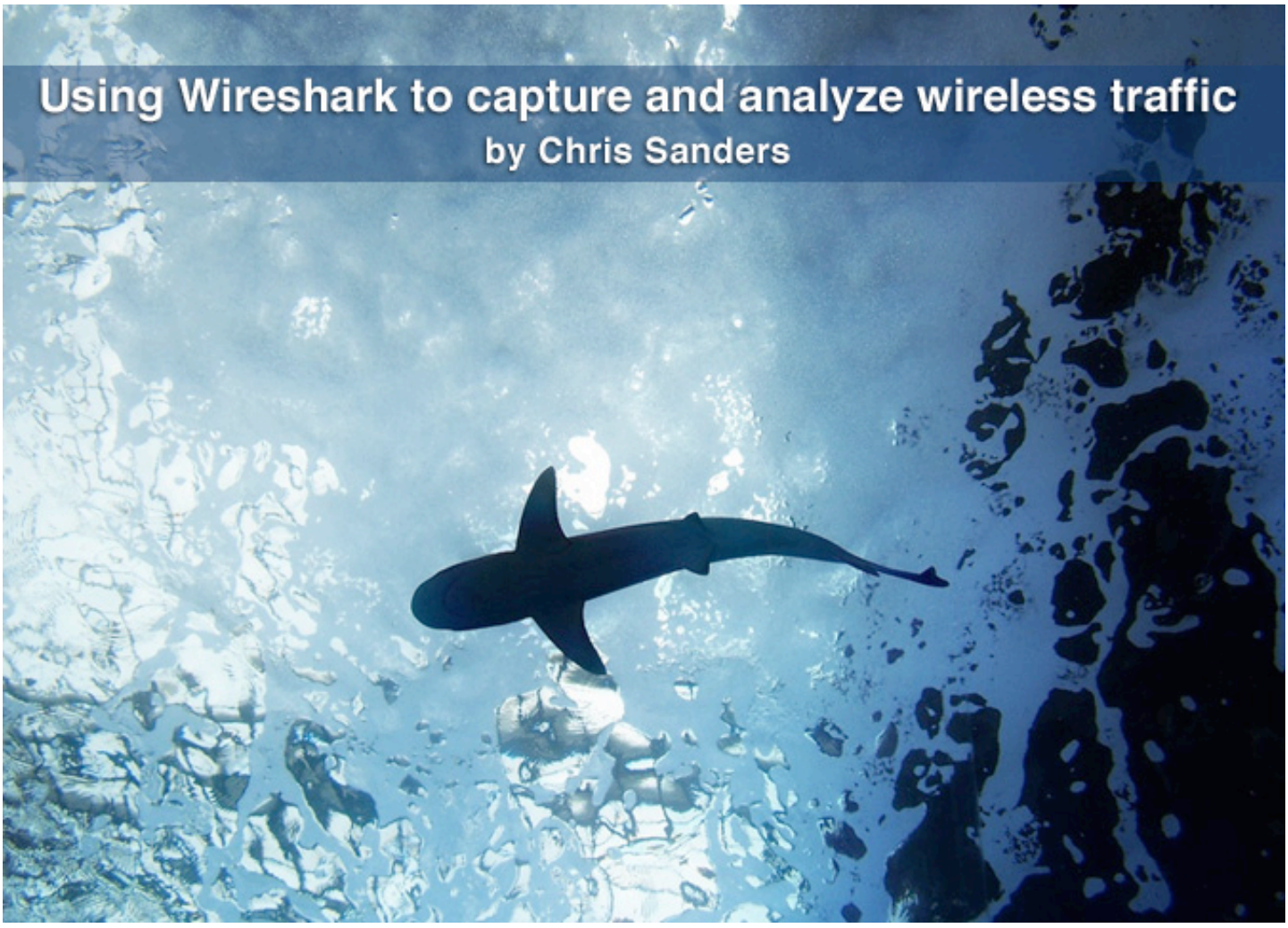


The tricky thing about a wireless network is that you can't always see what you're dealing with. In a wireless network, establishing connectivity isn't as simple as plugging in a cable, physical security isn't nearly as easy as just keeping unauthorized individuals out of a facility, and troubleshooting even trivial issues can sometimes result in a few expletives being thrown in the general direction of an access point. That being said, it shouldn't come as a surprise that analyzing packets from a wireless network isn't as uninvolved as just firing up a packet sniffer and hitting the capture button.

In this article I'm going to talk about the differences between capturing traffic on a wireless network as opposed to a wired network.

I'll show you how to capture some additional wireless packet data that you might not have known was there, and once you know how to capture the right data, I'm going to jump into the particulars of the 802.11 MAC layer, 802.11 frame headers, and the different 802.11 frame types.

The goal of this article is to provide you with some important building blocks necessary for properly analyzing wireless communications.

Wired vs. wireless networks

There are a lot of obvious differences between wireless and wired networks. On a wired network each node has its own individual cable allowing for predictable performance and a dedicated amount of bandwidth both upstream and downstream.

A wireless network is a shared medium meaning that all nodes on that network compete for bandwidth over a limited spectrum. It is because of this shared nature that a wireless network employs a different means of handling the transmission of data.

WIRED	WIRELESS
CSMA/CD	CSMA/CA
Dedicated bandwidth	Shared medium
Predictable	Performance decreases on load

The sharing of the wireless medium is done through an access method called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA is implemented as an alternative to Carrier Sense Multiple Access with Collision Detection (CSMA/CD), which is used in wired networks. An Ethernet network has the ability to transmit data while monitoring the network for collisions. At this point it can pause, wait a certain period of time, and resend the data again. In a wireless network, a wireless network interface card cannot transmit and receive data synchronously, so it must use collision avoidance rather than collision detection. This process is handled at layer two of the OSI model.

Layer 2: Where the meat is

The second layer of the OSI model, often called the Data Link layer or the MAC layer, is where 802.11 implements all of the features that make communication through the air possible. This includes tasks such as addressing, authentication and association, fragmentation, arbitration (CSMA/CA), and encryption. All of these things are what make the data link layer important to us, and what we will be spending our time together here examining.

The tricky thing about the wireless data link layer is that these frames aren't collected just by loading up Wireshark (our packet sniffer of choice for this article) and doing a standard capture.

I know what you're thinking: "I've capture packets from my wireless NIC before and it shows layer two information just like any other packet!" Well, you are correct in saying that Wireshark displays layer two frame information for packets captured from your wireless NIC. However, it only displays the components that it would display for an Ethernet network; your source and destination MAC addresses. There is a whole heap of information you are not seeing, and in order the get that information you have to make use of a feature called monitor mode.

Monitor mode is one of many modes that a wireless NIC can be set to. In monitor mode, a wireless NIC does not transmit any data, and only captures data on the channel it is configured to listen on. When set on monitor mode Wireshark will capture and display the entire contents of an 802.11 wireless layer two frame. How you utilize monitor mode is dependent upon the drivers available for your wireless NIC and your operating system of choice.

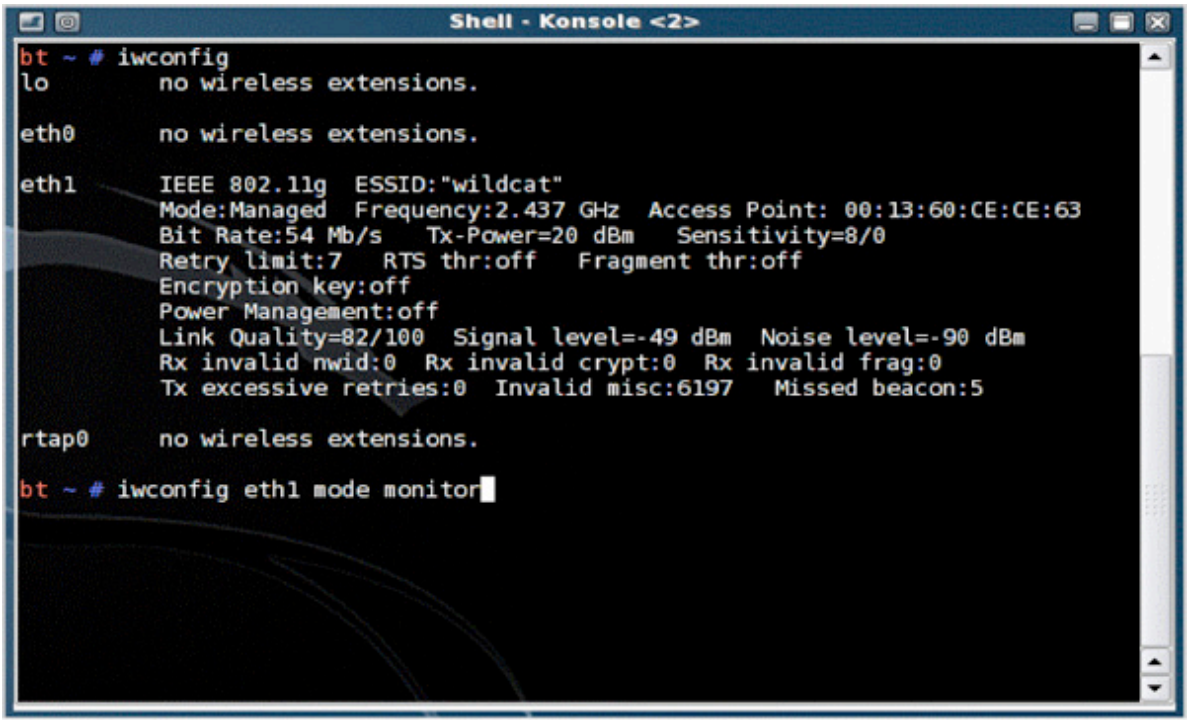
Using monitor mode in Linux

In Linux, a great majority of wireless drivers support monitor mode functionality, so changing your wireless NIC into monitor mode is a fairly simple process. Most of the wireless NIC drivers in Linux support the Linux Wireless Extensions interface so that you can configure them directly from a command shell with no additional software required. In order to determine if the wireless NIC you are using is supported by these wireless extensions, you can use the command iwconfig.

As you can see in the iwconfig output below, the eth1 interface supports Linux Wireless Extensions and displays information about the current configuration of the wireless NIC. We can easily see that the card is associated to a network with an SSID of "SANDERS" and that the card is in managed mode. In order to change the card to monitor mode, switch to a root shell and use this command:

```
# iwconfig eth1 mode monitor
```

You can verify the mode of the wireless NIC by running the iwconfig command once more. At this point you should be able to capture the appropriate data link layer wireless information.



It is important to note that not every Linux wireless NIC driver supports Linux Wireless Extensions. However, due to the open source nature of typical Linux drivers, most other drivers have been "modified" so that they can be put into monitor mode through some alternative means. If your wireless NIC doesn't support Linux Wireless Extensions then you should be able to do a quick Google search to find an alternative means of getting to monitor mode.

As you may remember reading earlier, one of the distinct differences between a wired and wireless connection is that the wireless connection operates on a shared spectrum.

This spectrum is broken up into several different channels in order to prevent interference from different systems in the same geographical area. This being the case, each node on a wireless network may only use one channel at a time to transmit or receive. This means that our wireless NIC in monitor mode must be explicitly configured to listen on whatever channel we want to grab packets off of. In order to set your wireless NIC to monitor on channel 6, you would use the command:

```
# iwconfig eth1 channel 6
```

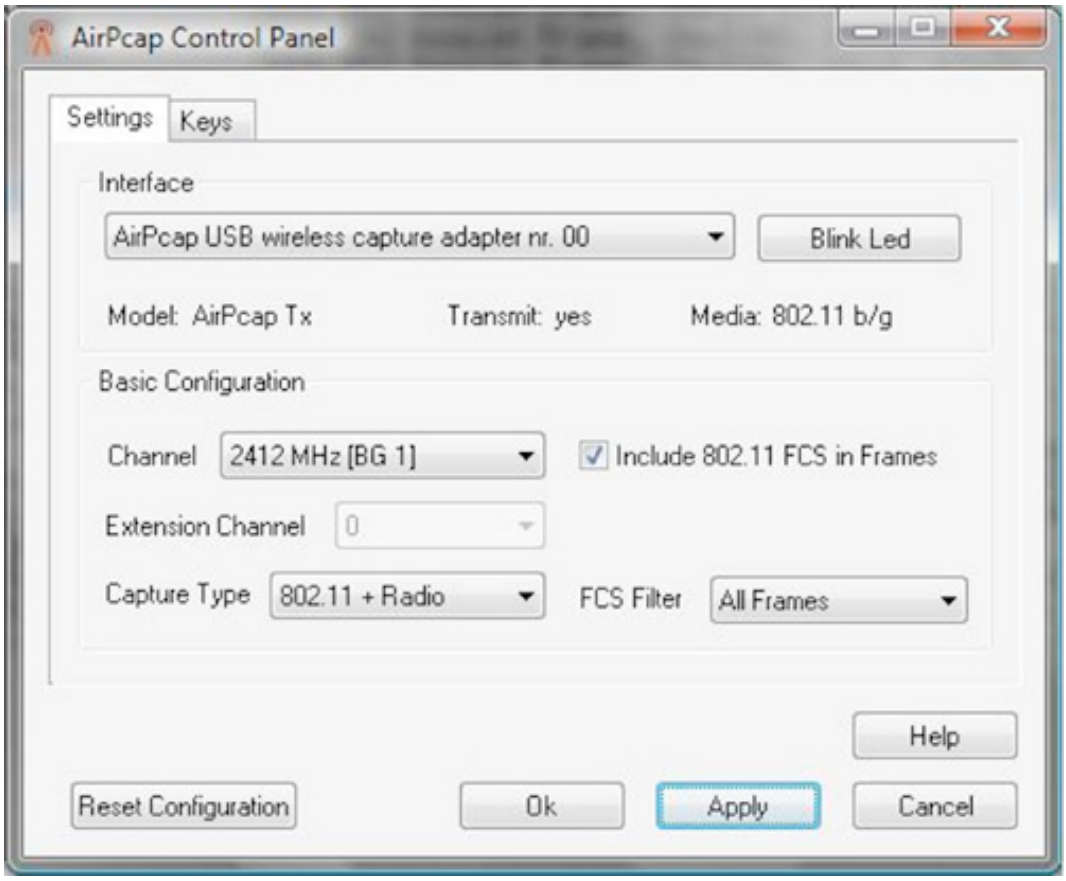
In this scenario, you would substitute whatever the assigned name for your wireless NIC interface is for eth1, and the numbers 1-11 (US) or 1-14 (International) in for the channel number.

Using an AirPcap device in Windows

Capturing wireless traffic in a Windows environment is unfortunately not as easy as a setting change. As with most Windows-based software, drivers in Windows are often not open source and do not allow for configuration change into monitor mode. With this in mind, we must use a specialized piece of hardware known as an AirPcap device.

Developed by CACE Technologies, employer of the original creator of Wireshark, an AirPcap device is essentially a USB 802.11 wireless adapter that is bundled with specialized software that will allow the device to be used in monitor mode.

Once you have obtained an AirPcap device you will be required to install the software on the accompanying CD to your analysis computer. The installation is a fairly straightforward accepting of the licensing agreement and clicking next a few times, so we won't cover that here. Once you have the software installed, you are presented with a few options you can configure in the AirPcap Control Panel.



As you can see from the screenshot above, there isn't an incredible amount of configuration to be done on the AirPcap device. These configuration options are stored on a per adapter basis.

The configurable options include:

- **Interface** - Select the device you are using for your capture here. Some advanced analysis scenarios may require you to use more than one AirPcap device to sniff simultaneously on multiple channels.

- **Blink LED** - Clicking this button will make the LED lights on the AirPcap device blink. This is primarily used to identify the specific adapter you are using if you are using multiple AirPcap devices.

- **Channel** - In this field, you select the channel you want AirPcap to listen on.

- **Extension Channel** - This option is only available on 802.11n capable AirPcap devices (AirPcap nX) and allows you to select an extension channel.

- **Capture Type** - The options are 802.11 Only, 802.11+Radio, and 802.11+PPI. The 802.11 Only option includes the standard 802.11 packet header on all capture packets. The 802.11 + Radio option includes this header and also a radiotap header, which contains additional information about the packet, such as data rate, frequency, signal level, and noise level. The 802.11+PPI option includes all of the previously mentioned data, along with information for multiple antennas when supported.

- **Include 802.11 FCS in Frames** - By default, some systems strip the last four checksum bits from wireless packets. This checksum, known as a Frame Check Sequence (FCS), is used to ensure that packets have not been corrupted during transmission. Unless the application you are using for interpreting packet captures has difficulty decoding packets with FCS, check this box to include the FCS checksums.

- **FCS Filter** - This option will allow you to filter out packets based upon whether they have a valid or invalid FCS.

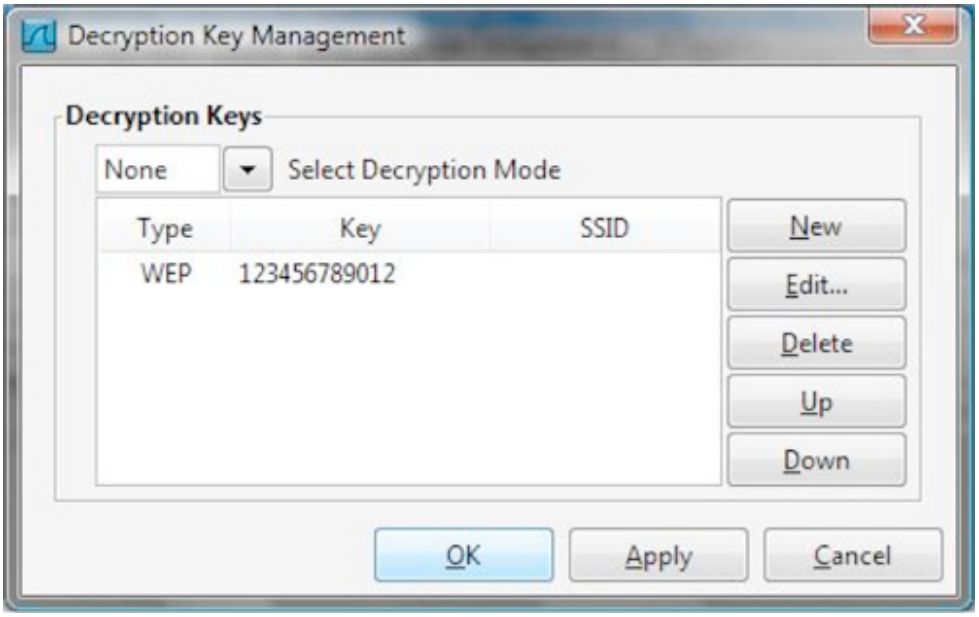
Aside from these configuration options you will also notice a Keys tab where you can enter and manage WEP keys for the decryption of WEP encrypted traffic. Most up-to-date wireless networks will not being using WEP for encryption, and because of this you may

initially come to the conclusion that the AirPcap device is limited and/or dated, but this is not the case. It is important to realize that AirPcap supports decryption of wireless traffic in two modes. Driver mode, configurable from the AirPcap Control Panel, only supports WEP.

That being the case, it is recommend that decryption keys be configured using Wireshark mode, which supports WEP, WPA, and WPA2, and is managed from the wireless toolbar inside of Wireshark.

The wireless toolbar is used to configure a lot of the options we have already learned about within the Wireshark program itself. You can enable this toolbar when you have an AirPcap adapter plugged into your analysis computer by opening Wireshark, going to the View drop-down menu, and placing a checkmark next to the Wireless Toolbar option.

As you can immediately determine, this toolbar makes a lot of the configuration options from the AirPcap device readily available from within Wireshark. The only major difference of any concern to us is the added functionality of the decryption section. In order to take advantage of this, you will need to set the Decryption Mode drop-down box to Wireshark, and add your appropriate encryption key by clicking the Decryption Keys button, clicking New, selecting the key type, and entering the key itself.



The 802.11 header

When you think about it, Ethernet really has it easy. All the MAC layer has to do is worry about a single source and destination address. An 802.11 MAC header on the other hand, has a lot more going on.

The illustration on the following page depicts the basic components of the MAC header.

- **Frame Control** - This section specifies the type and subtype of the MAC frame, as well as other options such as whether or not the packet is a fragment, whether power management is being used, or if WEP encryption is being used. There are three main types of MAC frames. First, management frames are used for tasks such as associating to an access point. Control frames are second and they are used to control the flow of data and handle things such as acknowledgement packets. Data frames are the final type and they contain the data being transmitted across the transmission medium.

• **Duration** - When this is used with a data frame this will specify the duration of the frame.

• **Address 1** - Source address

• **Address 2** - Destination address

• **Address 3** - Receiving station address (destination wireless station)

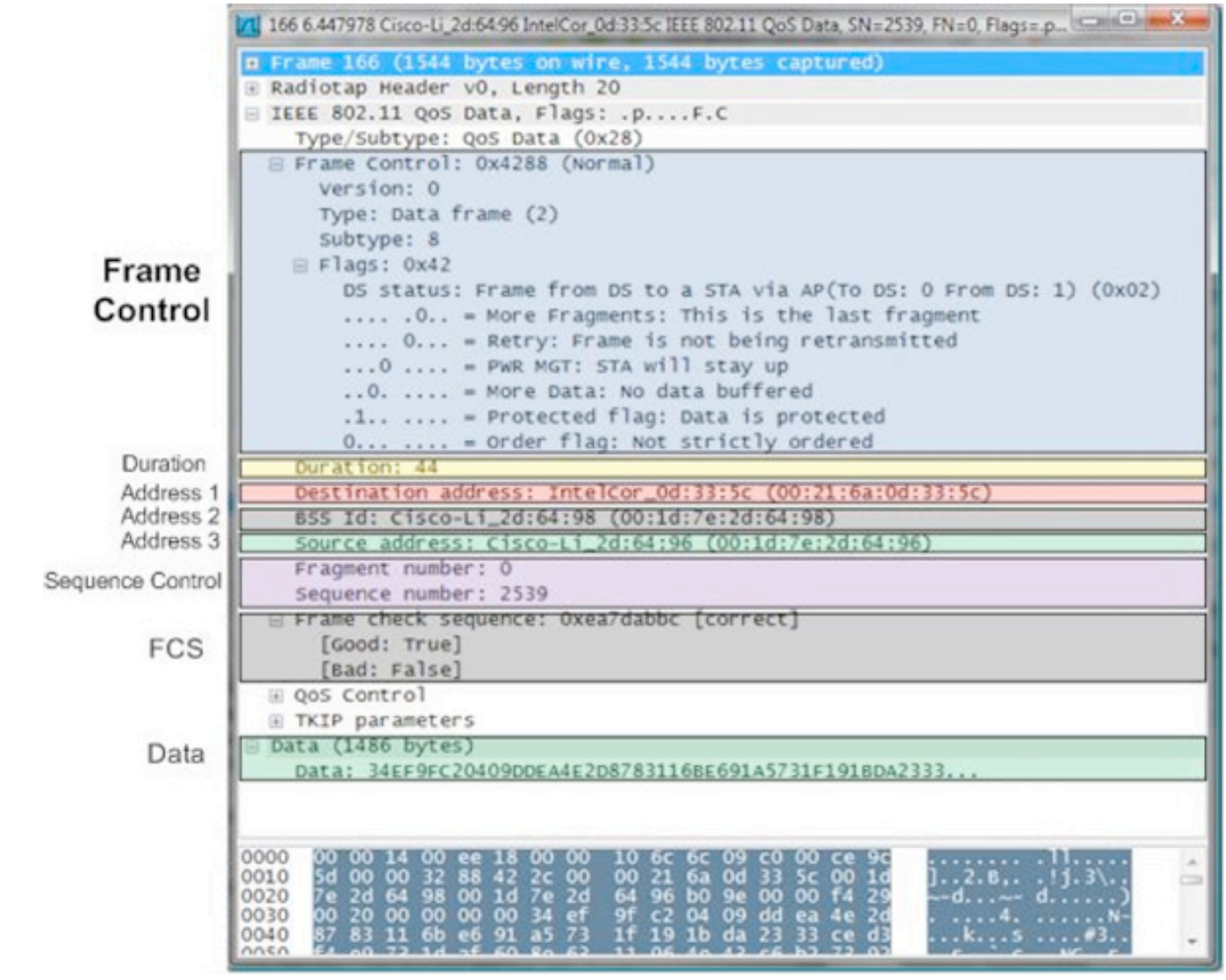
• **Address 4** - Transmitting wireless station

• **Frame Body** - Data contained in the frame

• **FCS** - The Frame Check Sequence discussed earlier.

Analyzing Wireshark dissection of the 802.11 header

With this background knowledge we can take a look at an individual packet that has been dissected by Wireshark and find the different components of the wireless header. The frame depicted below is a standard wireless data frame. We can immediately determine this by looking at the Type listing under the Frame Control section of the packet.



As I mentioned previously, the Frame Control section of the packet contains a lot of information and you can see all of these options here. Looking further into this packet you should be able to clearly find all of the sections of the packet.

The great thing about analyzing wireless packets is that what you see is what you get, and the packet you just looked at is what the great majority of wireless packets will look like. The defining difference between one packet and another is the type and subtype of that packet.

Management frames such as a Beacon will still contain all of the information listed above, but rather than the data portion of the packet they will contain the data specific to that frame type. You can view a complete listing of 802.11 frame types by viewing the 802.11 standards document (bit.ly/f2l0p).

A few frame types of interest include:

- Management Type 0
    - o Subtype 0000 – Association Request
    - o Subtype 0001 – Association Response
    - o Subtype 1000 – Beacon
    - o Subtype 1010 Disassociation
    - o Subtype 1011 Authentication
    - o Subtype 1100 De-authentication
- Control Type 01
    - o Subtype 1011 – Request to Send (RTS)
    - o Subtype 1100 – Acknowledgement
- Data Type 10
    - o Subtype 0000 - Data.

Wrap up

This is by no means a definitive guide on analyzing wireless traffic, but it should give you all of the information you need to get off on the right foot. We have covered why capturing layer two traffic is important to effectively analyzing wireless communications as well as the structure of these 802.11 frames.

The best thing you can do with the information presented here is to begin capturing packets on your own wireless networks. Once you start looking at common tasks such as associating to a network or completing an authentication request at the packet level, you should really get a sound grasp on what's happening in the air around you.

Chris Sanders is a network consultant based in western Kentucky. Chris writes and speaks on various topics including packet analysis, network security, Microsoft server technologies, and general network administration. His personal blog at www.chrissanders.org contains a great deal of articles and resources on all of these topics. Chris is also the founder and director of the Rural Technology Fund (www.ruraltechfund.org), a non-profit organization that provides scholarships to students from rural areas who are pursuing careers in information technology.