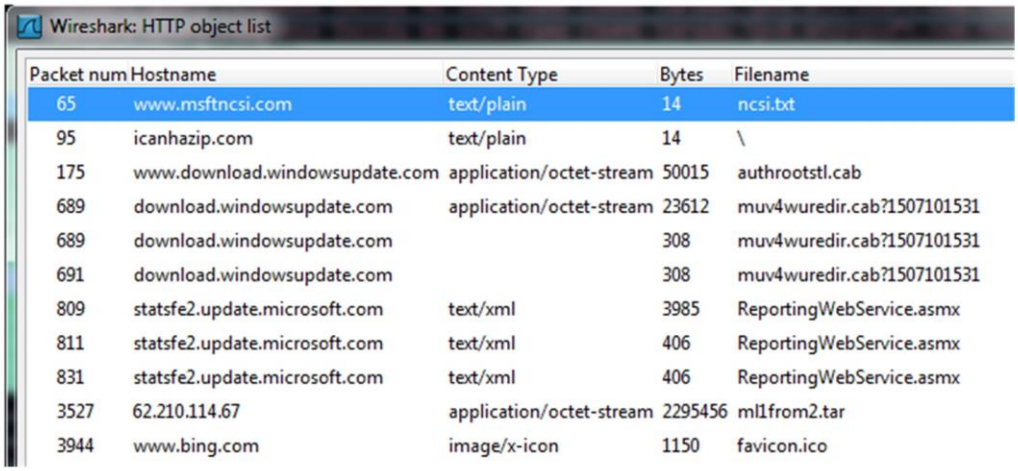
Ошибки — красный цвет
Предупреждения — желтый
Примечания — сине-зелёный (cyan)
Чат — серый

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.

## Извлечение данных из трафика

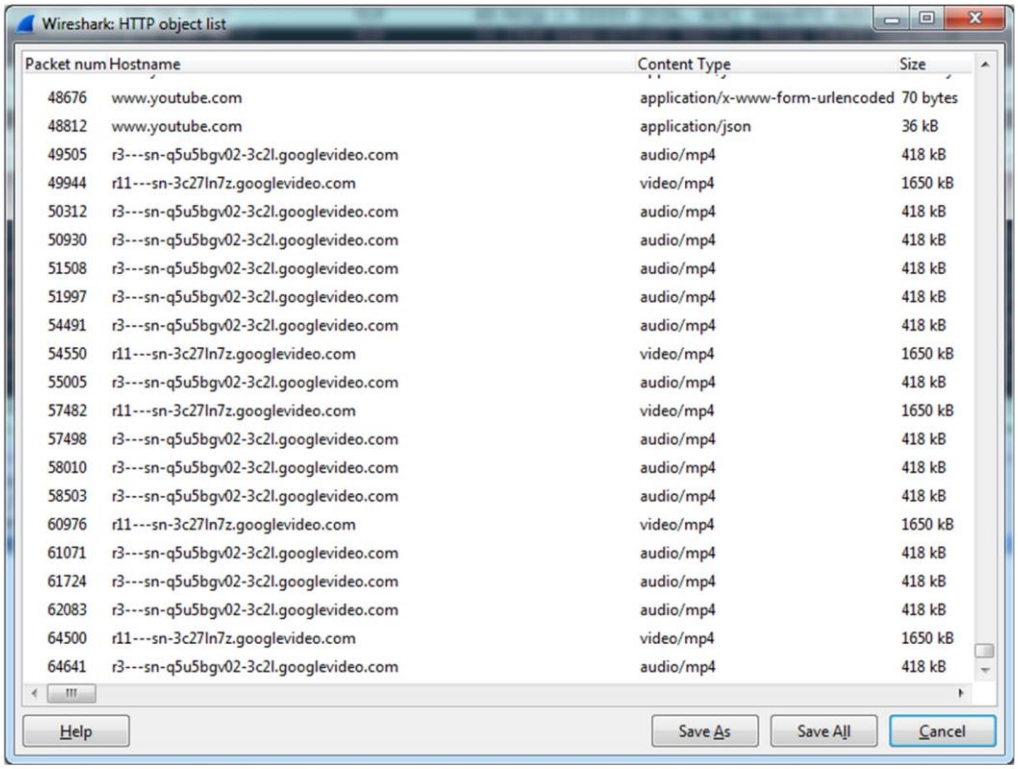
Вначале возьмем HTTP трафик. Сделаем правый клик по HTTP пакету — Protocol Preferences – и видим тут массу опций, которые непосредственно влияют на извлечение файлов из веб трафика. Для того чтобы увидеть, что можно извлечь из текущего дампа нужно перейти в меню File – Export Objects – HTTP.

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.



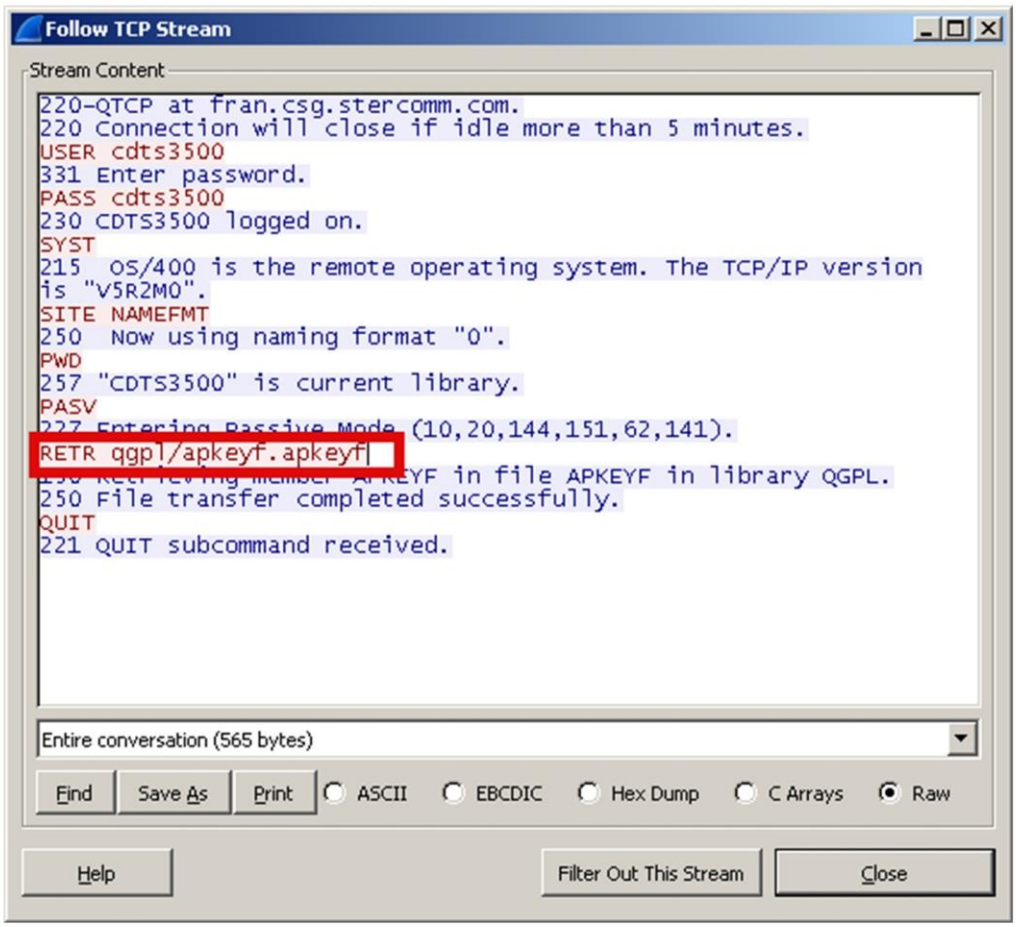

| Packet num | Hostname | Content Type | Bytes | Filename |
|---|---|---|---|---|
| 65 | www.msftncsi.com | text/plain | 14 | ncsi.txt |
| 95 | icanhazip.com | text/plain | 14 | \ |
| 175 | www.download.windowsupdate.com | application/octet-stream | 50015 | authrootstl.cab |
| 689 | download.windowsupdate.com | application/octet-stream | 23612 | muv4wuredir.cab?1507101531 |
| 689 | download.windowsupdate.com | | 308 | muv4wuredir.cab?1507101531 |
| 691 | download.windowsupdate.com | | 308 | muv4wuredir.cab?1507101531 |
| 809 | statsfe2.update.microsoft.com | text/xml | 3985 | ReportingWebService.asmx |
| 811 | statsfe2.update.microsoft.com | text/xml | 406 | ReportingWebService.asmx |
| 831 | statsfe2.update.microsoft.com | text/xml | 406 | ReportingWebService.asmx |
| 3527 | 62.210.114.67 | application/octet-stream | 2295456 | ml1from2.tar |
| 3944 | www.bing.com | image/x-icon | 1150 | favicon.ico |

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

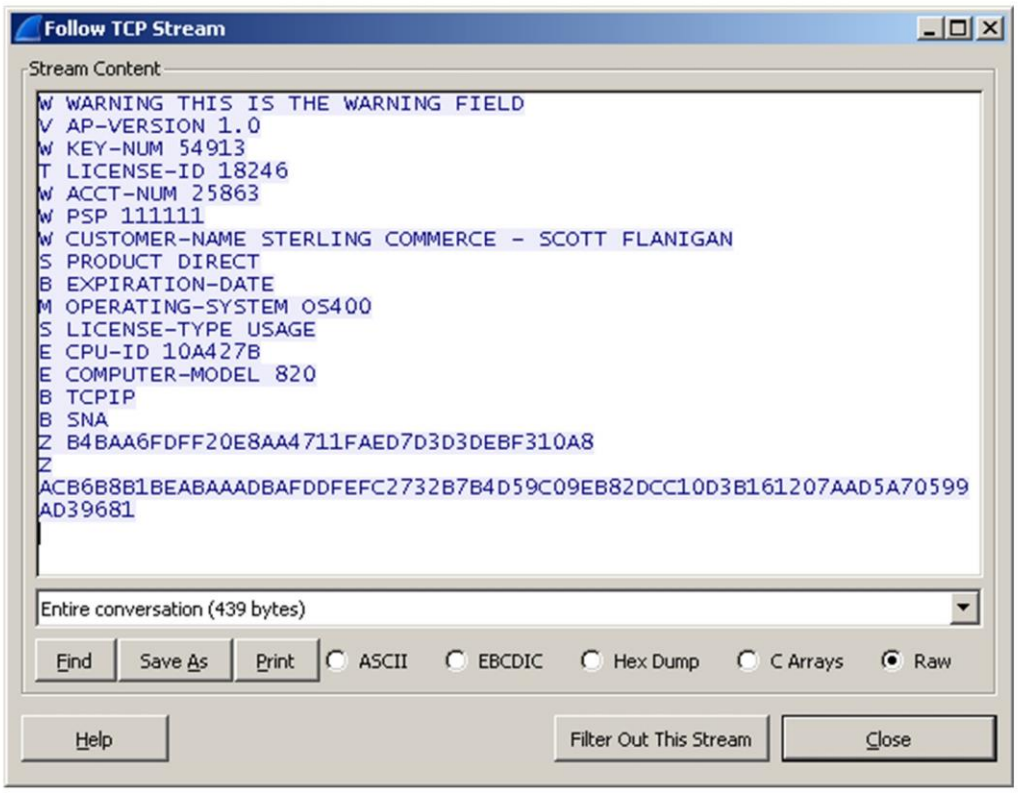



| Packet num | Hostname | Content Type | Size |
|---|---|---|---|
| 48676 | www.youtube.com | application/x-www-form-urlencoded | 70 bytes |
| 48812 | www.youtube.com | application/json | 36 kB |
| 49505 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 49944 | r11---sn-3c27ln7z.googlevideo.com | video/mp4 | 1650 kB |
| 50312 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 50930 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 51508 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 51997 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 54491 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 54550 | r11---sn-3c27ln7z.googlevideo.com | video/mp4 | 1650 kB |
| 55005 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 57482 | r11---sn-3c27ln7z.googlevideo.com | video/mp4 | 1650 kB |
| 57498 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 58010 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 58503 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 60976 | r11---sn-3c27ln7z.googlevideo.com | video/mp4 | 1650 kB |
| 61071 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 61724 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 62083 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |
| 64500 | r11---sn-3c27ln7z.googlevideo.com | video/mp4 | 1650 kB |
| 64641 | r3---sn-q5u5bgv02-3c2l.googlevideo.com | audio/mp4 | 418 kB |

Wireshark умеет вытаскивать файлы и с протокола FTP. Для этого можно использовать знакомый уже Follow TCP Stream. В итоге отобразится только обмен по протоколу FTP, в котором нужно будет найти строку RETR, что собственно и будет означать передачу файла.

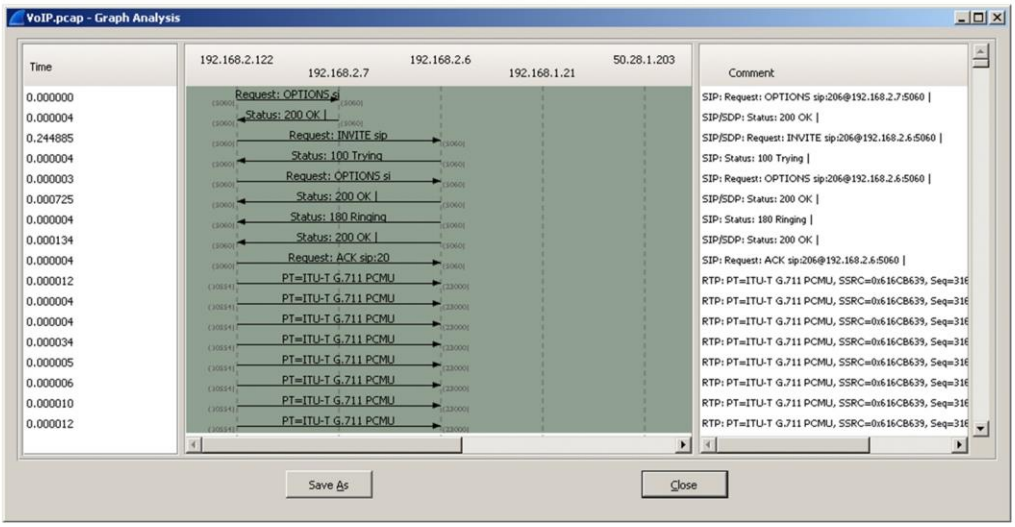



Затем опускаемся дальше, находим пакеты уже непосредственно с файлом (FTP-DATA) и опять выбираем Follow TCP Stream, видим содержимое файла, жмем Save As и сохраняем.

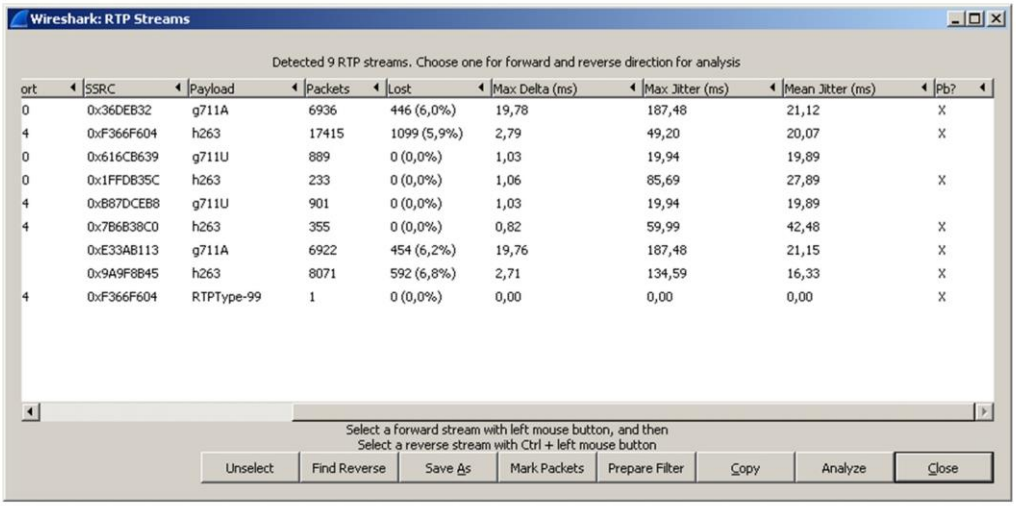



Wireshark имеет несколько встроенных функций для работы с VoIP. Он поддерживает массу голосовых протоколов — SIP, SDP, RTSP, H.323, RTCP, SRTP и другие. И, конечно же, умеет перехватывать и сохранять голосовой трафик для дальнейшего прослушивания.

Этот функционал как нельзя лучше подойдет для траблшутинга в сетях Voice over IP. Меню Statistics — Flow Graph покажет наглядную картину, как происходил весь обмен пакетами.



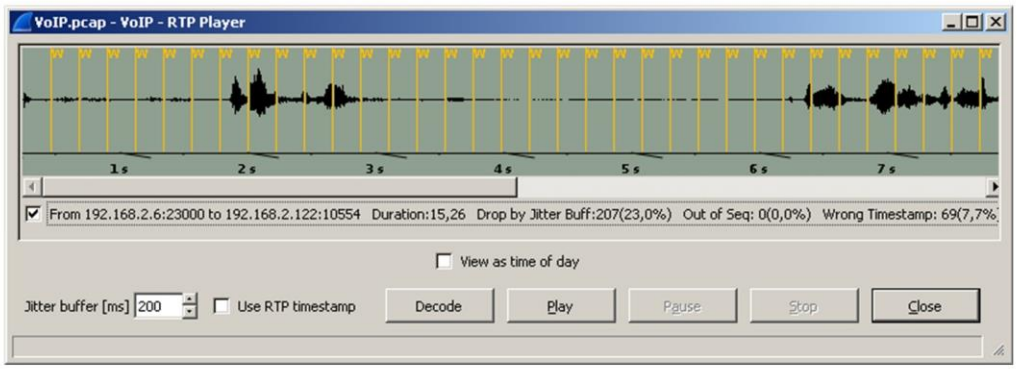

Меню Telephony отведено целиком для работы с голосовым трафиком. Например, Telephony – RTP – Show All Streams покажет подробно, что происходило с RTP, в частности jitter (параметр, который, вероятно, самый важный в голосе), что иногда сразу скажет о наличии проблем.



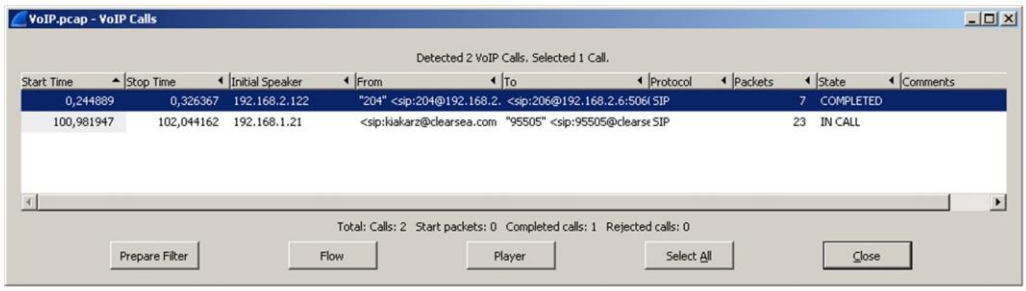

| ort | SSRC | Payload | Packets | Lost | Max Delta (ms) | Max Jitter (ms) | Mean Jitter (ms) | Pb? |
|---|---|---|---|---|---|---|---|---|
| 0 | 0x36DEB32 | g711A | 6936 | 446 (6,0%) | 19,78 | 187,48 | 21,12 | X |
| 4 | 0xF366F604 | h263 | 17415 | 1099 (5,9%) | 2,79 | 49,20 | 20,07 | X |
| 0 | 0x616CB639 | g711U | 889 | 0 (0,0%) | 1,03 | 19,94 | 19,89 | |
| 0 | 0x1FFDB35C | h263 | 233 | 0 (0,0%) | 1,06 | 85,69 | 27,89 | X |
| 4 | 0xB87DCEB8 | g711U | 901 | 0 (0,0%) | 1,03 | 19,94 | 19,89 | |
| 4 | 0x7B6B38C0 | h263 | 355 | 0 (0,0%) | 0,82 | 59,99 | 42,48 | X |
| | 0xE33AB113 | g711A | 6922 | 454 (6,2%) | 19,76 | 187,48 | 21,15 | X |
| | 0x9A9F8B45 | h263 | 8071 | 592 (6,8%) | 2,71 | 134,59 | 16,33 | X |
| 4 | 0xF366F604 | RTPType-99 | 1 | 0 (0,0%) | 0,00 | 0,00 | 0,00 | X |

Нажав на кнопку “Analyze”, можно открыть окно RTP stream Analysis – и, выбрав там поток, можно его даже проиграть, используя кнопку Player. Сначала отроется окно проигрывателя, в котором вначале нужно установить подходящее значение jitter и использовать кнопку decode.

Появится нечто похожее на анализатор спектра, в котором можно отметить требуемый разговор, и после этого кнопка Play станет активной.




Так же существует еще один способ прослушивания голосовых звонков — можно зайти в меню Telephony – VoIP Calls.




Откроется окно со списком совершенных звонков, где опять же можно нажать кнопку player, отменить нужные разговоры флажками и нажать play. Для того чтобы добиться приемлемого качества звучания, потребуется проиграться со значением поля jitter buffer, меняя его значение.

В конце рассмотрим как выглядит сканирование портов. Смотрим на дамп и видим, что вначале происходит ARP запрос и затем непосредственно начинается сканирование. Адрес нашего маршрутизатора 192.168.10.11, сканирование идет с адреса 192.168.10.101

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 14 | 24.963428000 | vmware_33:66:56 | Broadcast | ARP | 42 | Who has 192.168.10.11? Tell 192.168.10.101 |
| 15 | 24.972428000 | c2:00:15:4c:00:00 | vmware_33:66:56 | ARP | 60 | 192.168.10.11 is at c2:00:15:4c:00:00 |
| 17 | 37.985172000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > epmap [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 18 | 37.985172000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > imap [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 19 | 37.985172000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > ssh [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 20 | 37.985172000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > pptp [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 21 | 37.985172000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > ddi-tcp-1 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 22 | 37.985172000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > blackjack [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 23 | 37.986173000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > https [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 24 | 37.986173000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > sunrpc [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 25 | 37.986173000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > pop3 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 26 | 37.986173000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > ftp [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 27 | 38.023175000 | 192.168.10.11 | 192.168.10.101 | TCP | 60 | epmap > 41956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 28 | 38.026175000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > smux [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 29 | 38.026175000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > h323hostcall [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 30 | 38.033175000 | 192.168.10.11 | 192.168.10.101 | TCP | 60 | imap > 41956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 31 | 38.036175000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > pop3s [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 32 | 38.036175000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > smtp [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 33 | 38.043176000 | 192.168.10.11 | 192.168.10.101 | TCP | 60 | ssh > 41956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 34 | 38.046176000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > microsoft-ds [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 35 | 38.046176000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > ms-wbt-server [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 36 | 38.053176000 | 192.168.10.11 | 192.168.10.101 | TCP | 60 | pptp > 41956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 37 | 38.056177000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > ident [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 38 | 38.056177000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > mysql [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 39 | 38.063177000 | 192.168.10.11 | 192.168.10.101 | TCP | 60 | ddi-tcp-1 > 41956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 40 | 38.066177000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > domain [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 41 | 38.066177000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > rfb [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 42 | 38.073177000 | 192.168.10.11 | 192.168.10.101 | TCP | 60 | blackjack > 41956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |

Это, так называемое, SYN сканирование, когда идут SYN-пакеты на указанный диапазон портов. Так как большинство портов закрыто, маршрутизатор отвечает пакетами RST, ACK. Пролистав чуть ниже видим, что открыт telnet (tcp 23).

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 44 | 38.076178000 | 192.168.10.101 | 192.168.10.11 | TCP | 58 | 41956 > telnet [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 90 | 38.233187000 | 192.168.10.11 | 192.168.10.101 | TCP | 60 | telnet > 41956 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=1460 |
| 91 | 38.233187000 | 192.168.10.101 | 192.168.10.11 | TCP | 54 | 41956 > telnet [RST] Seq=1 Win=0 Len=0 |

На это указывает то, что маршрутизатор ответил пакетом SYN, ACK. К слову, для фильтрации портов в снифере можно использовать конструкции вида: tcp.srcport, tcp.dstport и tcp.port. Для протокола UDP всё аналогично — udp.srcport, udp.dstport, udp.port.

## Tcpdump

Tcpdump — главный инструмент сетевого анализа для специалистов по информационной безопасности. Утилита tcpdump относится к числу так называемых «сниферов» — программ предназначенных для перехвата сетевого трафика. Разработан Группой сетевых исследований (Network Reseach Group, NRG) Отдела информационных и вычислительных технологий (Information and Computing Sciences Division, ICSD) в Национальной лаборатории Лоренс Беркли (Lawrence Berkeley National Laboratory, LBNL).

Tcpdump работает при помощи интерфейса bpf (Berkeley Packet Filter). Если поддержку этого устройства отключить, сниффинг в BSD-UNIX окажется невозможен. Права на запуск программы tcpdump определяются правами доступа к устройсву bpf (/dev/bpf0). Эти права можно регулировать через devfs. Если вы предоставляете, например, группе operator права на чтение из этого устройства, то это значит, что все члены этой группы смогут перехватывать любой трафик, в том числе трафик суперпользователя.

Если программа tcpdump вызвана для прослушивания некоторого интерфейса, она переводит его в «promiscuous mode» — «неразборчивый режим». В этом режиме интерфейс ловит вообще все пакеты, которые до него добрались, а не только пакеты адресованные непосредственно ему. Таким образом, если сеть собрана не на коммураторах (switch), а на репитерах (hub), то tcpdump позволит перехватить трафик между посторонними машинами. Сказанное не означает, что перехват трафика невозможен в сети собранной на коммутаторах. Впрочем, интерфейс можно и не переводить в promiscous mode, если передать программе аргумент -p.

Ниже приведены несколько параметров, которые можно использовать при настройке tcpdump.

Опции

-i any: прослушивать все интерфейсы, чтобы увидеть, видите ли вы какой-либо трафик.

-i eth0: прослушивать интерфейс eth0.

-D: Показывать список доступных интерфейсов

-n: отображать IP адреса вместо имени хостов.

-nn: отображать IP адреса и номера портов вместо имени хостов и названия протоколов.

-q: показывает минимальное количество информации о пакете.

-t: не отображать метку времени в каждой строке.

-tttt: задает вывод временных меток в принятом по умолчанию формате для каждой строки.

-X: Показывать содержимое пакета как в шестнадцатеричной кодировке, так и в ASCII.

-XX: То же, что и -X, но также показывает ethernet header.

-v, -vv, -vvv: увеличить количество возвращаемой информации о пакетах.

-c: только получить x количество пакетов, а затем остановить.

-s: определить длину snaplength (размер) захвата в байтах. Используйте -s0, чтобы получить все, если только вы намеренно не захватили меньше.

-S: Печатать абсолютные порядковые номера.

-e: Получить ethernet header.

-q: Показать минимальную информацию о пакете.

-E: Расшифровать трафик IPSEC, предоставив ключ шифрования.

[Значение snaplength, по умолчанию для tcpdump 4.0, изменилось с 68 до 96 байтов. Хотя это даст вам увидеть больше информации о пакете, но это будет не вся информация. Используйте -s1514 или -s0, чтобы получить полный охват]

### Фильтрующие выражения tcpdump

В команде tcpdump возможно задать фильтрующее правило. Фильтрующее выражение состоит из параметра отбора пакетов и значения задаваемого параметра. В tcpdump выражения позволяют вам выделить различные типы трафика и найти именно то, что вы ищете. Освоение выражений и умение объединить их творчески — вот что делает по-настоящему мощным tcpdump.

Например, команда:

```
tcpdump -i eth0 host 10.0.0.1
```

имеет опцию -i eth0, задающую прослушиваемый интерфейс eth0 и фильтрующее выражение, заставляющее tcpdump отображать все пакеты, у которых в поле отправителя и в поле получателя стоит адрес 10.0.0.1. При этом, host - это параметр отбора, а 10.0.0.1 - это значение параметра.

Фильтрующие выражения можно комбинировать между собой с помощью дополнительных операторов and, or и not. С помощью этих операторов можно строить довольно сложные логические конструкции, группируя параметры в скобки. Например, команда:

```
tcpdump -i eth0 dst 10.0.0.5 and port 53
```

ограничит отображение только пакетами отправленные на хост 10.0.0.5 по протоколу DNS (порт 53).

Наиболее часто используемые фильтрующие параметры команды tcpdump:

dst хост

Проверяет, совпадает ли адрес получателя IP-пакета с указанным значением. Возможно задавать как IP, подсеть в формате 10.0.0.1/24, так и имя хоста.

src хост

Проверяет, совпадает ли адрес отправителя IP пакета с указанным значением. Возможно задавать как IP, подсеть в формате 10.0.0.1/24, так и имя хоста.

host хост

Проверяет, совпадает ли адрес отправителя или получателя с заданным значением. Возможно задавать как IP, подсеть в формате 10.0.0.1/24, так и имя хоста.

net имя_сети

Проверяется, находится ли адрес отправителя/получателя в заданной сети. Возможно указание сети в формате CIDR (например 10.0.0.1/22), либо указание имени сети, заданной в файле /etc/networks.

ip | arp | rarp | tcp | udp | icmp [хост]

Проверяет, принадлежит ли пакет одному из указанных протоколов и при указании адреса хоста проверяет, совпадает ли адрес отправителя\получателя с заданным. Возможно задавать как IP, подсеть в формате 10.0.0.1/24, так и имя хоста.

[tcp | udp] dst port номер_порта

Проверяется, принадлежит ли пакет протоколу TCP/UDP и равен ли порт назначения заданному. Можно указать номер порта, либо имя, заданное в файле /etc/services.

[tcp | udp] src port номер_порта

Проверяется, принадлежит ли пакет протоколу TCP/UDP и равен ли порт источника заданному. Можно указать номер порта, либо имя, заданное в файле /etc/services.

[tcp | udp] port номер_порта

Проверяется, принадлежит ли пакет протоколу TCP/UDP и равен ли порт назначения или источника заданному. Можно указать номер порта, либо имя, заданное в файле /etc/services.

ip broadcast

Проверяется, является ли IP пакет широковещательным.

ether { src | dst | host } MAC_адрес

Проверяется, принадлежит ли сетевой пакет источнику, назначению, источнику или назначению имеющему заданный MAC_адрес.

ether broadcast

Проверяется, является ли ARP-пакет широковещательным.

Далее рассмотрим следующие примеры команды для выполнения некоторых функций:
Вывод информации о трафике по всем интерфейсам или заданному интерфейсу:

```
# tcpdump -i any
# tcpdump -i eth0
```

Подробный вывод без разрешения имен хостов или номеров портов, абсолютных порядковых номеров и удобочитаемых временных меток.

```
# tcpdump -ttttnnvvS
```

Один из самых распространенных запросов, это покажет вам трафик из 1.2.3.4, будь то источник или место назначения.

```
# tcpdump host 1.2.3.4
```

Шестнадцатеричный вывод полезен, когда вы хотите увидеть содержимое рассматриваемых пакетов, и его часто лучше всего использовать, когда вы изолируете несколько кандидатов для более тщательного изучения.

```
# tcpdump -nnvXSs 0 -c1 icmp
tcpdump: listening on eth0, link-type EN10MB (Ethernet), 23:11:10.370321 IP
(tos 0x20, ttl 48, id 34859, offset 0, flags [none], length: 84)
69.254.213.43 > 72.21.34.42: icmp 64: echo request seq 0
0x0000: 4520 0054 882b 0000 3001 7cf5 45fe d52b E..T.+..0.|.E..+
0x0010: 4815 222a 0800 3530 272a 0000 25ff d744 H."..50'..%..D
0x0020: ae5e 0500 0809 0a0b 0c0d 0e0f 1011 1213 .^.............
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .............!"#
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()*+,-./0123
0x0050: 3435 3637 4567
1 packets captured
1 packets received by filter
0 packets dropped by kernel
```

Выделить трафик на основе источника или назначения очень просто, используя src и dst.

```
tcpdump src 2.3.4.5
tcpdump dst 3.4.5.6
```

Чтобы найти пакеты, идущие в или из определенной сети, используйте опцию net. Вы можете комбинировать это с опциями src или dst.

```
tcpdump net 1.2.3.0/24
```

Вы можете найти определенный порт трафика, используя опцию port, за которой следует номер порта.

```
tcpdump port 3389
```

tcpdump src port 1025

Если вы ищете определенный тип трафика, вы можете использовать tcp, udp, icmp и многие другие.

tcpdump icmp

Вы также можете найти весь трафик IP6, используя опцию протокола.

tcpdump ip6

Вы также можете использовать диапазон портов, чтобы найти трафик.

tcpdump portrange 21-23

Если вы ищете пакеты определенного размера, вы можете использовать эти параметры. Вы можете использовать маленький, большой или их соответствующие символы, которые вы ожидаете от математики.

tcpdump less 32
tcpdump greater 64
tcpdump <= 128

Часто полезно сохранять результат пакетов в файл для анализа в будущем. Эти файлы известны как файлы PCAP (PEE-cap), и их можно обрабатывать сотнями различных приложений, включая сетевые анализаторы, системы обнаружения вторжений и, конечно же, самим tcpdump. Здесь мы пишем файл с именем capture_file, используя ключ -w.

tcpdump port 80 -w capture_file

Вы можете читать файлы PCAP с помощью ключа -r. Обратите внимание, что вы можете использовать все регулярные команды в tcpdump при чтении в файле; вы ограничены только тем фактом, что вы не можете захватывать и обрабатывать то, чего не существует в файле.

tcpdump -r capture_file

Теперь, когда мы увидели, что мы можем сделать с основами с помощью некоторых примеров, давайте рассмотрим некоторые более сложные вещи.

Для комбинирования параметров поиска можно использовать следующие команды:

AND: and or &&
OR: or or ||
EXCEPT: not or !

Вот несколько примеров комбинированных команд.

Давайте найдем весь трафик с 10.5.2.3 к любому хосту на порте 3389.

tcpdump -nnvvS src 10.5.2.3 and dst port 3389

Давайте посмотрим на весь трафик, идущий от 192.168.x.x, и двигающийся к сетям 10.x или 172.16.x.x, и мы покажем шестнадцатеричный вывод без имени хоста и один уровень дополнительной детализации.

tcpdump -nvX src net 192.168.0.0/16 and dst net 10.0.0.0/8 or172.16.0.0/16

Следующая команда покажет нам весь трафик, идущий к 192.168.0.2, который не является ICMP.

tcpdump dst 192.168.0.2 and src net and not icmp

Это покажет нам весь трафик от хоста, который не является трафиком SSH (если предположить использование порта по умолчанию).

tcpdump -vv src mars and not dst port 22

Как вы можете видеть, вы можете создавать запросы, чтобы найти практически все, что вам нужно. Ключ должен сначала определить именно то, что вы ищете, а затем построить синтаксис, чтобы изолировать определенный тип трафика.

Сложная группировка и специальные символы

Также имейте в виду, что при создании сложных запросов вам, возможно, придется группировать свои параметры, используя одинарные кавычки. Одиночные кавычки используются для того, чтобы указать tcpdump, что нужно игнорировать некоторые специальные символы — в этом случае то, что в скобках «()». Этот же метод можно использоваться для группировки с использованием других выражений, таких как хост, порт, сеть и т.д. Посмотрите на приведенную ниже команду.

# Traffic that's from 10.0.2.4 AND destined for ports 3389 or 22 (неверно)
tcpdump src 10.0.2.4 and (dst port 3389 or 22)

Если вы попытались выполнить эту очень полезную команду в другом случае, вы получите ошибку из-за скобок. Вы можете исправить это, выйдя из скобок (поставив перед каждой из них \) или поставив всю команду в одинарные кавычки:

# Traffic that's from 10.0.2.4 AND destined for ports 3389 or 22 (correct)
# tcpdump 'src 10.0.2.4 and (dst port 3389 or 22)'

### Фильтрация по TCP-флагам

Вы также можете захватывать трафик на основе определенных флагов (-ов) TCP. Фильтры ниже находят различные флаги в TCP, т.к. tcp[13] позволяет считать данные TCP заголовка по смещению 13.

tcpdump 'tcp[13] & 32!=0' # Показать все URGENT пакеты (URG)

tcpdump 'tcp[13] & 16!=0' # Показать все ACKNOWLEDGE пакеты (ACK)

tcpdump 'tcp[13] & 8!=0' # Показать все PUSH пакеты (PSH)

tcpdump 'tcp[13] & 4!=0' # Показать все RESET пакеты (RST)

tcpdump 'tcp[13] & 2!=0' # Показать все SYNCHRONIZE пакеты (SYN)

tcpdump 'tcp[13] & 1!=0' # Показать все FINISH (FIN) пакеты

tcpdump 'tcp[13]=18' # Показать все SYNCHRONIZE / ACKNOWLEDGE пакеты (SYNACK)

Необходимо заметить, что только флаги PSH, RST, SYN и FIN отображаются в выводе поля tcpdump. Флаги URG и ACK также отображаются, но они показаны в другом месте на выходе, т.е. не в поле flags.

В следующем примере показан другой способ захвата пакетов со специфическими наборами TCP-флагов.

tcpdump 'tcp[tcpflags] == tcp-syn' # Выбирать пакеты с флагами RST

tcpdump 'tcp[tcpflags] == tcp-rst' # Выбирать пакеты с флагами RST

tcpdump 'tcp[tcpflags] == tcp-fin' # Выбирать пакеты с флагами FIN с помощью параметра

Наконец, есть несколько примеров, чтобы выделить некоторый специфический трафик, например, неправильные / вероятно-вредоносные пакеты.

tcpdump 'tcp[13] = 6' # Пакеты с установленными флагами RST и SYN (этого не должно быть в нормальном трафике)

tcpdump 'tcp[32:4] = 0x47455420' # Поиск HTTP GET запроса (0x47455420 = «GET »)

tcpdump 'tcp[(tcp[12]>>2):4] = 0x5353482D' # Поиск ssh-соединения на любой порт (через баннер) 0x5353482D = «SSH-»

tcpdump 'ip[8] < 10' # Пакеты с TTL меньше 10 (как правило показывает проблему или использует traceroute)

tcpdump 'ip[6] & 128 != 0' # Пакеты с установкой evil bit