Chapter 6

Wireless Sniffing with Wireshark

Solutions in this chapter:

- Techniques for Effective Wireless Sniffing
- Understanding Wireless Card Operating Modes
- Configuring Linux for Wireless Sniffing
- Configuring Windows for Wireless Sniffing
- Using Wireless Protocol Dissectors
- Useful Wireless Display Filters
- Leveraging Wireshark Wireless Analysis Features

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

Wireless networking is a complex field. With countless standards, protocols, and implementations, it is not uncommon for administrators to encounter configuration issues that require sophisticated troubleshooting and analysis mechanisms.

Fortunately, Wireshark has sophisticated wireless protocol analysis support to help administrators troubleshoot wireless networks. With the appropriate driver support, Wireshark can capture traffic "from the air" and decode it into a format that helps administrators track down issues that are causing poor performance, intermittent connectivity, and other common problems.

Wireshark is also a powerful wireless security analysis tool. Using Wireshark's display filtering and protocol decoders, you can easily sift through large amounts of wireless traffic to identify security vulnerabilities in the wireless network, including weak encryption or authentication mechanisms, and information disclosure risks. You can also perform intrusion detection analysis to identify common attacks against wireless networks while performing signal strength analysis to identify the location of a station or access point (AP).

This chapter introduces the unique challenges and recommendations for traffic sniffing on wireless networks. We examine the different operating modes supported by wireless cards, and configure Linux and Windows systems to support wireless traffic capture and analysis using Wireshark and third-party tools. Once you have mastered the task of capturing wireless traffic, you will learn how to leverage Wireshark's powerful wireless analysis features, and learn how to apply your new skills.

Challenges of Sniffing Wireless

Traditional network sniffing on an Ethernet network is fairly easy to set up. In a *shared environment*, an analysis workstation running Wireshark starts a new packet capture, which configures the card in promiscuous mode and waits until the desired amount of traffic has been captured. In a *switched environment*, you need to configure a span port that mirrors the traffic sent to other stations, before initiating the packet capture.

In both of these cases, it is easy to initiate a packet capture and start collecting traffic for analysis. When you switch to wireless analysis, however, the process of traffic sniffing becomes more complicated and requires additional decisions up front to best support the analysis you want to perform.

Selecting a Static Channel

Where a wired network offers a single medium mechanism for packet capture (i.e., the wire), wireless networks can operate on multiple wireless channels using different

frequencies in the same location. A table of wireless channel numbers and the corresponding frequencies is listed in Table 6.1. Even if two wireless users are sitting side-by-side, their computers may be operating on different wireless channels.

Table 6.1 Wireless Frequencies and Channels

Frequency	Channel Number	Frequency	Channel Number
2.412 GHz	1	2.484 GHz	14
2.417 GHz	2	5.180 GHz	36
2.422 GHz	3	5.200 GHz	40
2.427 GHz	4	5.220 GHz	44
2.432 GHz	5	5.240 GHz	48
2.437 GHz	6	5.260 GHz	52
2.442 GHz	7	5.280 GHz	56
2.447 GHz	8	5.300 GHz	60
2.452 GHz	9	5.320 GHz	64
2.457 GHz	10	5.745 GHz	149
2.462 GHz	11	5.765 GHz	153
2.467 GHz	12	5.785 GHz	157
2.472 GHz	13	5.805 GHz	161

If you want to analyze the traffic for a specific wireless AP or station, you must identify the channel or frequency used by the target device, and configure your wireless card to use the same channel before initiating your packet capture. This is because wireless cards can only operate on a single frequency at any given time. If you wanted to capture traffic from multiple channels simultaneously, you would need an additional wireless card for every channel you wanted to monitor.

Using Channel Hopping

If you want to capture traffic for a specific station, how do you locate the channel number that it is operating on? One technique is to use *channel hopping* to rapidly scan through all available wireless channels until the appropriate channel number is identified. With channel hopping, the wireless card is still only operating on a single frequency at any given time, but is rapidly switching between different channels, thus allowing Wireshark to capture any traffic that is present on the current channel. Fortunately, Wireshark operates independently of the current channel selection; therefore, it is not necessary to stop and restart the packet capture before each

channel hop. Change to the desired channel while Wireshark is running and Wireshark will continue to collect traffic.

Unfortunately, you cannot rely on channel hopping for all of your wireless traffic sniffing needs. Channel hopping will cause you to lose traffic, because you are rapidly switching channels. If your wireless card is configured to operate on channel 11 and you hop to another channel, you will not be able to "hear" any traffic that is occurring on channel 11 until you return as part of the channel-hopping pattern. As a result, channel hopping is not a useful technique for analyzing traffic for a specific AP or station, but it can be useful to identify the channel the network is operating on, which can be used to set a static channel assignment.

Range in Wireless Networks

Another unique characteristic of Wireshark is the range between the capture station and the transmitting device(s). When capturing wireless traffic, the range between the capture station and the transmitter is significant, and must be accounted for to provide the most reliable traffic collection.

If the capture station is too far away from one or more transmitters, it is unable to "hear" the wireless traffic. If the capture station is too close to another transmitting station, the radio interface may become overwhelmed with too much signal, thus resulting in corrupted traffic. Placing the station near the transmitter no closer than 3 feet is the most desirable location for achieving optimal traffic capture. You can achieve satisfactory results for a wireless packet capture from further away, but you will lose traffic from the capture if there is a significant distance between the capture station and the transmitter(s).

Interference and Collisions

Another challenge of sniffing wireless networks is the risk of interference and lost packets. Unlike an Ethernet network that can transmit and monitor the network simultaneously, wireless cards can only receive or transmit asynchronously. As a result, wireless networks must take special precautions to prevent multiple stations from transmitting at the same time. While these collision-avoidance mechanisms work well, it is still possible to experience collisions between multiple transmitters on the same channel, or to experience collisions with wireless local area networks (LANs) and other devices using the same frequency (e.g., cordless phones, baby monitors, microwave ovens, and so on).

When two devices transmit simultaneously within range of the sniffing station, the transmission becomes corrupted and is rejected by the receiver as an invalid packet. After waiting random back-off intervals, the two stations repeat their transmission, thus

indicating they are attempting to transmit the same information again. This is normal activity in a wireless LAN, but presents a challenge to the sniffing station.

When capturing traffic on a wireless network, there is no guarantee that you captured 100 percent of the traffic. Some traffic may have become corrupted in transit. In other cases, your capture station may be positioned such that it receives valid frames before they become corrupt en-route to the destination host. This forces the transmitting station to re-transmit the corrupted packets, which causes the capture station to have multiple copies of the same packet in the capture.

Recommendations for Sniffing Wireless

Now that you understand some of the limitations and challenges in sniffing wireless networks, you can apply some recommendations to achieve the best fidelity in wireless packet captures:

- **Locate the Capture Station Near the Source** When initiating a packet capture, locate the capture station close to the source of the wireless activity you are interested in (i.e., an AP or a wireless station).
- **Disable Other Nearby Transmitters** If you are using an external wireless card (e.g., a Personal Computer Emulator Card [PCCard]) for sniffing traffic, and you have a built-in card in your laptop, it is common to experience lost traffic on the sniffing card due to interference from the built-in card. To eliminate this factor and achieve a more accurate packet capture, disable any built-in wireless transmitters on the capture station during the packet capture, including Institute of Electrical & Electronics Engineers (IEEE) 802.11 interfaces and Bluetooth devices.
- **Reduce CPU Utilization While Capturing** If your host experiences excessive central processing unit (CPU) utilization during a packet capture, you may experience packet loss in the wireless capture (e.g., it is not a good idea to burn a DVD while capturing wireless traffic). To prevent packet loss, try to reduce your CPU utilization when capturing traffic with any sniffer software.
- **Match Channel Selection** If you take a comprehensive packet capture of a wireless network, make sure your wireless card is sniffing on the same channel as the target network. If you are channel hopping during a packet capture, you will inevitably lose traffic from your target network. Only use channel hopping to discover the available networks; focus your capture on a single channel. Note that while you may capture some traffic from a nearby

channel (e.g., you see traffic from channels 1 and 6 when listening on channel 3), the captured traffic will be sporadic and incomplete.

- **Match Modulation Type** With the progression of different IEEE 802.11 Physical layer standards, different modulation mechanisms have been developed to accommodate faster data rates. Ensure the supported modulation mechanism for your wireless card matches the target network you are targeting. For example, an IEEE 802.11b wireless card sniffing an IEEE 802.11g network will capture some backward-compatible modulated traffic, but may miss other traffic modulated for an 802.11g network. If in doubt, ensure the card you are using for traffic capture supports all the standard modulation mechanisms. Currently, this includes an IEEE 802.11a/b/g card, but will also include IEEE 802.11n cards with MIMO (multiple input, multiple output) technology in the future.

Understanding Wireless Card Modes

Before we start wireless sniffing using Wireshark, it is helpful to understand the different operating modes supported by wireless cards. Most wireless users only use their wireless cards as a station to an AP. In *managed mode*, the wireless card and driver software rely on a local AP to provide connectivity to the wireless network.

Another common mode for wireless cards is *ad-hoc mode* (or Independent Basic Service Set [IBSS] mode. Two wireless stations that want to communicate with each other directly can do so by sharing the responsibilities of an AP for a limited subset of wireless LAN services. Ad-hoc mode is used for short-term connectivity between stations, when an AP is not available to provide connectivity.

Many wireless cards also support *master mode*, where the wireless card provides the services of an AP when paired with the appropriate software. Managed mode allows you to configure your laptop or desktop system as an AP for providing connectivity to other wireless stations.

Finally, wireless cards support *monitor mode* functionality. When configured in monitor mode, the wireless card stops transmitting data and sniffs the currently configured channel, reporting the contents of any observed packets to the host operating system. This is the most useful mode of operation for analysis when using Wireshark, because a wireless card configured in monitor mode reports the entire contents of wireless packets, including header information and the encrypted or unencrypted data contents. When in monitor mode, the wireless card and driver reports the wireless frames "as-is," giving the most accurate view of the wireless activity for the selected channel.

In order to analyze a wireless network effectively using Wireshark, you need to configure your wireless card to operate in monitor mode on the appropriate channel, and then start a packet capture. Unfortunately, this is easier said than done. Because the majority of wireless card users use their wireless cards in managed or ad-hoc mode, wireless driver developers may not include support for monitor mode access. In the case of Linux, many drivers support monitor mode. Those Linux drivers that do not natively support monitor mode are often "patched" by other interested users or developers in order to access monitor mode functionality. However, in the case of Windows, drivers are closed-source, which prevents anyone except the driver developer from supplying monitor mode functionality. However, some commercial options exist for Windows that allow you to leverage the monitor mode support in your wireless card with custom driver software.

Next, we examine the steps necessary to configure your wireless card to support monitor mode access on Linux and Windows systems.

Getting Support for Monitor Mode - Linux

In order to begin sniffing wireless traffic with Wireshark, your wireless card must be in monitor mode. Wireshark does not do this automatically; you have to manually configure your wireless card before starting your packet capture. However, the commands you need in order to configure the card in monitor mode can differ based on the type of wireless card and driver that you are using. This section discusses how to complete this step based on the most common wireless card and driver combination for Linux.

TIP

Determining the type of wireless card you have isn't always easy. While there are only a handful of manufacturers that make the wireless chipset hardware, multiple vendors re-brand the cards, thus making it difficult to identify what the actual chipset is. One resource for identifying the chipset from the card manufacturer is available at *www.linux-wless.passys.nl*. If your specific card isn't listed here you can search using Google with the card name and keyword "chipset" (e.g., WPC55AG chipset).

Linux Wireless Extensions Compatible Drivers

Most wireless drivers for Linux systems use the Linux Wireless Extensions interface, thus providing a consistent configuration interface for manipulating the wireless card. First, let's identify the wireless driver interface name by running the wireless card configuration utility *iwconfig* with no parameters:

```
$ iwconfig
eth0      no wireless extensions.

lo        no wireless extensions.

eth1      IEEE 802.11b  ESSID:"Beacon Wi-Fi Network"
          Mode:Managed  Frequency:2.462 GHz  Access Point: 00:02:2D:8B:70:2E
          Bit Rate:11 Mb/s   Tx-Power=20 dBm   Sensitivity=8/0
          Retry limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=50/100  Signal level=-71 dBm  Noise level=-86 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:286   Missed beacon:5
```

NOTE

It is recommended that users take advantage of the Linux 2.6 kernel whenever possible. Most Linux distributions install their wireless tools packages for *iwconfig* and *iwpriv* by default; you will need to install these tools manually if they are not included with your default distribution. Use the package management utilities that come with your Linux distribution to search for packages with the name "wireless-tools" to identify installation options. Information specific to older Debian, SuSE, RedHat, and Mandrake distributions is available at *www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/DISTRIBUTIONS.txt.*

From this output, we determine that interfaces *eth0* and *lo* do not support Linux Wireless Extensions; however, Interface *eth1* does support wireless extensions. From the output, we can see that the card is currently in managed mode and is associated with an IEEE 802.11b network with the Service Set Identifier (SSID) "Beacon Wi-Fi Network" at 2.462 GHz (channel 11).

Since we want to use this wireless interface for wireless traffic sniffing, we need to place the card in monitor mode. In order to make changes to the wireless card configuration, we need to be the *root user*. Become the root user by running the *su* command and supplying the root user password:

```
$ su
Password: enter root password
#
```

After becoming the root user, you can use the *iwconfig* utility to configure the card for monitor mode, by specifying the interface name followed by mode monitor:

```
# iwconfig eth1 mode monitor
```

After placing the card in monitor mode, run the *iwconfig* utility with the interface name as the only command-line argument, to verify the configuration change:

```
# iwconfig eth1
eth1   unassociated ESSID:off/any
     Mode:Monitor Channel=0 Access Point: 00:00:00:00:00:00
     Bit Rate:0 kb/s  Tx-Power=20 dBm  Sensitivity=8/0
     Retry limit:7  RTS thr:off  Fragment thr:off
     Encryption key:off
     Power Management:off
     Link Quality:0 Signal level:0 Noise level:0
     Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
     Tx excessive retries:0 Invalid misc:7007  Missed beacon:0
```

In this output, we see that the mode has changed from managed to monitor. At this point, the wireless card is operating in monitor mode. Next, we need to make sure the interface is in the "up" state with the *ifconfig* utility, again using the interface name as the only command-line parameter:

```
# ifconfig eth1
eth1   Link encap:UNSPEC HWaddr 00-13-CE-55-B5-EC-BC-A9-00-00-00-00-00-00-
00-00
     BROADCAST MULTICAST MTU:1500 Metric:1
     RX packets:18176 errors:0 dropped:18462 overruns:0 frame:0
     TX packets:123 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
     Interrupt:11 Base address:0x4000 Memory:a8401000-a8401fff
```

The first indented line of text following the interface name and hardware address (HWaddr) reports the operating flags for the interface. In this example, the interface is configured to accept broadcast and multicast traffic. The interface is not currently in the *up* state, due to the lack of the UP keyword. Modify the interface configuration by placing the interface in the *up* state, then examine the interface configuration properties as shown below:

```
# ifconfig eth1 up
# ifconfig eth1
eth1   Link encap:UNSPEC HWaddr 00-13-CE-55-B5-EC-3C-4D-00-00-00-00-00-00-
00-00
     UP BROADCAST MULTICAST MTU:1500 Metric:1
     RX packets:34604 errors:0 dropped:34583 overruns:0 frame:0
     TX packets:232 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:18150 (17.7 Kb) TX bytes:0 (0.0 b)
     Interrupt:11 Base address:0x4000 Memory:a8401000-a8401fff
```

In this output we see that the interface is now in the *up* state and is ready to begin sniffing wireless traffic.

NOTE

Unlike the *iwconfig* tool, *ifconfig* does not understand the properties of an interface that is in monitor mode. When associated to a wireless network, the interface appears as a standard Ethernet interface; however, while in monitor mode, it appears as an unknown or unspecified link encapsulation mechanism. As a result, *ifconfig* displays a default of 16 bytes to represent the Media Access Control (MAC) address of the unspecified interface encapsulation (denoted with the string *UNSPEC*). In what appears to be a bug in the *ifconfig* tool, 8 bytes are printed to represent the MAC address, followed by 8 NULL bytes. The first 6 bytes represent the actual MAC address of the wireless card, followed by 2 bytes of uninitialized memory.

MADWIFI 0.9.1 Driver Configuration

The Multiband Atheros Driver for WiFi (MADWIFI) supports wireless cards based on the popular Atheros chipsets supporting IEEE 802.11a, IEEE 802.11b, and IEEE

802.11g wireless networks. While this driver supports monitor mode access, it does not support the configuration of monitor mode access using the *iwconfig* utility. Instead, the MADWIFI developers include a custom tool for configuring wireless card properties called the *wlanconfig* utility.

The MADWIFI drivers are unique in that they support multiple interfaces on the same wireless card known as Virtual Access Points (VAPs). Each VAP appears as its own interface name with a single default VAP configured in managed mode. In order to create an interface in monitor mode, however, we need to remove all VAPs on the local system with the *wlanconfig* utility. First, examine the list of wireless devices on the system using the *iwconfig* utility with no command-line arguments:

```
# iwconfig
wifi0   no wireless extensions.

ath0   IEEE 802.11b ESSID:""
     Mode:Managed Channel:0 Access Point: 00:00:00:00:00:00
     Bit Rate:0 kb/s  Tx-Power:0 dBm  Sensitivity=0/3
     Retry:off  RTS thr:off  Fragment thr:off
     Encryption key:off
     Power Management:off
     Link Quality=0/94 Signal level=-95 dBm Noise level=-95 dBm
     Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
     Tx excessive retries:0 Invalid misc:0  Missed beacon:0
```

NOTE

The MADWIFI drivers use a "master" interface with the naming convention *wifiX*, where *X* is *0* for the first wireless card, *1* for the second wireless card, and so on. The master interface is used to create one or more virtual interfaces with the *wlanconfig* utility. In most cases, you will only refer to the master interface when creating or destroying virtual interfaces. You will use the virtual interface for all other tasks, including sniffing wireless traffic with Wireshark, or accessing a wireless network as a station.

From this output we can see two interfaces; *wifi0* which does not support wireless extensions, and *ath0* which does. The *ath0* interface is named for the Atheros wireless chipset (*ath*) which is created by default in managed mode. In order to

configure an interface in monitor mode, we must delete or "destroy" this interface using the *wlanconfig* utility:

```
# wlanconfig ath0 destroy
# iwconfig
wifi0   no wireless extensions.
```

From the output of the *iwconfig* utility, we see that the *ath0* interface is no longer present. Next, we re-create the *ath0* interface with the *wlanconfig* utility, this time indicating that the interface should be created in monitor mode, referencing the *wifi0* interface as the master interface:

```
# wlanconfig ath0 create wlandev wifi0 wlanmode monitor
ath0
# iwconfig
wifi0   no wireless extensions.

ath0   IEEE 802.11b ESSID:""
     Mode:Monitor Channel:0 Access Point: 00:00:00:00:00:00
     Bit Rate:0 kb/s  Tx-Power:0 dBm  Sensitivity=0/3
     Retry:off  RTS thr:off  Fragment thr:off
     Encryption key:off
     Power Management:off
     Link Quality=0/94 Signal level=-95 dBm Noise level=-95 dBm
     Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
     Tx excessive retries:0 Invalid misc:0  Missed beacon:0
```

Next, we must ensure the *ath0* interface is in the up state using the *ifconfig* utility, as shown below:

```
# ifconfig ath0 up
# ifconfig ath0
ath0   Link encap:UNSPEC HWaddr 00-20-A6-4F-01-40-BC-9D-00-00-00-00-00-00-
00-00
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
     RX packets:0 errors:0 dropped:0 overruns:0 frame:0
     TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:0
     RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

From the output of the *ifconfig* utility we see that the interface is now in the *up* state and is ready to start sniffing wireless traffic.

Capturing Wireless Traffic - Linux

Once your wireless card in Linux has been placed in monitor mode, you are ready to start capturing wireless traffic. Recall that wireless cards can only capture traffic on a single channel at any given time. If you know the wireless channel you want to capture traffic on, configure your wireless card to listen on that channel using the *iwconfig* utility:

```
# iwconfig ath0 channel 1
# iwconfig ath0
```

Replace *ath0* with the name of your wireless interface, and the number *1* with the channel number you want to capture traffic on. As seen from the output of the *iwconfig* command, the card is currently configured to listen on 2.412 Gigahertz (GHz) (channel 1).

If you don't know the target channel number you want to use to capture traffic, you can configure your wireless card to perform channel hopping. Unfortunately, Linux doesn't come with a built-in tool for channel hopping; however, you can configure channel hopping manually with a short shell script. Enter the text found in Code 6.1 into a short shell script using your favorite text-editor. Line numbers have been added for clarity; do not enter the line numbers when creating this script.

Code 6.1 Channel Hopping Shell Script

```
 1. #!/bin/bash
 2. IFACE=ath0
 3. IEEE80211bg="1 2 3 4 5 6 7 8 9 10 11"
 4. IEEE80211bg_intl="$IEEE80211b 12 13 14"
 5. IEEE80211a="36 40 44 48 52 56 60 64 149 153 157 161"
 6. IEEE80211bga="$IEEE80211bg $IEEE80211a"
 7. IEEE80211bga_intl="$IEEE80211bg_intl $IEEE80211a"
 8.
 9. while true ; do
10.   for CHAN in $IEEE80211bg ; do
11.     echo "Switching to channel $CHAN"
12.     iwconfig $IFACE $CHAN
13.     sleep 1
14.   done
15. done
```

After saving the shell script, change the permissions on the file to make it an executable program:

```
# chmod 755 chanhop.sh
```

Change the interface name *ath0* on line 2 to reflect the name of your wireless interface. Also, change the channel designator *$IEEE802.11bg* on line 10 to reflect the channels that are supported by your wireless card. To start the channel-hopping script, run the shell script from the directory where it was created:

```
# ./chanhop.sh
Switching to channel 1
Switching to channel 2
```

When you want to stop the channel-hopping script, press **Ctrl+C**.

NOTE

If creating shell scripts for channel hopping isn't appealing, you can download a more sophisticated copy of this script from the Wireshark web site wiki at http://wiki.wireshark.org/CaptureSetup/WLAN.

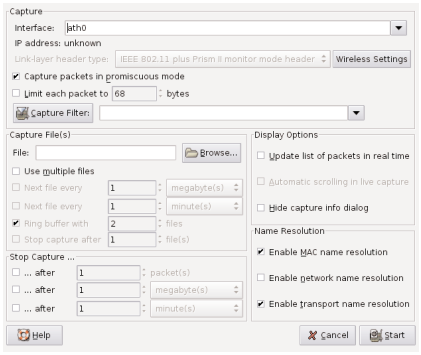
Starting a Packet Capture - Linux

Whether you have specified a single channel for capturing wireless traffic or are currently channel hopping, the process for capturing wireless traffic on Linux remains the same. Start Wireshark by running the *wireshark* executable with no command-line arguments as the root user, and initiate a new packet capture by pressing **Capture | Options**. This opens the "Wireshark Capture" options dialog box (see Figure 6.1).

Choose the wireless interface that has been placed in monitor mode by selecting the drop-down box labeled "Interface:," and then specify the desired capture options. Next, click **Start** to initiate the packet capture.

At this point, you've configured your system to capture wireless traffic in monitor mode. The next step is to utilize the information contained in the packets you are capturing. Fortunately, Wireshark has sophisticated analysis mechanisms that can be used for wireless traffic analysis. Let's examine the steps for configuring monitor mode support on Windows systems.

Figure 6.1 Wireshark Capture Options Dialog Box - Linux



Getting Support for Monitor Mode - Windows

Unfortunately, Windows drivers for wireless cards do not normally include support for monitor mode access, instead restricting users to operating the card in managed mode. Fortunately, through a combination of commercial and open-source software, we can overcome this limitation to use Windows hosts for wireless traffic analysis with Wireshark.

Introducing AirPcap

In order to overcome the limitations with most wireless drivers for Windows systems, the engineers at CACE Technologies have introduced a commercial product called AirPcap. A combination of a USB IEEE 802.11b/g adapter, supporting driver software, and a client configuration utility, AirPcap provides a simple mechanism to capture wireless traffic in monitor mode on Windows workstations at a reasonable cost. AirPcap is available at *www.cacetech.com*.

After obtaining the AirPcap CD and Universal Serial Bus (USB) wireless adapter, follow the installation instructions detailed in the AirPcap User's Guide. Ensure you have installed the appropriate version (WinPcap 4.0 beta 1) of WinPcap to support the AirPcap.

NOTE

Unfortunately, at the time of this writing, there are no free software solutions that allow Windows users to capture wireless traffic reliably, and without violating other software license restrictions. If you need to perform wireless traffic analysis with a Windows workstation, Wireshark is an effective tool; however, you would have to purchase a driver and hardware combination that supports monitor mode.

If you want to avoid any costs associated with drivers for monitor mode packet capture, you are encouraged to use a Linux option that bundles monitor mode support with the free wireless drivers. Using a bootable Linux CD such as Backtrack from *www.remote-exploit.org*, you can create an easily accessible Linux environment by booting from the Linux CD and plugging in a supported wireless card.

TIP

Another option for Windows users is to use the licensed AiroPeek NX software to collect packet captures. Since Wireshark can read AiroPeek NX's *.apc* files, you can use Wireshark to augment the features you get from AiroPeek NX. Unfortunately, the demo version of AiroPeek NX does not allow you to save packet captures.

Specifying the Capture Channel

After installing the AirPcap drivers, start the AirPcap control panel tool by navigating to **Start | All Programs | airpcap | Airpcap Control Panel** (see Figure 6.2). Using this utility, you can manipulate the following settings for the wireless capture, as described in Table 6.2.

Figure 6.2 AirPcap Control Panel

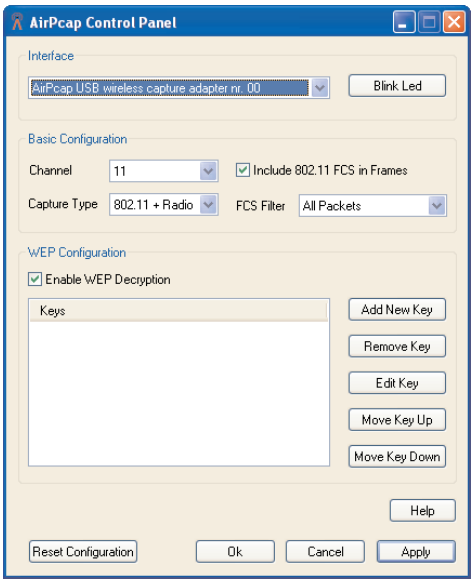


Table 6.2 AirPcap Control Panel Settings

Parameter	Options	Description
Blink LED	On, Off	Blinks the Light Emitting Diode (LED) on the Airpcap USB adapter; useful when using multiple AirPcap dongles on the same host.
Channel	1–14	Specifies the channel that Wireshark will capture traffic on with the specified AirPcap adapter. Because the AirPcap adapter is listen-only, it allows users to capture on all supported IEEE 802.11b/g channels, even those that are not permitted for use by the Federal Communications Commission (FCC). At the time of this writing, AirPcap does not include a tool to perform channel hopping during a packet capture.
Include 802.11 FCS in Frames	On, Off	The last 4 bytes of every packet on a wireless network is known as the Frame Check Sequence (FCS), which is a 32-bit checksum that is used to identify whether a packet was accidentally corrupted in transmission. This information is often stripped from monitor

Continued

Table 6.2 AirPcap Control Panel Settings

Parameter	Options	Description
		mode packet captures on Linux systems, but can be useful to validate the integrity of a packet if present. The recommended value is to set this option to "On" to record the FCS information in each packet.
Capture Type	802.11 Only, 802.11 + Radio	Each Promiscuous Capture Library (libpcap) packet capture file or interface has a capture link type assigned to it that tells Wireshark and other sniffer tools what to expect from the sniffer. The AirPcap Control Panel allows you to specify 802.11 Only or 802.11 + Radio as the link type. The 802.11 Only link type will produce a packet capture where each packet begins with the IEEE 802.11 header contents. The 802.11 + Radio link type will prepend a header before the start of the IEEE 802.11 header, known as the *Radiotap* header. This header allows the capture to store additional information from the driver for each packet that is not part of the 802.11 header information (e.g., signal strength, signal quality, modulation type, channel type [802.11b, 802.11g], the data rate, channel number and other useful information). The recommended value is to set this option to 802.11 + Radio to record the additional information with each packet.
FCS Filter	All Packets, Valid Packets, Wrong FCS Packets	Regardless of whether the Frame Check Sequence (FCS) is recorded for each frame in the packet capture, the AirPcap adapter will check the FCS of each frame to determine if it is valid or corrupted when received. AirPcap allows users to specify if they want to receive both valid and invalid packets (All Packets), only valid packets (correct FCS), or invalid packets (wrong FCS). For most uses of AirPcap, it is recommended you select "Valid Packets," since any packets

Continued

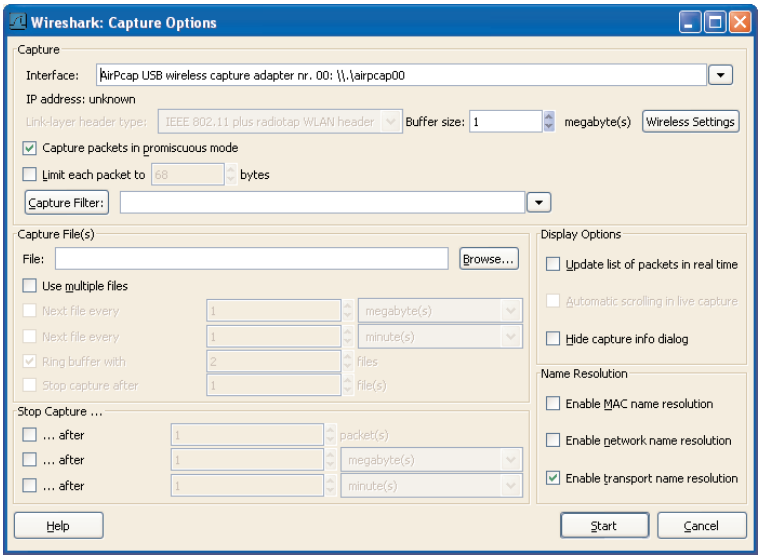
Table 6.2 AirPcap Control Panel Settings

Parameter	Options	Description
		that are invalid were likely not properly received by the station they were directed to. However, it may be useful to capture packets with a wrong FCS to determine how many packets are being corrupted in transit.
WEP Settings	Multiple	The AirPcap Control Panel allows users to specify static Wired Equivalent Privacy (WEP) keys to use for decrypting traffic with Wireshark. This option is also available from the Wireshark graphical user interface (GUI), and is examined later in this chapter. After selecting the desired options, press the **OK** button to activate and save your preferences.

Capturing Wireless Traffic - Windows

After specifying your capture preferences on the AirPcap Control Panel, start Wireshark and initiate a new packet capture by navigating to **Capture | Options**. This opens the Wireshark capture options dialog box (see Figure 6.3).

Figure 6.3 Wireshark Capture Options - Windows



Choose the AirPcap interface by selecting the drop-down box labeled "Interface:," and then specify the desired capture options. Next, click **Start** to initiate the packet capture. Stop the capture after you have collected the desired amount of traffic by clicking on the **Stop** button, or go to **Capture | Stop** in the capture dialog box.

At this point, you are capturing wireless traffic in monitor mode on Windows. Next comes the challenging part: extracting useful information from the packet capture contents. The following section examines the many Wireshark features that make this analysis easier.

Analyzing Wireless Traffic

Regardless of whether you are reading a packet capture from a stored file or from a live interface on a Windows or Linux host, Wireshark's analysis features are nearly identical. Wireshark offers many useful features for analyzing wireless traffic, including detailed protocol dissectors, powerful display filters, customizable display properties, and the ability to decrypt wireless traffic. Each of these features are examined in detail.

Navigating the Packet Details Window

One of the most impressive Wireshark features is the ability to dissect the contents of traffic and present it in a collapsible "tree-like" manner. For wireless traffic, Wireshark presents the Frame Dissector window starting with frame statistics, and then the 802.11 MAC layer contents. If additional data follows for the 802.11 header, Wireshark logically divides each of the protocols that follow into a new window.

Frame Statistics

The first group in the Packet Details window detailed summary information about the currently selected frame. The Frame window doesn't display any of the selected frame's contents, but rather general information contained in the packet capture for the selected frame (see Table 6.3).

Table 6.3 Frame Statistical Detail

Field Name	Description	Display Filter Reference Name
Arrival Time	The "Arrival Time" reflects the timestamp recorded by the station that is capturing traffic when the packet arrived. The accuracy of this field is only as accurate as the time on the receiving station. Note that packet captures from Windows systems are only represented with accuracy in seconds; no support for representing fractional seconds is available.	*frame.time*
Time Delta from Previous Packet	The "Time Delta" field identifies the elapsed time between the selected frame and the frame immediately before this frame. This field is updated when a display filter is applied to reflect the time from the previously displayed frame. This feature can be very useful when analyzing traffic that is transmitted with a consistent time interval (such as beacon frames) to identify interference causing dropped frames.	*frame.time_delta*
Time Since Reference or First Frame	The "Time Since Reference" or "First Frame" field indicates the amount of time that has elapsed since the start of the packet capture for the currently selected frame. This field is not updated when a display filter is applied.	*frame.time_relative*
Frame Number	The "Frame Number" field is a sequential counter starting with 1, uniquely representing the current frame. This field is useful for applying a display filter where one or more frames need to be selected or excluded from the display.	*frame.number*
Packet Length	The "Packet Length" reflects the actual length of the entire packet, regardless of how much of the packet was captured. By default, the entire frame is captured with Wireshark and Airodump.	*frame.pkt_len*

Continued

Table 6.3 Frame Statistical Detail

Field Name	Description	Display Filter Reference Name
Capture Length	The "Capture Length" reflects how much data was captured based on the specified number of bytes the user wanted to capture for each frame (known as the "snap length"). By default, Wireshark uses a snap length of 65,535 bytes to capture the entire frame contents. When an alternative snap length is specified, the capture length can be smaller if the frame size is smaller than the snap length.	*frame.cap_len*
Protocols in Frame	The "Protocols in Frame" field specifies all the protocols that are present, starting with the IEEE 802.11 header.	*frame.protocols*

TIP

Maintaining accurate host time is important for many kinds of protocol analysis, and especially important if you want to correlate events across multiple systems. Consider using the Network Time Protocol (NTP) on you Linux or Windows clients to ensure your local system time is always accurate.

IEEE 802.11 Header

Following the frame statistics data, Wireshark starts to dissect the protocol information for the selected packet. The IEEE 802.11 header is fairly complex; unlike a standard Ethernet header, it is between 24 and 30 bytes (compared to the standard Ethernet header of 14 bytes), has three or four addresses (compared to Ethernet's two addresses), and has many more fields to specify various pieces of information pertinent to wireless networks. What's more, wireless frames can have additional protocols appended to the end of the IEEE 802.11 header, including encryption options, Quality of Service (QoS) options, and embedded protocol identifiers (IEEE 802.2 header), all before actually getting any data to represent the upper-layer Network layer protocols.

Fortunately, Wireshark makes this analysis simple by intelligently representing this data in an easy-to-navigate form. We'll use many of these data fields when we start using display filters on wireless traffic and analyzing real-life packet captures, so it's beneficial to start with an analysis of each of the fields in the IEEE 802.11 header as shown in Table 6.4 below.

Figure 6.4 IEEE 802.11 Header Fields

Field Name	Description	Display Filter Reference Name
Type/Subtype	The Type/Subtype field value is not represented as data in the IEEE 802.11 header; rather, it is included as a convenience mechanism to uniquely identify the type and subtype combination that is included in the header of this frame. This field is commonly used in display filters.	*wlan.fc.type_subtype*
Frame Control	The Frame Control field is a 2-byte field that represents the first 2 bytes of the IEEE 802.11 header. Wireshark further dissects this field into four additional fields, as described below.	*wlan.fc*
Version	The Version field is included in the frame control header and specifies the version of the IEEE 802.11 header. At the time of this writing, this value is *0*.	*wlan.fc.version*
Type	The Type field is included in the frame control header and specifies the type of frame (data, management, or control).	*wlan.fc.type*
Subtype	The Subtype field is included in the frame control header and specifies the function for the specified frame type. For example, if the frame is a type management frame, the subtype field indicates the type of management frame (e.g., a beacon frame, authenticate request, or disassociate notice).	*wlan.fc.subtype*
Flags	The Flags field is a 1-byte field in the frame control header that specifies eight	*wlan.fc.flags*

Continued

Figure 6.4 IEEE 802.11 Header Fields

Field Name	Description	Display Filter Reference Name
	different options of the frame. Wireshark further dissects this field into each unique option, as described below.	
DS status	The Distribution System (DS) Status field represents the direction the frame is traveling in. Wireshark represents two unique fields as one display entry: *From DS* and *To DS*. When From DS is set to *1* and To DS is set to *0*, the frame is traveling from the AP to the wireless network. When From DS is set to *0* and To DS is set to *1*, the frame is traveling from a wireless client to the AP.	*wlan.fcds*
More Fragments	The More Fragments field in the flags header is used to indicate if additional fragments of a frame must be reassembled to process the entire frame. This field is not used often.	*wlan.fc.flag*
Retry	The Retry field in the flags header is used to indicate if the current frame is being retransmitted. The first time a frame is transmitted, the retry bit is cleared. If it is not received properly, the transmitting station retransmits the frame and sets the retry bit to indicate this status.	*wlan.fc.retry*
Power Management	The Power Management field in the flags header is used to indicate if the station is planning to enter a "dozing" state where they will reduce their participation in the network in an attempt to conserve power.	*wlan.fc.pwrmgmt*
More Data	The More Data field in the flags header is used by an AP to indicate that the station receiving frames has more packets waiting in a buffer for delivery. The More Data field is often used when a station awakens from a power-conservation mode to deliver all pending traffic.	*wlan.fc.moredata*

Continued

Figure 6.4 IEEE 802.11 Header Fields

Field Name	Description	Display Filter Reference Name
Protected	The Protected field in the flags header is used by an AP to indicate that an IEEE 802.11 encryption mechanism is used to encrypt the contents of the frame. At the time of this writing, the protected field indicates that the payload of the frame is encrypted with the Wired Equivalence Privacy (WEP) protocol, Temporal Key Integrity Protocol (TKIP), or the Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP).	*wlan.fc.protected*
Order	The Order field in the flags header is used to indicate that the transmission of frames should be handled in a strict order, preventing a station from re-ordering the delivery of frames to improve performance or operational management. This field is not used often.	*wlan.fc.order*
Duration	The Duration field follows the frame control header and serves one of two functions. In most frames, the duration field specifies the amount of time required to complete the transmission of the frame in a quantity of microseconds. When associating to the AP, however, the duration field identifies the association identifier (i.e., a unique value assigned to each station connected to the AP).	*wlan.duration*
Address Fields	The IEEE 802.11 header contains one Address Field (receiver or destination address) if the type of frame is a control message, and three Address Fields for normal data or management traffic (source, destination, and basic SSID [BSSID]). Wireless LANs that bridge multiple networks together also include a fourth address. Complicating things,	*wlan.da (destination), wlan.sa (source), wlan.bssid (BSSID), wlan.ra (receiver)*

Continued

Figure 6.4 IEEE 802.11 Header Fields

Field Name	Description	Display Filter Reference Name
	the order of these addresses isn't consistent, and changes depending on the To DS and From DS flag settings in the frame control header. Fortunately, Wireshark correctly represents all of these fields, allowing us to apply filters using the appropriate display name.	
Fragment Number	The Fragment Number (FN) field is a sequential number that is used to uniquely identify a fragment of a frame, starting at *0*. This field is not used often.	*wlan.frag*
Sequence Number	The Sequence Number (SN) field is a sequential number that is used to identify the entire frame, starting at *0*. Each frame transmitted by a station should have a SN that is one greater than the previous frame, until the counter wraps at 4,095.	*wlan.seq*

As mentioned previously, there are additional header fields that follow the IEEE 802.11 header, and Wireshark also dissects the contents of these fields. We will use our understanding of the fields in the IEEE 802.11 header in the next section, where we apply useful display filters to a traffic capture.

Leveraging Display Filters

One of the most powerful and useful features in Wireshark is the ability to apply inclusive or exclusive display filters to a packet capture, in order to narrow down the number of packets to those containing useful data. When capturing traffic on a wireless network, it is easy to become overwhelmed by the sheer quantity of data that is captured. (At an absolute minimum, a wireless network transmits 10 frames per second, before a single station connects to the network.) Using display filters, you can exclude uninteresting traffic to reveal useful information, or search through a large packet capture for a specific set of information.

In this section, we demonstrate several useful display filters for analyzing wireless traffic. We focus on using our knowledge of the IEEE 802.11 header and frame statistic contents to apply wireless-specific filters that can be applied in real-world analysis scenarios.

Traffic for a Specific Basic Service Set

An IEEE 802.11 wireless network with an AP providing connectivity to one or more client systems is known as a Basic Service Set (BSS). This is the most common wireless LAN implementation, and is used everywhere from corporate networks to hotspot environments and high-security government institutions.

Each wireless AP is uniquely identified by the Basic Service Set Identifier (BSSID). Recall that the BSSID is one of the addresses found in the IEEE 802.11 header, and is present in every data or management frame transmitted by a wireless station or an AP to uniquely identify the wireless LAN.

When traffic is captured in monitor mode, the wireless card reports all valid IEEE 802.11 frames for the specified channel, regardless of the BSSID or the network name being used. This can also include traffic from other nearby channels, because many wireless cards also have sufficient radio sensitivity to capture traffic from other nearby frequencies (e.g., it's not uncommon for a wireless card on channel 3 to capture traffic from channels 1, 2, 3, 4, and 5).

TIP

The BSSID address is often the same Medium Access Control (MAC) address as the wireless card on the AP, when there is a single network name configured. When multiple network names or virtual APs are configured, the BSSID may be similar to the MAC address of the AP's wireless card with minor variations (often in the last byte of the address).

When doing troubleshooting analysis, however, you usually want to limit the analysis to traffic to and from a specific AP that is servicing the problematic client. Using display filters, you can easily exclude traffic from nearby APs and focus the analysis on a specific AP. In this display filter, the goal is to identify all of the traffic for a single AP.

Identify the Station MAC Address

We start by obtaining the MAC address of the station that we are troubleshooting, or any station that is connected to the target BSS. On a Windows system, we can extract this information by running the *ipconfig /all* utility from a command shell (see Figure 6.4). On a Linux system, use *ifconfig -a* to determine the MAC address (see Figure 6.5).

Figure 6.4 Windows MAC Address Information

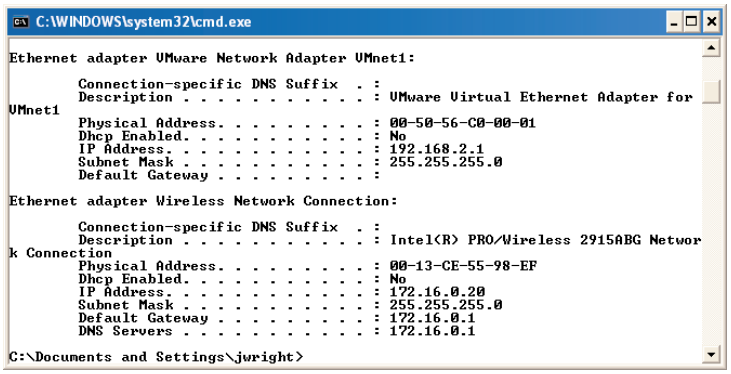
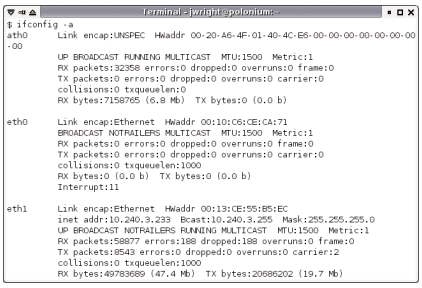


Figure 6.5 Linux MAC Address Information



Once we have identified the correct station address, you can use it to apply a display filter to your packet capture.

Filter for Station MAC

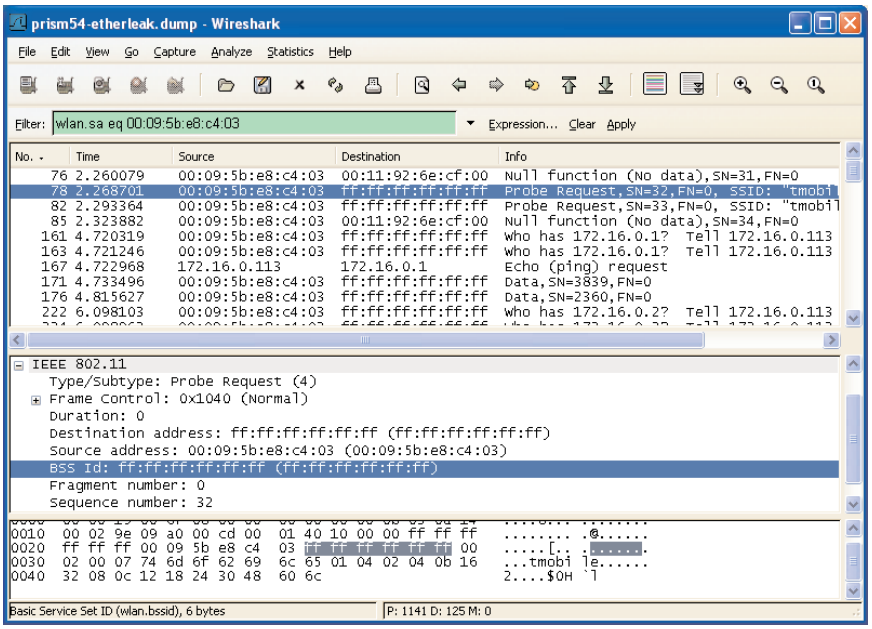
With the packet capture open, apply a display filter to display only traffic from the client station using the *wlan.sa* display field name. Assuming the station MAC address is *00:09:5b:e8:c4:03*, the display filter would be applied as:

```
wlan.sa eq 00:09:5b:e8:c4:03
```

A sample packet capture showing the results of this filter are shown in Figure 6.6.

From the Display Filter window, we see that 125 frames were returned from a packet capture of 1,141 total. However, when we examine the Packet Details window for the selected frame, we see that the BSSID is the broadcast address (*ff:ff:ff:ff:ff:ff*). This is because the selected frame is a *probe request* packet, which the client uses as a mechanism to discover networks in the area. We can refine our display filter to return only

Figure 6.6 Filtering on Source MAC Address



traffic destined specifically for the AP, by amending the display filter to return only frames with our station MAC address as the source that are not destined to the broadcast BSSID. The display filter now becomes:

```
wlan.sa eq 00:09:5b:e8:c4:03 and wlan.bssid ne ff:ff:ff:ff:ff:ff
```

This updated filter is shown in Figure 6.7.

Filter on BSSID

From the previous filter, we see that the BSSID for the station with the specified source address is *00:11:92:6e:cf:00*. We can use this information to apply a filter for only this BSSID, to exclude traffic from any other APs:

```
wlan.bssid eq 00:11:92:6e:cf:00
```

This final filter excludes any traffic not specifically destined to this AP, which will allow us to focus our analysis on this specific network.

Figure 6.7 Filtering on Source MAC Address and BSSID

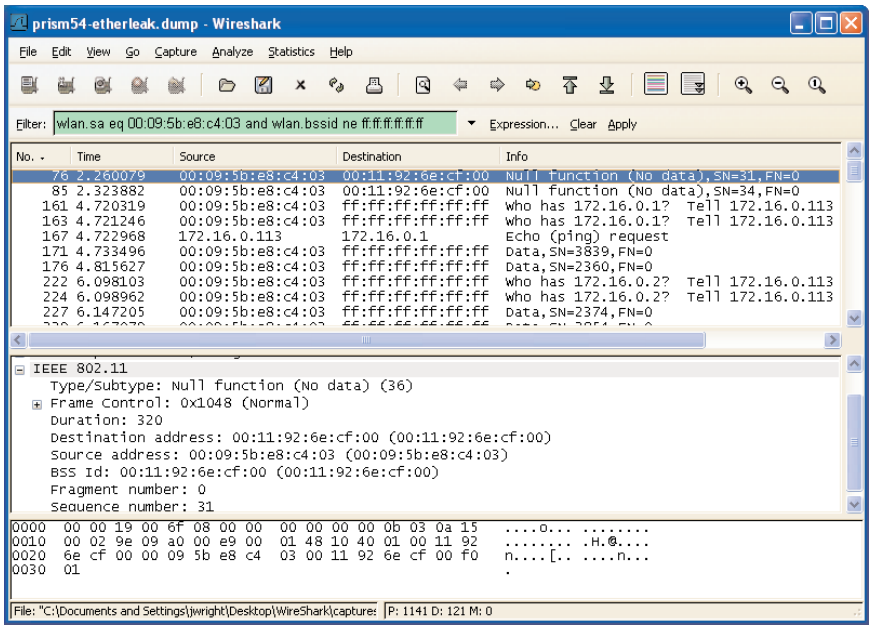
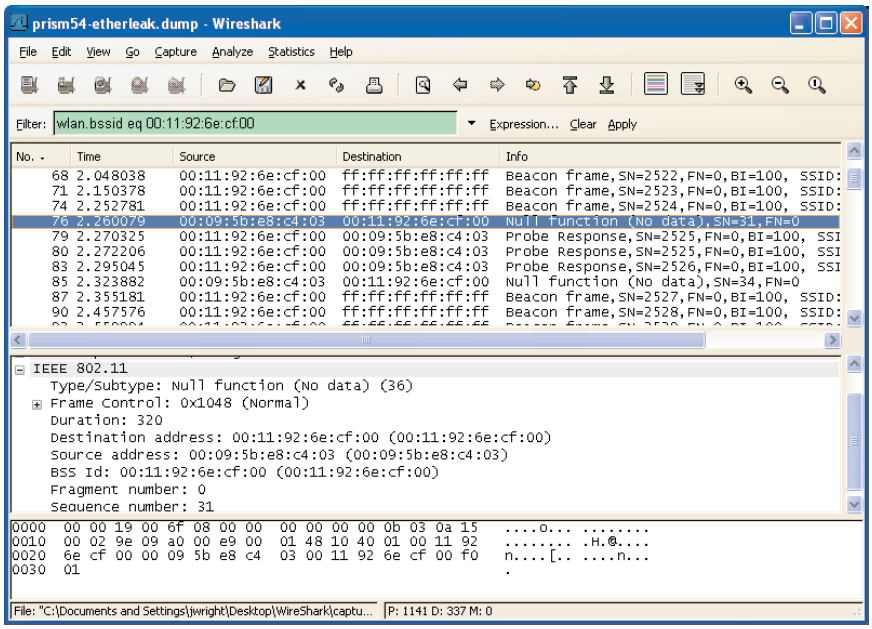


Figure 6.8 Filtering on BSSID



Traffic for a Specific Extended Service Set

Filtering for a specific BSS is useful if you can narrow your troubleshooting down to a specific AP; however, initially, you may need to take a broader look at your wireless network and assess traffic for all of the APs in your capture file. Indeed, many of the problems in wireless networking have to do with roaming between APs, which forces us to assess traffic from multiple APs. Fortunately, Wireshark display filters come to the rescue.

When you configure and deploy a wireless network, each AP is configured with one or more network names (or SSIDs), also known as an Extended SSID (ESSID). When you deploy multiple APs that facilitate a client's ability to roam between APs, all of the APs with the same SSID are referred to as participating in an Extended Service Set (ESS).

In the display filter example, our goal is to identify all of the traffic for a specific ESS identified by the SSID or network name. Unfortunately, we cannot apply a filter to identify all frames for a given SSID, as many management frames and all data and control frames do not include the SSID information. Instead, we need to enumerate all the BSSIDs for a specified SSID to develop an inclusive filter.

Filter on SSID

The first step is to apply a filter for a target SSID. As mentioned previously, this only returns management frames that include the SSID information element; however, it will present a list of all of the APs that use this SSID for additional filtering.

The SSID is included in the payload of beacon frames, probe response frames, and associate request frames. Navigate to this field by selecting any beacon frame, go to the Packet Details window, and then go to **IEEE 802.11 Wireless LAN Management Frame | Tagged Parameters | SSID Parameter Set | Tag Interpretation.** The display name for this field will be revealed as *wlan_mgt.tag.interpretation* (see Figure 6.9).

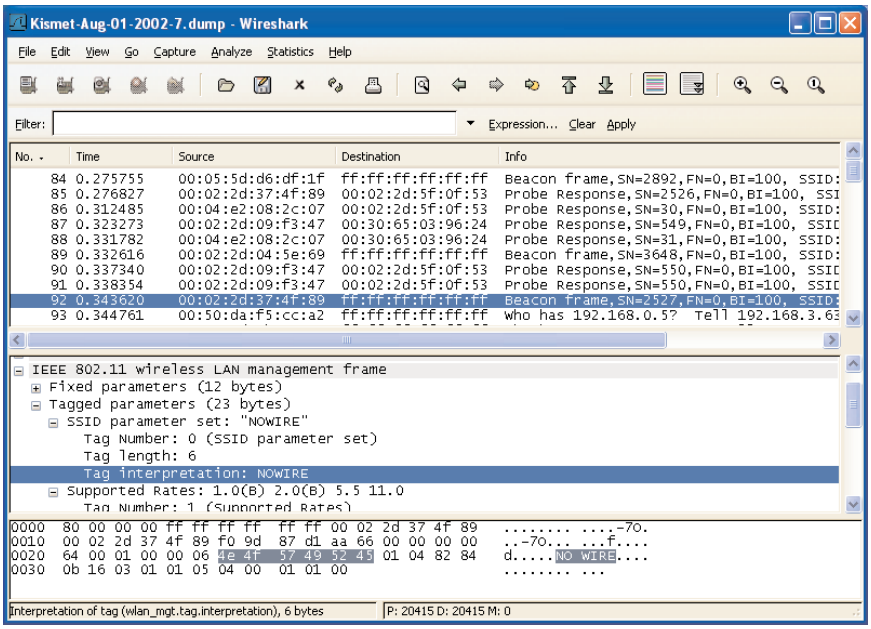
We can apply a display filter to identify all packets that includes the SSID *"NOWIRE"* as shown:

```
wlan_mgt.tag.interpretation eq "NOWIRE"
```

WARNING

All string references in Wireshark, including the SSID, are case-sensitive. When applying any filter that includes a string, ensure that you specify the proper case for a successful filter expression.

Figure 6.9 Displaying the SSID Tagged Parameter



Exclude Each BSSID

Once we have applied the filter on the SSID, the capture has been reduced to management frames (mostly beacon frames). Since our goal is to identify all of the traffic for the ESS, we need to modify this filter to identify all traffic for each BSS. Fortunately, the display filter for the SSID has revealed a list of all the APs configured with the specified SSID, which allows us to identify the BSSID for each AP.

In order to ensure that we have a complete list of all the BSSIDs, we start by applying an exclusive filter for each BSSID. Click on any beacon frame and navigate to the BSSID field by typing **IEEE 802.11 | BSS Id**. Using the display field name, *wlan.bssid*, add an exclusion display filter to the existing display filter for the given BSSID. For example, if the BSSID is *00:02:2d:37:4f:89*, our display filter becomes:

```
wlan_mgt.tag.interpretation eq "NOWIRE" and !(wlan.bssid eq 00:02:2d:37:4f:89)
```

Note that in this display filter, we are using the exclamation point as a negation operator, and testing for the BSSID equal to the specified address. This effectively returns all frames with a matching SSID except for the specified BSSID. Since our ultimate goal is to include only traffic from these BSSIDs, we negate the display filter with a leading exclamation point, which will make it easy to reverse the effect of the display filter simply by removing the exclamation point.

Next, we repeat this step for each of the remaining frames in the packet capture, selecting another BSSID and adding it to the exclusion list. For example, if the next BSSID is *00:40:05:df:93:c6*, it is added to our exclusion list:

```
wlan_mgt.tag.interpretation eq "NOWIRE" and !(wlan.bssid eq
00:02:2d:37:4f:89 or wlan.bssid eq 00:40:05:df:93:c6)
```

Repeat this process until there are no packets remaining in the capture display.

Invert Filter

At this point, our display filter should have no packets displayed. We have effectively identified each AP in the packet capture that is associated with the specified SSID. Now, we can modify the packet capture to invert the exclusion filter on the *wlan.bssid* field to include all of the specified addresses. For example, if our packet capture looks like this:

```
wlan_mgt.tag.interpretation eq "NOWIRE" and !(wlan.bssid eq
00:02:2d:37:4f:89 or wlan.bssid eq 00:40:05:df:93:c6 or wlan.bssid eq
00:40:96:36:80:f0)
```

We can modify it by removing the filter on the *wlan_mgt.tag.interpretation* field, and the exclamation point before the list of BSSIDs:

```
(wlan.bssid eq 00:02:2d:37:4f:89 or wlan.bssid eq 00:40:05:df:93:c6 or
wlan.bssid eq 00:40:96:36:80:f0)
```

Applying this filter will return all traffic for the specified BSSIDs, effectively excluding any traffic from neighboring networks that are not part of the initially specified SSID. This allows us to focus our analysis only on traffic to and from the networks associated with the initial SSID.

TIP

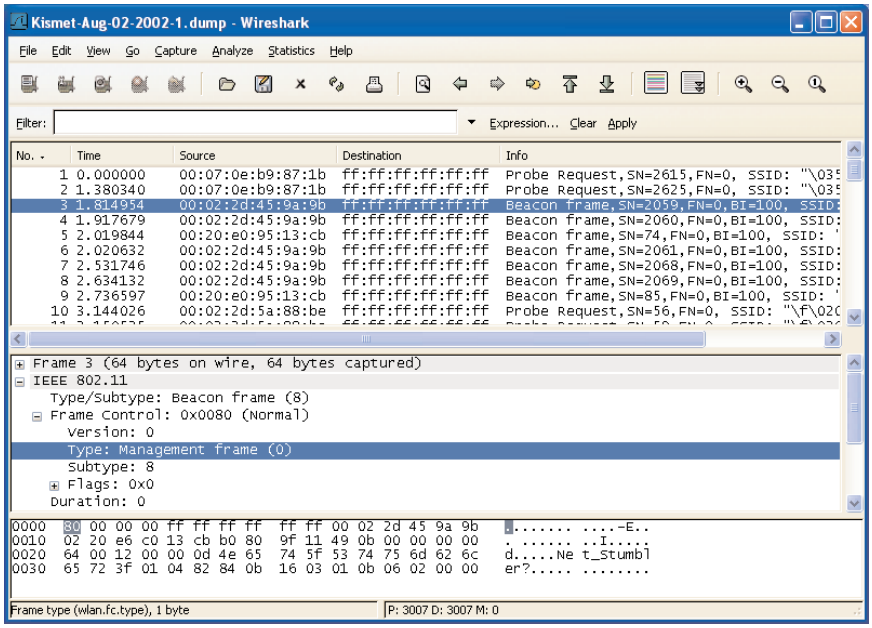
After applying a significant display filter (as shown in this example), it is wise to save the resulting packets in a new packet capture file. This way, you can assess the results of your filter at a later time without having to repeat the filtering process. To save an extract of packets, click **File | Save As**, and then click on the **Displayed** button. Enter an appropriate filename for the results of the display filter and click on **Save**.

You can also save the display filter itself by clicking **Analyze | Display Filters**. Enter a name for the display filter in the "Filter name" text box and click on **Save**. When you want to recall the filter, go to **Analyze | Display Filters** and double-click on the desired display filter name.

Even when there are no stations participating on the network, an AP will transmit at least ten packets a second to advertise the presence and capabilities of the network. These beacon frames are a vital component of any wireless network, but they can be difficult to assess a packet capture if these frames aren't particularly interesting to you.

Fortunately, we can easily apply a display filter to exclude these frames. In the Packet Details window, Wireshark identifies the type and subtype fields in the IEEE 802.11 header. By selecting a beacon frame, we can see that the type has a value of *0*, and the subtype has a value of *4* (see Figure 6.10).

Figure 6.10 Beacon Frame Type/Subtype



We can exclude these frames by applying a display filter as shown below:

```
!(wlan.fc.type eq 0 and wlan.fc.subtype eq 8)
```

Wireshark also gives us the "Type and Subtype Combined" field that can also be used for filtering. Instead of applying a filter on the Type and Subtype fields, we can apply a filter on the Combined Type and Subtype field as follows:

```
wlan.fc.type_subtype ne 8
```

Tools & Traps

Representing Wireless Frame Types

When assessing a wireless packet capture with Wireshark, it is common to apply display filters to look for or exclude certain frames based on the IEEE 802.11 frame type and frame subtype fields. If you are trying to exclude frames from a capture, it is easy to identify the Type and Subtype fields by navigating the Packet Details window and using the values for your filter. If you are looking for a specific frame type, however, you have to remember either the Frame Type and Subtype values, or the Combined Type/Subtype value assigned by Wireshark.

Instead of expecting you to memorize the 35+ values for different frame types, we've included them here for easy reference.

Frame Type/Subtype	Filter
Management Frames	*wlan.fc.type eq 0*
Control Frames	*wlan.fc.type eq 1*
Data Frames	*wlan.fc.type eq 2*
Association Request	*wlan.fc.type_subtype eq 0*
Association response	*wlan.fc.type_subtype eq 1*
Reassociation Request	*wlan.fc.type_subtype eq 2*
Reassociation Response	*wlan.fc.type_subtype eq 3*
Probe Request	*wlan.fc.type_subtype eq 4*
Probe Response	*wlan.fc.type_subtype eq 5*
Beacon	*wlan.fc.type_subtype eq 8*
Announcement Traffic Indication Map (ATIM)	*wlan.fc.type_subtype eq 9*
Disassociate	*wlan.fc.type_subtype eq 10*
Authentication	*wlan.fc.type_subtype eq 11*
Deauthentication	*wlan.fc.type_subtype eq 12*
Action Frames	*wlan.fc.type_subtype eq 13*
Block Acknowledgement (ACK) Request	*wlan.fc.type_subtype eq 24*
Block ACK	*wlan.fc.type_subtype eq 25*
Power-Save Poll	*wlan.fc.type_subtype eq 26*
Request to Send	*wlan.fc.type_subtype eq 27*

Continued

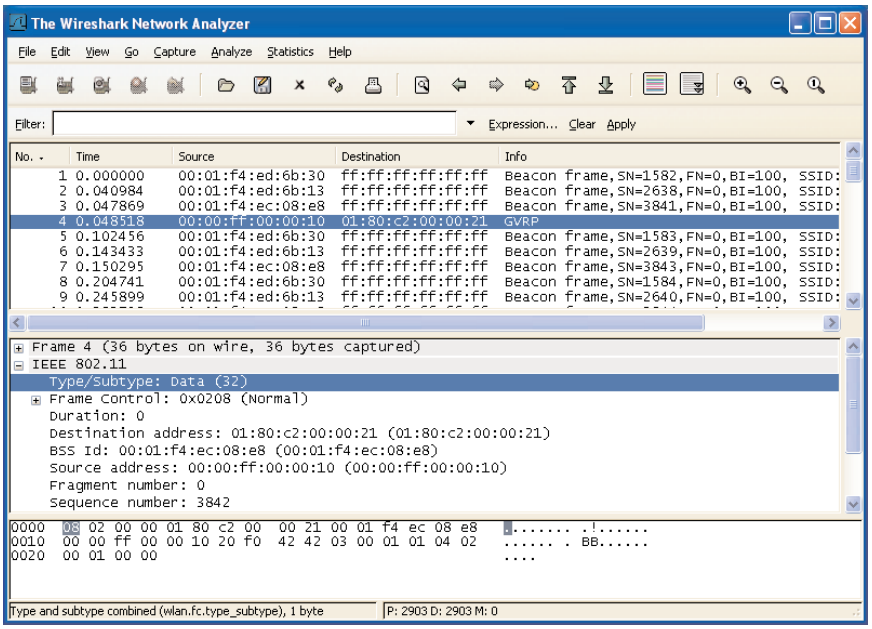
Frame Type/Subtype	Filter
Clear to Send	*wlan.fc.type_subtype eq 28*
ACK	*wlan.fc.type_subtype eq 29*
Contention Free Period End	*wlan.fc.type_subtype eq 30*
Contention Free Period End ACK	*wlan.fc.type_subtype eq 31*
Data + Contention Free ACK	*wlan.fc.type_subtype eq 33*
Data + Contention Free Poll	*wlan.fc.type_subtype eq 34*
Data + Contention Free ACK + Contention Free Poll	*wlan.fc.type_subtype eq 35*
NULL Data	*wlan.fc.type_subtype eq 36*
NULL Data + Contention Free ACK	*wlan.fc.type_subtype eq 37*
NULL Data + Contention Free Poll	*wlan.fc.type_subtype eq 38*
NULL Data + Contention Free ACK + Contention Free Poll	*wlan.fc.type_subtype eq 39*
QoS Data	*wlan.fc.type_subtype eq 40*
QoS Data + Contention Free ACK	*wlan.fc.type_subtype eq 41*
QoS Data + Contention Free Poll	*wlan.fc.type_subtype eq 42*
QoS Data + Contention Free ACK + Contention Free Poll	*wlan.fc.type_subtype eq 43*
NULL QoS Data	*wlan.fc.type_subtype eq 44*
NULL QoS Data + Contention Free Poll	*wlan.fc.type_subtype eq 46*
NULL QoS Data + Contention Free ACK + Contention Free Poll	*wlan.fc.type_subtype eq 47*

Data Traffic Only

Excluding beacon frames will reduce the amount of traffic in your wireless packet capture, but it will also leave many other types of packets including other management frames, control frames, and data frames. In some cases, you may only want to examine data traffic to assess potential information disclosure risks on the network, or as a measurement of efficiency for client traffic.

The process of applying a display filter for data traffic is similar to filtering beacon frames. Navigate to a data packet and inspect the Packet Details window to inspect the packet Type and Subtype Combined field (see Figure 6.11).

Figure 6.11 Data Frame Type/Subtype



In Figure 6.11, we see that the Type and Subtype Combined field has a value of 32. We can use this field to apply a display filter that displays only this type of packet:

```
wlan.fc.type_subtype eq 32
```

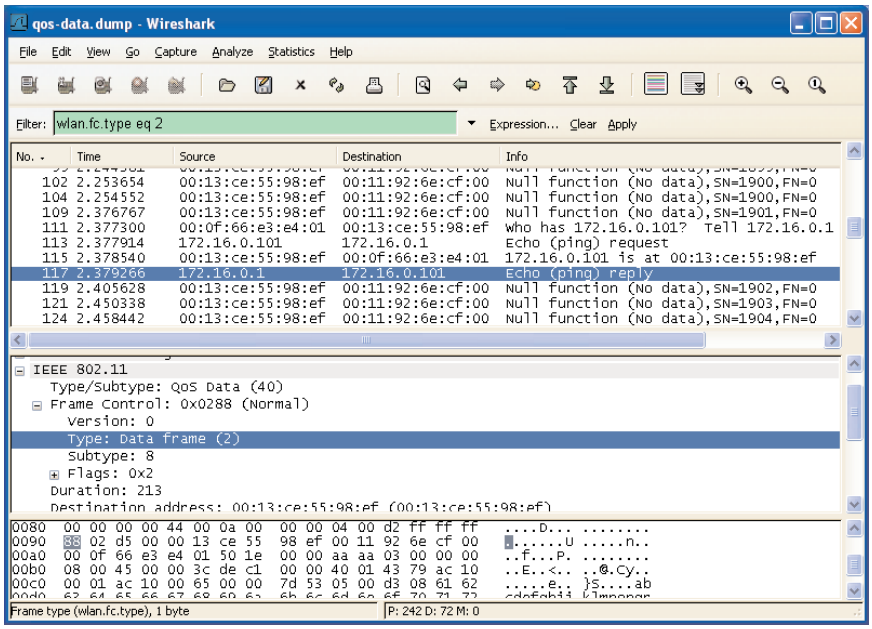
While this display filter is effective at excluding traffic, it can be too restrictive for some analysis needs. Remember that that Type and Subtype Combined field is a unique identifier for both field values. When we apply a filter to display only frames with a Type and Subtype combined value of 32, we exclude other types of data frames including QoS marked wireless frames. An alternative display filter is to examine only the IEEE 802.11 type field without referencing the subtype field as well:

```
wlan.fc.type eq 2
```

A sample of this display filter is shown in Figure 6.12.

With this modified display filter, we can see all of the data frames, regardless of the Subtype field. In this example, we can see normal data traffic (such as the Internet Control Message Protocol [ICMP] request and reply frames), but we also have NULL data frames. NULL data frames are used by some APs and station cards to enter power conservation mode, or are used right before switching frequencies to

Figure 6.12 Limiting Data Frame Type Filter



scan for other nearby networks. If NULL data frames aren't useful in your analysis, you can exclude them by modifying the display filter:

```
wlan.fc.type eq 2 and !(wlan.fc.subtype eq 4)
```

If this display filter reveals more data subtypes than are necessary for your analysis, add additional exclusion filters inside the parenthesis, separated by the *or* keyword.

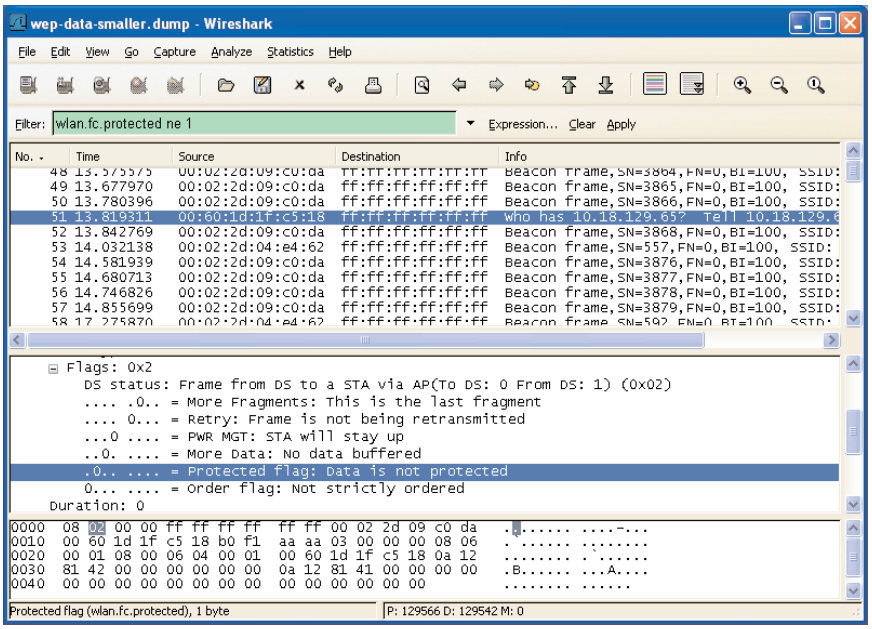
Unencrypted Data Traffic Only

Another common analysis technique is to identify wireless traffic that is not encrypted. This may be in an effort to identify misconfigured devices that could be disclosing sensitive information over the wireless network, or as part of an audit to ensure wireless traffic is encrypted, or to identify rogue APs, since most rogue devices are deployed with no encryption.

As seen in the IEEE 802.11 header analysis, one of the bits in the frame control header is known as the *protected bit* (formerly known as the WEP bit, or the privacy bit). The protected bit is set to *1* when the packet is encrypted using an IEEE 802.11 encryption mechanism such as WEP, TKIP, or CCMP; otherwise it is set to *0*. We can apply a filter using this field to identify all unencrypted wireless traffic:

```
wlan.fc.protected ne 1
```

Figure 6.13 Excluding Encrypted Frames



A sample packet capture with this display filter applied is shown in Figure 6.13.

While this filter shows the unencrypted wireless traffic, it is not the most effective display filter because it also reveals unencrypted management and control frames. Since these frames are always unencrypted, we can extend the display filter to identify unencrypted data frames only to get the most effective analysis:

```
wlan.fc.protected ne 1 and wlan.fc.type eq 2
```

NOTE

At the time of this writing, the available encryption mechanisms for IEEE 802.11 wireless networks only apply to data frames, and do not provide any confidentiality for management or control frames. However, this is slated for change with the ratification of the IEEE 802.11w amendment that was designed to extend security to management traffic as well as data traffic. The IEEE 802.11w task group is scheduled to ratify the *Protected Management Frames* amendment in April 2008.

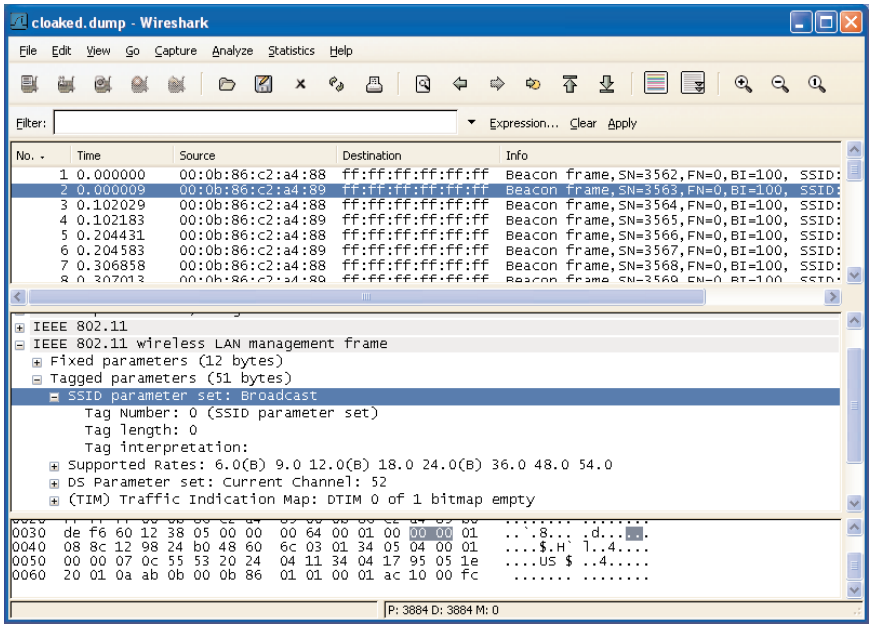
Identifying Hidden SSIDs

Many organizations have adopted SSID cloaking, or prevented their APs from advertising their SSIDs to anyone who asks. While this provides a minimal measure of security, it is an ineffective mechanism for controlling access to the network and should only be used in conjunction with a strong encryption and authentication mechanism.

When an AP wants to obscure the SSID of the network, it does not respond when it receives a request for the network name, and it removes the SSID advertisement from beacon frames. Because it is mandatory to include some indicator of the network name (whether legitimate or not) in beacon frames, vendors have adopted different conventions for obscuring the SSID by replacing it with one or more space characters or NULL bytes (one or more *0*s) or an SSID with a length of *0*. An example of a cloaked SSID represented by Wireshark is shown in Figure 6.14.

In this case, the SSID for this network has been replaced with an empty value *0*

Figure 6.14 Cloaked SSID Tagged Parameter

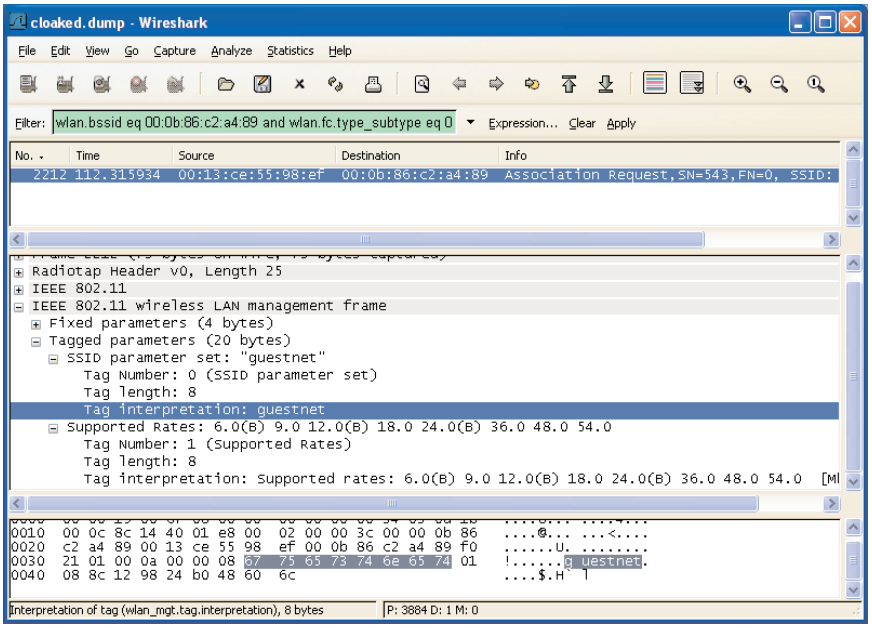


bytes in length. While this may prevent the disclosure of the SSID to the casual observer, stations will still send the SSID in plaintext over the network each time they associate to the wireless network. In this example, we see that the BSSID of the network is *00:0b:86:c2:a4:89*; we can apply a display filter for this network BSSID and associate request frames to examine the SSID name sent by the client:

```
wlan.bssid eq 00:0b:86:c2:a4:89 and wlan.fc.type_subtype eq 0
```

By applying this filter, we reveal any association requests for the specified BSSID. By clicking **IEEE 802.11 Wireless LAN Management Frame | Tagged Parameters | SSID Parameter Set**, we can see the SSID specified by the client station, revealing the SSID for the network as *guestnet* (see Figure 6.15).

Figure 6.15 Revealed SSID on a Cloaked Network



Examining EAP Exchanges

So far, we've limited our usage of display filters to the IEEE 802.11 header and management payload data. Wireshark can also identify and apply display filters to other wireless-related protocols including the Extensible Authentication Protocol (EAP).

EAP is used in conjunction with the IEEE 802.1x network authentication mechanism to authenticate users to a wireless network by using one of several EAP methods. Common EAP methods include the Protected Extensible Authentication Protocol (PEAP), the Extensible Authentication Protocol with Transport Layer Security (EAP/TLS), Tunneled Transport Layer Security (TTLS) and the Lightweight Extensible Authentication Protocol (LEAP). By examining the exchange of EAP data with Wireshark, we can troubleshoot user authentication issues, evaluate potential security risks, and discover architecture components used by the wireless network.

To identify any EAP traffic in a capture file, apply a display filter for the EAP Over LAN protocol:

```
eapol
```

This filter will return any EAP traffic present in the capture file, including authentication requests, identity negotiation, key and encryption negotiation exchanges, and success or failure messages. Next, we examine each data exchange mechanism in the EAP exchange.

Identifying the EAP type

If you are auditing a wireless network or trying to identify potentially misconfigured client systems, you may need to identify the EAP method used by those client systems. The EAP method is reported in an EAP exchange in the EAP type field. We can use a display filter to identify frames that report this information:

```
eap.type
```

After applying this filter, select a frame and navigate to the EAP type field by clicking **802.1x Authentication | Extensible Authentication Protocol | Type** in the Packet Details window. Wireshark will identify the numeric value for the EAP type, the name of the EAP type, and the last name of the primary author who developed the Internet Engineering Task Force (IETF) draft to describe the EAP type. In Figure 6.16, we can see that the EAP type has a value of 25, which indicates the use of the PEAP protocol.

Evaluating Username Disclosure

EAP methods that rely on username and password authentication include PEAP, TTLS and LEAP. These methods may disclose user identity information (e.g., a username) in plaintext over the wireless network. This can be an information disclosure risk for some organizations, because it allows an attacker to enumerate valid usernames, which can be the basis for additional attacks against the wireless network.

Wireshark allows you to identify usernames that are disclosed on the network by examining the EAP Type field for a special value indicating identity information is present in the frame. This EAP mechanism uses an initial *EAP Identity Request* from the AP or the client to request the identity information, followed by an *EAP Identity Response* that contains the identify information. We can apply a display filter to return only EAP Identify Response frames by filtering on the EAP type and EAP code fields:

```
eap.type eq 1 and eap.code eq 2
```

In the sample packet capture displayed in Figure 6.17, we see that the identity information has been disclosed as the username *jwright*.

Figure 6.16 Identifying the EAP Type

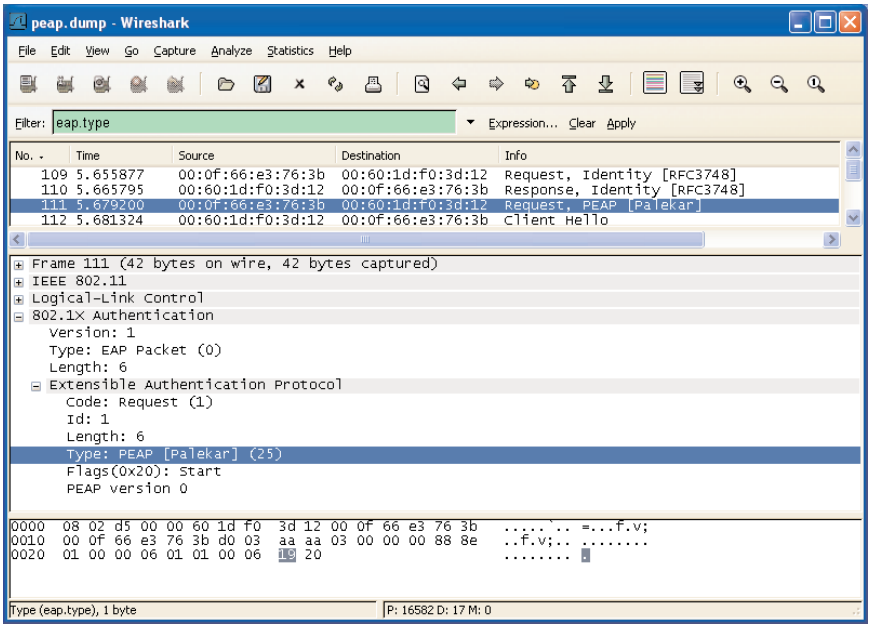
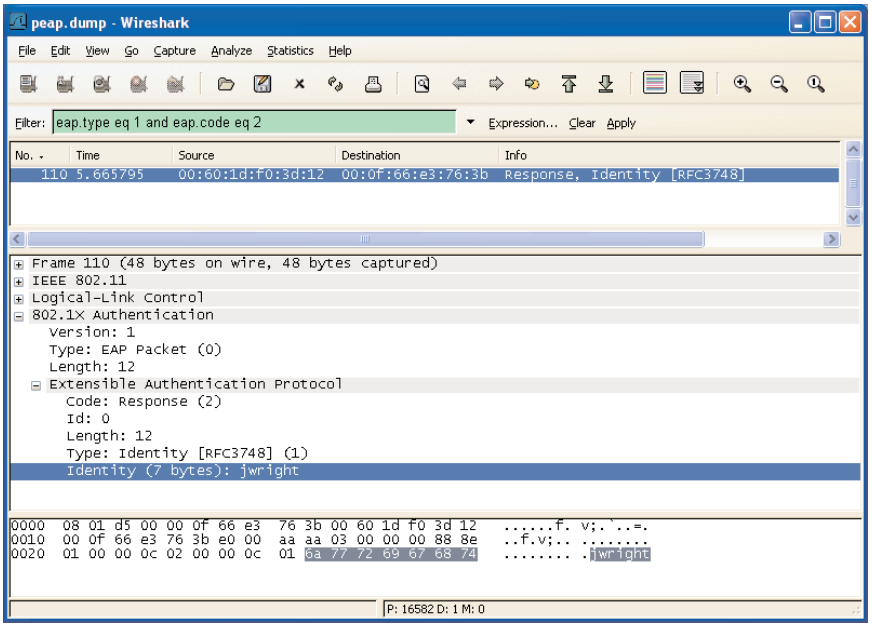
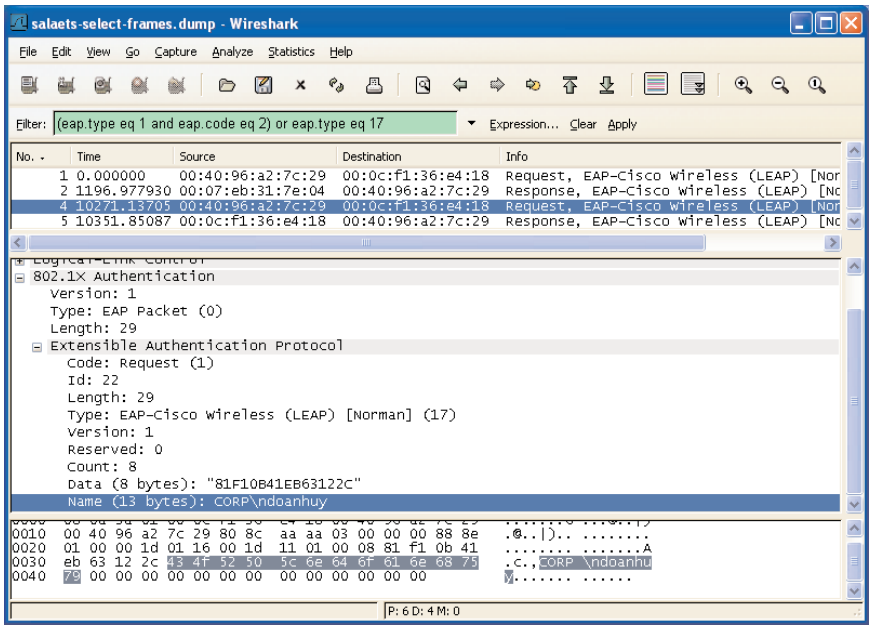


Figure 6.17 EAP Identity Disclosure



A notable exception to this rule for identifying EAP identity information is the case of non-standard EAP types; specifically, the LEAP protocol. In a LEAP exchange, identify information is not exchanged using the EAP Identity type; rather, it is included in EAP request and response frames in the data payload. We can modify our display filter to accommodate for this inconsistency by also identifying any LEAP traffic with an additional clause for the LEAP EAP type (see Figure 6.18).

Figure 6.18 Cisco LEAP Identity Disclosure



In Figure 6.18, we see that the EAP type is LEAP, and the identity information is disclosed as *CORP\ndoanhuy*. From this identity information, we can also ascertain that the target network is using a Microsoft Windows infrastructure with the domain name *CORP*.

Identifying EAP Authentication Failures

Troubleshooting authentication problems on the wireless network can be challenging, and often requires a packet sniffer to determine if the failure is happening on the client or over the network. Wireshark can assist in providing this information by identifying EAP authentication failure messages.

The EAP code field is present in all EAP packets, and is used to indicate the content of the message that follows. Presently there are four EAP codes:

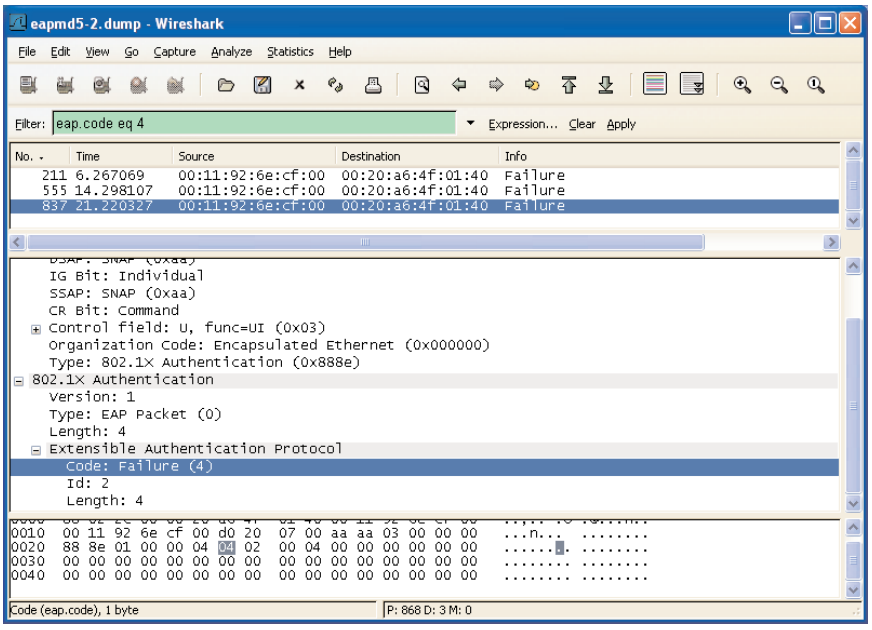
- **Code 1 - EAP Request** A value of 1 in the EAP Code field indicates that the EAP frame is requesting information from the recipient. This can be identity information, encryption negotiation content, or a response-to-challenge text.
- **Code 2 - EAP Response** A value of 2 in the EAP Code field indicates that the EAP frame is responding to an EAP Request frame.
- **Code 3 - EAP Success** A value of 3 in the EAP Code field indicates that the previous EAP Response was successful. This is primarily used as a response to authentication messages.
- **Code 4 - EAP Failure** A value of 4 in the EAP Code field indicates that the previous EAP Response failed authentication.

We can apply a display filter on the EAP Code field to identify EAP failures:

```
eap.code eq 4
```

The result of this display filter on a sample packet capture is displayed in Figure 6.19.

Figure 6.19 EAP Failure Notification



In Figure 6.19, we can see three authentication failures at approximately 8 seconds and 7 seconds apart. We can also see that the From DS bit is set in the IEEE 802.11 header, indicating that the failure message is coming from the AP. From this, we can determine that the failure message is coming from the client system, not from the network.

Identifying Key Negotiation Properties

Some EAP methods negotiate a Transport Layer Security (TLS) tunnel before exchanging authentication information to protect weak authentication protocol data. In order to establish the TLS tunnel, at least one digital certificate is transmitted from the AP to the station. We can use Wireshark to examine this certificate information, and possibly determine other sensitive information about the network including the organization name and address.

First, apply a display filter to identify the portions of the EAP exchange that are exchanging Secure Sockets Layer (SSL) digital certificate content:

```
eap and ssl.handshake.type eq 11
```

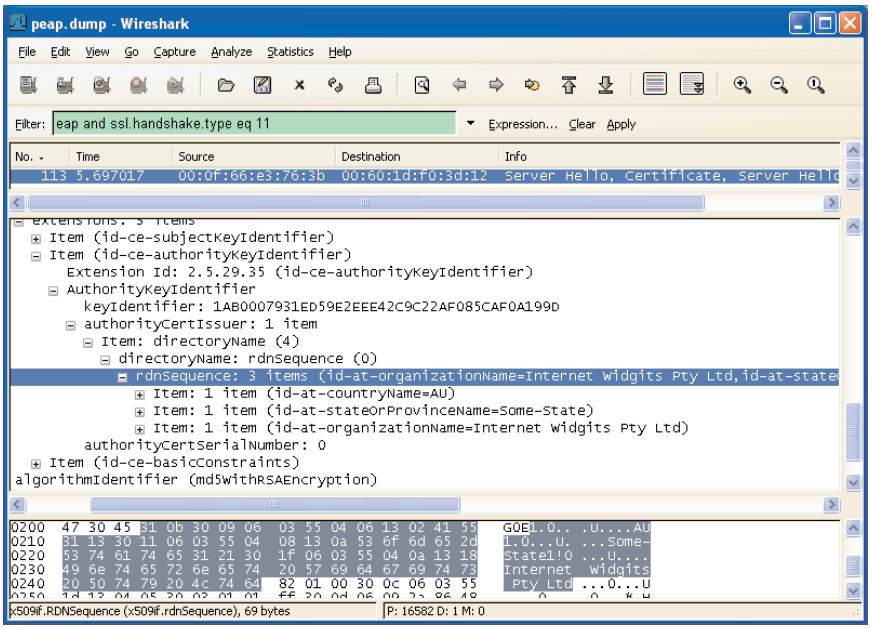
This filter will display only EAP traffic with embedded SSL information that includes a SSL handshake exchange type that includes a digital certificate (type 11). After selecting the frame, navigate to the Packet Details window and click **802.1X Authentication | Extensible Authentication Protocol | Secure Socket Layer | TLS Record Layer: Handshake Protocol: Certificate | Handshake Protocol: Certificate | Certificate | Certificate | signedCertificate | Extensions | AuthorityKeyIdentifier | Item | directoryName | rdnSequence**. This will reveal the digital certificate content indicating the country name, state, and possibly address information and the organization name to which the certificate was issued. A sample packet capture that reveals certificate content is shown in Figure 6.20.

In Figure 6.20, we see that the certificate content reveals the organization name as *Internet Widgits Pty Ltd*, with the country identifier *AU* (Australia) and the state or province name *Some-State*.

Identifying Wireless Encryption Mechanisms

The IT industry analysis group Gartner published a report indicating that 70 percent of successful attacks against wireless LANs will be due to the misconfiguration of APs and wireless clients. This prediction is easy to believe; many organizations deploy wireless networks without auditing their post-deployment environment with a tool like Wireshark. With the complexity of wireless APs and client systems, it is easy to

Figure 6.20 SSL Digital Certificate Content



make a configuration mistake that exposes devices to weak encryption mechanisms, or no encryption.

We can assess wireless packet captures with Wireshark to identify the security mechanisms used to protect the network. We've learned how to identify the authentication mechanisms that are in place by looking for EAP traffic; we can also assess the encryption mechanism using display filters.

Common encryption mechanisms on wireless networks include standard IEEE wireless LAN encryption protocols such as WEP, TKIP and CCMP, as well as upper-layer encryption mechanisms such as Secure Internet Protocol (IPSec)/Virtual Private Network (VPN). We examine each of these mechanisms and how we can assess a packet capture to identify the encryption protocol in use.

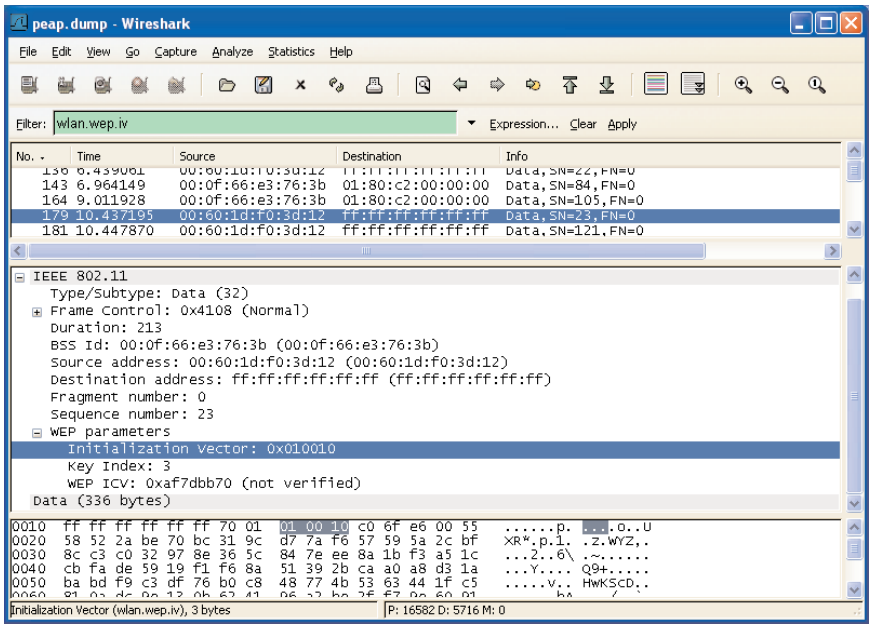
Identifying WEP

WEP is the most prevalent encryption mechanism used to protect wireless networks; however, it is also widely known as an insecure protocol. Wireshark uniquely identifies WEP-encrypted traffic by decoding the 4-byte WEP header that follows the IEEE 802.11 header. We can identify WEP traffic by identifying any frames that include the mandatory WEP Initialization Vector (IV):

```
wlan.wep.iv
```

A sample packet capture with this filter applied is shown in Figure 6.21. Wireshark identifies the WEP header and the IV value, along with the key index value and the WEP integrity check value (ICV).

Figure 6.21 Identifying WEP Traffic



Identifying TKIP and CCMP

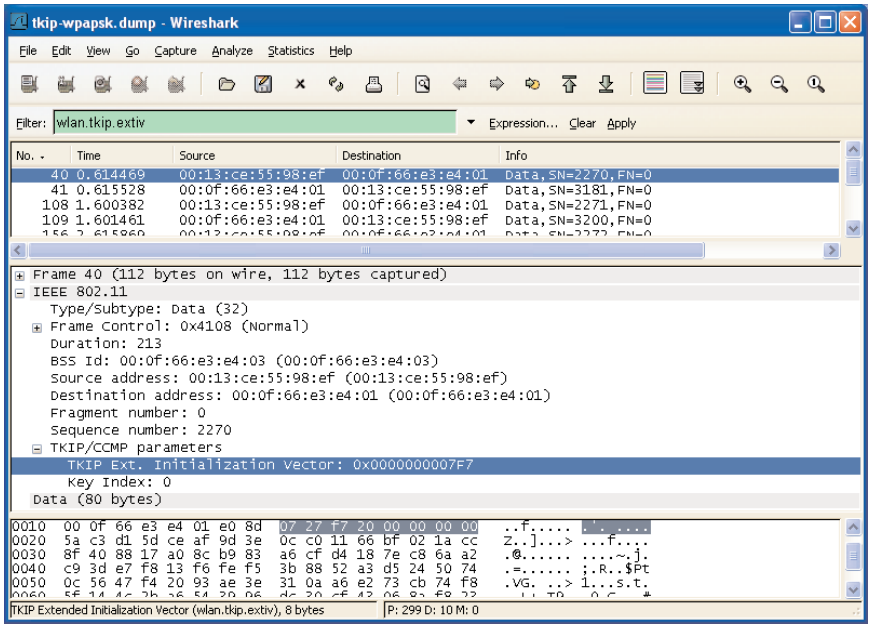
TKIP is the successor to WEP, and is designed to be a software upgrade for hardware built only to support WEP. Since TKIP was designed to work on legacy WEP hardware, it retained the use of the same underlying encryption protocol, Ron's Code 4 (RC4). And while RC4 is still considered safe for current use, it is no longer an acceptable encryption mechanism for use by U.S. government agencies. Another alternative is to use the CCMP protocol, which uses the Advanced Encryption System (AES) cipher.

Like WEP, both TKIP and CCMP use an encryption protocol header that follows the IEEE 802.11 header. This header is modified from the legacy WEP header, allowing us to identify whether TKIP or CCMP are in use, but does not allow us to differentiate TKIP from CCMP; we can only determine that one or the other is currently in use by looking at this header. We can use a display filter to identify this header by filtering on the extended IV field:

```
wlan.tkip.extiv
```

Despite the use of *tkip* in this display filter, it's not possible to differentiate between TKIP or CCMP by looking at the encryption header. A sample packet capture that displays this filter and the TKIP/CCMP header is shown in Figure 6.22.

Figure 6.22 Identifying TKIP or CCMP Traffic



By applying this filter, we know that either TKIP or CCMP is in use for this BSS. To further differentiate whether TKIP or CCMP is in use, we need to inspect data in a beacon frame. To identify a beacon frame for this network, apply a display filter using the BSSID identified with this filter, looking for packets with the type/subtype for a beacon frame.

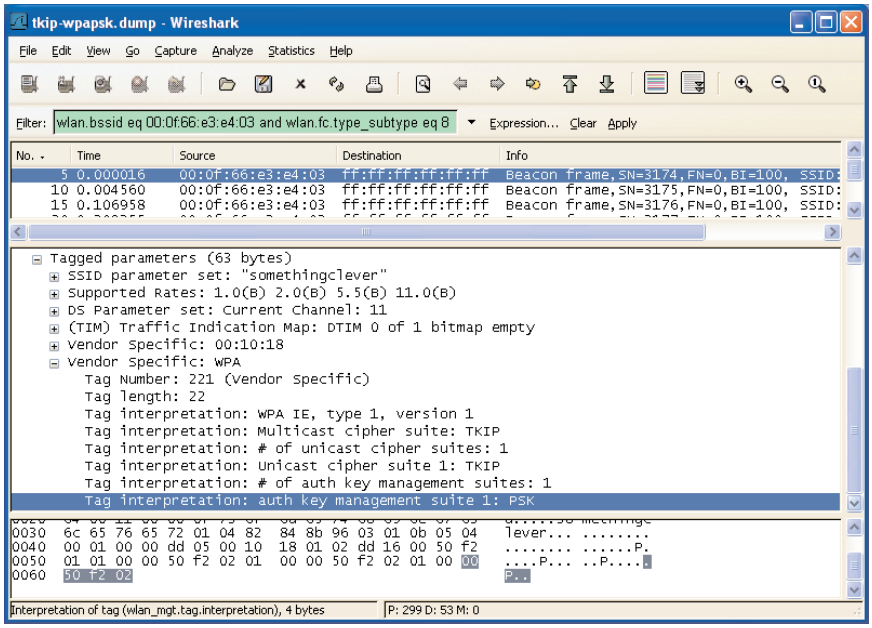
```
wlan.bssid eq 00:0f:66:e3:e4:03 and wlan.fc.type_subtype eq 8
```

After applying this filter, navigate to the beacon frame's tagged information element data by clicking **IEEE 802.11 Wireless LAN Management Frame | Tagged Parameters**. Look for an information element labeled "Vendor Specific: WPA" or "RSN Information." A Wi-Fi Protected Access (WPA) information element indicates that the AP has passed the testing certification program designed by the WiFi Alliance (WFA), known as WiFi Protected Access (WPA), while a RSN information element tag indicates that the AP has implemented the robust security network (RSN) standards in the IEEE 802.11i amendment. In either case, expand the information element to identify the encryption mechanism used for Unicast and

multicast traffic (either AES indicating the CCMP cipher or TKIP). We can also determine the key derivation mechanism (how the dynamic keys are generated) used for the network, by examining the value next to the *auth key management suite* string; either PSK indicating a pre-shared key, or WPA indicating key derivation from a Remote Authentication Dial-In User Service (RADIUS) server over IEEE 802.1x.

A sample packet capture shown in Figure 6.23, demonstrates a beacon frame's information element indicating encryption information. In this example, we see that the "Vendor Specific: WPA" information element, which indicates both the multicast and Unicast cipher suites, are using the TKIP algorithm, with Pre-Shared Key (PSK) as the authentication key management suite mechanism.

Figure 6.23 Identifying Multicast and Unicast Cipher Suites - Authentication Mechanism



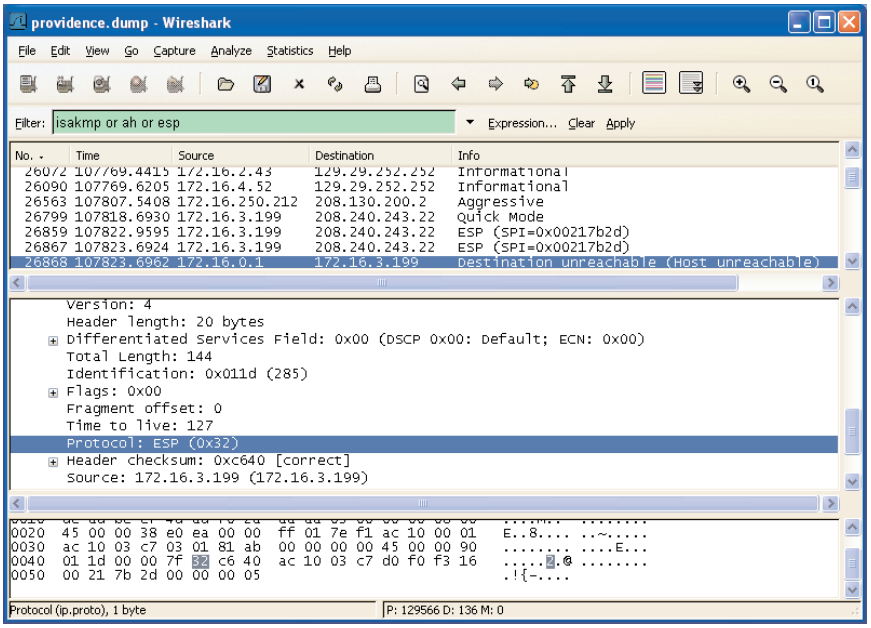
Identifying IPSec/VPN

Some wireless networks will not use the standard IEEE 802.11 encryption mechanisms, instead opting for an upper-layer encryption mechanism such as IPSec. Wireshark can identify this type of encryption mechanism by applying a display filter for any of the associated IPSec protocols such as the Internet Security Association and Key Management Protocol (ISAKMP), the Encapsulating Security Payload (ESP), or the Authentication Header (AH) protocol. To identify IPSec traffic, apply a display filter as follows:

```
isakmp or ah or esp
```

This filter will return any of the associated IPSec protocols, as shown in Figure 6.24.

Figure 6.24 Identifying IPSec/VPN Traffic



Note in Figure 6.24 that an ICMP Destination Unreachable packet is also returned. This is because Wireshark also decodes the embedded protocol within the ICMP packet, which includes ESP information.

So far, we have seen how powerful Wireshark's display filter functionality can be. The usefulness of display filters is only limited to the fields that can be identified by display name, and your creativeness in taking advantage of this powerful feature. What's more, display filters can be used for other classification and identification mechanisms as well, including the ability to colorize packets matching arbitrary display filters.

Leveraging Colorized Packet Displays

Looking at a packet trace of any more than a handful of packets can be intimidating. If you aren't sure exactly what it is you are looking for in the packet capture, you're often left to blindly click on packets to examine the contents, or to start applying predefined display filters in the hope of identifying something useful.

In order to make it easier to examine and assess a packet capture at a glance, Wireshark allows you to customize the color of packets in the Packet List window. This often overlooked feature can be very useful for assessing a packet capture, and to simplify the process of troubleshooting a wireless connection issue when applied with useful display filters.

Marking From DS and To DS

When examining a packet capture, it is helpful to identify if the traffic is originating from the wired network or the wireless network. For example, if you see traffic coming from a local IP address, it may not necessarily be traffic from a wireless station; it may be a wired station that is communicating with a wireless station or a wired station sending broadcast traffic.

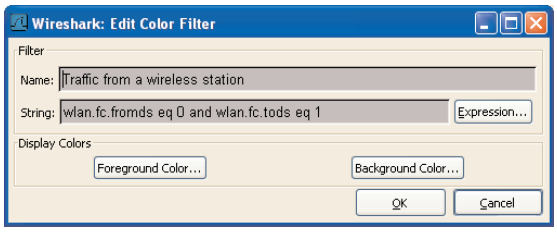
We can determine if traffic is originating from the wireless network by examining the flags in the frame control header, looking for the From DS bit and the To DS bit set. If the From DS bit is set and the To DS bit is clear, we know that the traffic originated from the wired network (or the AP).

Applying this logic to a coloring rule that marks traffic from the wired network using one color and traffic from the wireless network using a different color, allows us to determine where the traffic originated. To access the coloring rules dialog box click **View | Coloring rules.** Click **New** to create a new coloring rule with the following properties:

- **Filter Name:** Traffic from a wireless station
- **Filter String:** *wlan.fc.fromds eq 0* and *wlan.fc.tods eq 1*

Next, select a foreground color and a background color to uniquely identify traffic from a wireless station. The Name and String dialog boxes will update to reflect the colors you select (see Figure 6.25).

Figure 6.25 Color Filtering Traffic From a Wireless Station



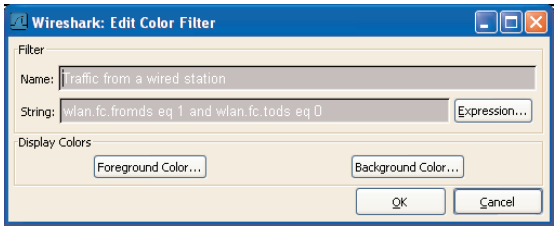
When you are happy with your selection, press **OK**.

Next, create a second coloring rule to mark traffic originating from the wired network using the following properties:

- **Filter Name:** Traffic from a wired station
- **Filter String:** *wlan.fc.fromds eq 1* and *wlan.fc.tods eq 0*

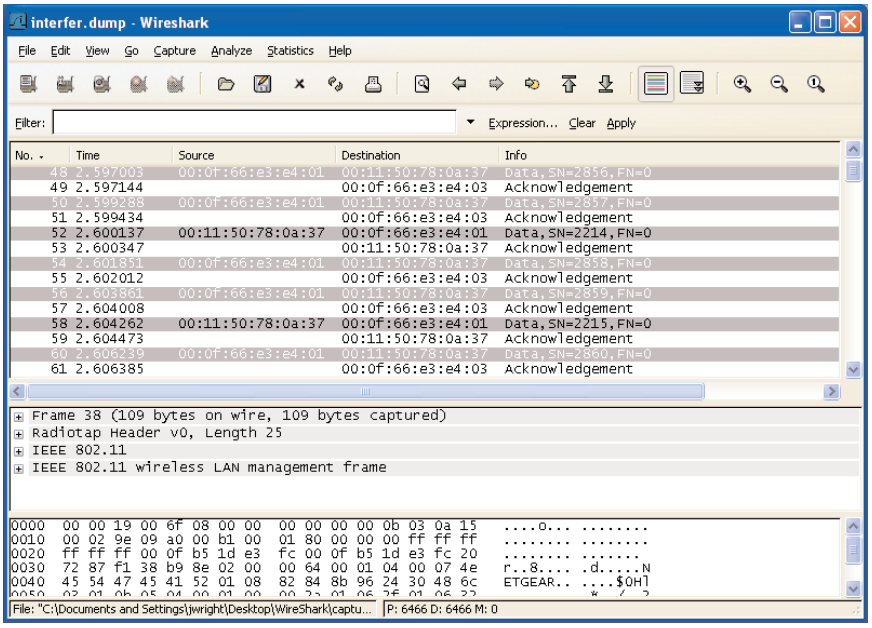
Select a foreground color and a background color to uniquely identify this traffic (see in Figure 6.26).

Figure 6.26 Color Filtering Traffic From a Wired Station



Press **OK** to accept the new filter, and then press **Apply**. If you want to save this color filter for later analysis, press **Save**; otherwise press **OK** to close the Coloring Rules dialog box. Wireshark will automatically update your packet display to apply the new coloring rules (see Figure 6.27).

Figure 6.27 Applied Wired/Wireless Traffic Coloring Rules



In Figure 6.27, we can identify frames 52 and 58 as traffic originating on the wireless network, and frames 48, 50, 54, 56, and 60 as originating on the wired network. The remaining frames have neither the From DS nor To DS bits set, which is appropriate for management and control frames.

Marking Interfering Traffic

As more organizations deploy wireless networks, the amount of interference from neighboring networks grows, which can have an adverse affect on the performance of wireless LANs. While many organizations go to significant trouble to select channel plans that minimize interference for a given BSS, it's not uncommon for an AP from a neighboring organization to occupy similar frequencies and interfere with your network.

Earlier in this chapter, we learned how to create a display filter to identify all of the APs for a specific BSS. We can apply this filter to Wireshark's coloring rules using an inverse display filter, to easily identify traffic from interfering networks. Assuming our list of BSSIDs includes *00:0f:66:e3:e4:03* and *00:0f:66:e3:25:92*, create a new coloring rule with the following properties:

- **Filter Name:** Interfering networks
- **Filter String:** *!(wlan.bssid eq 00:0f:66:e3:e4:03* or *wlan.bssid eq 00:0f:66:e3:25:92*) and *!wlan.fc.type eq 1*

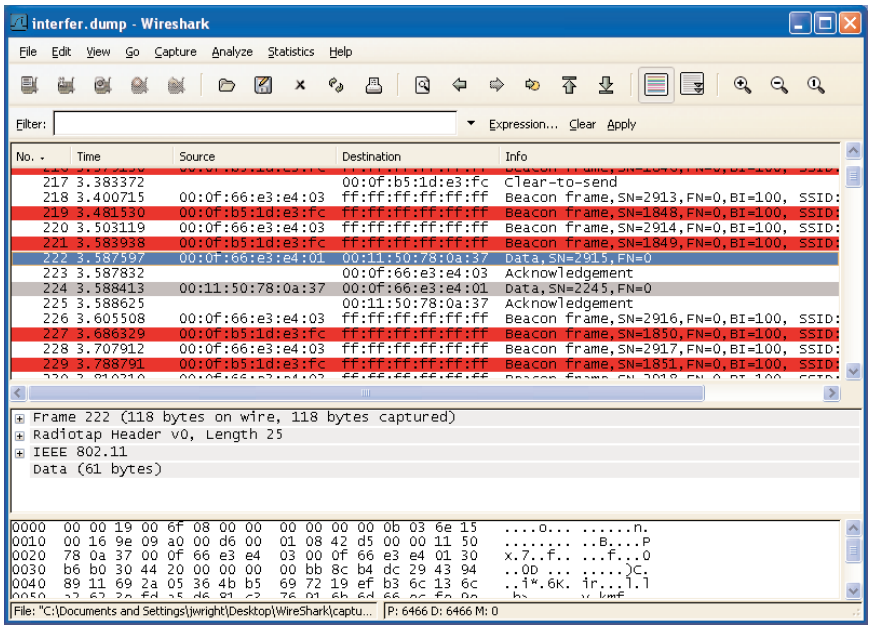
In this display filter, we exclude any traffic that is from one of the specified BSSIDs, as well as any control frames, since control frames do not specify a BSSID in the IEEE 802.11 header. Next, assign a foreground color and a background color to uniquely identify this coloring rule, then press **OK** and **Apply**. Wireshark will update the display to reflect the new coloring rule and allow you to identify interfering networks (see Figure 6.28).

Marking Retries

For each data frame transmitted by a wireless station of the AP, the recipient must transmit an ACK frame to indicate the successful delivery of the packet. If the packet was not received or was received in a corrupted state, the recipient waits for the source to retransmit the packet. In all retransmitted packets, the retransmit bit in the frame control header is set.

Evidence of retransmitted frames can indicate interference on the network that is causing the initial delivery of packets to fail. We can create a coloring rule to help us identify retransmitted frames using the following properties:

Figure 6.28 Marking Interfering Network Traffic



- **Filter Name:** Retransmitted frames
- **Filter String**: *wlan.fc.retry eq 1*

Once we apply this color filter, Wireshark will highlight retransmitted frames (see Figure 6.29).

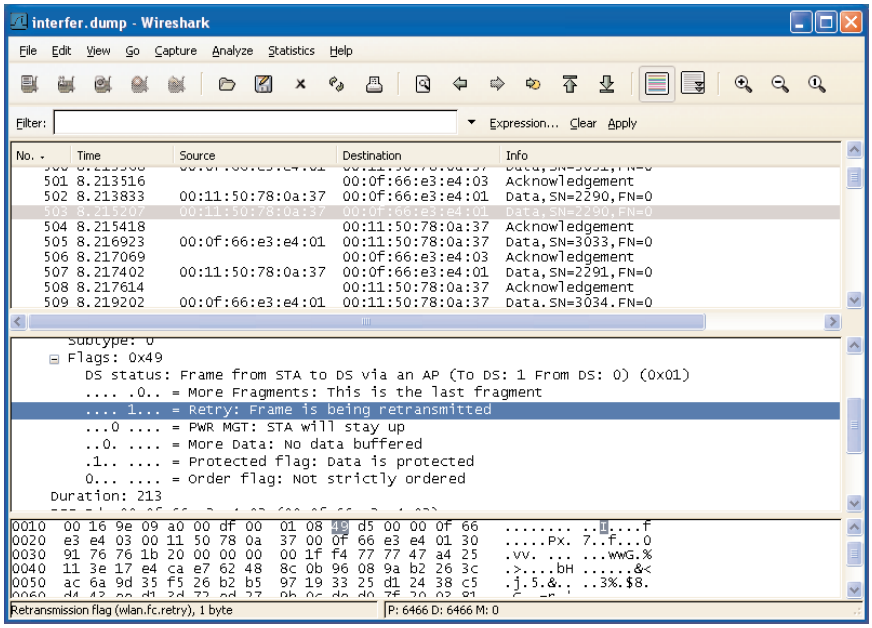
In Figure 6.29, frame 503 is a retransmit of frame 502. Notice that frame 500 was sent to the station at *00:11:50:78:0a:37* and then acknowledged within 2/1000th of a second. The frame at 502 was not positively acknowledged, or there was interference that caused the loss of the ACK frame, which then required a retransmit.

Creative use of custom coloring rules can make analyzing a packet capture much easier. Remember that you can use any Wireshark display filter to create a custom coloring rule, making this feature very flexible and effective at easily identifying important traffic characteristics.

Adding Informative Columns

By default, Wireshark displays six columns in the Packet List window, including the frame number, time, source, destination, protocol, and information string. Wireshark allows you to customize this view, including the ability to add two additional columns that are pertinent to wireless packet captures: the IEEE 802.11 RSSI and IEEE 802.11 TX Rate columns.

Figure 6.29 Marking Retransmitted Frames



The IEEE 802.11 Received Signal Strength Indication (RSSI) column gives you an indicator as to the radio signal strength for the selected packet, while the IEEE 802.11 RX Rate column indicates the data rate that was used for transmission of this packet. Note that this information is not present in any standard IEEE 802.11 header information; rather, it is supplied in the Radiotap header information or in the Linux Prism AVS header contents. As such, this feature will not work with packet captures that do not supply this additional information, including packet captures using only the standard IEEE 802.11 link type.

To add these columns to your Packet List window, click **Edit | Preferences** and then select **Columns** under the "User Interface" menu selection. Click **New** and type **RSSI** in the "Title" text box, then click on the Format drop-down list and select the IEEE 802.11 RSSI item. Repeat this step to add the data rate column using the title "Rate," and select the IEEE 802.11 TX rate item from the Format drop-down list (see Figure 6.30).

Next, click **Save | OK**. Unlike other Wireshark preferences, adding a new column requires you to restart Wireshark in order for the change to take effect. Close Wireshark and your capture file by clicking **File | Quit**, and then restart Wireshark and open a wireless capture file. If the RSSI and TX Rate information is present in your capture file, Wireshark will populate these new columns with the appropriate information (see Figure 6.31).

Figure 6.30 Wireshark Column Editor

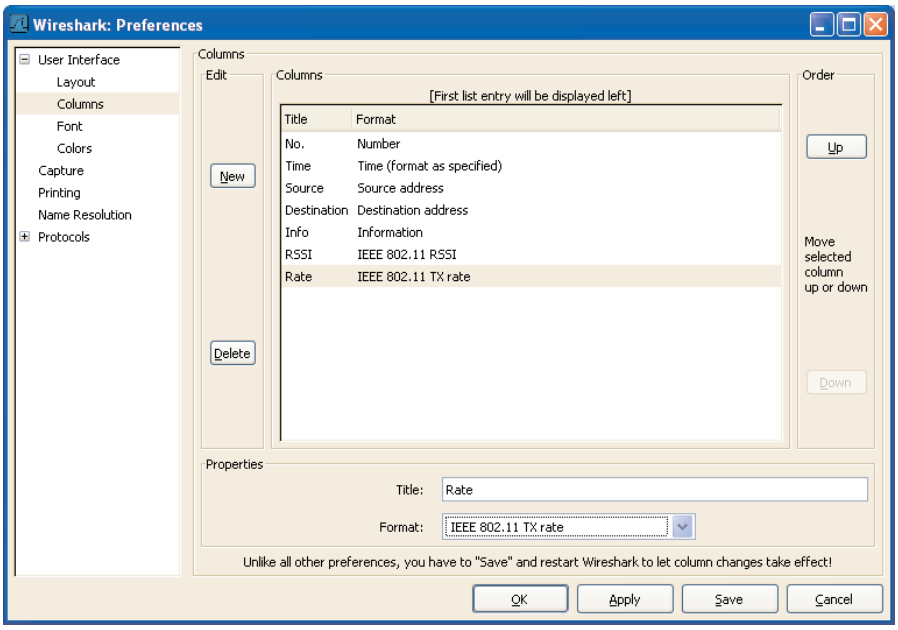
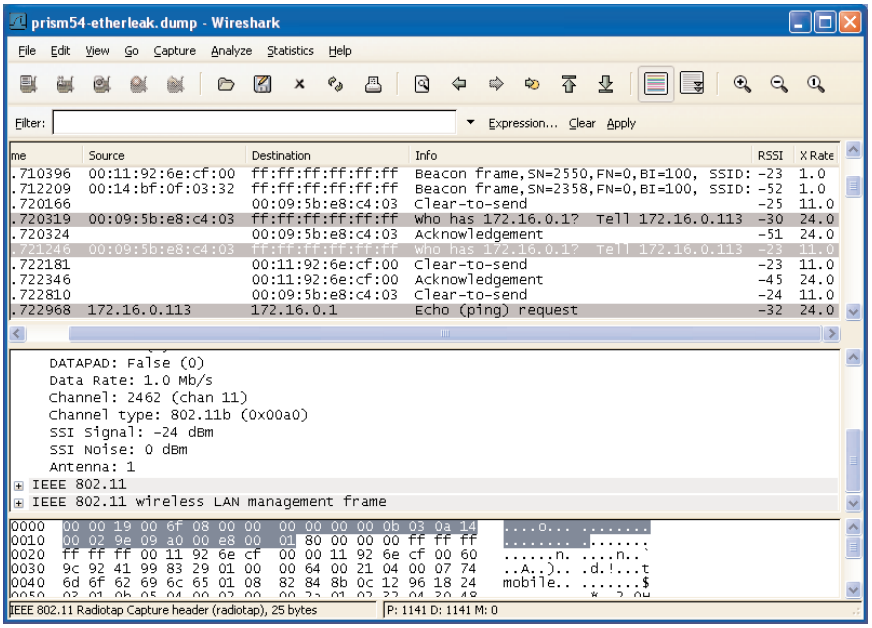


Figure 6.31 Displaying RSSI and Rate Columns



The ability to view these fields can be useful for troubleshooting and wireless intrusion detection purposes. As a station gets farther away from the AP, the TX rate for data frames drops in order to sustain connectivity to an AP. Observing a large number of stations transmitting below the optimal 11 Megabits per second (Mbps) for IEEE 802.11b networks or 54 Mbps for IEEE 802.11g or IEEE 802.11a networks, is an indicator of poor AP selection on behalf of the client (there may be a more optimal AP available), or poor deployment or configuration of APs. Inspecting the RSSI information allows you to identify drops in the signal strength for a client, which can be an indicator of interference or other Radio Frequency (RF) loss characteristics, which can also affect network performance.

Decrypting Traffic

One of the challenges of wireless traffic analysis is the ability to inspect the contents of encrypted data frames. While Wireshark has the ability to decode many different Network layer and higher protocols, encrypted traffic limits your ability to analyze packets and troubleshoot network problems.

Fortunately, Wireshark offers some options to analyze WEP-encrypted data. When configured with the appropriate WEP key, Wireshark can automatically decrypt WEP-encrypted data and dissect the plaintext contents of these frames. This allows you to use display filters, coloring rules, and all other Wireshark features on the decrypted frame contents.

In order for Wireshark to decrypt the contents of WEP-encrypted packets, it must be given the appropriate WEP key for the network. Wireshark does not assist you in breaking WEP keys or attacking the WEP protocol. If you are the legitimate administrator of the wireless network, you can configure Wireshark with the appropriate WEP key by clicking **Edit | Preferences**, and then expanding the "Protocols" menu and selecting IEEE 802.11. In the Wireshark Preferences window, supply one or more WEP keys in hexadecimal form separated by colons (see Figure 6.32). After entering one or more WEP keys, select the **Enable Decryption** checkbox. Click **OK** when finished.

Wireshark will automatically apply the WEP key to each WEP-encrypted packet in the capture. If the packet decrypts properly, Wireshark will add a tabbed view to the Packet Bytes window, allowing you to choose between the encrypted and decrypted views. Wireshark will also dissect the contents of the unencrypted frame, allowing you to view the embedded protocol information as if the frame were unencrypted in its original state.

Unfortunately, at the time of this writing, Wireshark does not support decrypting TKIP or CCMP packets. However, you can use external tools such as the

Figure 6.32 Specifying WEP Keys

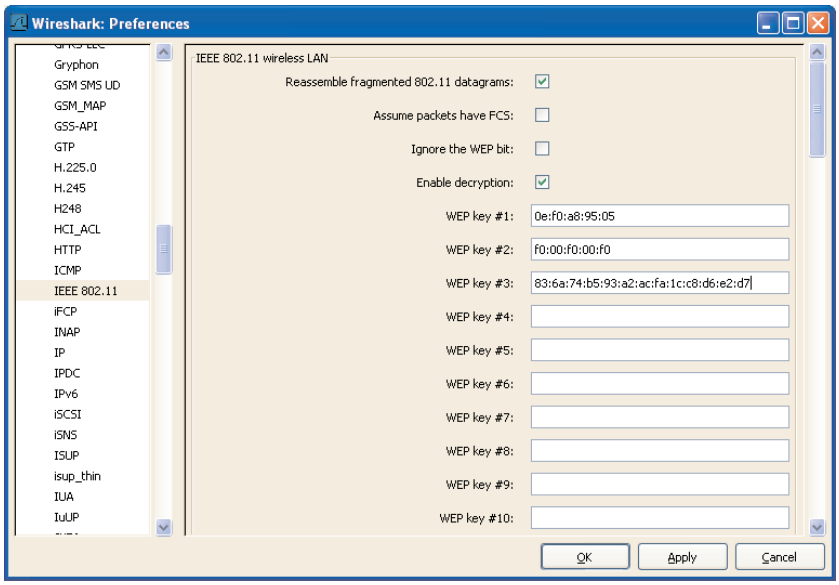
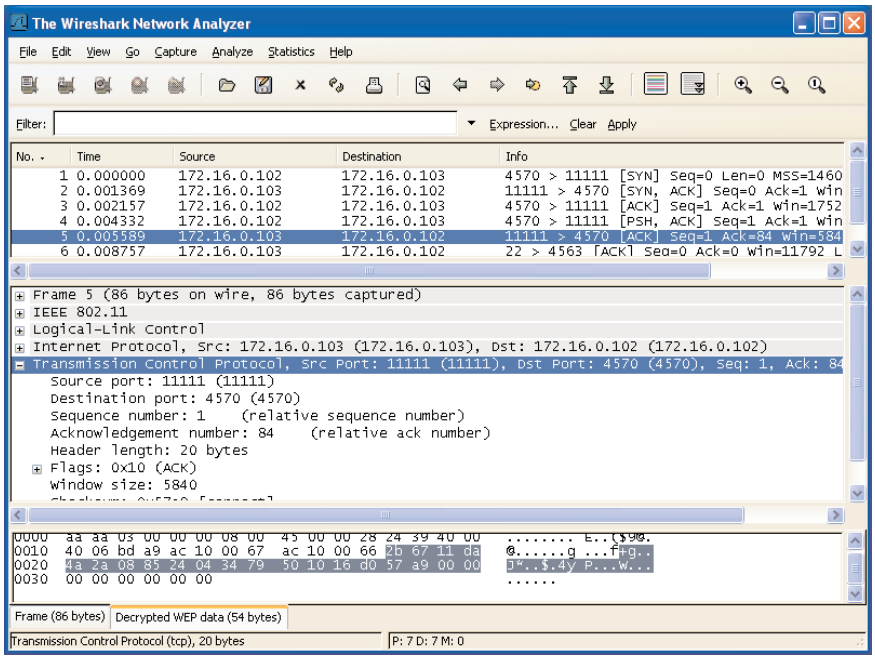


Figure 6.33 Viewing Encrypted and Unencrypted WEP Traffic

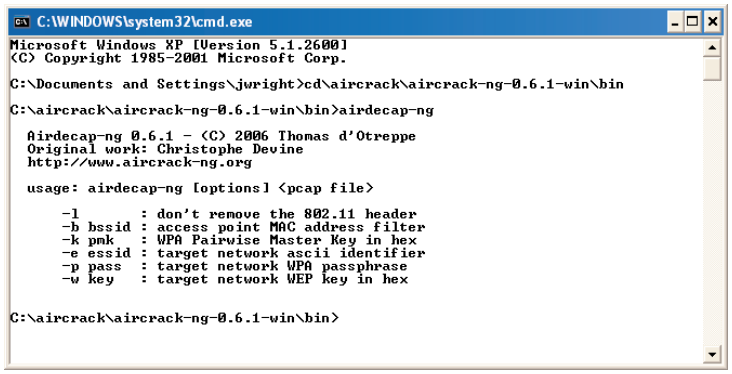


airdecap-ng utility (included in the open-source *Aircrack-ng* suite of tools) to rewrite a packet capture that uses the TKIP protocol. Similar to Wireshark's ability to decrypt WEP traffic, *airdecap-ng* requires you to have knowledge of either the PSK or the Pairwise Master Key (PMK) in order to decrypt TKIP traffic.

To install *airdecap-ng* on your system, you must download and complete the installation instructions for the *Aircrack-ng* tools. Download the latest version of *Aircrack-ng* from *www.aircrack-ng.org*. For Windows systems, download the *Aircrack-ng* zip file for Windows and extract it to a directory of your choosing. For Linux users, you must build the software using a C compiler, or obtain a precompiled binary from your Linux distribution vendor.

Once *Aircrack-ng* is installed, you can use the *airdecap-ng* tool to decrypt WEP or TKIP traffic, generating a new libpcap output file containing unencrypted traffic. There is no GUI interface for *airdecap-ng*, therefore, it is necessary to open a command shell and execute *airdecap-ng* from the command prompt (see Figure 6.34).

Figure 6.34 *Airdecap-ng* Command Parameters



You can decrypt WEP traffic by specifying the WEP key in hexadecimal format using the *-w* flag. We'll also supply the *-l* flag to retain the IEEE 802.11 header data. By default, *airdecap-ng* strips the IEEE 802.11 header, making the traffic appear to be a wired packet capture. *Airdecap-ng* will decrypt the traffic in the identified capture file, generating a new file with *-dec* appended after the filename and before the file extension (see Figure 6.35).

Similarly, you can decrypt a TKIP packet capture using the same technique, by specifying the TKIP PMK with the *-k* parameter or by specifying the PSK with the *-p* parameter. When decrypting TKIP traffic, you must also specify the network SSID (see Figure 6.36).

In Figure 6.36, *Airdecap-ng* creates the output file *wpapsk-dec.dump*, which contains the unencrypted data frames.

Figure 6.35 Decrypting WEP Traffic with *Airdecap-ng*

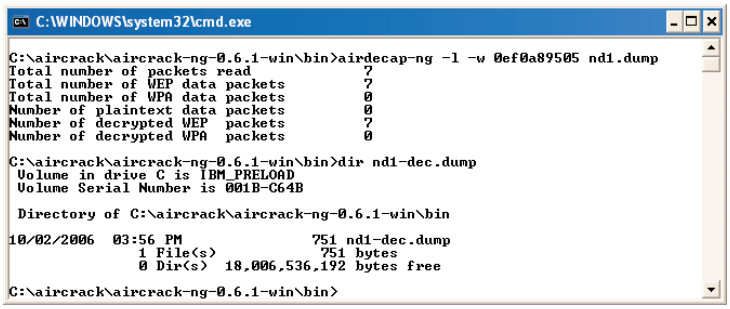
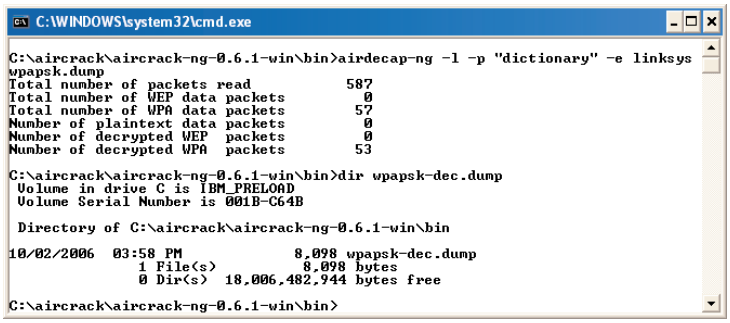


Figure 6.36 Decrypting TKIP Traffic with *Airdecap-ng*



Once you have decrypted the packet captures with *airdecap-ng*, you can open and inspect the unencrypted packet contents with Wireshark.

Real-world Wireless Traffic Captures

Now that you have learned how to leverage the wireless analysis features of Wireshark, you can examine real-world wireless traffic captures. Each of the captures reviewed in this section were selected to help reinforce the concepts learned in this chapter while demonstrating techniques that you can use to assess your own wireless network.

Identifying a Station's Channel

Introduction

Many administrators would agree that the wireless network configuration and management interface in Windows XP has improved steadily with each XP service pack. One of the remaining frustrations with the Windows Zero Configuration (WZC) interface for wireless networking, is the inability to report the current wireless channel

that the client is operating on. This is necessary information for troubleshooting connectivity problems or intermittent performance issues on the wireless LAN.

Identifying the channel number requires you to analyze information elements transmitted by the AP. While it is possible to estimate the channel number by switching through the channels manually with the *iwconfig* utility on Linux systems, wireless cards often receive frames from off-channels. In these cases, you might configure the wireless card on channel 1 and see traffic from the wireless station; however, the station could be operating on channel 3 instead.

In this packet real-world wireless traffic capture, we examine how to identify the current channel that a target wireless station is operating on.

Systems Affected

This traffic capture applies to all operating systems as a general analysis mechanism that is useful for network troubleshooting. The devices involved include the target wireless station and the AP. The wireless capture station will channel hop for this analysis, because it does not know which channel the target wireless station is using.

Breakdown and Analysis

In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-1.cap*. In this case, we need to identify the operating channel for the station with the MAC address *00:60:1d:1f:c5:18*.

After opening the capture file in Wireshark, apply a display filter to identify data traffic for the target station MAC address:

```
wlan.sa eq 00:60:1d:1f:c5:18 and wlan.fc.type eq 2
```

This excludes all traffic except that which originates from the target station. We apply this filter so we can examine the IEEE 802.11 protocol header information to determine the BSSID address. Click on any frame from this station and then navigate to the Packet Details window and click **IEEE 802.11 | BSS Id**. The BSSID reflects the unique identifier for this network, which we'll use to continue our analysis.

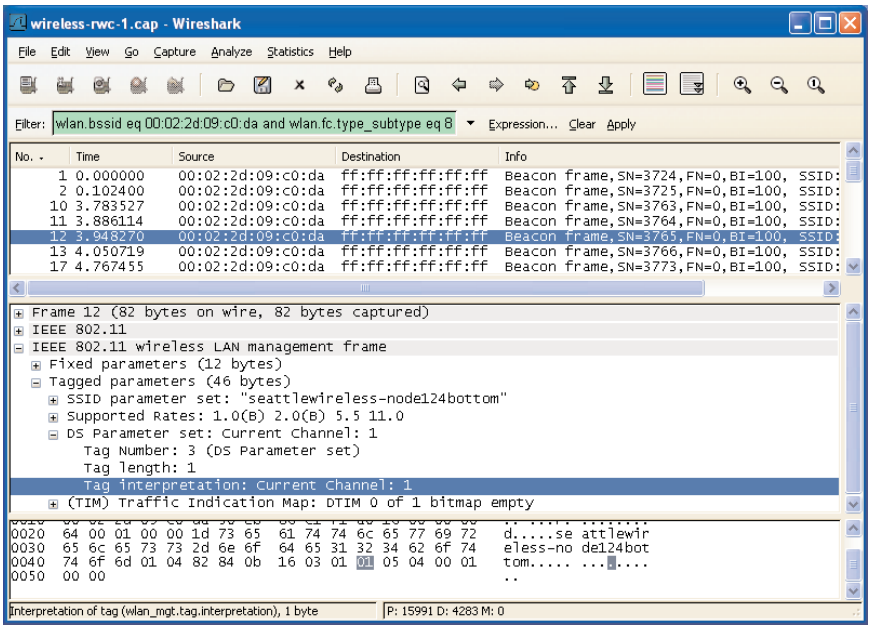
Once we know the BSSID for the network, we can clear the existing display filter and create a new filter to identify beacon frames from the AP servicing the identified station:

```
wlan.bssid eq 00:02:2d:09:c0:da and wlan.fc.type_subtype eq 8
```

Applying this filter will display beacon frames from the AP. On the summary line, we can see the source and destination address information and the summary information including the network SSID. By navigating to the Packet Details window and clicking **IEEE 802.11 Wireless LAN Management Frame | Tagged**

Parameters | DS Parameter Set | Tag Interpretation, we can examine the contents of this information element. This tag represents the current channel number for the AP (see Figure 6.37).

Figure 6.37 Tagged Parameters - Current Channel



By examining this capture, we can see that the beacon contents from the AP indicate that it is operating on channel 1. By association with the BSSID, we know that the station is also on channel 1.

Wireless Connection Failures

Introduction

Connection problems create common troubleshooting tasks for wireless LAN administrators. Often, the errors that are observed on the wireless network don't make their way to the client system in a mechanism that allows the end user (or administrator) to identify the problem. Fortunately, Wireshark can be used to help you gain more visibility into the problems "on the air" that prevent users from establishing connectivity to the wireless network.

Because this is such an important and recurring issue for administrators, we examine three different real-world packet captures to troubleshoot an authentication issue at the IEEE 802.11 layer, and two issues at the IEEE 802.1x/EAP layer.

Systems Affected

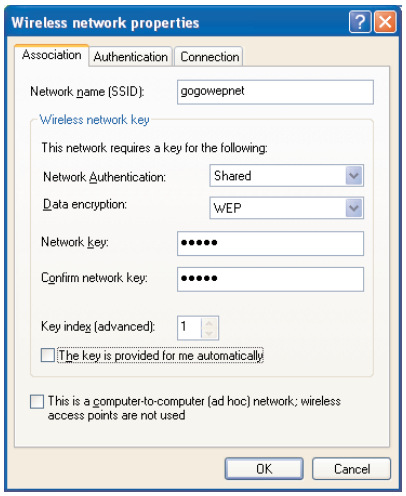
These traffic captures apply to all client and server operating systems that support wireless networking. One capture deals with an issue affecting a WEP-based network, and the other two captures deal with issues in LEAP networks. These principles also apply to other encryption mechanisms (e.g., TKIP, CCMP, and EAP).

Breakdown and Analysis

Capture 1

In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-2a.cap*. In this case, we are responding to a Windows XP Signal Processor 2 (SP2) station that is unable to connect to the wireless network using WEP encryption. After examining standard logging mechanisms (e.g., the Windows event log) on the XP workstation, there are no apparent error messages that indicate the source of the problem. A cursory glance of the client configuration appears correct, and the WEP key was re-keyed to verify that it is correct. The Wireless Network Properties Configuration window for this network is shown in Figure 6.38.

Figure 6.38 Station Wireless Network Configuration Properties



In an effort to troubleshoot this network, a wireless packet capture has been taken while the client was attempting to connect to the wireless network. Open the capture file *wireless-rwc-2a.cap* with Wireshark to begin analyzing the traffic.

In all wireless networks, the connection process starts with the station sending probe request frames to identify available APs in the area. The AP responds with a

probe response frame (unless configured otherwise), which informs the station that the AP is available. After identifying an available AP, the station continues the connection process to authenticate and associate to the AP.

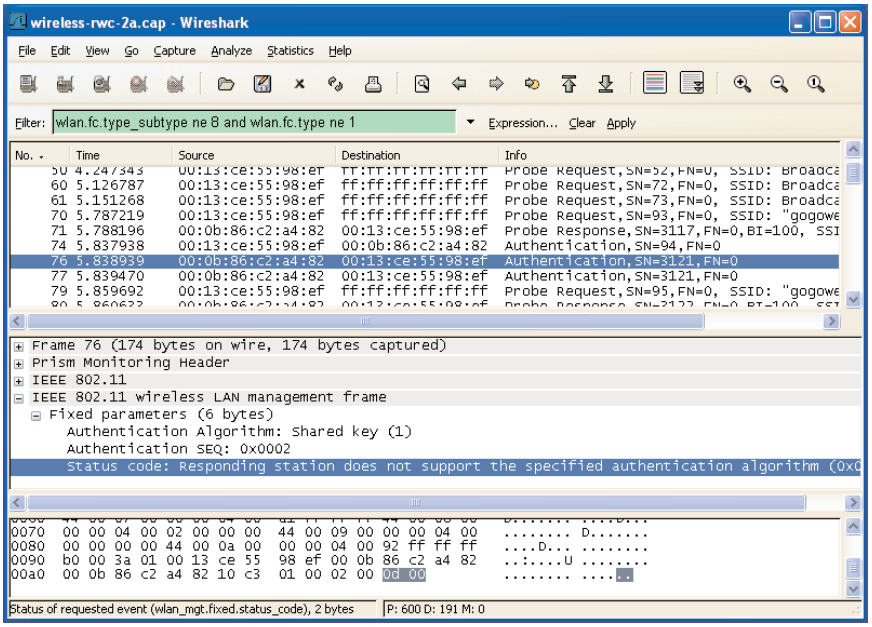
In WEP networks, the client sends an authenticate request to the AP, which elicits an authenticate response. In the case of shared-key network authentication, the AP sends a random challenge value that the client encrypted with their WEP key, and returns to the AP to verify before receiving an authenticate success or failure message. In the case of open network authentication, the AP skips the challenge/response step and issues a success message. Following authentication, the station associates to the AP by transmitting an association request packet. The AP responds with an association response message, after which the station can communicate on the wireless network.

As a logical troubleshooting step, it makes sense to verify each of these steps to identify where the connection process is failing. To reduce the number of frames displayed in the packet capture, apply the following display filter to exclude beacon frames and all control frames:

```
wlan.fc.type_subtype ne 8 and wlan.fc.type ne 1
```

The results of this display filter are shown in Figure 6.39.

Figure 6.39 Filtering Beacons and Control Frames - Real-world Capture 2a



In frame 34, we see the station sending probe requests for the SSID *<No current ssid>*; another probe request in frame 35 is targeting the broadcast SSID. These repeat without response until frame 70, which probes for the *gogowepnet* SSID, gets a response from the AP in the form of a probe response frame. This is appropriate behavior, because the station needs to know the BSSID and other capability information contained in the probe response frame before connecting to the AP.

Following the probe response in frames 74, 76, and 77, we see authentication traffic. We can tell that frame 77 is a retransmit of frame 76, because the SN (3121) listed in the information column is the same for both packets. We start our detailed analysis with frame 74; click on this frame and expand the management parameters by clicking **IEEE 802.11 Wireless LAN Management Frame | Fixed Parameters**. This will reveal the authentication algorithm as shared key, the authentication SN, and the status code. Because this is the first packet in the authentication exchange, the status code value is irrelevant.

Now that we know the authentication algorithm information that is being requested, look at frame 77. In the management parameters information we see the authentication algorithm is still shared key, but the status code has a message indicating "Responding station does not support the specified authentication algorithm." This error is preventing the client from connecting to the AP.

As a security feature, modern APs using WEP only support open authentication with WEP encryption, because shared key authentication introduces additional vulnerabilities to the network. Since this client is requesting shared-key authentication, the AP is rejecting the request with an error. To resolve this problem, reconfigure the client system to use open authentication with WEP instead of shared authentication.

Capture 2

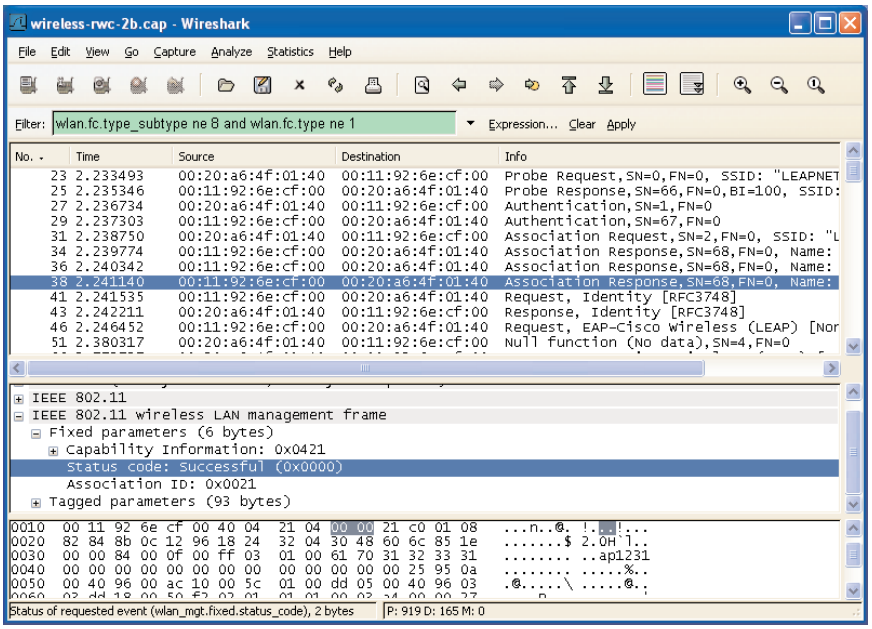
In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-2b.cap*. This is another example of a client that is unable to connect to the network. Open the capture file *wireless-rwc-2b.cap* and use the same display filter as in the previous capture to exclude beacon frames and control frames from the display:

```
wlan.fc.type_subtype ne 8 and wlan.fc.type ne 1
```

The result of this display filter is shown in Figure 6.40.

We can identify that the station is connecting to the AP successfully by looking at the Packet List window Information column. In frames 23 and 25, we see the probe request and response exchange, followed by two authentication frames. Clicking on frame 29 to select the second authentication frame, and clicking on **IEEE 802.11 Wireless LAN Management Frame | Fixed Parameters | Status Code** reveals that the authentication exchange was successful.

Figure 6.40 Filtering Beacons and Control Frames - Real-world Capture 2b



Following the authentication exchange, there is an association request in frame 31, and three authentication responses in frames 34, 36, and 38. Looking at the information column in the Packet List window for these three frames indicates that the SN is 68 for each packet, and that that they are retransmissions that were not properly acknowledged by the recipient. We can select frame 36 or 38 and click **IEEE 802.11 | Frame Control | Flags | Retry** to verify that these frames are retransmissions, by examining the value of the retransmit flag in the frame control header. This isn't unusual activity; however, it could indicate that some other interference source on the network is preventing the earlier frames from being received properly.

Examining the contents of the status code in the last association frame (frame number 38) by clicking **IEEE 802.11 Wireless LAN Management Frame | Fixed Parameters | Status Code**, indicates that the association was successful in completing the IEEE 802.11 authentication and association exchange.

At this point, we don't know exactly what we need to troubleshoot in this capture; therefore, it is helpful to use the coloring rules to assess the traffic capture. Apply a coloring rule to mark packets from the wired network with one color, and packets from the wireless network with a second color. This will allow you to easily assess the remainder of the frames to identify the traffic that is coming from the AP or from client systems.

Following the association response frame, we see that the beginning of the EAP authentication exchange in frame 41 is coming from the AP; this frame is requesting identity information from the station, which is returned in frame 43 with an identity response frame. In frame 46, we see a new EAP request frame indicating that the EAP type is the Cisco LEAP protocol. We can inspect the EAP details of this frame by clicking **802.1X Authentication | Extensible Authentication Protocol**. Inspecting the details of this frame, we see that the payload of the EAP packet includes an 8-byte random value and the name of the user authentication to the network. The 8-byte random value represents the challenge value that must be encrypted and returned by the authenticating station.

Frame 51 indicates a NULL data frame. This frame is not part of the EAP exchange. Rather, it is a mechanism that is used by the station to enter power-conservation mode while advertising to the AP that it should save any pending traffic for that station until it returns to the network. This can be confirmed by inspecting the power management bit in the frame control header, and by clicking **IEEE 802.11 | Frame Control | Flags | PWR MGT**. Since this value is set to *1*, we know that the station is entering power management mode. This is normal activity for some stations, especially Intel Centrino wireless cards, which are more aggressive at power conservation than other chipset manufacturers.

The station returns from power conservation mode in frame 66 with an EAP response frame. Again, we can inspect the contents of the EAP payload by clicking **802.1X Authentication | Extensible Authentication Protocol**. In the EAP payload contents, we see something labeled "Peer Challenge [8] Random Value"; this is an incorrect representation by Wireshark. Instead of being a peer challenge value, this is the actual peer response. Further, the peer response value is 24 bytes in length, not 8 bytes as indicated. This frame represents the response from the wireless station following the earlier challenge value.

Following the EAP response from the station, we would normally expect to see an EAP Success message from the AP. In this capture, we see an additional NULL data frame from the station indicating additional power management activity, followed by a multicast data frame for the Spanning Tree Protocol (STP) in frames 71 and 77, respectively. Instead of an EAP Success message, we see several deauthentication messages from the AP to the wireless client. This indicates that the LEAP authentication exchange was not successful, and that the AP is notifying the station that it has been disconnected from the network. The deauthentication frame is transmitted multiple times, because it is not properly acknowledge by the wireless station, possibly because the station is in power conservation mode.

In this packet capture, we see that the station has successfully completed the IEEE 802.11 authentication and association exchange, but was unable to complete

the IEEE 802.1X authentication exchange. This failure is repeated several times by the client and the AP in the capture file, starting at frames 376 and again at frame 724. The lack of an EAP Success message indicates that there was an authentication problem that caused the EAP exchange to fail (probably the result of an incorrect password entered by the user). While Wireshark cannot confirm this, we can use other sources of information to troubleshoot this issue, including logging messages on the AP and on the RADIUS server used to perform user authentication.

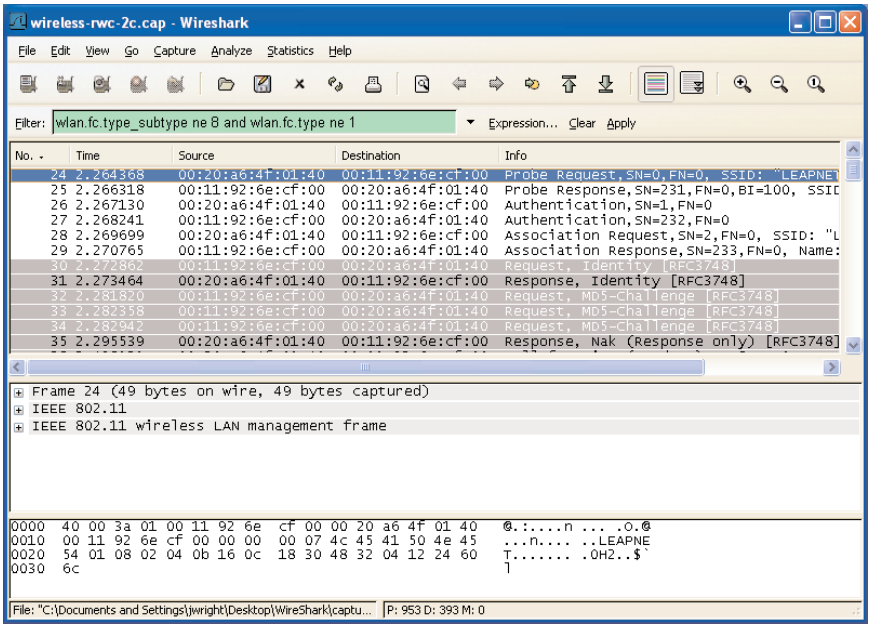
Capture 3

In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-2c.cap*. This is another example of a client that is unable to connect to the network. Open the capture file *wireless-rwc-2c.cap* and use the same display filter as used in the previous capture to exclude beacon frames and control frames from the display:

```
wlan.fc.type_subtype ne 8 and wlan.fc.type ne 1
```

Also apply the coloring rules to identify traffic from the AP or from a station with different colors. The result of this display filter and coloring rule is shown in Figure 6.41.

Figure 6.41 Filtering Beacons and Control Frames - Real-world Capture 2c

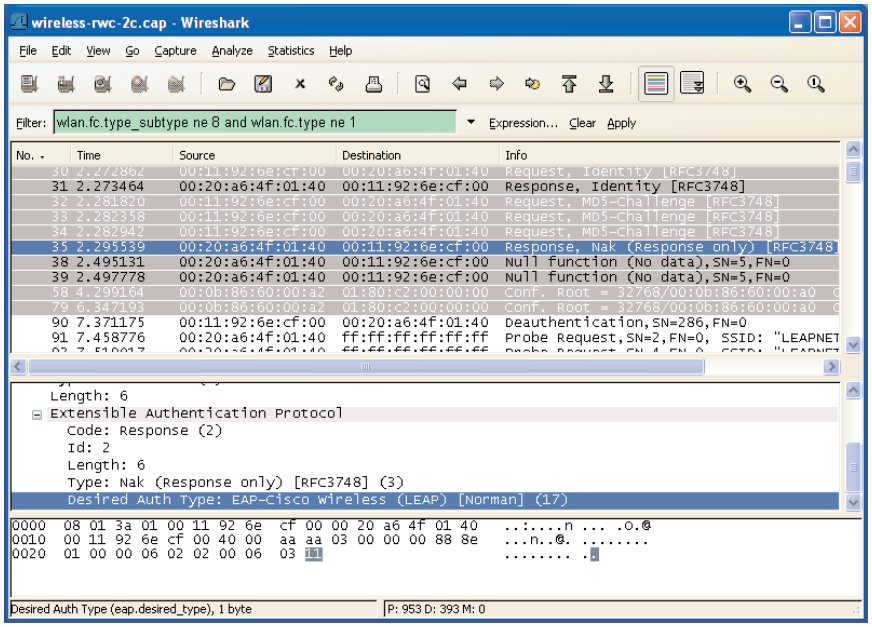


Like the previous packet capture, we determine that the station at *00:20:a6:4f:01:40* is able to complete the IEEE 802.11 authentication and association process by examining the contents of the information column in the Packet List window for frames 24 through 29. Following the association response frame, we see the beginning of the EAP exchange in frame 30 with an EAP Identity Request, followed by the EAP Identity Response.

In frames 32 through 34, we see an EAP request from the AP multiple times. This is another example of the station not immediately replying with an ACK frame, thereby causing the AP to retransmit the frame until a response is received. In the information column for these frames, we see the EAP type of Message Digest 5 (MD5) Challenge, also known as EAP-MD5. This indicates that the network is configured to use EAP-MD5 authentication on the RADIUS server, and that the AP is issuing an EAP-MD5 challenge for the station to encrypt as part of the authentication exchange.

In frame 35, we see a response from the station indicating an EAP negative ACK or Negative Acknowledgement (NAK) response. We can view the contents of the EAP payload for frame 35 by clicking **802.1X Authentication | Extensible Authentication Protocol**. We can see that the EAP type is a NAK message, which indicates that there is an error in the EAP exchange. Following the Type field, we see that the EAP payload indicates the desired authentication type of Cisco LEAP (see Figure 6.42).

Figure 6.42 Identifying the EAP Type - Real-world Capture 2c



In this packet capture, the station failed to connect to the wireless network, because it was configured to use LEAP authentication when the infrastructure network was configured to use EAP-MD5 authentication. Because there was no common EAP mechanism that was acceptable to both the client and the RADIUS server, authentication failed, which resulted in the AP issuing a deauthenticate message in frame 90. Visiting the client system and reconfiguring it to use the proper authentication mechanism would solve this problem, allowing the client to successfully authenticate to the network.

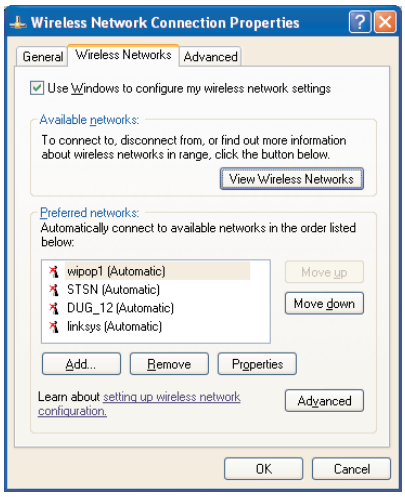
Wireless Network Probing

Introduction

Modern wireless client software is designed to make it easier for end users to maintain a list of preferred wireless networks. Users often connect to more than one wireless network (e.g., when in the office, a user may connect a corporate wireless network called "CORPNET"; when at home, they may connect to a home wireless network called "HOMENET"). When on the road, users may connect to hotel wireless networks such as STSN or hhonors, or public hotspot networks such as PANERA or T-Mobile.

In order to simplify the process of connecting to any of these networks, most wireless clients store a list of preferred networks in a Preferred Network List (PNL). On Windows XP systems, the PNL is available by right-clicking on the wireless adapter in the Network Connections window and selecting Properties (see Figure 6.43).

Figure 6.43 Windows XP PNL



In Figure 6.43, we can see that networks *linksys*, *DUG_12*, *STSN*, and *wipop1* are all preferred networks for this client system, allowing the station to easily connect to any of these available networks without interaction from the end user.

In order to identify if the networks in the PNL are available, wireless stations regularly transmit probe request frames with the SSID specified in the payload of the frame, and wait for responses from any available networks matching the SSID. This can be a potential information disclosure threat, because it allows an attacker to monitor and identify all of the network names configured in the station's PNL, by enumerating probe request frames.

In this real-world packet capture, we examine a mechanism to enumerate the networks configured in the PNL to evaluate potential information disclosure threats, or to identify stations that are connecting to wireless networks in an unauthorized manner, by identifying suspicious SSIDs.

Systems Affected

Both Windows and Mac OS X stations include support for PNLs, and regularly probe for all of these networks. Standard Linux systems do not include support for PNLs, although third-party applications may include support for this functionality with desktop environments such as K Desktop Environment (KDE) or GNU Network Object Model Environment (GNOME).

Breakdown and Analysis

In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-3.cap*. Open this packet capture file with Wireshark to examine the contents.

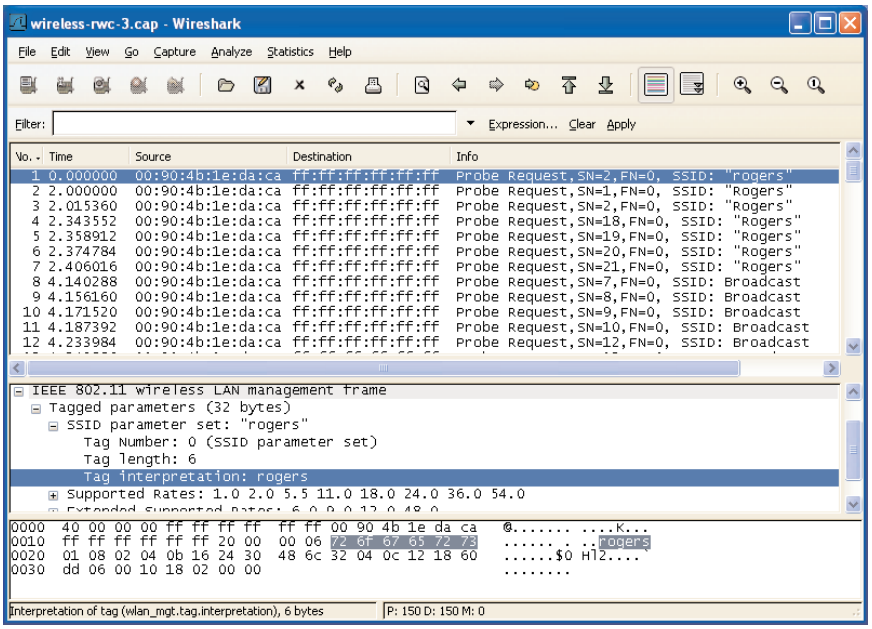
In order to identify network names from the PNL, we need to examine probe request frames coming from client systems. The packet capture file for this example has already been filtered to include only probe request frames, but we could use a display filter to identify only this frame type:

```
wlan.fc.type_subtype eq 4
```

The packet capture is displayed in Figure 6.44.

In frame 1, we see traffic from a station at *00:90:4b:1e:da:ca* sending a probe request frame to the broadcast destination address. In the information column in the Packet List window, we see that the desired SSID name for the probe request is *rogers*. In frame 2, we see another probe request, this time looking for the SSID *Rogers.* (SSID names are case sensitive.) The "Rogers" SSID is repeated until frame 8 where the SSID changes to the broadcast SSID. The broadcast SSID is indicated by

Figure 6.44 Examining Probe Request Content - Real-world Capture 3



the lack of an SSID or a 0-length SSID. After selecting frame 8, we can confirm this by examining the contents of the SSID field by clicking **IEEE 802.11 Wireless LAN Management Frame | Tagged Parameters | SSID Parameter Set | Tag Length**.

We can continue to examine the SSIDs specified in the packet capture file by scrolling through the entire packet capture. Unfortunately, Wireshark display filters do not include the ability to apply a "unique" filtering mechanism where only one of each unique SSID value is specified. We can effectively get the same results from using the text-based Wireshark tool using some common UNIX text-processing utilities.

From a shell prompt, examine the contents of the *wireless-rwc-3.cap* packet capture file with the tshark tool as shown:

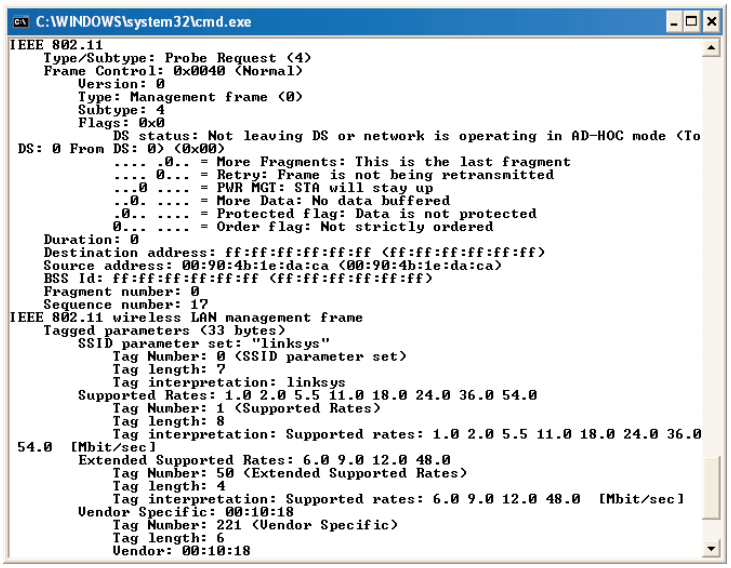
```
tshark -r wireless-rwc-3.cap -R "wlan.fc.type_subtype eq 4" -V
```

This syntax instructs tshark to read (*-r*) from the packet capture file using the display filter (*-R*) *wlan.fc.type_subtype eq 4* (display only probe request frames) with verbose decoding output (*-V*). Tshark processes and displays the contents of the packet capture file (see Figure 6.45).

TIP

UNIX operating systems are distributed with several text-processing tools that make parsing and extracting data from text-based output simple. You can download many of the most common and useful text-processing tools for Windows systems by visiting the GNU utilities for Win32 project Web site at http://unxutils.sourceforge.net. Download the *UnxUtils.zip* file and extract it to a directory in your local execution path such as *C:\WINDOWS*, or create a new directory such as *C:\BIN* for these tools and add this new directory to your system path. You can modify the system path by right-clicking on **My Computer** and then selecting **Properties | Advanced | Environment Variables**. In the System Variables section, scroll to the path variable, double-click on the value for this variable and append the new directory with a leading semi-colon to the end of the path list (e.g. *;C:\BIN*).

Figure 6.45 TShark Output - Real-world Capture 3



In this output, we see that the line beginning with *SSID parameter set* indicates the SSID in the probe request packet. The text processing tool *grep* can be used to filter the output from tshark to list only this line, and pass the output from *grep* into

the *sort* utility. It then passes the output from *sort* to the *uniq* tool to remove duplicates using the following command-line argument:

```
tshark -r wireless-rwc-3.cap -R "wlan.fc.type_subtype eq 4" -V | grep "SSID
parameter set:" | sort | uniq
```

By processing the output from tshark with the *grep*, *sort*, and *uniq* tools, we can get a unique list of the SSIDs identified from probe request frames:

```
C:\wireshark>tshark -r wireless-rwc-3.cap -nV | grep "SSID parameter set:" |
sort | uniq
    SSID parameter set: "hhonors"
    SSID parameter set: "linksys"
    SSID parameter set: "matrix"
    SSID parameter set: "rogers"
    SSID parameter set: "Rogers"
    SSID parameter set: "turbonet"
    SSID parameter set: "wldurel"
    SSID parameter set: Broadcast

C:\wireshark>
```

Using this technique, you can enumerate all of the SSIDs being probed by clients for the specified capture file. If you are interested in the PNL for a specific client, modify the display filter specified with the *-R* command-line argument to specify the target client MAC address (e.g.,, if the client MAC address you want to assess is *00:90:4b:1e:da:ca*, modify the display filter used in the previous example:

```
wlan.fc.type_subtype eq 4 and wlan.sa eq 00:90:4b:1e:da:ca
```

This analysis can be useful for identifying misconfigured client systems that have deprecated wireless networks still listed in the PNL, or to identify stations that have possibly violated organizational policy by connecting to unauthorized networks.

EAP Authentication Account Sharing

Introduction

Password-based EAP types are the most popular IEEE 802.1x authentication mechanism for wireless networks. Many of these EAP types, including PEAPv0, LEAP, and EAP-MD5, can disclose username information in plaintext as part of the

authentication exchange. This can be potentially advantageous to an attacker, but is also advantageous to an administrator to assess the identities of users on the wireless network.

Systems Affected

This analysis applies to wireless networks using IEEE 802.1x authentication for wireless networks, with an EAP type that discloses username information in plaintext as part of the authentication exchange. Examples of EAP types that disclose this information include EAP-MD5, LEAP, and PEAPv0.

Breakdown and Analysis

In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-4.cap*. Open this packet capture file with Wireshark to examine the contents.

NOTE

It was necessary to sanitize the *wireless-rwc-4.cap* contents before being allowed to include it as a reference for this book. Please disregard the timestamp information for each frame, as it is not valid for this analysis. Other sources of information in the capture have also been modified that do not affect the outcome of the analysis.

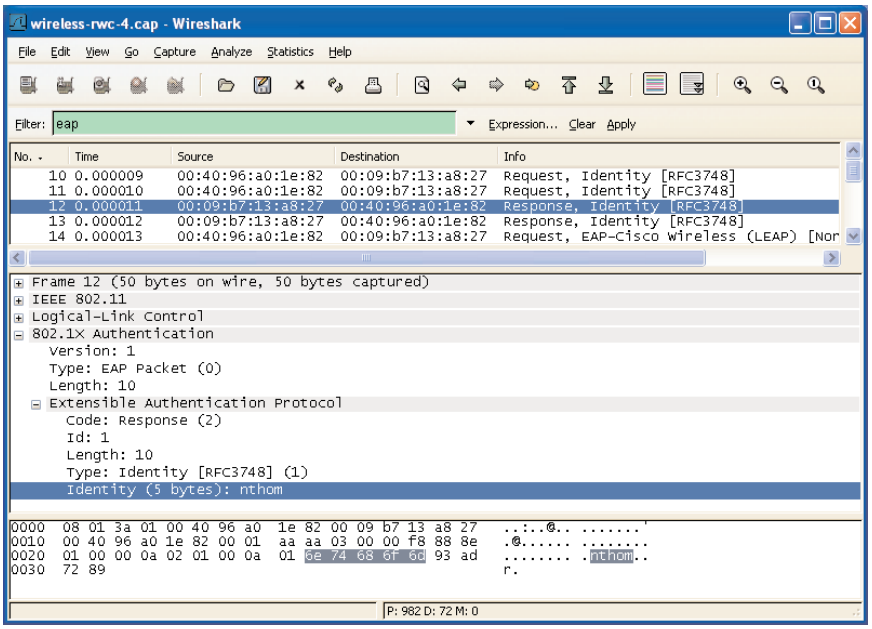
In order to examine username information disclosed in plaintext, we are primarily concerned with EAP traffic. Apply the following display filter to examine all EAP traffic in the capture file:

```
eap
```

The initial display after applying the filter for this packet capture is displayed in Figure 6.46.

Looking at the information column, we can see that this is a capture of Cisco LEAP (EAP-Cisco) traffic. While the first two frames in the display filter results don't disclose username information, selecting frame 12 (the third frame of the display filter results) displays the string *nthom* in the Packet Bytes window. Clicking **802.1X Authentication | Extensible Authentication Protocol | Identity** confirms that this is the username of the person at this workstation who is authenticating to the wireless network.

Figure 6.46 Displaying EAP Traffic - Real-world Capture 4



Examining the contents of the EAP header, we can reduce the number of packets returned in our display filter to include only EAP traffic of type identity and code response by applying the following display filter:

```
eap.code eq 2 and eap.type eq 1
```

The results of this updated filter allows us to focus on the usernames reported for each packet. Scrolling through each packet, we see the username *nthom* in frames 12 and 13 for the station at *00:09:b7:13:a8:27*, and the username *plynn* in frames 35 and 36 for the same station. This could indicate multiple users sharing a single workstation, or it could indicate a single user attempting to authenticate with multiple different usernames. Frequent occurrences of this type of activity or attempts for multiple usernames should be investigated for a potential security breach.

Continuing to examine the results of the display filter, we see the username *hbonn* is used from the station at *00:0a:8a:47:db:7b* in frames 77, 78, 84, 85, 101, 102, 210, and 211. Even more interesting is the reoccurrence of the username *nthom* in frame 210 from the station at *00:40:96:42:db:08*. This indicates that a single username (*nthom*) is being used from multiple stations, which is the result of multiple users sharing the same username and password. If your organization has a policy against this kind of activity, you could use this analysis to identify the offending stations and users.

IEEE 802.11 DoS Attacks

Introduction

IEEE 802.11 networks are vulnerable to a wide range of DoS attacks, allowing an attacker to indefinitely prevent one or more users from being able to access the medium for an indefinite amount of time. When under a DoS attack, the victim only knows that they are unable to access the wireless network and unable to identify that their loss of connectivity is the result of a malicious action or a network anomaly. Using Wireshark to analyze a traffic capture, the administrator can determine if the DoS condition is the result of malicious or non-malicious activity.

Systems Affected

This analysis applies to any wireless network that is potentially susceptible to DoS attacks, including all wireless networks where a potential adversary can be near the physical proximity of the network.

Breakdown and Analysis

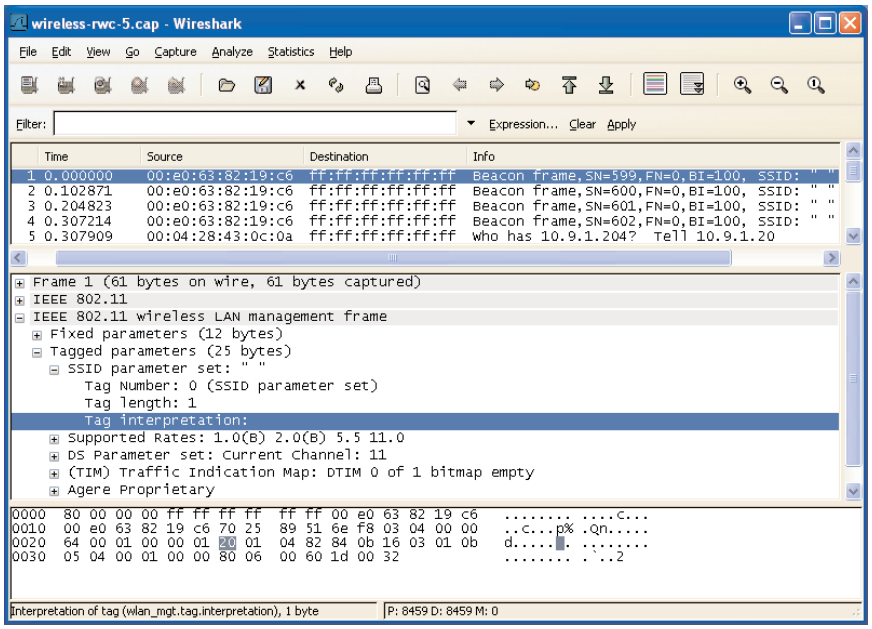
In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-5.cap*. Open this packet capture file using Wireshark to examine the contents.

After opening the packet capture, we see that the first frame is a beacon frame from the AP. By examining the packet list row for this frame, we determine that the source MAC address of the AP is *00:e0:63:82:19:c6*, and that the AP is attempting to hide or cloak the network SSID by replacing the legitimate SSID with a single space character (*0x20*). The information column for the Packet List window also gives us other information, including the packet SN, FN, and beacon interval (BI. Of particular interest is the SN information (see Figure 6.47).

All management and data frames on an IEEE 802.11 network are transmitted with a SN in the 802.11 header contents. The SN is a 12-bit field used for fragmentation. If a transmitting station needs to fragment a large packet into multiple smaller packets, the receiving station knows which fragments belong together by the SN value. When an AP boots, it will start using the SN *0*, incremented by *1* for each packet transmitted. Once the SN reaches 4,095, the sequence returns to *0* and repeats.

When examining a packet capture, each packet from a single source should have a SN that is a positively incrementing integer value, modulo 4,095. In some cases, there may be gaps in the SN (usually when a transmitter goes off-channel to scan for other networks, and your capture card misses those frames while remaining on a single channel), but the value should always be incremented by a positive value until

Figure 6.47 Information Column Content Analysis - Real-world Capture 5



it wraps. We can observe this behavior in the first several frames of this packet capture, where three beacons are transmitted sequentially, each with a new SN that is incremented by one (599, 600, 601).

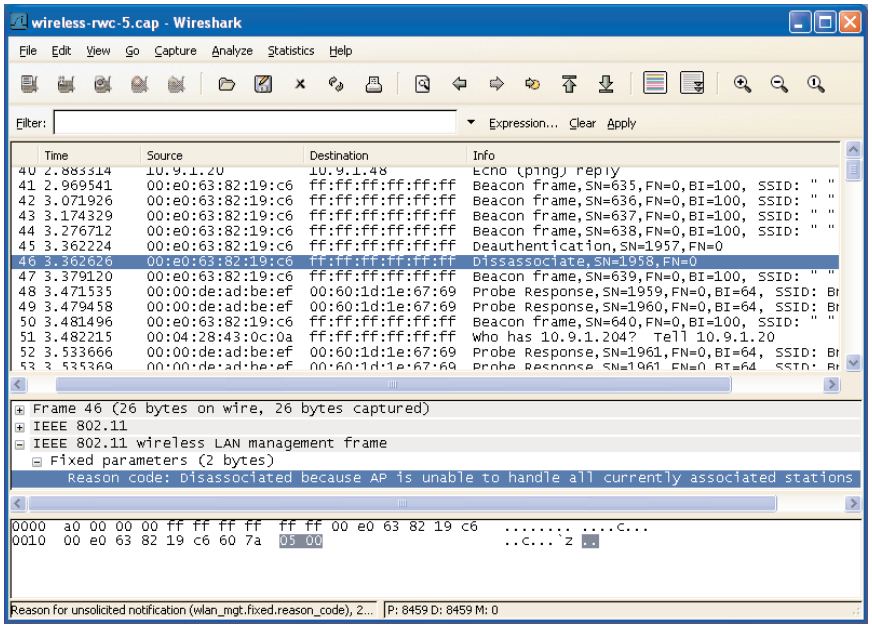
TIP

The concept of monitoring the activity of SNs for a transmitter is an important characteristic of wireless Intrusion Detection Software (IDS), and is used for a variety of analysis mechanisms.

Continuing to scroll through the packets listed in the Packet List window, we see regular beacon frame activity from the AP, as well as unencrypted data frames from multiple stations, including a regular ICMP Echo Request and Response pair between the stations at *10.9.1.48* and *10.9.1.20*, respectively. At frames 45 and 46, however, we see deauthentication and disassociate frames transmitted by the AP to the broadcast address. These frames are a legitimate part of the IEEE 802.11 specification, and are used by the AP to inform stations that they have been disconnected from the network. Both deauthentication and disassociate frames include a parameter

in the payload of the management frame known as the *reason code*, which identifies why the station was disconnected from the network. Selecting frame 45 and clicking **IEEE 802.11 | IEEE 802.11 Wireless LAN Management Frame | Fixed Parameter** reveals the reason code for the deauthenticate frame as *0x0002*, which indicates that the previous authentication is no longer valid. Selecting frame 46 and navigating to the reason code indicates that the station was disassociated, because the AP is unable to handle all currently associated stations (*0x0005* (see Figure 6.48).

Figure 6.48 Deauthentication and Disassociate Reason Code Analysis - Real-world Capture 5



By carefully inspecting the Packet List window, we can spot several unusual conditions with this packet capture:

- A beacon in frame 47 follows the deauthentication and disassociate frames with an anomalous SN. Starting with the two beacons prior to the deauthentication frame, the SN pattern is 637 (beacon), 638 (beacon), 1957 (deauthentication), 1958 (disassociate), and 639 (beacon). The deauthentication and disassociate frames were transmitted with the same source MAC address as the beacon frames, but do not follow the standard convention for selection of the SN.

- Following frame 47, we have two probe response frames with SNs that follow the previous deauthentication and disassociate frames, but conflict with the beacon frame. This is unusual from a SN perspective, because we are observing probe responses without a prior probe request frame. However, it is possible that our sniffer dropped the probe request frame, which must be taken into consideration.
- The source MAC address of the probe response frames is *00:00:de:ad:be:ef*. While this is a valid MAC address, it is not the original MAC address assigned to the station that transmitted this packet.
- The SSID contents of the probe response frame are sent to the broadcast SSID (a *0*-length SSID indicates the broadcast SSID). This is also unusual, because the AP must include an SSID in the probe response frame, even if it is the cloaked SSID. (In this network, the cloaked SSID is represented with a single space character.)

These factors all point to the notion that the deauthentication and disassociate frames were spoofed and not transmitted by the legitimate source, and that the probe response frames were sent in an effort to otherwise manipulate the network to the attacker's goals.

Examining the contents of the packet capture further, we see more similar deauthentication and disassociate activity with anomalous SN values, as well as additional unsolicited probe responses with the unusual source MAC address. With this analysis, we can determine that any DoS conditions users are experiencing are the effect of an attack against the network, not from the result of misconfigured clients or other legitimate network anomalies.

NOTE

The technique used in this packet capture is known as the *NULL SSID DoS* attack, where an attacker is attempting to get client stations to process malformed probe response frames in an effort to manipulate firmware on legacy wireless LAN cards. This is particularly effective as a DoS attack, because it renders the victim wireless card inoperable until the card has been power-cycled. More information on this bug is available in the Wireless Vulnerabilities and Exploits database as WVE-2006-0064 (www.wve.org/entries/show/WVE-2006-0064).

IEEE 802.11 Spoofing Attacks

Introduction

A significant security weakness in IEEE 802.11 networks is the lack of a cryptographically secure integrity check mechanism for traffic on the wireless network. While more modern encryption protocols such as TKIP or CCMP provide a secure integrity check over the payload of a data frame, it does not prevent at least partial analysis of a frame transmitted by an attacker with a spoofed source MAC address. This weakness exposes a wireless network to several classes of attacks, including packet spoofing attacks where an attacker impersonates a legitimate station on the network.

Systems Affected

This analysis focuses on a vulnerability in a WEP network, but the principles of analysis for identifying spoofed traffic, apply to all IEEE 802.11 wireless networks.

Breakdown and Analysis

In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-6.cap*. Open this packet capture file with Wireshark to examine the contents.

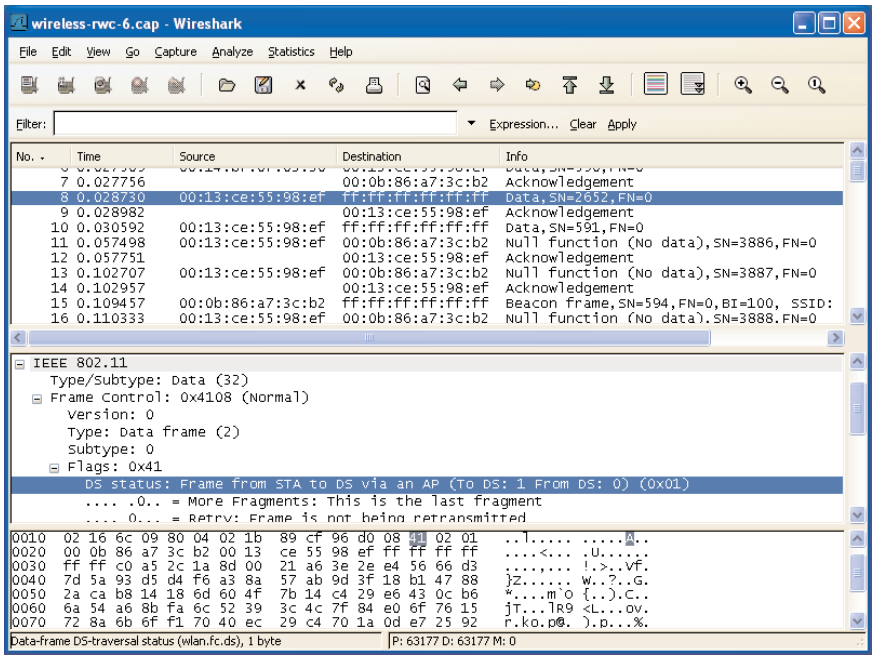
Upon opening the packet capture, we see traffic from multiple stations and beacon frames for the *WEPNET* SSID. After examining the first few frames, you may notice an anomalous SN combination for the station at *00:13:ce:55:98:ef* in frames 4 (SN=2651), 8 (SN=2652), and 10 (SN=591). However, when assessing the contents of frames, it is important to examine the status of the From DS and To DS flags to determine if the frame is coming from a station on the wireless network, or if it is being retransmitted by an AP to other stations on the network. Select frame 8 and click **IEEE 802.11 | Frame Control | Flags** to examine the DS status information (see Figure 6.49).

Examining the *DS status* line, we see that the frame is being transmitted to the distribution system (To DS). This indicates that frame 8 is being transmitted by a wireless station to the AP. Selecting frame 10 and navigating to the DS status line indicates that the frame is being transmitted from the distribution system (From DS) by the AP. This is not indicative of a spoofing attack; rather, the AP is retransmitting the frame from the station to be received by other stations on the AP.

In order to inspect the SN patterns for signs of possible spoofing activity, we can apply a display filter to examine only traffic sent to the DS from wireless stations:

```
wlan.fc.tods eq 1 and wlan.fc.fromds eq 0
```

Figure 6.49 SN Analysis - Real-world Capture 6



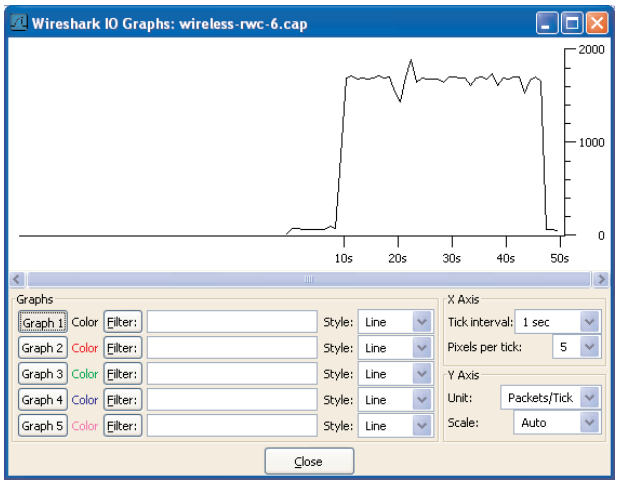
After applying this display filter, we can focus our attention on traffic from wireless stations. In the first packet, we see a NULL function frame from the wireless station with SN 3885. (Recall that this is often used for power management on wireless clients, to indicate to the AP that the client is entering a power-conservation state.) In frame number 4, we have a data frame with SN 2651 sent to a Unicast address, followed by a frame with that next SN in the series sent to the broadcast address. Next, we have another NULL function frame, returning to the original SN series.

Continuing to look at the packet capture contents, we see additional data frames sent to the broadcast address with SNs that are not part of the series used by the NULL function frames. By selecting one of these anomalous frames (e.g., frame number 11) and clicking **IEEE 802.11 | WEP Parameters**, we determine that this is a WEP-encrypted network. At this point, we have determined that there is a station that is transmitting spoofed WEP-encrypted packets sent to the broadcast address; our next task is to evaluate what the potential impact is to the network.

Wireshark can produce a simple but effective Input/Output (IO) graph to illustrate the behavior of traffic on the network. Click **Statistics | IO Graphs** to open the IO Graphs window (see Figure 6.50).

In Figure 6.50, Wireshark illustrates the characteristics of the packet capture based on our analysis preferences. By default, Wireshark plots the time on the *X* axis

Figure 6.50 IO Graph Analysis - Real-world Capture 6



and the number of packets on the *Y* axis. This allows us to determine that there was little activity on the wireless network for the first 10 seconds of the packet capture, which increased to a rate of approximately 1,000 packets per second for approximately 38 seconds before the activity returned to a minimal level.

We can refocus the graph by modifying the *X* axis and *Y* axis values to give the best view of the network activity. Change the Tick interval on the X axis to *0.1* seconds and change the pixels per tick to *2*. This will extend the width of the graph, forcing us to scroll to see the activity of the entire capture.

NOTE

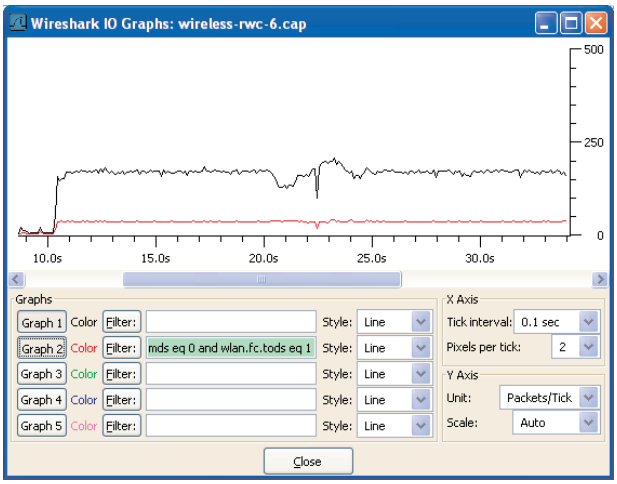
To obtain a better view of the graph content, you can expand the size of the IO Graphs window to any resolution supported by your video card.

By default, the IO graph illustration shows the analysis for all traffic in the capture file. We can add additional graphs to this view based on any criteria we specify with Wireshark display filters. For this example, it is useful to identify exactly how much traffic is originating from wireless stations (To the DS), and how much traffic is originating from the AP (From the DS). Click on the Graph 2 line in the Filter text box and enter the following display filter to identify all traffic from wireless stations to the DS:

```
wlan.fc.fromds eq 0 and wlan.fc.tods eq 1
```

Next, click the **Graph 2** button to update the IO illustration (see Figure 6.51).

Figure 6.51 IO Graph/Wireless Station Traffic - Real-world Capture 6



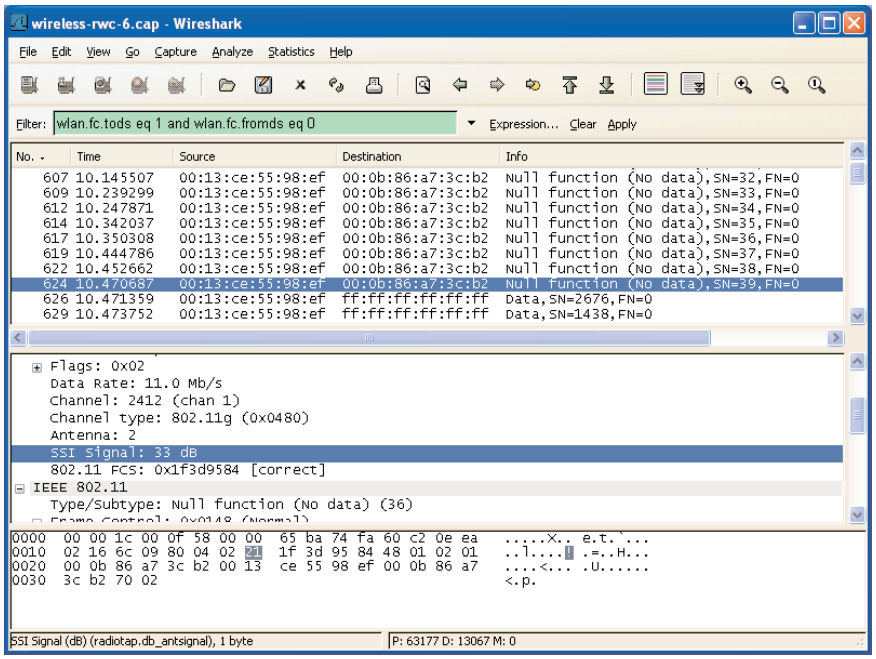
The new graph filter line illustrates the quantity and frequency of packets being transmitted from wireless stations to the DS. Approximately 30 percent of the traffic is from wireless stations; the remaining traffic is made up of traffic from the AP to the stations or management or control frames that do not set the From DS or To DS bits in the 802.11 frame control header.

In order to focus on the spoofed packets, we want to identify a pattern in the packets and apply a display filter to display only those frames. We have determined that the significant increase in activity on the network started at approximately 10 seconds into the packet capture, therefore, we can switch back to the Packet List window and scroll to this point in the capture to examine the traffic activity (see Figure 6.52).

Fortunately, this packet capture was taken with the Radiotap Link layer header information, which allows us to identify additional information about the characteristics of the traffic beyond the 802.11 header contents (e.g., the signal strength indicator is recorded with each packet that is captured, as well as the channel type and data rate information. Selecting packet 624 and clicking **Radiotap Header | SSI Signal** reveals the signal strength as 33 decibels (dB) for this NULL function packet, which is believed to be from the legitimate station. Repeating the process for frame 629 reveals the signal strength as 60 dB for the data frame that is believed to be spoofed. Sampling additional packets reveals similar information; legitimate frames have a signal level between 31 dB and 46 dB, while illegitimate (spoofed) frames have a signal level between 54 dB and 67 dB. This characteristic can be described in a display filter to display only spoofed traffic:

```
wlan.fc.tods eq 1 and wlan.fc.fromds eq 0 and wlan.sa eq 00:13:ce:55:98:ef
and radiotap.db_antsignal > 50
```

Figure 6.52 Examining Increasing Traffic Activity - Real-world Capture 6



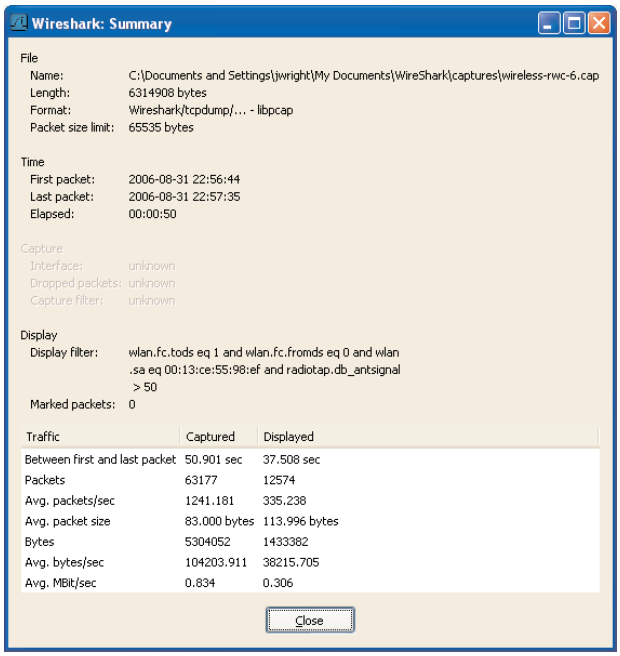
This new display filter returns 12,574 frames, all of which appear to be transmitted by an attacker through packet-spoofing techniques. While it may not be a comprehensive list of all of the spoofed frames in the packet capture (it's conceivable that some frames were transmitted with lower signal levels), it is sufficient to use for additional analysis.

With the display filter applied that only displays traffic suspected as being spoofed, we can use Wireshark's analysis and summarization features to glean additional information about this activity. Click **Statistics | Summary** to examine the summary information (see Figure 6.53).

The bottom of the Wireshark Summary window identifies several metrics regarding the traffic for all of the frames in the capture, and for packets returned with a display filter. In this case, our display filter is showing 37.5 seconds of traffic for a total of 12,574 frames at a rate of over 335 packets per second. This gives us an idea as to the rate of the attack, which appears to be aggressive based on the number of packets per second.

Returning to the IO Graphs window, we can add this new display filter and graph a third line to illustrate the traffic that is spoofed, compared to traffic sent from wireless stations. Enter the same display filter in the Filter text box for graph 3

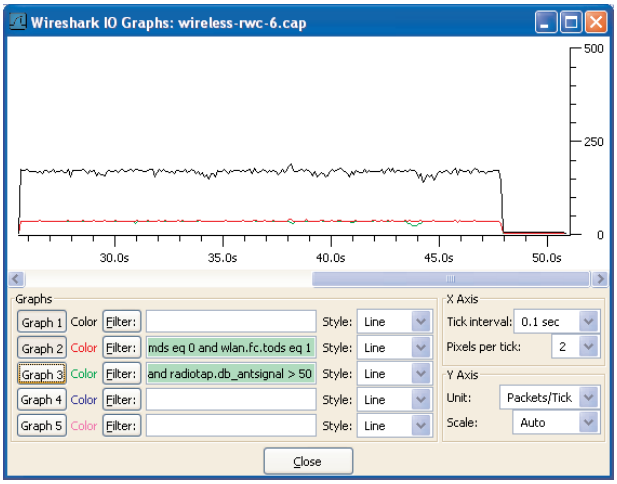
Figure 6.53 Frame Statistics Summary - Real-world Capture 6



and click the **Graph 3** button. Wireshark processes the new display filter and returns a new IO graph line (see Figure 6.54).

With this new graph line, we see that nearly all of the traffic sent from the wireless network is spoofed traffic from the attacker.

Figure 6.54 IO Graph/Spoofed Traffic Comparison - Real-world Capture 6



At this point in the analysis, we've determined several factors that are useful for our analysis:

- The wireless network was relatively quiet until 10 seconds into the packet capture
- An attacker started transmitting illegitimate WEP-encrypted frames into the wireless network, spoofing the source address of a legitimate station
- The attacker represents nearly 100 percent of the wireless frames sent to the DS
- The attacker is transmitting frames at approximately 335 packets per second
- In response to their spoofed traffic, the attacker's activity is generating a significant number of packets from the AP to the wireless network

With this information and some background knowledge in the weaknesses of WEP networks, we can determine that an attacker is manipulating the wireless network to accelerate the amount of traffic on the network. This is a common technique used for WEP cracking; an attacker may require several hundred-thousand packets on the wireless network to recover a WEP key. With a single station that is not regularly transmitting any encrypted traffic, it may take an attacker weeks to recover a sufficient quantity of traffic to recover the WEP key. By manipulating the network in this fashion, the attacker has increased the traffic level from a minimal number of frames to over 300 frames per second. At this rate, an attacker will have collected a sufficient number of packets (approximately 150,000 unique encrypted packets is a useful quantity for WEP cracking) in less than 10 minutes.

As the administrator of this network, you may have knowledge of the WEP key used to decrypt traffic. In this example, the WEP key for the network is *f0:00:f0:00:f0*. We can supply Wireshark with this encryption key and Wireshark will display both the encrypted and unencrypted content for each packet.

To enter the encryption key for this capture click **Edit | Preferences | Protocols | IEEE 802.11**. Type **f0:00:f0:00:f0** in an available WEP key slot, and check the **Enable decryption** checkbox (see Figure 6.55). Click **OK** when finished.

After Wireshark reloads the packet capture and decrypts each packet, any packets that are successfully decrypted with the specified WEP key include two tabs at the bottom of the Packet Bytes window labeled "Frame" and "Decrypted WEP data." With this new information, we can identify the activity that was generated by the attacker. Select frame 663 (one of the spoofed packets) and inspect the Packet Details window to identify the nature of the traffic (see Figure 6.56).

Figure 6.55 Supplying WEP Encryption Keys - Real-world Capture 6

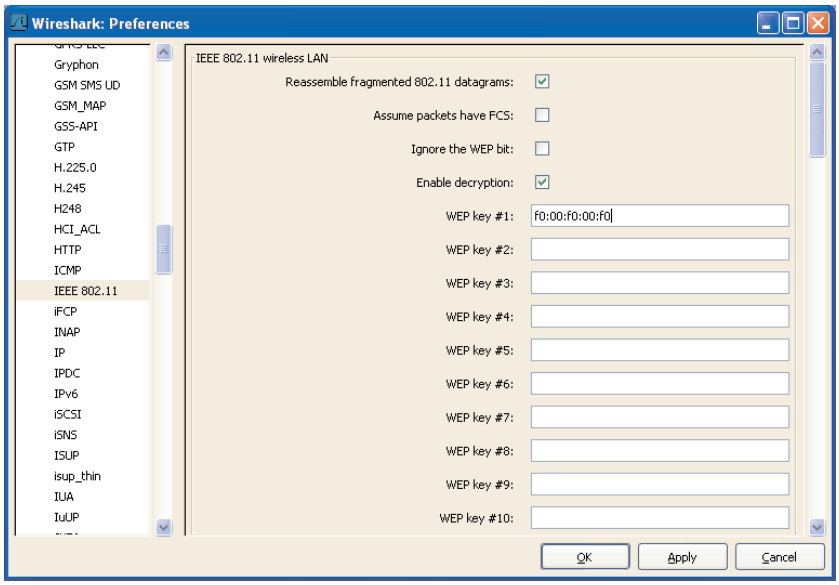
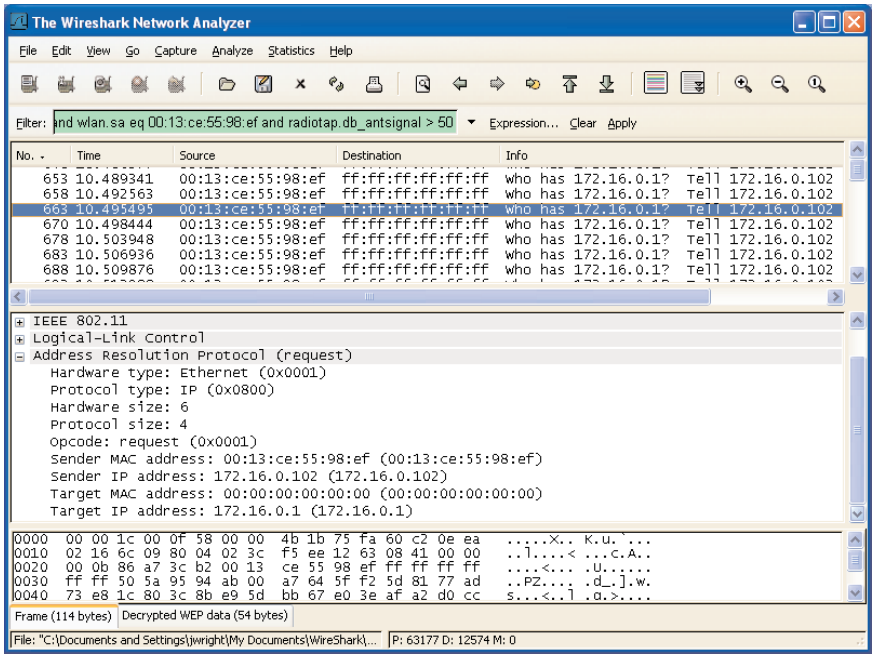


Figure 6.56 Examining Decrypted Frame Contents - Real-world Capture 6



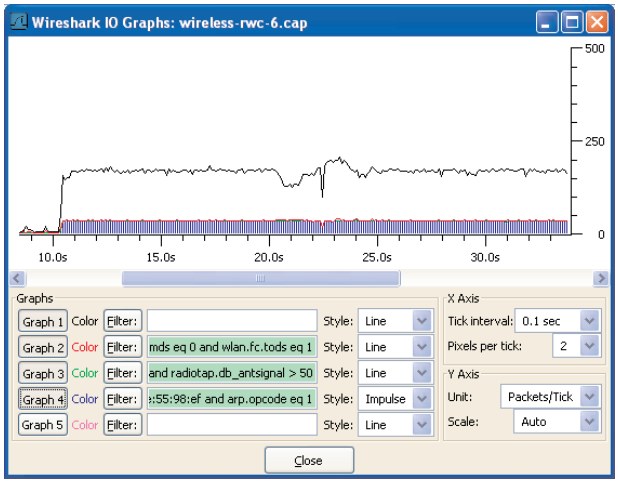
We can see that the traffic being transmitted by the attacker is a repetitive series of Address Resolution Protocol (ARP) Request packets that elicit ARP Response packets from a station on the network. This activity reinforces our analysis that the attacker is attempting to increase traffic levels on the network in order to collect enough packets to use a WEP cracking tool. We can return to the IO Graphs view and add another display filter to evaluate our earlier signal strength indicator display filter, verifying if our initial analysis of spoofed traffic matches the series of ARP request frames on the network.

Return to the IO Graphs window and add a third (and final) display filter to identify only ARP request packets in the graph 4 line:

```
wlan.fc.tods eq 1 and wlan.fc.fromds eq 0 and wlan.sa eq 00:13:ce:55:98:ef
and arp.opcode eq 1
```

Modify the line style for this graph from "Line" to "Impulse" to make the graph easier to see in context with the other graphs (see Figure 6.57).

Figure 6.57 IO Graph/ARP Request Traffic Comparison - Real-world Capture 6



We can see that the lines from graphs 3 and 4 match very closely, indicating that our analysis of the attacker's activity based on signal strength indicators and the decrypted protocol activity are both correct.

NOTE

This attack activity matches the mechanism implemented in the Aireplay tool that ships with the *Aircrack-ng* suite of WEP and WPA-PSK cracking tools. This attack tool is assigned the identifier WVE-2005-0015 by the Wireless Vulnerabilities and Exploits group; visit www.wve.org/entries/show/WVE-2005-0015 for additional information about this attack tool.

Malformed Traffic Analysis

Introduction

A recent development in the saga of wireless LAN security is the use of IEEE 802.11 protocol *fuzzing* against wireless stations to identify bugs in driver software. Fuzzing is a technique where an attacker sends malformed packets that violate the specification of a protocol to a client or a server. If the server or client software is not written to expect invalid packets, it can sometimes trigger flaws in software that can be exploited by an attacker.

Notes from the Underground…

IEEE 802.11 Protocol Fuzzing

A recent advancement in the list of techniques that can be used to compromise the security of a wireless network, is the use of IEEE 802.11 protocol fuzzing. Fuzzing is a technique used by security researchers and attackers to identify software weaknesses when presented with unexpected data. This technique is frequently used to identify potential security flaws in software that can be successfully exploited to the attacker's gain. Once an attacker identifies a sequence of data that causes a victim to behave in a way the target software's author did not intend, the technique is developed into an exploit that can be used repeatedly against vulnerable stations. In the case of IEEE 802.11 wireless LANs, fuzzing is being used to identify weaknesses in device driver software when presented with malformed or unexpected wireless frames. These frames can be malformed by violating the framing rules stated in the 802.11 specification or by violating the expected order and timing of otherwise legitimate packets.

Continued

The use of 802.11 protocol fuzzing is not necessarily a bad thing. If a researcher uses this technique to identify potential software flaws in station drivers, and uses ethical disclosure practices to communicate these flaws to the vendor, all wireless users benefit from improving the quality of otherwise buggy software. However, if the intention of the researcher is to turn them into exploits for their own gain (potentially by exploiting systems or by selling their exploits to others who will use them for ill gain), the risk to vulnerable organizations is significant.

In a wireless LAN, any attacker that gets within physical proximity of the victim network can inject packets that will be received by wireless stations, regardless of the encryption or authentication mechanisms used. If an attacker can identify a driver vulnerability in the processing of these packets, there is little that can be done to protect the vulnerable station. This is amplified because there is little security software designed to protect the integrity of client systems at the wireless LAN layer (OSI model layer 2). Most firewalls and other security tools (e.g., host-based intrusion prevention tools) start assessing traffic at layer 3 and higher, often leaving stations vulnerable and blind to any attacks at layer 2.

Fortunately, independent security researchers are actively looking for, identifying, and communicating these driver flaws to the appropriate vendors, in an effort to resolve them before they become actively exploited by attackers. Concerned organizations should take care to ensure driver software on client stations remains current, and that upper-layer analysis tools (such as intrusion detection systems) are used to identify questionable activity from compromised stations (including Internet Relay Chat [IRC] information, or signs of spyware infections and other unauthorized network use) to monitor for potentially compromised stations.

Systems Affected

This analysis applies to all wireless LANs where an attacker can get within physical range of the wireless network.

Breakdown and Analysis

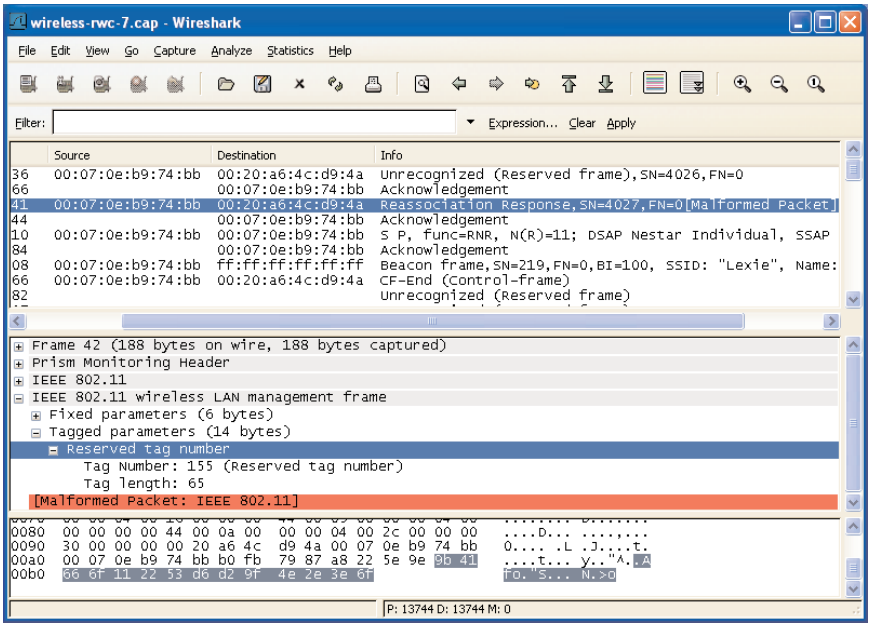
Capture 1

In this real-world traffic capture analysis, we reference the capture file *wireless-rwc-7.cap*. Open this packet capture file with Wireshark to examine the contents.

This packet capture includes the 4-byte FCS at the end of each frame; however, Wireshark has no way of knowing that the FCS is present. In order to successfully analyze the contents of this capture, instruct Wireshark to expect the FCS information by clicking **Edit | Preferences | Protocols | IEEE 802.11** and ensure "Assume packets have FCS" is selected. Click **OK**.

Upon opening the packet capture, we see traffic from networks with the SSIDs *Lexie* and *NETGEAR*, as well as some unencrypted data frames in the form of ICMP echo requests and responses. Scrolling through the packet capture, we notice the information column for frame 42 is labeled "Reassociation Response,SN=4027,FN=0[Malformed Packet]." This is Wireshark's mechanism to indicate that this packet does not comply with the packet framing rules of the IEEE 802.11 specification. As soon as Wireshark comes to the point in the packet where it evaluates the content as invalid, it will stop processing the remainder of the frame and insert the "Malformed Packet" label. We can navigate the Packet Details window to identify the content that caused the frame to be marked as invalid, by going to the end of the Packet Details window. In frame 42, click **IEEE 802.11 Wireless LAN Management Frame | Tagged Parameters | Reserved Tag Number**. Wireshark will attempt to decode this information, as shown in Figure 6.58:

Figure 6.58 Assessing Malformed Frames - Real-world Capture 7



In this case, we see that the management frame payload information element is not properly evaluated by Wireshark. Each of the tagged parameters in IEEE 802.11 management packets is consistently formatted as shown below:

Tag Type (1 byte)	Tag Length (1 byte)	Data (variable length, corresponding to tag length, between 0 and 255 bytes)

In this example, Wireshark identifies the tag number as *155* or *0×9b* with a length is 65 bytes (*0×41*). However, only 8 bytes remain at the end of the packet before the FCS, not 65 as indicated by the frame length. This prompts Wireshark to identify the packet as malformed.

NOTE

Wireshark identifies this packet as malformed, because the reported tag length exceeds the number of remaining bytes in the packet, and because Wireshark knows it has received 100 percent of the bytes in the packet, as indicated by the frame packet length and capture length information. However, Wireshark also identifies this tagged parameter as using a reserved tag number.

Throughout the development of Wireshark, the authors of various dissectors maintained the software to stay current with the supported protocols and the values used in these protocols. However, over time, standards bodies such as the IEEE and IETF may allocate previously reserved values for new uses of existing protocols. As such, Wireshark doesn't assume the packet is invalid simply because it does not know how to interpret a value that it observed, such as the tag number 155 in this example.

Observing a single malformed frame does not suggest that the capture is the product of protocol fuzzing techniques or other potentially hostile activity; it is possible that this frame was accidentally corrupted when it was transmitted, or that the station that is sending this data is flawed and is sending invalid frames. We can easily determine if the first condition caused the frame to be malformed by checking the contents of the FCS field. In frame 42, click **IEEE 802.11 | Frame Check Sequence**. We see that Wireshark reports the FCS as *0×4e2e3e6f*, which it reports as correct for this packet. While it is possible that the packet could be modified by retaining a valid FCS, it is highly unlikely. As such, we can assume the packet was received in this capture exactly as it was transmitted.

The second possibility of a flawed implementation remains, which would also suggest that we would see multiple packets that shared the characteristic of the reserved tag number with a length that is longer than the number of bytes available in the capture. We can use a display filter to identify other frames that are similar to frame 42, in an attempt to prove/disprove this theory. Right-click on **Reserved**

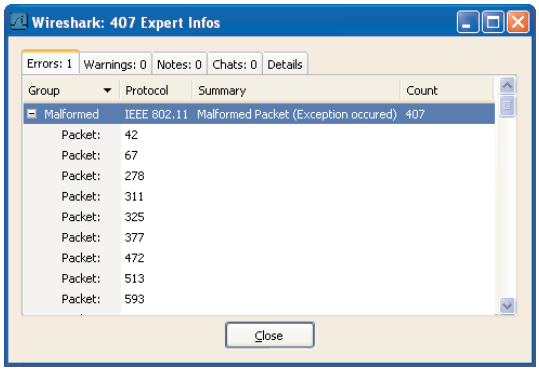
Tag Number and select **Prepare a Filter | Selected**. This will place a display filter in the Filter text-box:

```
frame[174:10] == 9b:41:66:6f:11:22:53:d6:d2:9f
```

This filter instructs Wireshark to start looking at the 174-byte offer in this packet for a 10-byte sequence matching *9b:41:66:6f:11:22:53:d6:d2:9f* (note that *0×9b* is the reserved tag or 155, *0×41* is the malformed length [65], and the remaining bytes represent the actual data following this tag. Clicking **Apply** will prompt Wireshark to process this display filter and display only frames that match this characteristic. When the display filter is applied, we see that only a single frame has this characteristic, which makes the possibility of a flawed implementation generating this malformed frame less of a possibility.

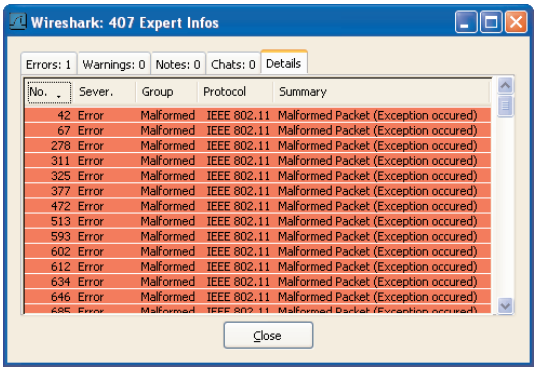
Fortunately, Wireshark has a facility for performing expert information analysis in order for the packet capture to identify anomalies such as malformed frames. Instead of scrolling through the capture looking for information lines that indicate malformed frames, we can use the Expert Information feature by clicking **Analyze | Expert Info Composite**. Wireshark assesses the contents of the packet capture and opens a new window (see Figure 6.59).

Figure 6.59 Expert Information Analysis - Real-world Capture 7



We see that Wireshark has detected 407 malformed frames in this packet capture. Expanding the list of malformed frames by clicking on the plus (+) sign in the group column, reveals the list of packets that are malformed. Clicking on any of the packet's identifiers will update the Packet List window to display the contents for the selected packet. Clicking on the Details tab will display additional information for each of the errors detected (see Figure 6.60).

Figure 6.60 Expert Information Analysis - Detail Window - Real-world Capture 7



Here we see that each of the malformed frames has an exception in the IEEE 802.11 protocol analysis. Select frame 67 and click **IEEE 802.11 Wireless LAN Management Frame | Tagged Parameters** to view the tagged parameter list. Again, Wireshark attempts to dissect the contents of the tagged parameters, but characterizes each tag as a reserved tag number. Expanding the last tag in the management payload reveals that it is using tag number 64 with a length of 62 bytes, even though only 28 bytes are remaining in the packet payload (excluding 4 bytes for the trailing FCS).

Returning to the main Wireshark window, we can use the display filter function to display only malformed frames with the following filter:

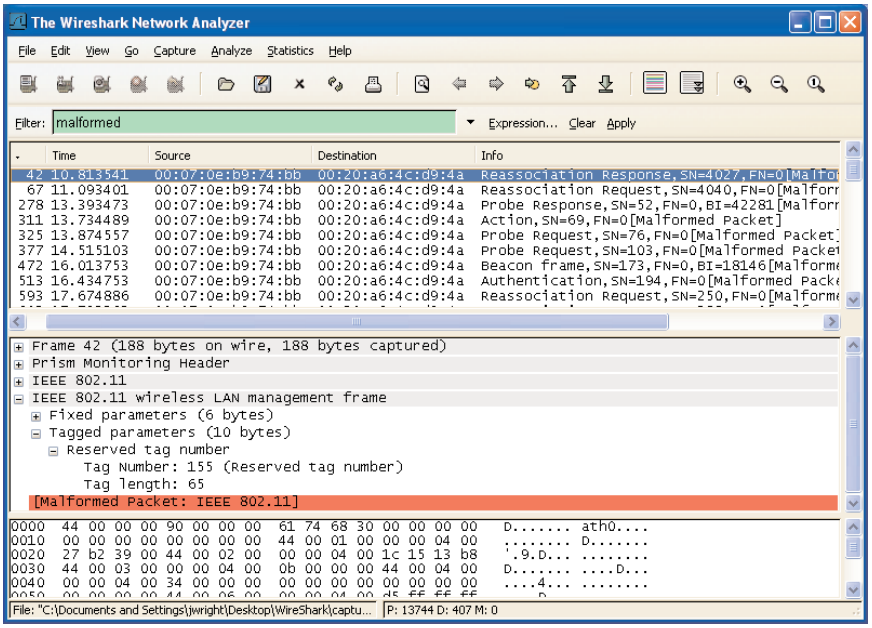
```
malformed
```

Enter this display string in the Filter text-box and click **Apply**. Wireshark will display a list of all the frames that were identified as malformed (see Figure 6.61).

In this display we are examining only the malformed frames, which gives us some curious information about the packet capture:

- Each malformed frame has a consistent source MAC address and destination address.
- The frame types vary including reassociation response, reassociation request, probe response, action, probe requests, beacons, and so on. This is unusual, because frames that should only be transmitted by an AP (beacons, reassociation response, probe response) are mixed with frames that should only be transmitted by stations (probe request, reassociation request, association request).

Figure 6.61 Filtering on Malformed Frames - Real-world Capture 7



- Individual frames include values that are not reasonable; frame 278 indicates the beacon interval is 42,281 millisecond (msec) (*BI=42281*), which means the AP is transmitting beacons once every 43.3 seconds, as opposed to the standard convention of 10 times per second. Similarly, frame 472 reports a beacon interval of 18,146, or one beacon every 18.1 seconds.

Selecting individual frames reveals more anomalous activity (e.g., frame 311 is identified as an action frame, a new type of management frame designed to support the IEEE 802.11h, IEEE 802.11k, and IEEE 802.11r working groups. The Flags byte in the frame control header for this frame has the To DS bit set and the From DS bit clear, which indicates it is a wireless station and not an AP, but it also has the power management bit set (indicating the station is going to enter a power-conservation mode) and the more data bit set (indicating it is an AP which has buffered packets waiting to be delivered to a station). Further, the strict-order bit is set, which is generally not used in IEEE 802.11 implementations and should always be clear.

TIP

A great place to get information about upcoming IEEE 802.11 standards is the IEEE 802.11 timelines page, where each working group provides a short summary of the activity of the working group and the estimated schedule for the ratification of the standard or amendment. Linked to each task group is the project authorization request form and approval letter, which documents the intentions of the working group and the detailed status page for the activity of the working group. The IEEE 802.11 timelines page is available at http://grouper.ieee.org/groups/802/11/Reports/802.11_Timelines.htm.

Our analysis suggests evidence of IEEE 802.11 protocol fuzzing; our "malformed" display filter revealed over 400 packets that have conflicting field settings and reserved field values. However, further analysis also indicates that these 400 frames are not the only packets that appear to be the result of protocol fuzzing. Apply the following display filter to identify all frames with the consistent source and destination address we have identified for this traffic.

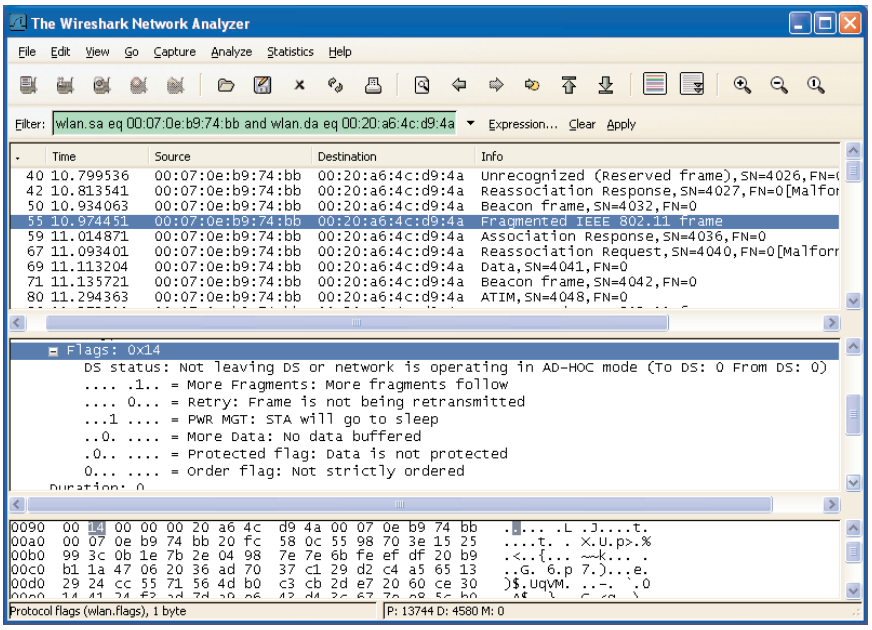
```
wlan.sa eq 00:07:0e:b9:74:bb and wlan.da eq 00:20:a6:4c:d9:4a
```

Applying this filter returns 4,580 frames (see Figure 6.62).

Even though many of these frames aren't recognized as malformed by Wireshark, they appear to have similar characteristics that would lead us to believe that they are also the result of IEEE 802.11 protocol fuzzing. For example, frame 55 is reported as a fragmented packet, a feature that is seldom-used in wireless LANs. However, it is also indicating that the station is going to sleep, effectively saying, "Here's the first part of a packet, now I'm going to sleep, so hold on to this for me."

From a security researcher's perspective, Wireshark is an indispensable tool for analyzing the results of protocol fuzzing activity, assisting in narrowing down the activity that caused misbehavior in the target station. From an intrusion analysis perspective, it's not likely you'll see this kind of activity on your network, because most of these packets don't elicit a response from the target station; rather, a Wireless Local Area Network (WLAN) IDS system may observe the few frames sent by the attacker to reproduce an identified vulnerability in an effort to exploit a victim system.

Figure 6.62 Filtering on Consistent Source and Destination - Real-world Capture 7



Summary

Packet sniffing on wireless networks has unique challenges that are different than the challenges of capturing traffic on wired networks. Fortunately, many wireless cards support the ability to capture wireless traffic without needing to connect to a network with the monitor mode feature. By leveraging available tools and drivers for Windows and Linux systems, you can use a standard wireless card to capture traffic on the wireless network for analysis.

Wireshark's wireless analysis features have grown to be a very powerful tool for troubleshooting and analyzing wireless networks. Leveraging Wireshark's display filters and powerful protocol dissector features, you can sift through large quantities of wireless traffic to identify a specific condition or field value you are looking for, or exclude undesirable traffic until you are left with only a handful of traffic to assess. In this chapter, we examined several examples of display filters taken from practical analysis needs that you can apply to your own network analysis needs.

Wireshark doesn't limit itself to display filters for wireless analysis; we can also take advantage of other analysis features built into Wireshark to simplify wireless network analysis. Features like Wireshark's coloring rules allow us to leverage display filters to uniquely color-code packets in the Packet List window, which allows you to assess the contents of a packet capture by looking at the number of packets. If your packet capture includes radio signal strength information or transmission rate information, Wireshark can make that information visible as well, giving extra visibility into the health and robustness of wireless client connectivity. Finally, when configured with the appropriate encryption keys, Wireshark can decrypt traffic dynamically, to further simplify the task of network troubleshooting.

Finally, we examined several packet captures taken from production and lab wireless environments, to demonstrate Wireshark's wireless analysis and troubleshooting capabilities. Without a doubt, Wireshark is a powerful assessment and analysis tool for wireless networks that should be a part of every auditor, engineer, and consultant toolkit.

Solutions Fast Track

Techniques for Effective Wireless Sniffing

- ☑ Wireless cards can sniff on one channel at a time.
- ☑ Channel hopping is used to rapidly change channels and briefly capture traffic.
- ☑ Interference can result in lost traffic and incomplete packet captures.
- ☑ Locate the capture station near the station being monitored, while disabling any local transmitters and minimizing CPU utilization.

Understanding Wireless Card Operating Modes

- ☑ Wireless card operating modes include managed, master, ad-hoc, and monitor.
- ☑ Monitor mode causes the card to passively capture wireless traffic without connecting to a network.
- ☑ Wireless cards do not normally transmit while in monitor mode.

Configuring Linux for Wireless Sniffing

- ☑ Linux Wireless Extensions compatible drivers use the *iwconfig* utility to configure monitor mode.
- ☑ The Linux MADWIFI drivers for Atheros cards use the *wlanconfig* utility to configure monitor mode.
- ☑ Linux Wireless Extensions compatible drivers and the MADWIFI drivers use the *iwconfig* utility to specify the channel number.

Configuring Windows for Wireless Sniffing

- ☑ Windows does not have a built-in mechanism for using a wireless driver in monitor mode.
- ☑ The commercial AirPcap drivers and USB wireless dongle can be used to capture traffic in monitor mode.

Using Wireless Protocol Dissectors

- ☑ Frame statistic information is included as the first group of fields in the Packet Details window.
- ☑ Protocol dissectors extract and enumerate fields in the IEEE 802.11 header and payload.
- ☑ The IEEE 802.11 header and payload data can be very complex, but the data is easily assessed with protocol dissectors.

Useful Wireless Display Filters

- ☑ Display filters can be applied to any of the fields in the IEEE 802.11 header and payload data.
- ☑ Complex display filters can be built by adding to a filter with *AND* or *OR* conditions.
- ☑ You can apply inclusive filters when looking for a specific set of data, or to remove uninteresting data from the packet list.

Leveraging Wireshark Wireless Analysis Features

- ☑ Coloring rules leverage display filters to identify matching display filter conditions.
- ☑ Applying a handful of helpful coloring rules can make it easier to analyze large quantities of frames.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: Can I use Wireshark for wireless intrusion analysis?

A: Wireshark's display filter functionality can be used to identify some attacks on wireless networks, such as deauthenticate DoS attacks (*wlan.fc.type_subtype eq 12*), FakeAP (*wlan_mgt.fixed.timestamp < "0×000000000003d070"*) and NetStumbler (*wlan.fc.type_subtype eq 32*, *llc.oui eq 0x00601d*, and *llc.pid eq 0x0001*), but it is not a replacement for a sophisticated WLAN IDS system.

Q: Can I use Wireshark to crack wireless encryption keys?

A: No, Wireshark does not include any key cracking functionality. Wireshark can decrypt WEP traffic, but only when configured with the correct WEP key.

Q: Can I use Wireshark to analyze traffic captured with Kismet?

A: Yes, Kismet generates several output file types including libpcap files with a *.dump* extension. Wireshark can open and assess libpcap files generated with Kismet.

Q: Can I use Wireshark to analyze traffic captured with NetStumbler?

A: No, NetStumbler does not capture traffic in monitor mode and is unable to create libpcap files for use with Wireshark.

Q: What is the best card to get for wireless analysis?

A: Wireless cards with an Atheros chipset are known to be effective at capturing wireless traffic on Linux systems, often allowing you to select between IEEE 802.11a, 802.11b and 802.11g traffic. Visit the Atheros Product Database at http://customerproducts.atheros.com/customerproducts/ResultsPageBasic.asp to identify if a card is based on an Atheros chipset. For Windows hosts, the AirPcap adapter is functional and well-supported by Wireshark, but does not yet support IEEE 802.11a channels.

Q: Can I use Wireshark to capture traffic while connected to an AP?

A: When associated to an AP, the wireless card is working in managed mode. Some drivers allow you to capture traffic in managed mode, but the data is reported as if it were coming from a standard Ethernet interface. This prevents you from seeing management frames, control frames, and data destined for other networks.

Q: Can Wireshark sniff IEEE 802.11a/b/g/n networks?

A: Wireshark isn't limited in its ability to sniff any Physical layer wireless network type, as long as the driver is compatible with libpcap/winpcap and the wireless card supports monitor mode for the desired spectrum. At the time of this writing we are just starting to see pre-802.11n networks; if your wireless card and driver support monitor mode for IEEE 802.11n modulation, Wireshark can be used to analyze the traffic.

Q: How can I examine signal strength information in a Wireshark capture?

A: Wireshark will display signal strength information in any packet capture that includes this information in the frame header contents. Some packet captures only contain the IEEE 802.11 header contents, which does not include signal strength information. When capturing traffic for a wireless network, use the Radiotap of Prism AVS link types to record signal strength information.