TAP All The Things

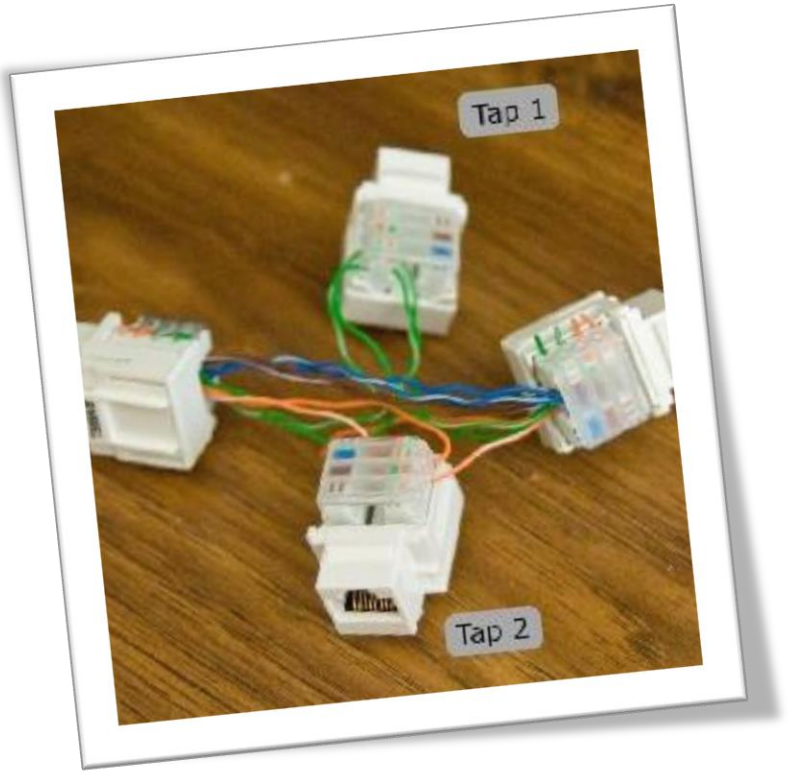
Using the BSidesDFW 9 Badge to View Network Traffic

BSIDES DFW 9

Presented by @alt_bier

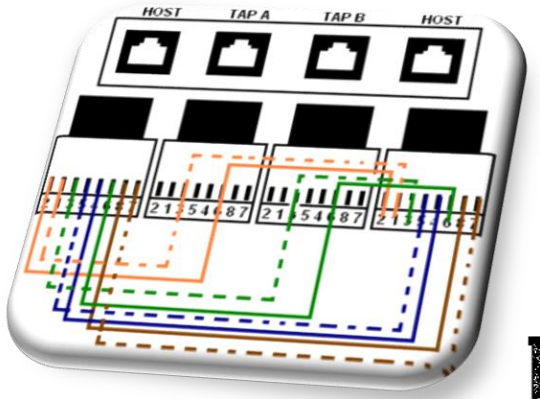
HARDWARE HACKING HV VILLAGE

What is a TAP?



A network TAP is an external monitoring device that mirrors the traffic that passes between two network nodes. A TAP (test access point) is a hardware device inserted at a specific point in the network to monitor data.

A passive Ethernet TAP like the BSidesDFW badge is limited to speeds of 100Mb since the 1Gb protocol would require active components in the TAP. The BSidesDFW badge uses capacitors to force the inline devices to auto-negotiate a speed of 100Mb or below.

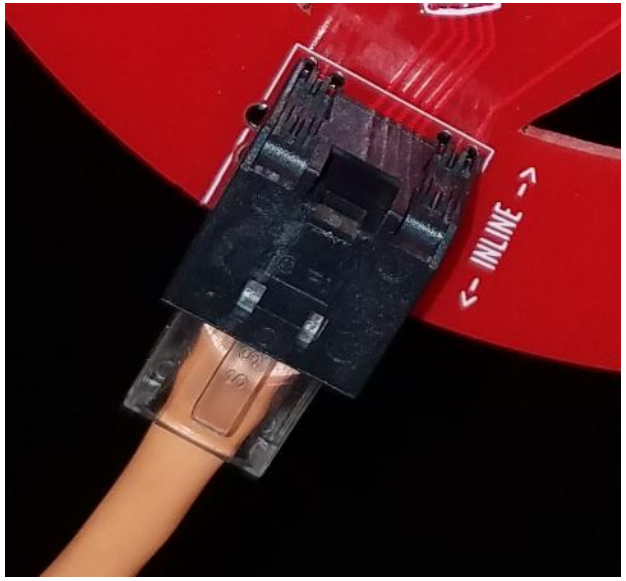


Connecting Devices to the TAP

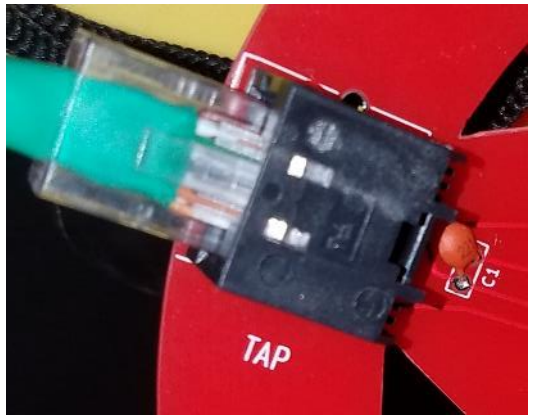
The BSidesDFW badge has two "Inline" RJ45 ports that should be connected between the two network nodes that you wish to monitor.

You should not need to modify speed settings on these "Inline" node connections as they will auto-negotiate to 100Mb or less.

For the Demo we will be connecting two laptops together using static IP's. In the real world you would most likely place the TAP in-between a single device and the switch port it connects to.



Connecting Devices to the TAP



The BSidesDFW badge has two "TAP" RJ45 ports that are hardwired to see traffic in one direction. To view traffic from both directions you must monitor both.

Since the "TAP" ports are only receiving traffic, auto-negotiation will not work. The monitor interfaces must have the port speed manually configured.

For the Demo we will be monitoring both "TAP" ports connected to the same laptop. In the real world you would most likely monitor one or the other direction, moving the monitor port as needed.

Wireshark Protocol Analyzer

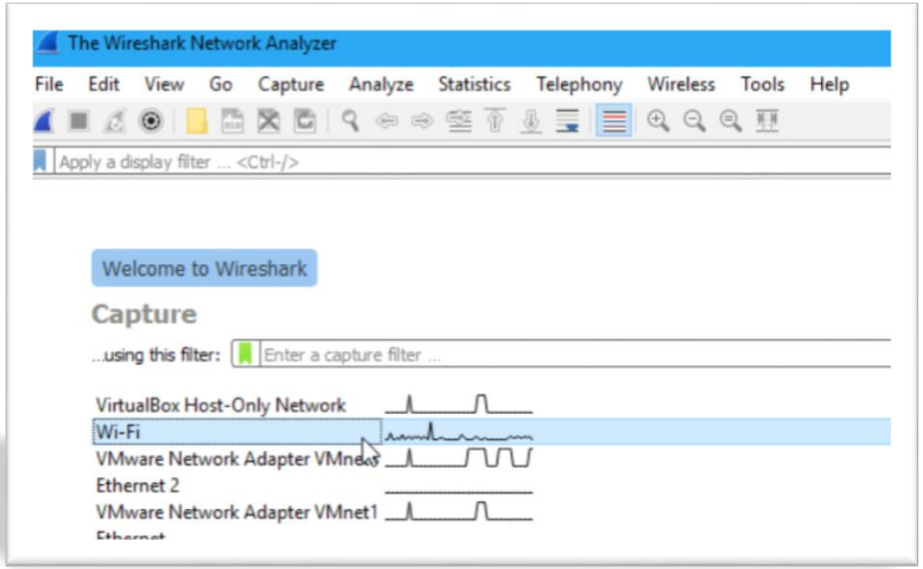
Wireshark is a free open source network protocol analyzer (formerly known as Ethereal) that captures packets in real time and display them in human-readable format. It is available for download here:
https://www.wireshark.org/

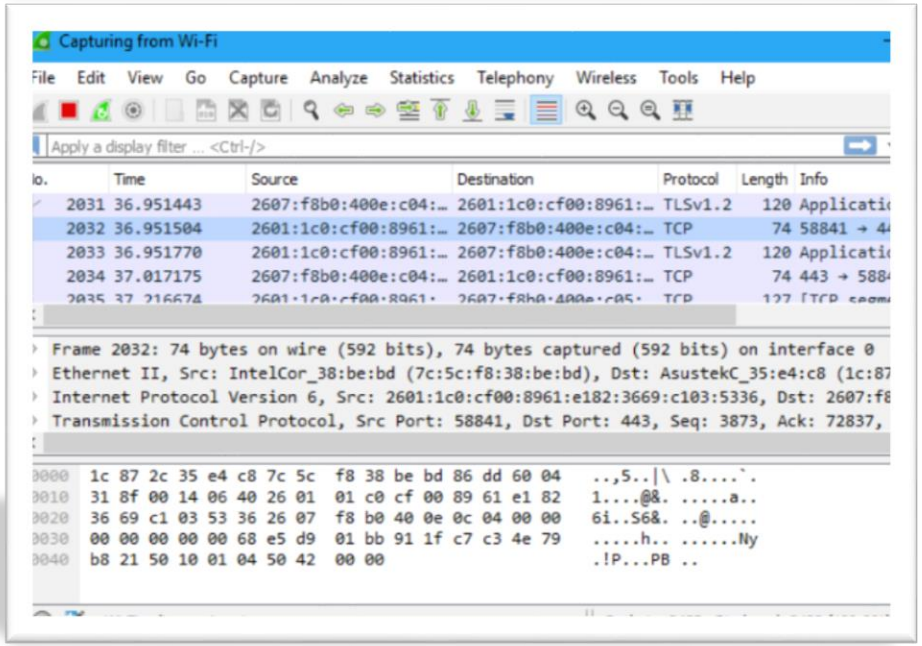
Wireshark will display the packets seen on one or more interfaces and “decode” this information providing details about the protocols being used and the state of the traffic in real time.

Wireshark Basics

After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface.

As soon as you click the interface's name, you'll see the packets start to appear in real time.

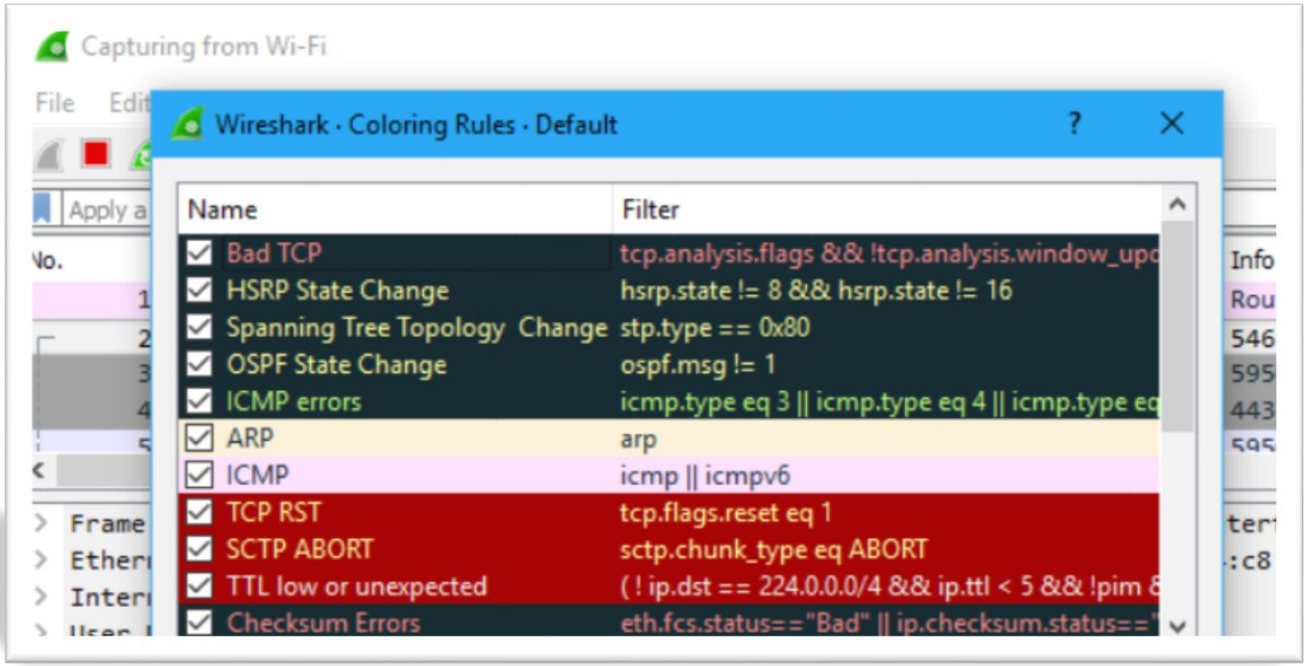




Wireshark Basics

Wireshark color coding: You'll probably see packets highlighted in a variety of different colors.

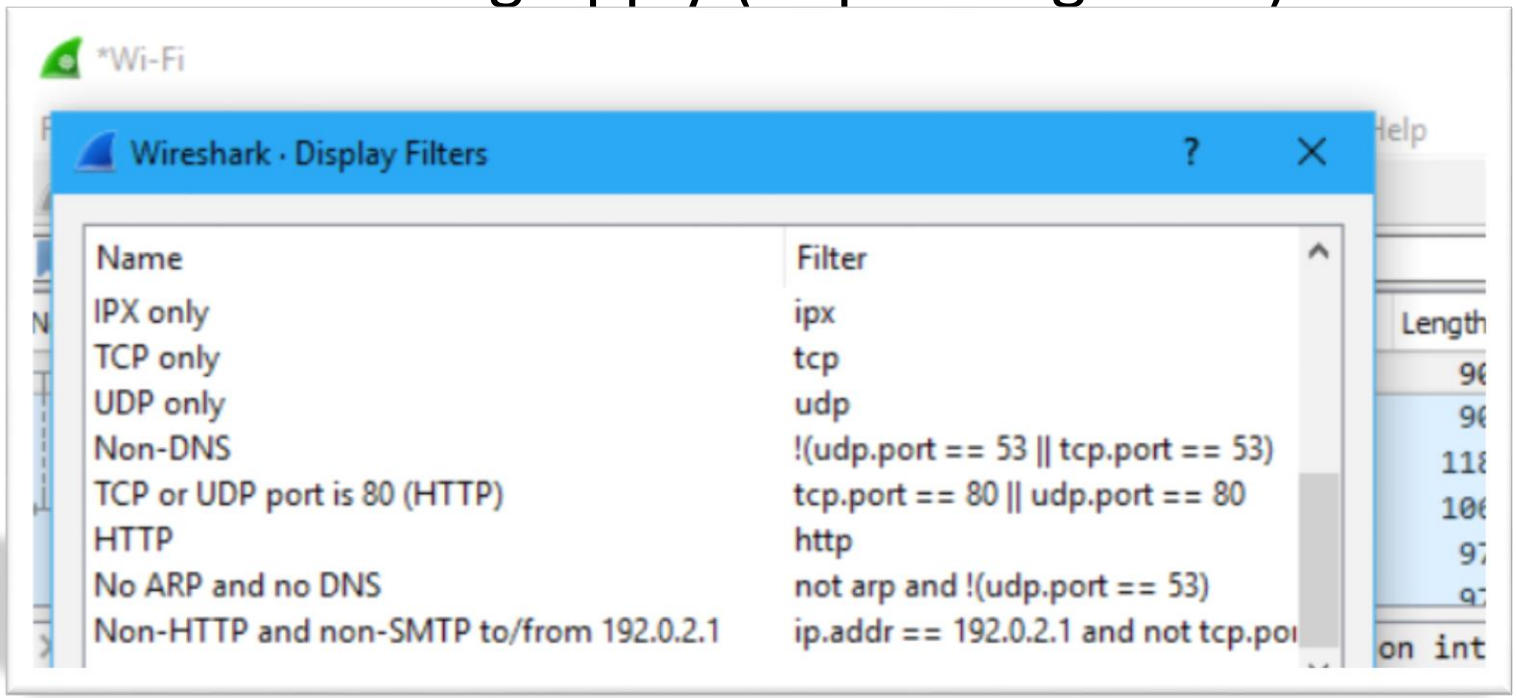
Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors



Wireshark Basics

Packet filtering: If you want to inspect something specific, such as an application or destination, you can apply a filter to only show this data.

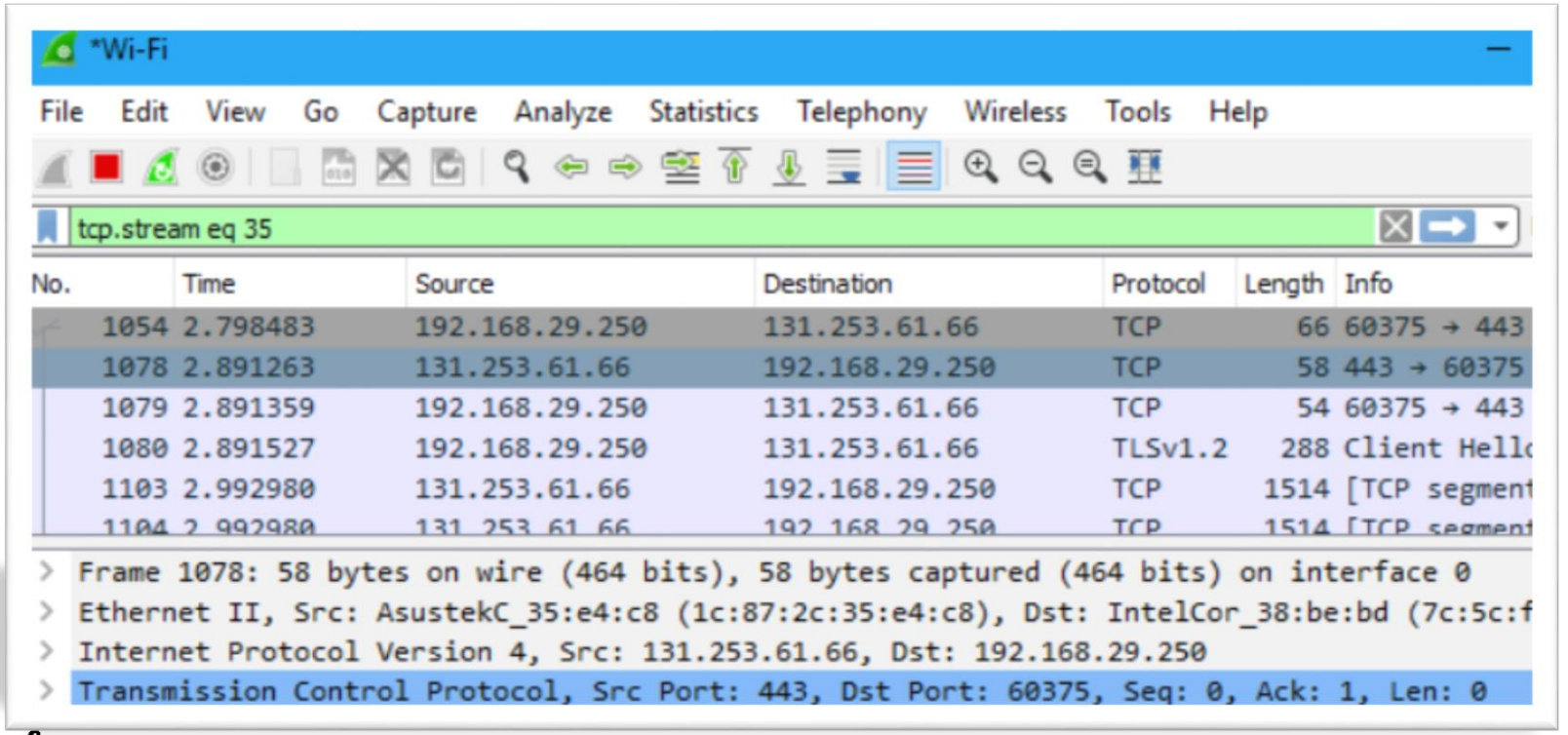
The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter).



Wireshark Basics

Follow a TCP Stream: Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

You'll see the full TCP conversation between the client and the server.

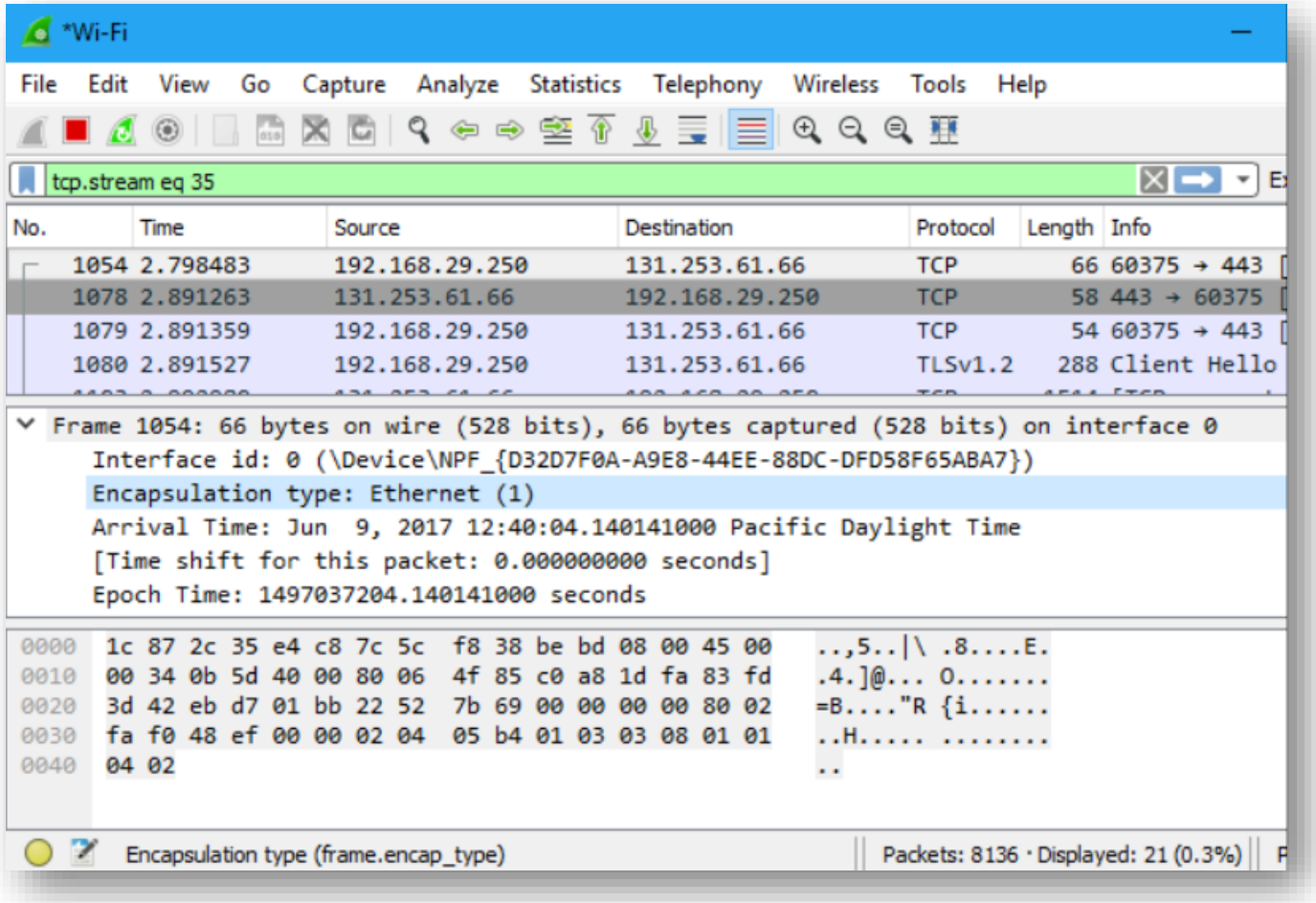


Wireshark Basics

Inspect a Packet: If you click on a packet to select it you can dig down to view its details in the middle portion of the screen.

Clicking on any of those detail lines expands its view and provides any additional detail available.

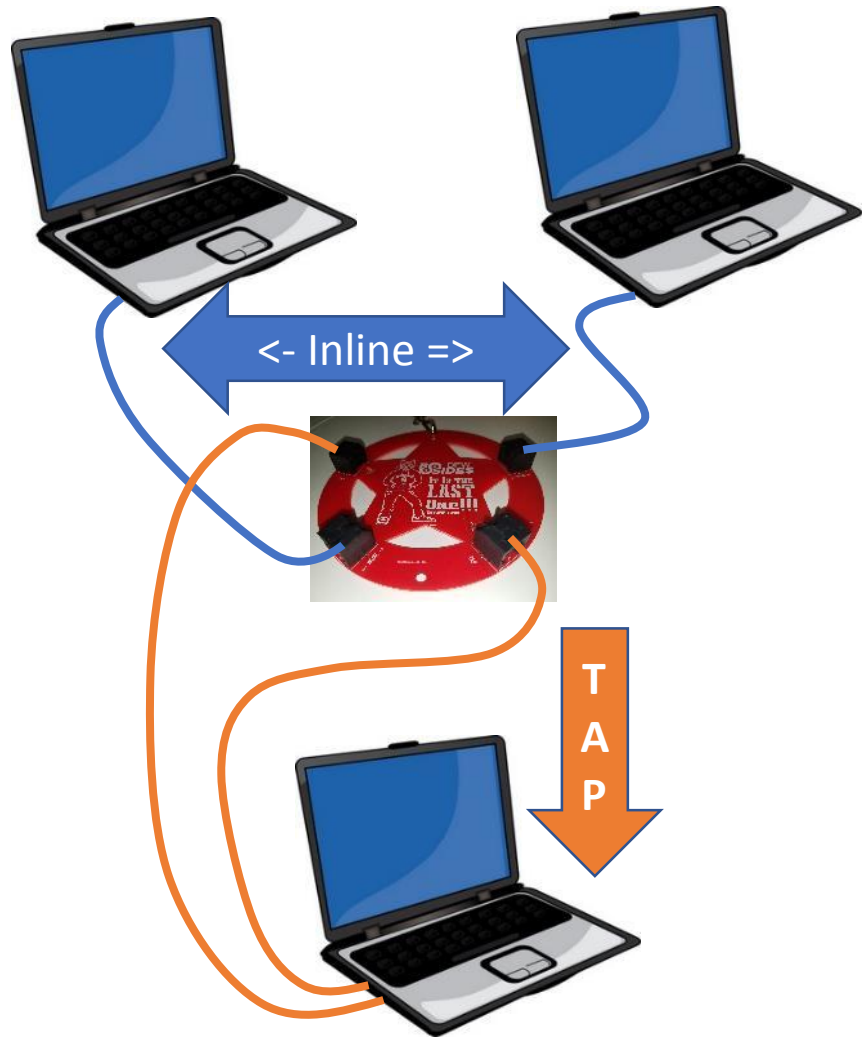
The bottom portion of the screen shows the raw packet data being decoded.



Demo Configuration

The "Inline" nodes are two laptops configured with static IP's using normal Ethernet auto-negotiation. The first is set up as a web server and the second as a client.

The "TAP" ports are connected to a monitor laptop that has been configured with no IP address information and manually assigned port speed with auto-negotiation disabled.



Demo Configuration - Web Server

The web server is an Ubuntu laptop running Apache/PHP to serve a simple web page that displays a random image.

The web page has been configured to prevent caching which might otherwise cause the client to serve local content instead of pulling from the server.

A bash script will display the web server logs during the demo.

index.php

```
<html>
 <head>
  <title>Test page</title>
  <meta Http-Equiv="Cache-Control" Content="no-cache">
  <meta Http-Equiv="Pragma" Content="no-cache">
  <meta Http-Equiv="Expires" Content="0">
  <meta Http-Equiv="Pragma-directive: no-cache">
  <meta Http-Equiv="Cache-directive: no-cache">
 </head>
 <body>

 <h1>Test Page</h1>

<?php
// display random image with inline php for a random image number
and time added to prevent caching
//
// Downloaded random images from https://picsum.photos/
// and stored them in a local images dir so this can work while
disconnected from internet
?>
<img src="/images/400/picsum.photos.<?php $random = rand(1,20);
echo $random; ?>.jpeg?nocache=<?php echo time(); ?>" />


 </body>
</html>
```

Demo Configuration - Web Client

The web client is an Ubuntu laptop running a Firefox web browser.

A Python script will be running to automate the web page calls to the server and display them in the Firefox browser during the demo.

Since a simple kill would cause Firefox to enter safe mode I used xdotool to close it cleanly.

testweb.sh

```
#!/bin/bash
fftabs=6
delay=10
while true
do
for ((i=1;i<=$fftabs;i++))
  do
    echo '-------------------------'
    date
    echo 'python -m webbrowser -t "http://192.168.192.10" 2>/dev/null'
    python -m webbrowser -t "http://192.168.192.10" 2>/dev/null
    echo "sleep $delay"
    sleep $delay
  done
  echo '-------------------------'
  date
  echo "Close Firefox after $fftabs tabs"
  ./xdotool-closeff.sh 2>/dev/null
  echo "sleep $delay"
  sleep $delay
done
```

xdotool-closeff.sh

```
#!/bin/bash

WID=`xdotool search "Mozilla Firefox" | head -1`
xdotool windowactivate --sync $WID
xdotool key --clearmodifiers ctrl+q
```

Demo Configuration - TAP Monitor

The TAP Monitor is an Ubuntu laptop running Wireshark to view the network traffic.

Wireshark will be running on both "TAP" monitor interfaces thus seeing both directions of the connection.

The application ethtool will be used to disable auto-negotiation and set the port speed on the monitor interfaces.

setnet100.sh

```
#!/bin/bash
echo "---------------"
sudo ethtool enp0s25
sudo ethtool -s enp0s25 autoneg off
sudo ethtool -s enp0s25 speed 100 duplex full
sudo ethtool enp0s25
echo "---------------"
sudo ethtool enx001060230c4e
sudo ethtool -s enx001060230c4e autoneg off
sudo ethtool -s enx001060230c4e speed 100 duplex full
sudo ethtool enx001060230c4e
echo "---------------"
```

Demo

The Demo will be set up in the Hardware Hacking Village area.

We encourage everyone to check out this demo. Since the laptops must be kept unlocked, we ask that this be done with eyes only and not hands.

The demo should look something like the picture to the right.