CHAPTER 10

WIRELESS HACKING

Wireless technology hit the American market more than 60 years ago during the World War I, World War II era. However, due to the perceived threats to national security it was deemed for military use only. Today, wireless computing is in the steep upside climb toward its peak in the marketplace; likewise are the technology hype, feature development, and insecurities surrounding wireless. In 1999, approximately 1.4 million wireless local area network (WLAN) transceivers were distributed worldwide. Only one year later in 2000, the number nearly quadrupled to 4.9 million, and the numbers are expected to keep growing until 2006, when nearly 56 million WLAN transceivers are projected to be distributed. This growth would represent a predicted $4.5 billion market, according to recent Allied Business Intelligence reports.

802.11 wireless networks should not be confused with their cousin Bluetooth, which was developed by a commercial coalition, including Ericsson, Motorola, and Microsoft. 802.11 networks currently transmit on the 2GHz and 3GHz bands, although development and prototypes have been created to work on the 5GHz band. Due to the relatively quick development time and the initial specification for the 802.*x* protocols and the Wired Equivalent Privacy (WEP) algorithm, numerous attacks, cracks, and easy-to-use tools have been released to irritate such technology innovators.

In this chapter, we will discuss the more important security issues, countermeasures, and core technologies publicly identified in the 802.11 realm to date, from the perspective of the standard attack methodology we have outlined earlier in the book: footprint, scan, enumerate, penetrate, and, if desired, deny service. Because wireless technology is somewhat different in attack techniques when compared to wired devices, our methodology combines the scan and enumerate phases into one cohesive stage.

You can expect to see the latest tools and techniques that hackers use during their war-driving escapades to identify wireless networks, users, and authentication protocols, in addition to penetration tactics for cracking protected authentication data and leveraging poorly configured WLANs. Also, numerous vendor configurations and third-party tools will be highlighted so that site administrators will gain a step up in defending their wireless users and networks.

At the end of this chapter you should be able to design, implement, use a modern war-driving system capable of executing most of the latest attacks on your wireless network as well as defending against such attacks.

WIRELESS FOOTPRINTING

Wireless networks and access points (APs) are some of the easiest and cheapest types of targets to footprint (or "war-drive") and ironically some of the hardest to detect and investigate. War-driving once was synonymous with the simple configuration of a laptop, a wireless card, and Network Stumbler (or NetStumbler). Now it is a much more sophisticated setup that can utilize multiple types of high-powered antennas, wireless cards, and palm-sized computing devices, including the ever-popular iPAQ and Palm.

We use the term "war-driving" loosely in the realm of the hacking methodology and "footprinting" mainly because you do not have to be driving. You may walk around a technology park, downtown area, or simply through the halls of your own building with your laptop if you are performing an internal audit. Footprinting wireless devices, particularly APs, start with the simple task of locating them via the passive method of listening for AP broadcast beacons or the more aggressive method of transmitting client beacons in search of AP responses. Understand that all WLAN footprinting can be done remotely as long as you are in range to receive or transmit beacons and packets to the AP. With this said, a huge advantage would be to have a better antenna than what usually comes with the card you purchase.

As you will see, the proper equipment makes all the difference in footprinting a WLAN. Numerous types of wireless cards exist, with different chipsets. Some allow you to put the card in promiscuous mode (that is, to sniff the traffic) and others will not. Likewise, certain cards inherently work better because they provide support for different operating systems. Antenna strength and direction are also equipment factors. You may want to use an omnidirectional antenna if you are just driving through crowded streets and a directional antenna if you're targeting a specific building, location, or AP. Oh yes, let us not forget about the global positioning system (GPS). GPS will prove to be a wonderful addition to your equipment list if you wish to track APs, monitor their transmitting range, and potentially retest them in the future.

Equipment

Certain types of equipment will be necessary to execute a subset of the presented attacks in addition to the required software. Wireless cards, antennas, GPS devices, as you will notice, play a large role in what kinds of attacks and at what range these attacks will be successful.

Cards

Be aware, all wireless cards are not created equal. It is important to understand the requirements and limitations of the cards you plan to use. Some cards require more power, are less sensitive, and might not have an available antenna jack for expanding the range with an additional antenna. You should also know that the ramp-up times to use a card with particular operating systems are significantly different. If you choose to use Linux or BSD, you will have to recompile the kernels with the proper pcmcia-cs drivers, which may not be an easy task if you have little to no UNIX experience. Windows, on the other hand, is a much easier setup process, but you will notice there are far fewer tools, exploits, and techniques you can use from the Win32 console.

AiroPeek NX is the only wireless sniffer worth mentioning for the Windows environment. NetStumbler, a tool that often gets mistaken as a wireless sniffer, only parses wireless packet headers and uses a nice GUI for real-time reporting on access point location, identification, and a few other particulars. The AiroPeek NX application supports packet capturing via 802.11a and 802.11b. It also supports non-U.S. channel surfing. The United States has provisioned for 802.11 wireless networks to utilize channels 1 through 11 for communication; however, other countries outside the U.S. commonly utilize channels 1 through 24. One particularly useful feature of AiroPeek NX, if you are an international traveler, is that it can support up to all 24 channels. The link listed here provides a full listing of the cards supported by the AiroPeek NX suite:

Windows WLAN Sniffer Driver Compatibility	http://www.wildpackets.com/support/hardware/airopeek_nx

The most widely supported OS in regard to wireless attack tools, drivers, and sniffers is by far Linux. The Linux community has invested significant time and resources into developing a collection of PCMCIA drivers (pcmcia-cs) that are compatible with most vendor releases of the 802.11b Prism2 chipset. As stated earlier, you must compile these drivers into the kernel.

Installing the drivers is quite easy and extremely similar to just installing about all other Linux-based applications and drivers. The following installation instructions are current for version 3.2.3 of the pcmcia-cs drivers. Obviously, if a later version is out and you attempt to install it, make sure you change the version number in the file name and directory structures. You can download the current pcmcia-cs drivers from http://sourceforge.net/project/showfiles.php?group_id=2405.

The following are general installation directions:

1. Untar and extract the pcmcia-cs-3.2.3.tar.gz files into /usr/src.
2. Run ``make config'' in /usr/src/pcmcia-cs-3.2.3/.
3. Run ``make all'' from /usr/src/pcmcia-cs-3.2.3/.
4. Run ``make install'' from /usr/src/pcmcia-cs-3.2.3/.

Depending on your WLAN, system configuration, or target networks, you may need to customize the startup script and the option files in /etc/pcmcia directory.

You can certainly find the drivers you need for your card with a quick query on Google.com, but it is always nice to have the information given to you. Therefore, listed next are some of the best locations to get your wireless card drivers for Linux. As you can see, they are divided by chipset.

Orinoco	http://airsnort.shmoo.com/orinocoinfo.html
Prism2	http://www.linux-wlan.com/linux-wlan/
Cisco	http://airo-linux.sourceforge.net/

Last, but definitely not least, let's tackle the driver issue for all you who like the new OpenBSD kernel on the Mac laptops (or any other laptop you use that's loaded with OpenBSD.) The OpenBSD kernel is very similar to Linux for the types of procedures required to get the system up and running in a wireless mode, specifically promiscuous wireless mode. Because of this, here's a good link where you can get drivers and more information on the BSD tools if your heart so desires.

OpenBSD Wireless Drivers	http://www.dachb0den.com/projects/source-mods.html

Antennas

Be prepared. Finding and installing the proper antenna may prove to be the most cumbersome task in setting up your war-driving "giddyap." You must first decide what type of war-driving you are going to do (see Figure 10-1). Is it going to be in a major city such as New York, Boston, or San Francisco? Maybe you are going to drive around an area that is less dense, such as the "Silicon Valley of the East Coast," Northern Virginia, or the suburbs of Los Angeles, where you need to drive at high speeds and may be 30 to 40 yards from the target buildings and their access points. These considerations must go into the decision for the antenna you are going to use.

To completely understand the differences in antennas, you need to get a little primer on some of the behind-the-scenes technology for the antennas. First and foremost, you need to understand antenna direction. Basically there are three types of direction when it comes to classifying antennas: directional, multidirectional, and omnidirectional. In general, directional antennas are used when communicating or targeting specific areas and are not very effective for war-driving (if you are actually driving). Directional antennas are also the type of antennas that are most effective in long-range packet capturing

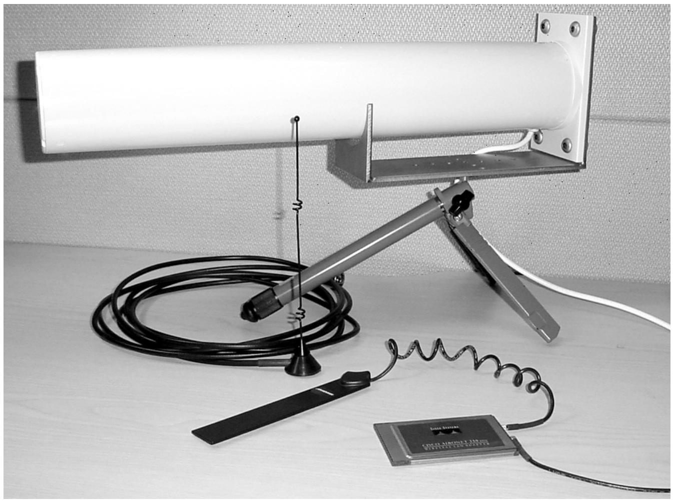


Figure 10-1. Typical war-driving antennas

because the power and waves are tightly focused in one direction. Multidirectional antennas are similar to directional antennas in the sense that both use highly concentrated and focused antennas for their transceivers. In most cases, multidirectional antennas are bidirectional (a front and back configuration) or quad-directional. Range is usually a bit smaller when compared to equally powered unidirectional antennas because the power must be used in more than one direction. Lastly, omnidirectional antennas are what most think of when they think of antennas. An omnidirectional antenna is the most effective in close city driving because it transmits and receives signals from all directions, thereby providing the largest angular range. Just for all you who learn in a pictorial manner, car antennas are omnidirectional.

Now that you understand the different terms for antenna direction, it is pertinent that you understand a few of the common types of antennas and how to distinguish a good antenna from a bad one. The wireless term "gain" is used to describe the energy of a directionally focused antenna. Realize that all transceiver antennas have gain in at least two directions—the direction they are sending information and the direction they are receiving it. If your goal is to communicate over long distances, you will want a narrow-focus, high-gain antenna. Yet, if you do not require a long link, you may want a wide-focus, low-gain antenna (omni).

Very few antennas are completely unidirectional because in most cases this would involve a stationary device communicating with another stationary device. One common type of unidirectional antenna would be a building-to-building wireless bridge. A yagi antenna uses a combination of small horizontal antennas to extend its focus. A patch or panel antenna has a large focus that is directly relational to the size of the panel. It appears to be a flat surface and focuses its gain in one general direction. A dish is another type of antenna that can be used, but it's only good for devices that need to transmit in one general direction, because the back of the dish is not ideal for transmitting or receiving signals. For all practical purposes, you will most likely need an omnidirectional antenna with a wide focus and small gain that can easily connect to your wireless card without the need of an additional power supply.

Numerous vendors and distributors are out there that you could use to get the proper equipment to go war-driving. Listed next are some of our favorites. Each will sell you some of the general stuff you will need; however, Wireless Central is well known for its actual "war-driving bundles," and HyperLinkTech is known for its high-powered and long-range antennas.

HyperLinkTech	http://www.hyperlinktech.com
Wireless Central	http://www.wirelesscentral.net
Fleeman, Anderson, and Bird Corporation	http://www.fab-corp.com/

GPS

A global positioning system (GPS) is the wireless equivalent of using a network-mapping tool or application on wired network assessments (see Figure 10-2). Most GPS devices wrap into the war-driving software via timestamp comparisons. The GPS software keeps a real-time log of the device's position by mapping the longitude and latitude coordinates with corresponding timestamps into a simple text file. These text files are easily imported into a variety of mapping software programs that you can use to create colorful and accurate maps for identified access points and their range.

GPS units are relatively easy to purchase and install on your laptop, especially if you are a Windows user. Numerous vendors are available, and most of the actual devices are relatively similar when it comes to their technology aspects. The main differences between the competing products involve aesthetics—the look and feel of the units—and the software that comes packaged with the products. Good software comes with a good amount of rural and suburban maps, up-to-date streets, and most important an excellent direction algorithm. These features all come into use when you attempt to route future war-drives to ensure you don't backtrack as well as when you are profiling large areas.

Installing the drivers and the GPS unit is more or less straightforward; however, there are a few considerations you should make before the actual installation takes place. You will need to determine where your setup will go and how you will actually do your war-driving. For example, a serial cable is needed for connecting your GPS to your laptop in most cases, plus you will find out that your GPS unit gets better and more accurate location readings if it has direct access to the sky. All of you who are fortunate enough to have convertible Boxters and Jeeps need not worry; everyone else may want to consider purchasing a long enough cable for their GPS unit to sit on the dashboard of their car or rigging the unit with a magnet and affixing it to the roof.

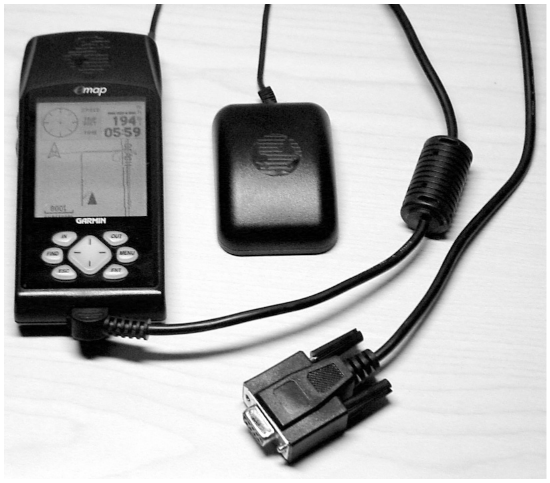


Figure 10-2. GPS unit

NOTE Don't forget a GPS unit will do you little good if you don't have proper range with your wireless card to begin with. Hence, if you are going to spend the time, effort, and money to get set up with a war-driving package, including one with GPS mapping software, you should purchase a decent antenna. Refer to the previous section for details and specifics about antennas, their features, and other war-driving specifics.

As with earlier sections in this chapter, we have listed a few of our personal favorites when it comes to finding and purchasing from a GPS vendor. We realize there are many other vendors you can choose from, but the following vendors are our recommendations because of unique products such as the Magellan line of GPS devices. Besides, the goal is that by the end of the chapter you will be able to properly design, implement, and use a top-of-the-line war-driving system that even your friends will be jealous of.

Garmin International	http://www.garmin.com/
Magellan	http://www.magellangps.com/

War-Driving Software

Setting up your war-driving software can be a bit more complicated due to its prerequisite hardware and software installations, mentioned previously. Because war-driving software requires a GPS unit to locate the position of the laptop by the AP as well as the use of AP identification software, setup may be prove to be a challenge. However, war-drivers allowing for the implementation of GPS units is one of the most useful features you will need. This is true simply because it allows you to map out vulnerable APs for future use or to pinpoint them for hardening.

Because wireless technology (and technology in general) tends to rely on acronyms, you need to be aware of a few simple terms before heading into this section and the rest of the chapter. These terms include SSID, MAC, and IV. The Service Set Identifier (SSID) is used as an identifier to distinguish one access point from another (or in macro-cases, one organization from another). You can think of it as something similar to a domain name for wireless networks. The Media Access Control (MAC) address is the unique address that identifies each node of a network. In WLANs, it can be used as a source for client access control. The Initialization Vector (IV) of a Wired Equivalent Privacy (WEP) packet is included after the 802.11 header and is used in combination with the shared secret key to cipher the packet's data.

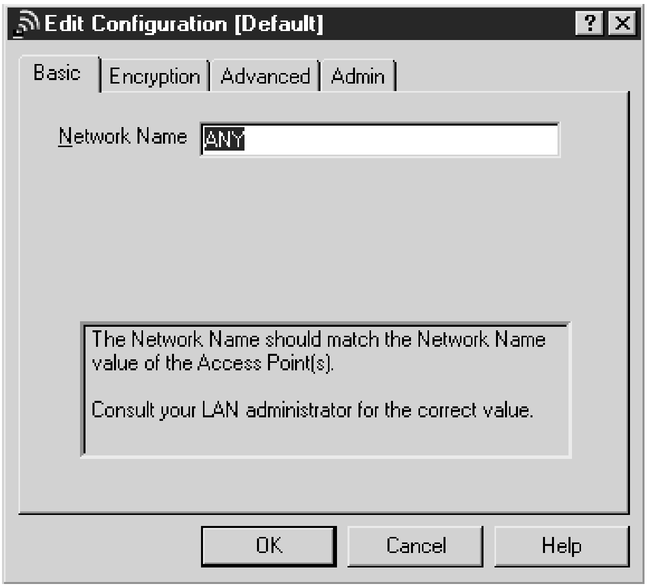
NetStumbler, the first publicly available war-driver application, was released as a tool that analyzed the 802.11 header and IV fields of the wireless packet in order to determine the SSID, MAC address, WEP usage, WEP key length (40 or 128 bit), signal range, and potentially the access point vendor. Soon after, a few Linux and UNIX-based tools came out that had similar tactics but also allowed for WEP key cracking and actual packet data cracking. Most of these cracking tools made use of Tim Newsham's discovery and implementation of exploiting key weaknesses in the WEP algorithm and key scheduling algorithm (KSA). Some of the industry-standard war-drivers are listed next. All are different; hence, each has a unique tool feature that you may need in the field.

NetStumbler

Popularity:	*9*
Simplicity:	*9*
Impact:	*9*
Risk Rating:	***9***

NetStumbler (http://www.netstumbler.com/) is a Windows-based war-driving tool that will detect wireless networks and mark their relative position with a GPS. NetStumbler uses an 802.11 Probe Request sent to the broadcast destination address, which causes all access points in the area to issue an 802.11 Probe Response containing network configuration information, such as their SSID and WEP status. When hooked up to a GPS, NetStumbler will record a GPS coordinate for the highest signal strength found for each access point. Using the network and GPS data, you can create maps with tools such as StumbVerter and Microsoft MapPoint. NetStumbler supports the Hermes chipset cards on Windows 2000, the most popular being the Lucent (now Proxim) Orinoco branded cards. On Windows XP the NDIS 5.1 networking library has 802.11 capabilities itself, which allows NetStumbler to be used with most cards that support it.

To use NetStumbler, insert your wireless card and set your SSID or network name to ANY. For Orinoco cards, this can be found in the Client Manager utility, as shown next. If NetStumbler doesn't detect access points you know are present, check this first before performing other troubleshooting. Setting the Network Name field to ANY tells the driver to use a zero-length SSID in its Probe Requests. By default, most access points will respond to Probe Requests that contain their SSID or a zero-length SSID.



Network Name: ANY

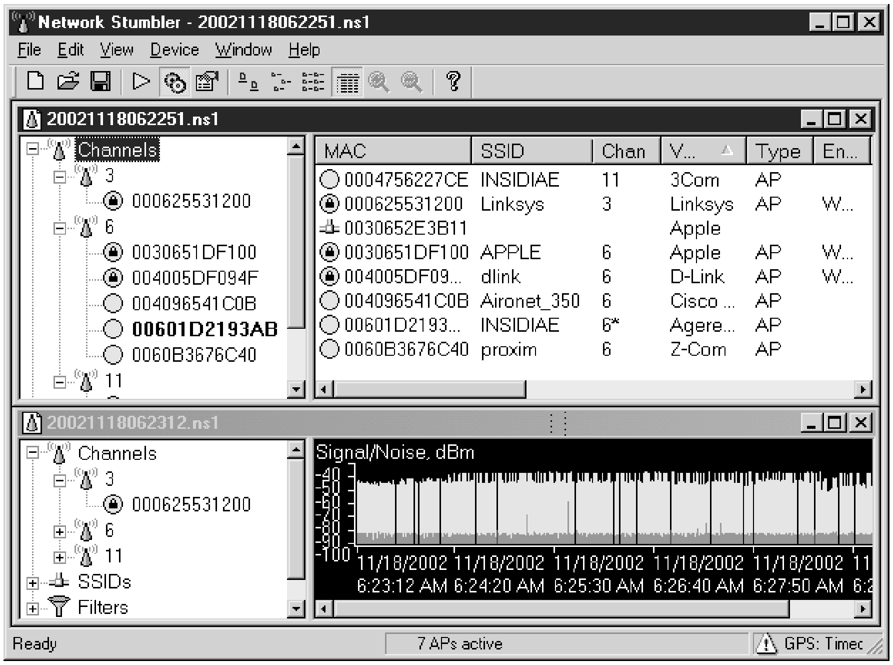
The Network Name should match the Network Name value of the Access Point(s).

Consult your LAN administrator for the correct value.

Once the card is configured correctly, start up NetStumbler and click the green arrow on the toolbar (if not depressed already). If there are any access points in the area that will respond to a Broadcast Probe Request, they should respond and be shown in the window. You can use the Filters option to quickly sort multiple networks on criteria such as WEP usage or whether the network is an IBSS or BSS type network. Because an IBSS (Independent BSS) network is a group of systems operating without an access point like a BSS network, an attacker would only be able to access the systems in that network and not necessarily use the wireless network as a bridge to the internal LAN. Selecting any of the networks by their circle icon will also show a signal-to-noise ratio graph (see Figure 10-3).

NetStumbler Countermeasures

NetStumbler's primary weakness is that it relies on one form of wireless network detection, the Broadcast Probe Request. Wireless equipment vendors will usually offer an option to



MAC	SSID	Chan	V...	Type	En...
0004756227CE	INSIDIAE	11	3Com	AP	
000625531200	Linksys	3	Linksys	AP	W...
0030652E3B11			Apple		
0030651DF100	APPLE	6	Apple	AP	W...
004005DF09...	dlink	6	D-Link	AP	W...
004096541C0B	Aironet_350	6	Cisco ...	AP	
00601D2193...	INSIDIAE	6*	Agere...	AP	
0060B3676C40	proxim	6	Z-Com	AP	

Figure 10-3. Network Stumbler

disable this 802.11 feature, which effectively blinds NetStumbler. Other war- driving software available now, such as Kismet, will also use this method but have other detection mechanisms to back them up if they fail. That said, there is still no shortage of networks that can be detected by NetStumbler, and the feature to respond to a Broadcast Probe Request is still enabled by default for many vendors.

Kismet

Popularity:	*8*
Simplicity:	*7*
Impact:	*9*
Risk Rating:	***8***

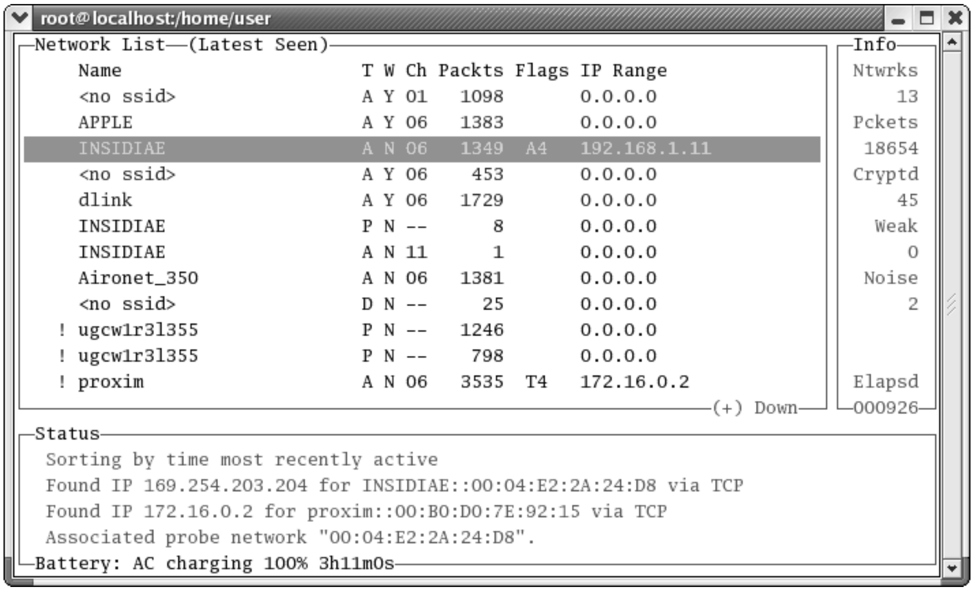
Kismet (http://www.kismetwireless.net/) is a Linux and BSD-based wireless sniffer that has war-driving functionality. It allows you to track wireless access points and their GPS locations like NetStumbler, but offers many other features as well. Kismet is a passive network-detection tool that will cycle through available wireless channels looking for 802.11 packets that indicate the presence of a wireless LAN, such as Beacons and Association Requests. Kismet can also gather additional information about a network if it can, such as IP addressing and Cisco Discovery Protocol (CDP) names.

Included with Kismet is a program called GPSMap, which generates a map of the Kismet results. Kismet supports most of the wireless cards available for Linux or OpenBSD.

To use Kismet, you will first have to install the custom drivers required for monitor mode operation. This can vary depending on the chipset your card uses, but Kismet comes with a single way to enable all of them for monitor operation. Before starting Kismet, run the `kismet_monitor` script to place your card into monitor mode. Be sure you are in a directory that the Kismet user has access to before starting Kismet:

```
[root@localhost user]# kismet_monitor
Using /usr/local/etc/kismet.conf sources...
Enabling monitor mode for a cisco card on eth1
Modifying device eth1
```

This will place the wireless card configured in your kismet.conf file into monitor mode. Once Kismet is loaded, the interface will display any networks in range. By default, Kismet will sort the networks in an “Autofit” mode that doesn’t let you step through them. Press `S` to bring up the sort menu and then choose one of the available options; “l” (or latest time seen) works well in most cases. The main window, shown next, displays the network name (SSID). The T column displays the type of network, W signifies whether or not WEP is enabled, and Ch stands for "channel number." The IP Range column shows any detected IP addresses found, either via ARP requests or normal traffic.



```
root@localhost:/home/user
Network List—(Latest Seen)                                                    Info
    Name                        T W Ch Packts Flags IP Range                 Ntwrks
    <no ssid>                   A Y 01   1098       0.0.0.0                      13
    APPLE                       A Y 06   1383       0.0.0.0                  Pckets
    INSIDIAE                    A N 06   1349   A4  192.168.1.11              18654
    <no ssid>                   A Y 06    453       0.0.0.0                  Cryptd
    dlink                       A Y 06   1729       0.0.0.0                      45
    INSIDIAE                    P N --      8       0.0.0.0                    Weak
    INSIDIAE                    A N 11      1       0.0.0.0                       0
    Aironet_350                 A N 06   1381       0.0.0.0                   Noise
    <no ssid>                   D N --     25       0.0.0.0                       2
  ! ugcw1r31355                 P N --   1246       0.0.0.0
  ! ugcw1r31355                 P N --    798       0.0.0.0
  ! proxim                      A N 06   3535   T4  172.16.0.2               Elapsd
                                                              (+) Down       000926
Status
 Sorting by time most recently active
 Found IP 169.254.203.204 for INSIDIAE::00:04:E2:2A:24:D8 via TCP
 Found IP 172.16.0.2 for proxim::00:B0:D0:7E:92:15 via TCP
 Associated probe network "00:04:E2:2A:24:D8".
Battery: AC charging 100% 3h11m0s
```

Kismet Countermeasures

As far as countermeasures to Kismet go, there aren't many. Kismet is currently the best war-driving tool available and will find networks that NetStumbler routinely misses. In addition to its network-discovery capabilities, it can also automatically log WEP packets with weak IVs for use with AirSnort as well as detect IP addresses in use on the WLAN.

Dstumbler

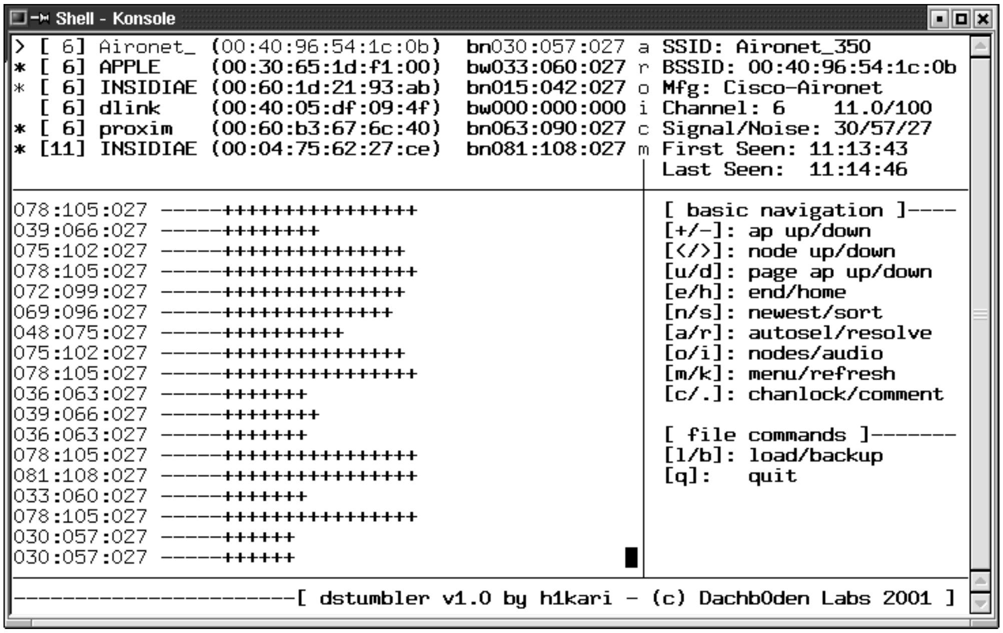
Popularity:	5
Simplicity:	6
Impact:	9
Risk Rating:	***7***

Dstumbler (http://www.dachb0den.com/projects/dstumbler.html) is part of the BSD-Airtools package for the OpenBSD, NetBSD, and FreeBSD operating systems. Dstumbler is a war-driving application that supports logging access point locations with a GPS. Support for both Orinoco and Prism2 cards is provided, although the monitor mode support that allows it to detect access points which don't respond to a Broadcast Probe Request is for Prism2 cards only. Dstumbler will also report if an access point is using a default SSID and has some capabilities to detect whether a network is using 40 or 104-bit WEP.

Dstumbler requires kernel patches for monitor mode support on the Prism2 cards for Net and Free-BSD. OpenBSD 3.2, however, includes these modifications in the default kernel. After the bsd-airtools package is installed, you can start Dstumbler by specifying the wireless interface to use.

```
foo# dstumbler wi0 -o -m 30 -l log.txt
```

This will start up Dstumbler on the wi0 interface, in monitor mode (`-o`), randomly changing the MAC address every 30 seconds (`-m 30`) and logging to an output file called log.txt (`-l log.txt`). Once the application has loaded, you are presented with an ncurses interface with three main windows, as shown next. At the upper left is a display of detected networks, to the right are the details for the selected network, and below is the real-time signal strength of the selected network. You can move up and down the network list with the up- and down-arrow keys.



```
> [ 6] Aironet_ (00:40:96:54:1c:0b)  bn030:057:027 a SSID: Aironet_350
* [ 6] APPLE    (00:30:65:1d:f1:00)  bw033:060:027 r BSSID: 00:40:96:54:1c:0b
* [ 6] INSIDIAE (00:60:1d:21:93:ab)  bn015:042:027 o Mfg: Cisco-Aironet
  [ 6] dlink    (00:40:05:df:09:4f)  bw000:000:000 i Channel: 6    11.0/100
* [ 6] proxim   (00:60:b3:67:6c:40)  bn063:090:027 c Signal/Noise: 30/57/27
* [11] INSIDIAE (00:04:75:62:27:ce)  bn081:108:027 m First Seen: 11:13:43
                                                   Last Seen:  11:14:46

078:105:027 -----++++++++++++++++                  [ basic navigation ]----
039:066:027 -----++++++++                          [+/-]: ap up/down
075:102:027 -----+++++++++++++++                   [</>]: node up/down
078:105:027 -----++++++++++++++++                  [u/d]: page ap up/down
072:099:027 -----+++++++++++++++                   [e/h]: end/home
069:096:027 -----++++++++++++++                    [n/s]: newest/sort
048:075:027 -----+++++++++++                       [a/r]: autosel/resolve
075:102:027 -----+++++++++++++++                   [o/i]: nodes/audio
078:105:027 -----++++++++++++++++                  [m/k]: menu/refresh
036:063:027 -----+++++++                           [c/.]: chanlock/comment
039:066:027 -----++++++++
036:063:027 -----+++++++                           [ file commands ]-------
078:105:027 -----++++++++++++++++                  [l/b]: load/backup
081:108:027 -----++++++++++++++++                  [q]:   quit
033:060:027 -----+++++++
078:105:027 -----++++++++++++++++
030:057:027 -----++++++
030:057:027 -----++++++

-----------------------[ dstumbler v1.0 by h1kari - (c) Dachb0den Labs 2001 ]
```

Dstumbler Countermeasures

When scanning with an Orinoco card, Dstumbler can be blocked by disabling the response to Broadcast SSID requests. In monitor mode, however, you will likely not be able to prevent the tool from detecting your SSID. Because Dstumbler can highlight the fact that you are using a default SSID, you should at least change it do something other than the OEM initial setting.

Wireless Mapping

Once you've discovered the available access points, one thing you can do with this data is create maps based on the results of the network and GPS data. War-driving tools will log the current GPS location, signal strength, and attributes of each access point. Based on this data, these tools can guess where the access point is on the assumption that the closer you get to an AP, the stronger the signal will be. Previously you would need to convert the results from your war-driving tool to a format that a mapping system such as Microsoft MapPoint or the MapBlast website could use to interpret the GPS coordinates. Now there is software available that automates this process for you and reads in the data straight from the war-driving tool. In addition to using your own data, some groups have established sites such as http://www.wifimaps.com and http://www.gigle.net to accumulate the information in a large database.

StumbVerter

StumbVerter (http://www.sonar-security.com/) is an application that uses MapPoint 2002 to plot data from files in the NetStumbler format. This saves you the hassle of manually inputting this information into MapPoint or another mapping tool. It also creates NetStumbler-style icons on the map for each access point. Green icons represent non-encrypted networks, and red icons indicate networks using WEP.

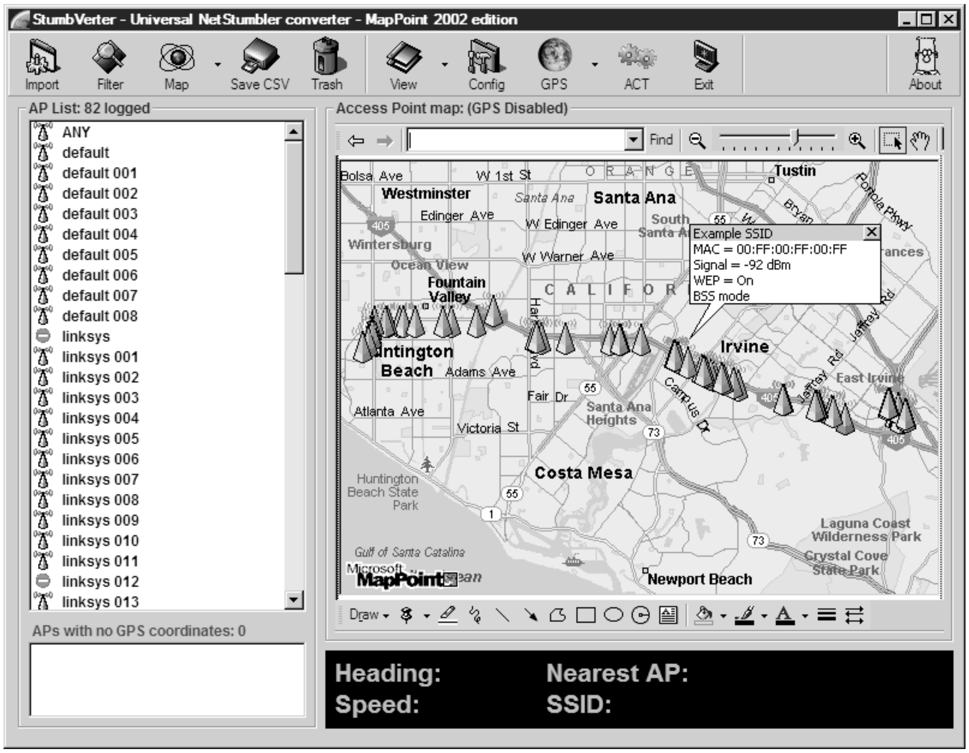
To use StumbVerter, click the Import button and select a saved NetStumbler scan (be sure it's one with GPS data; otherwise, StumbVerter will not be able to plot the AP locations). Once the map is loaded you can select View/Show All AP Names and Info to get additional information about each network, including the SSID and MAC address. The normal MapPoint 2002 controls are available, so you can zoom and edit the map just like you would in MapPoint. If you are satisfied with the map, you can save it off to a MapPoint file, bitmap, or HTML page (see Figure 10-4).

GPSMap

GPSMap is included with the Kismet wireless monitoring package. It imports Kismet gps and network files and then plots the network locations on maps from a variety of sources. GPSMap is probably the most versatile war-driving map generator available and supports many drawing options for each access point. Maps can be made based on the estimated range of each network, the power output, a scatter plot, or all of these options together. Although it is extremely flexible, GPSMap can be a bit command-line intensive. To create a map with GPSMap, you'll need some saved Kismet results with GPS data. This would be at least a .network and .gps file for a given date and scan. Here's an example:

Kismet-07-2002-1.network and Kismet-07-2002-1.gps.

Once you know which result files you want to use, you'll need to run GPSMap against those files with the right options. The major arguments are the name of the output file (`-o`), what source to take the background map image from (`-S`), and your draw options.


StumbVerter - Universal NetStumbler converter - MapPoint 2002 edition
Import
Filter
Map
Save CSV
Trash
View
Config
GPS
ACT
Exit
About
Access Point map: (GPS Disabled)
Find
Draw

AP List: 82 logged

```
ANY
default
default 001
default 002
default 003
default 004
default 005
default 006
default 007
default 008
linksys
linksys 001
linksys 002
linksys 003
linksys 004
linksys 005
linksys 006
linksys 007
linksys 008
linksys 009
linksys 010
linksys 011
linksys 012
linksys 013
```

APs with no GPS coordinates: 0

```
Example SSID
MAC = 00:FF:00:FF:00:FF
Signal = -92 dBm
WEP = On
BSS mode
```

Heading: Nearest AP:
Speed: SSID:

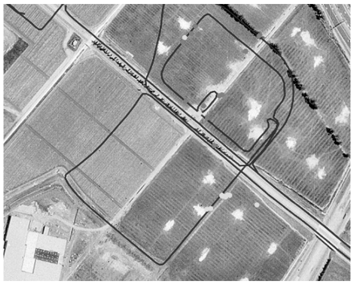
Figure 10-4. StumbVerter

Because GPSMap uses ImageMagick, your output file can be in almost any imaginable format, such as JPEG, GIF, or PNG. The background image sources are three vector map services—MapBlast, MapPoint, and Tiger Census maps—and one photographic source using United States Geological Survey (USGS) maps from Terraserver (http://terraserver.homeadvisor.msn.com/). Map sources or drawing options depend on your personal preferences and what you want to do with the map. It's best to try them all out and see which ones best fit your needs.

In the following example, we are creating a PNG map called newmap.png (`-o newmap.png`) using a USGS map as the background (`-S 2`) to a scale of 10 (`-s 10`). The drawing options are set to color the networks based on WEP status (`-n 1`), draw a track of the driven route (`-t`) with a line width of 4 (`-Y 4`), and map each access point with a dot at the center of the network range (`-e`), making the circle five units wide (`-H 5`). The last argument will be the name of the .gps file you wish to use for input.

```
[root@localhost user]# gpsmap -o newmap.png -s 10 -S 2 -n 1 -t -Y 4
 -e -H 5 Kismet-Nov-07-2002-1.gps
```

The resulting map will look something like this:



JiGLE

JiGLE (http://www.wigle.net/) is a Java client for viewing data from the WiGLE.net database of wireless networks (see Figure 10-5).

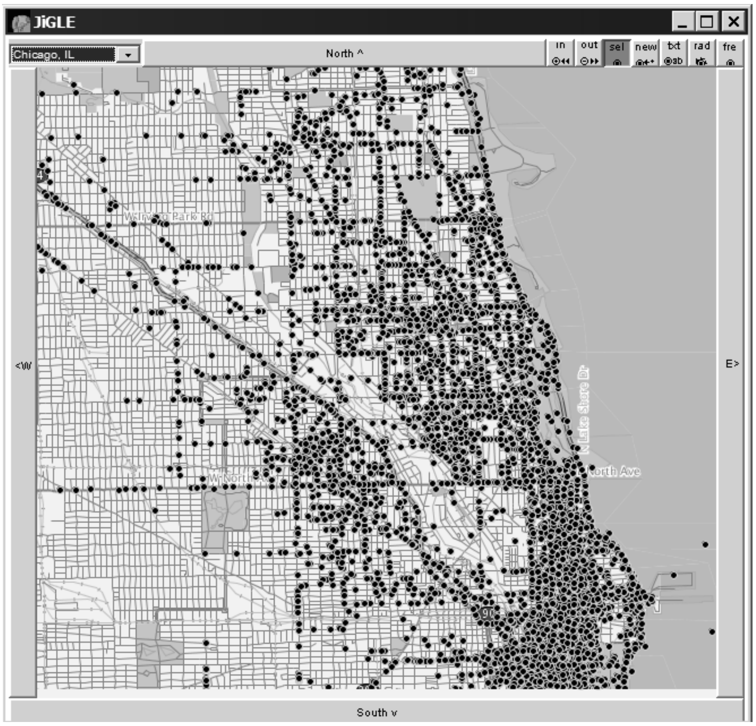


Figure 10-5. JiGLE

WiGLE.net currently has over 160,000 wireless networks listed in its database, which means that if you live in a area with WiGLE data, people wouldn't even have to go war-driving themselves to find your network. JiGLE reads in network and GPS data from WiGLE map packs. By default, it comes with a map pack for Chicago, but you just need to register to download any other available pack for other parts of the country. The client itself can also read in your own NetStumbler or Kismet results file and plot the network points on a map you provide.

To use JiGLE, make sure you have the Java JRE 1.3.0 or above installed and click the run.bat file in the JiGLE directory. Then select from the available map packs with the drop-down menu on the left side of the toolbar. If you'd like to get additional map packs, you'll need to download them from http://www.wigle.net/gps/gps/GPSDB/mappacks/.

If you're performing a wireless assessment, it would be a good idea to check the WiGLE database or other online databases such as http://www.netstumbler.com/ for the presence of your access point. Most of the DBs will honor your request to remove your AP.

WIRELESS SCANNING AND ENUMERATION

Following the *Hacking Exposed* attack methodology, the second and third stages of properly targeting and penetrating a system are scanning and enumeration. As you probably know by now, wireless technology is significantly different from most other technologies you have learned about in this book. Hence, it is the only technology that can be compromised without jumping on the wire. Wireless scanning and enumeration are combined in the sense that in general these stages of penetration are conducted simultaneously. Remember, the goal of the scanning and enumeration phases is to determine a method to gain system access.

After you have gone war-driving, identified target access points, and captured loads of WEP-encrypted and non-encrypted packets, it is time to start the next stage of the penetration process. Although installing the antenna may be the most difficult stage in preparing to war-drive, packet analysis is the most technically demanding aspect of wireless hacking because it requires you to be able to use and understand a packet sniffer and in some cases decipher the transmission itself.

During the initial war-driving expedition you must first undergo, you will have identified access points and some pertinent information about them. Such information could include an AP's SSID, MAC address, WEP usage, IP address, and different network transmissions. As with any attack, the more information you have at the onset of attempted penetration, the higher the probability of success and the more predictable the outcome of the attack.

Initially the single most important piece of data you should have about your identified access point is its SSID. In just about all cases this is how you will reference the identified AP. After you gain the SSID, the next goal is to determine and classify the types of data you've sniffed off the WLAN. The data can be logically divided by access point and then further subdivided by AP client. During packet analysis, you will quickly notice if the data you received from the initial war-drive is encrypted. If so you must determine whether the data is encrypted via a WEP-implementation schema or an additional layered schema such as SSL over HTTP. If a WEP-based encryption schema is being used, the next step is identifying whether it is a 40- or 128-bit key.

The initial step of scanning and enumerating a wireless network involves passively sniffing traffic and conducting analysis for further aggressive probes and attacks.

Wireless Sniffers

A preface for this chapter: Wireless sniffers are no different from "wired" sniffers when it comes to actual packet deciphering and analysis. The only difference is the wireless sniffer can read and categorize the wireless packet structure with 802.11 headers, IVs, and so on. Sniffers capable of capturing 802.11 packets will be heavily used within this section. If you have never used a sniffer or conducted packet analysis (or it has been a while since you have), it is highly recommended that you brush up your skills before moving on to this section.

Packet-Capture and Analysis Resources

The following resources when used together provide a thorough overview of the techniques and technical "know-how" behind packet-capturing and analysis.

- **http://www.robertgraham.com/pubs/sniffing-faq.html** A comprehensive site that could probably answer just about all your questions. Make this your first stop for information.
- **http://grc.com/oo/packetsniff.htm** A great source for specific packet analysis, commercial sniffers, identifying promiscuous-mode nodes, and thwarting unauthorized sniffers.
- **http://cs.ecs.baylor.edu/~donahoo/tools/sniffer/sniffingFAQ.htm** A good introductory site covering the basics of packet sniffing and the overall architecture requirements of a sniffer.

Many network sniffers exist for promiscuous card packet capturing, yet very few exist for the wireless side of the world due to the age of the technology. Basically you have three different setups you can run with, depending on your platform of choice: Windows, Linux, and OpenBSD. Granted, if you are a pro, you may be able to write your own drivers and sniffer modes to get your sniffer software to work under different platforms, but these three are currently the most supported via drivers and tools.

Flipping (a.k.a. switching) your wireless card into promiscuous mode is completely automated under Windows; however, under Linux it is a bit more complicated, which is exactly why we have included a guide for getting sniffer software working under Linux. Configuring the OpenBSD kernel and software is similar, so we apologize for not listing the redundancies. If you would like OpenBSD-specific information, go to www.dachb0den.com.

Configuring Linux Wireless Cards for Promiscuous Mode

By following these instructions, it should be rather simple for you to set up your Linux laptop and get to wireless sniffing in under an hour (not including tool and file download time).

Step 1: Get Prepared First and foremost, you will need a wireless PCMCIA network card with the Prism2 chipset. Now that you have your card, as with any new installation it is recommended that you back up your important data in case something were to cause your files to be irretrievable. Although this is not an overly risky installation, precautions should be taken. The following are examples of wireless cards that use the Prism2 Chipset:

- Compaq WL100
- SMC2632
- Linksys WPC11

Step 2: Get the Files When you have completed the first step and ready to start, you will need to download a few files if you don't already have them on your system. If the following links become broken because of new releases, it should not be difficult to find any of them via a Google search:

Linux PCMCIA Card Services Package	http://pcmcia-cs.sourceforge.net
Linux WLAN Package (linux-wlan-ng-0.1.10)	http://www.linux-wlan.com/linux-wlan
Prismdump Utility	http://developer.axis.com/download/tools/
CVS PCAP and CVS TCPDUMP	http://cvs.tcpdump.org
WLAN Drivers Patch (Tim Newsham's Patch)	http://www.lava.net/~newsham/wlan
Ethereal (Optional but Highly Recommended)	http://www.ethereal.com

Step 3: Compile and Configure Once you have downloaded the preceding files, you are ready to actually start configuring your system. In general, most apps use the `./configure | make | make install` installation setup, but for specific compilation instructions, refer to the individual Readme files for each of the applications.

NOTE It is extremely important that you execute the WLAN Drivers Patch (a.k.a. Newsham's Patch) before you compile the WLAN package on your system. It will not function properly otherwise.

Step 4: Flip the Card After compilation, you need to restart all your card services and ensure that all the modifications have been implemented. Most wireless sniffing and cracking tools have built-in functionality for flipping (changing) your card into promiscuous mode; however, you may wish to simply capture the packets without automated cracking or other features included within the tools. Whatever the case may be, the command to flip your card is shown here:

Enable Sniffing
```
%root%> wlanctl-ng wlan0 lnxreq_wlansniff channel=# enable=true
```

Disable Sniffing
```
%root%> wlanctl-ng wlan0 lnxreq_wlansniff channel=# enable=false
```

You should understand that when your card is in promiscuous mode, it is unable to send packets, thereby disallowing it to communicate on a wired or wireless network.

NOTE The pound sign (#) equals the channel number on which you wish to sniff packets. Most access points default to channels 6 and 10, meaning you will probably capture the most traffic while sniffing these channels.

Step 5: Start Sniffing The last step for manual wireless sniffing is to start capturing the packets to ensure you have completed the setup correctly. A simple tool you can use to test this is Prismdump, a tool you should have download and compiled in Steps 2 and 3. Prismdump simply manipulates the captured packets into the industry-standard format, PCAP. PCAP, a.k.a. the Packet Capture format, is often a common format for saving raw packet data.

Run Prismdump
```
%root%> prismdump > wlan_packets
```

A quick no-brainer: When your wlan_packets file is over one byte in size, you know you have started to capture 802.11 packets, which means you may start to use your WEP-cracking software or packet-analysis software, such as Ethereal.

Wireless Monitoring Tools

Wireless monitoring tools, as previously stated, are extremely similar to their wired "counterparts." Most of the tools are relatively easy to install and run with the analysis being the complicated aspect of the tool. Additional information for the presented tools can be found at the homepage for each of the corresponding applications and tools.

Prism2dump

Popularity:	3
Simplicity:	5
Impact:	7
Risk Rating:	**5**

Prism2dump (http://www.dachb0den.com/projects/prism2dump.html) comes with the BSD-Airtools suite and works with Prism2 chipset cards. It will output 802.11 frames with three levels of details, specified by the `-v` flag.

To use Prism2dump, you'll need to have a Prism2 kernel driver that supports monitor mode operation. Newer BSD versions such as OpenBSD 3.2 will support this in the default kernel, but others will need the kernel patches included in the BSD-Airtools package.

To use Prism2dump, first place your card in monitor mode with `prism2ctl <interface> -m`, as shown here:

```
foo# prism2ctl wi0 -m
```

After the command is issued, you can check the status by running `prism2ctl` and only specifying the interface. In the following example, the line for monitor mode indicates it is active:

```
foo# prism2ctl wi0
Sleep mode:                          [ Off ]
Suppress post back-off delay:        [ Off ]
Suppress Tx Exception:               [ Off ]
Monitor mode:                        [ On ]
LED Test:                            [ ]
Continuous Tx:                       [ ]
Continuous Rx:                       [ Off ]
Signal State:                        [ ]
Automatic level control:             [ Off ]
```

Once monitor mode is enabled, run Prism2dump by specifying the interface name and level of verbosity requested. Here, we are using the wi0 interface and `-v 2`, which prints all 802.11 protocol information.

The dump results show an 802.11 Management Probe Response from an access point with an SSID of APPLE on channel 6:

```
foo# prism2dump wi0 -v 2
prism2dump: listening on wi0
```

```
- [0:5:5d:a7:36:53 <- 0:30:65:1d:f1:0 <- 0:30:65:1d:f1:0]
- port: 7 ts: 151.143336 1:81 20:0
- sn: 3200 (d4:ec:cc:dc:8c:4c) len: 36
 - ** mgmt-proberesp ** ts: 17.605513 int: 100 capinfo: ess priv
   + ssid: [APPLE]
   + rates: 1.0 2.0 5.5 11.0
   + ds ch: 6
```

Tcpdump

Popularity:	*7*
Simplicity:	*6*
Impact:	*7*
Risk Rating:	***7***

Tcpdump (http://www.tcpdump.org/) is a standard UNIX network monitoring tool that supports decoding 802.11 frame information in newer versions. Because basic tcpdump usage is covered elsewhere in this book, we won't describe general information here, just the 802.11-specific items. To use tcpdump to decode 802.11 traffic, you'll need to install versions of libpcap and tcpdump that support it. As of this writing, the "current" rev of each package supports decoding 802.11 frames. Usage on wireless networks is basically the same as other types of networks, but you will need to place your card in monitor mode first to read the management frames. Outside of the various commands for each card and OS, the easiest way to flip the card to monitor mode is using the `kismet_monitor` script included with Kismet. Using tcpdump on a wireless network without putting the card in monitor mode will show broadcasts and traffic destined for the localhost, like a switched Ethernet network.

One option to note is `-e`, which will print out the frame-control fields, the packet length, and all the addresses in the 802.11 header that show the BSSID and destination MAC address. Also for parsing purposes, "wlan" can be used in place of "ether" for arguments such as `wlan protocol ip`. In the following example, we have already enabled monitor mode on the wireless card and are running tcpdump by specifying the wireless interface (`-i eth1`), getting the extra 802.11 information (`-e`), and printing out hex and ASCII data from the packets (`-X`):

```
[root@localhost root]# tcpdump -i eth1 -e -X
```

In the following packet you can see that BSSID is 00:60:b3:67:6c:40, the DA (or destination) is the broadcast address (FF:FF:FF:FF:FF:FF), and the source address is the same as the BSSID, the MAC address of the access point. The frame type is a Beacon, and it's using an SSID of proxim. The access point is capable of establishing an 802.11 link at speeds of 1, 2, 5.5, and 11 Mbps on channel 6.

```
16:13:52.974207 BSSID:00:60:b3:67:6c:40 DA:Broadcast SA:00:60:b3:67:
6c:40 Beacon (proxim) [1.0 2.0 5.5 11.0 Mbit] ESS CH: 6
0x0000   18e2 3540 1300 0000 6400 0100 0006 7072        ..5@....d.....pr
0x0010   6f78 696d 0104 0284 0b16 0301 0605 0400        oxim............
0x0020   0300 00                                        ...
```

Ethereal

Popularity:	*9*
Simplicity:	*6*
Impact:	*7*
Risk Rating:	***8***

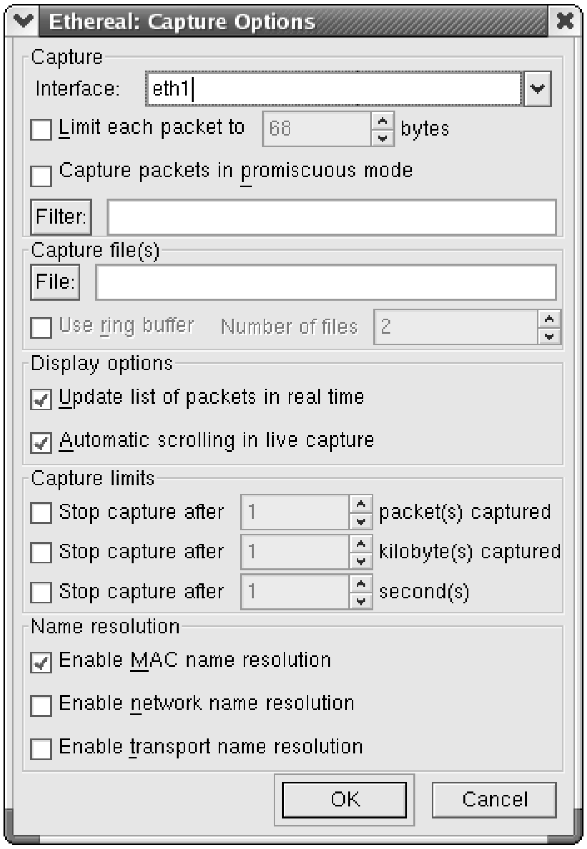
Ethereal (http://www.ethereal.com) is a UNIX- and Windows-based network monitoring tool. Although not specifically designed for 802.11 analysis, it does support capturing and decoding 802.11 packets with libpcap on UNIX systems. For Windows systems, it does not have the ability to directly capture 802.11 packets, but it can read the same capture file format that is generated by the UNIX versions of tcpdump or Ethereal. This means you could gather the data on a UNIX system and then later analyze it on your Windows machine.

We'll use Ethereal for most of the enumeration section because it does offer good filtering capabilities and is cross-platform enough to the degree that we can view packet data the same way across UNIX and Windows systems.

Ethereal requires drivers capable of monitor-mode operation and that the card be placed in monitor mode before you start capturing packets.

To use Ethereal to capture 802.11 packets, place your card into monitor mode with `kismet_monitor` or the card-specific command and start Ethereal. Press CTRL-K or select the Capture/Start menu to bring up the Capture Options window (shown next). Check the drop-down adapter list for your wireless interface (if it is not present, type in the interface name). You can configure the rest of the options as per your needs. Note that you do not need to be concerned with the "Capture Packets In Promiscuous Mode" box.

The card is placed in monitor mode before Ethereal is run, so this switch will not have an effect on the captured results.


Ethereal: Capture Options

Capture
Interface: eth1
[ ] Limit each packet to 68 bytes
[ ] Capture packets in promiscuous mode
Filter:

Capture file(s)
File:
[ ] Use ring buffer   Number of files 2

Display options
[✓] Update list of packets in real time
[✓] Automatic scrolling in live capture

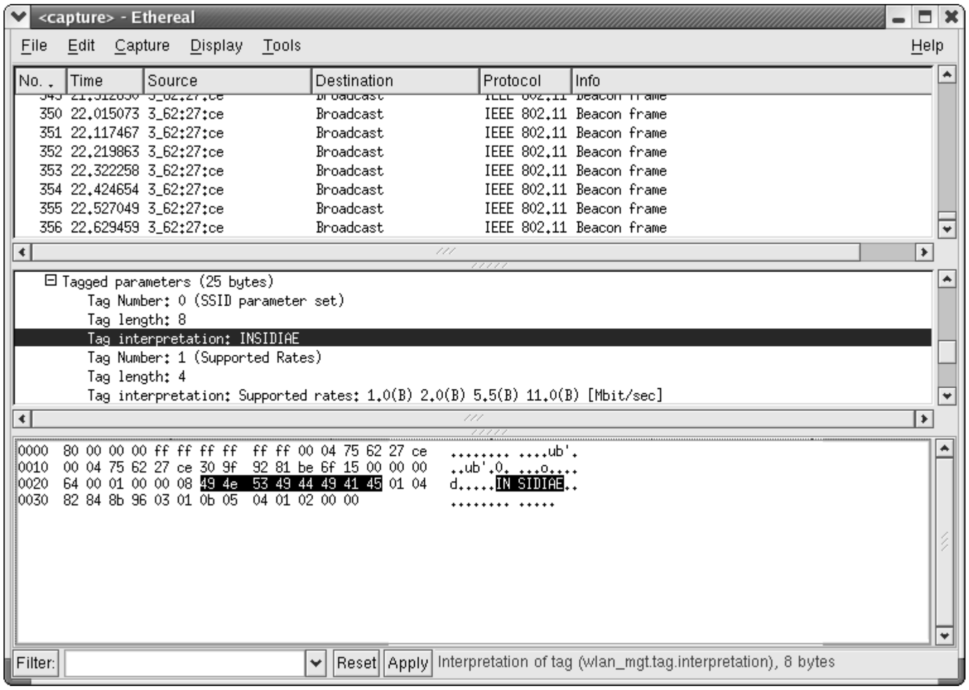
Capture limits
[ ] Stop capture after 1 packet(s) captured
[ ] Stop capture after 1 kilobyte(s) captured
[ ] Stop capture after 1 second(s)

Name resolution
[✓] Enable MAC name resolution
[ ] Enable network name resolution
[ ] Enable transport name resolution

The Ethereal interface is divided into three panes, as shown in Figure 10-6. The top window is the packet list pane and provides a summarized list of the packets captured. The middle pane shows a detailed breakdown of the packet selected in the packet list, and the bottom pane is a raw hex and ASCII dump called the data view pane.

You've probably used Ethereal to view packets on Ethernet networks before. Using it on 802.11 networks is similar, but you are given some new options to the existing Ethereal filtering rules using the wlan category.

Consult the Ethereal documentation for a complete listing of the wlan filter subcategories.



```
No.  Time      Source      Destination  Protocol     Info
350 22.015073 3_62:27:ce   Broadcast    IEEE 802.11  Beacon frame
351 22.117467 3_62:27:ce   Broadcast    IEEE 802.11  Beacon frame
352 22.219863 3_62:27:ce   Broadcast    IEEE 802.11  Beacon frame
353 22.322258 3_62:27:ce   Broadcast    IEEE 802.11  Beacon frame
354 22.424654 3_62:27:ce   Broadcast    IEEE 802.11  Beacon frame
355 22.527049 3_62:27:ce   Broadcast    IEEE 802.11  Beacon frame
356 22.629459 3_62:27:ce   Broadcast    IEEE 802.11  Beacon frame

⊟ Tagged parameters (25 bytes)
      Tag Number: 0 (SSID parameter set)
      Tag length: 8
      Tag interpretation: INSIDIAE
      Tag Number: 1 (Supported Rates)
      Tag length: 4
      Tag interpretation: Supported rates: 1.0(B) 2.0(B) 5.5(B) 11.0(B) [Mbit/sec]

0000  80 00 00 00 ff ff ff ff  ff ff 00 04 75 62 27 ce   ........ ....ub'.
0010  00 04 75 62 27 ce 30 9f  92 81 be 6f 15 00 00 00   ..ub'.0. ...o....
0020  64 00 01 00 00 08 49 4e  53 49 44 49 41 45 01 04   d.....IN SIDIAE..
0030  82 84 8b 96 03 01 0b 05  04 01 02 00 00           ........ .....

Interpretation of tag (wlan_mgt.tag.interpretation), 8 bytes
```

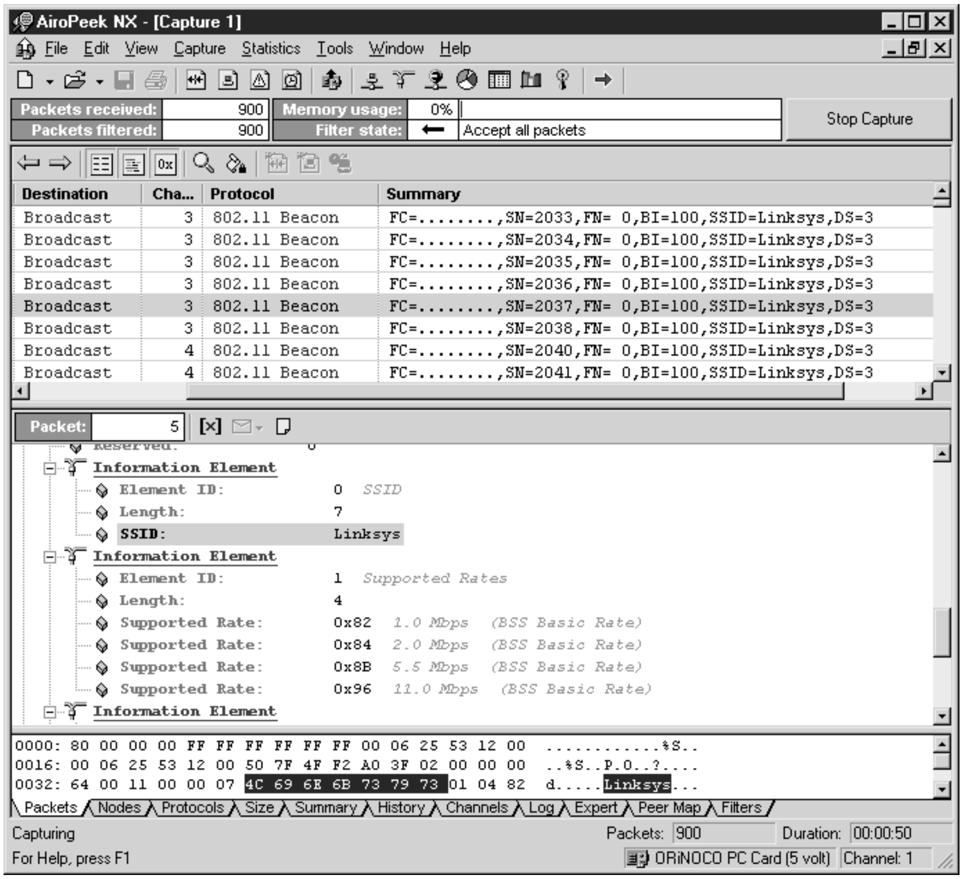
Figure 10-6. The three panes of the Ethereal interface

AiroPeek NX

Popularity:	*4*
Simplicity:	*8*
Impact:	*7*
Risk Rating:	***6***

AiroPeek NX (http://www.wildpackets.com) is a commercial 802.11 monitoring and analysis tool available for Windows 2000 and XP. A few other commercial solutions for 802.11 packet captures are available on Windows, but AiroPeek NX is the most usable and is priced the lowest. Unfortunately, there are no free tools available to perform packet capturing on Windows operating systems, so if you are stuck in Windows, using AiroPeek NX or another commercial product is your only option. AiroPeek supports Lucent and Cisco 802.11b cards and also has support for some of the newer 802.11a cards. AiroPeek NX is primarily designed for wireless network troubleshooting and analysis, but it does have some security friendly options as well (see Figure 10-7).

AiroPeek NX supports channel scanning at a user-defined interval as well as decrypting traffic on the fly with a provided WEP key. AiroPeek NX's filtering is also very easy to configure, and you can save off filter combinations to template files. This gives you the ability to quickly switch between filter groups you may use for network discovery and other groups you may use for in-depth analysis. AiroPeek NX also provides a useful Nodes view, which groups detected stations by their MAC address and will also show IP addresses and protocols observed for each. The Peer Map view presents a matrix of all


AiroPeek NX - [Capture 1]
File Edit View Capture Statistics Tools Window Help
Stop Capture
Packets Nodes Protocols Size Summary History Channels Log Expert Peer Map Filters

Packets received: 900 Memory usage: 0%
Packets filtered: 900 Filter state: Accept all packets

Destination	Cha...	Protocol	Summary
Broadcast	3	802.11 Beacon	FC=........,SN=2033,FN= 0,BI=100,SSID=Linksys,DS=3
Broadcast	3	802.11 Beacon	FC=........,SN=2034,FN= 0,BI=100,SSID=Linksys,DS=3
Broadcast	3	802.11 Beacon	FC=........,SN=2035,FN= 0,BI=100,SSID=Linksys,DS=3
Broadcast	3	802.11 Beacon	FC=........,SN=2036,FN= 0,BI=100,SSID=Linksys,DS=3
Broadcast	3	802.11 Beacon	FC=........,SN=2037,FN= 0,BI=100,SSID=Linksys,DS=3
Broadcast	3	802.11 Beacon	FC=........,SN=2038,FN= 0,BI=100,SSID=Linksys,DS=3
Broadcast	4	802.11 Beacon	FC=........,SN=2040,FN= 0,BI=100,SSID=Linksys,DS=3
Broadcast	4	802.11 Beacon	FC=........,SN=2041,FN= 0,BI=100,SSID=Linksys,DS=3

Packet: 5

```
      Reserved:                 0
   Information Element
      Element ID:               0   SSID
      Length:                   7
      SSID:                     Linksys
   Information Element
      Element ID:               1   Supported Rates
      Length:                   4
      Supported Rate:           0x82  1.0 Mbps   (BSS Basic Rate)
      Supported Rate:           0x84  2.0 Mbps   (BSS Basic Rate)
      Supported Rate:           0x8B  5.5 Mbps   (BSS Basic Rate)
      Supported Rate:           0x96  11.0 Mbps  (BSS Basic Rate)
   Information Element
```

```
0000: 80 00 00 00 FF FF FF FF FF FF 00 06 25 53 12 00   ............%S..
0016: 00 06 25 53 12 00 50 7F 4F F2 A0 3F 02 00 00 00   ..%S..P.O..?....
0032: 64 00 11 00 00 07 4C 69 6E 6B 73 79 73 01 04 82   d.....Linksys...
```

Capturing   Packets: 900   Duration: 00:00:50
For Help, press F1   ORiNOCO PC Card (5 volt)   Channel: 1

Figure 10-7. The AiroPeek NX Packets tab

hosts discovered on the network by their connections to each other. This can make it very easy to visualize access point and client relationships.

Identifying Wireless Network Defenses and Countermeasures

Do not confuse this section with network hardening or a guide to locking down your access points. It is merely a section dedicated to identifying any implemented WLAN countermeasures and potentially leveraging those defenses. Just as with any other network or system target, it is imperative to determine the types of systems, where they are located, and their configurations. WLANs, APs, and wireless clients are no different.

The information presented will help you learn to identify systems and determine what type of security measures have been implemented. For instance, you will be able to quickly determine whether a system is without security and considered to be "Open System Authentication." You will also learn to determine the difference between a system with WEP implemented and the implemented bit-length for the shared secret via analysis of the 802.11 header and initialization vector. In addition to infrastructure-based controls, you will be able to determine whether common vendor-implemented security features such as MAC-based access control lists (ACLs) have been defined on the access points or protocol or firmware upgrades have been made to the WEP algorithm or 802.11b. Lastly, we will cover methods for leveraging multiple layers of encryption, such as embedded PKI schemas, gateway-based IPSec, and application-layer VPNs, including SSL tunnels.

There are a few prerequisites for this chapter if you want to get the most out of it. In addition to packet analysis (covered in the previous section), you should be able to understand the basics of encryption technologies and cryptography key management.

Here's a list of basic encryption technology resources:

- **http://www.crypto.com** Matt Blaze's cryptography resource page, an excellent source for research papers, cryptography algorithm analysis, and overall knowledge transfer.
- **http://developer.netscape.com/docs/manuals/security/pkin/contents.htm** Sun has provided a good resource as an introduction to public-key cryptography.
- **http://www-cs.engr.ccny.cuny.edu/~csmma/** An excellent academia resource provided by Professor Michael Anshel that has links to nearly all types of cryptography technologies.

SSID

The SSID is the first piece of information required to connect to a wireless network. 802.11 networks use the SSID to distinguish BSSs from each other. By itself the SSID is not intended to be used as a password or access control measure, but users are often led to believe this by vendors. Gathering the SSID is simple; all war-driving software shown earlier in the chapter will report a network's SSID or "network name." If the target access point responds to a Broadcast SSID Probe, then most wireless card drivers configured with an SSID of ANY will be able to associate with the wireless network. Having the SSID set to ANY usually makes the driver send a probe request to the broadcast address with a zero-length SSID. This, in turn, causes any access point that will respond to these requests (most do by default) to issue a response with its SSID and info. In the intended case, this makes it easier on the user because the user doesn't have to remember the SSID to connect to the wireless LAN—but, of course, it makes it much simpler for attackers to gather this data.

SSIDs can be found in a variety of 802.11 traffic:

- **Beacons** Beacons are sent continually by the access point and can be observed with a wireless sniffer. The Ethereal filter string to see only beacons is

  ```
  wlan.fc.type==0 and wlan.fc.subtype==8
  ```

  If you would like to filter out the beacon's frames (they are transmitted constantly and get in the way), just enclose the previous statement in `!()`, like so:

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

- **Probe Requests** Probe Requests are sent by client systems wishing to connect to the wireless network. If the client is configured with an SSID, it will be shown in the request. A Probe Request with a null SSID likely indicates a network name of "ANY" configured for the card.
- **Probe Responses** Probe Responses are sent in response to a Probe Request. The Probe Request can either have a blank SSID or the SSID of the network the client wishes to connect to.
- **Association and Reassociation Requests** The requests are made by the client when joining or rejoining the network. Reassociations requests are meant to support wireless clients roaming from access point to access point within the same ESS, but they can also be issued if the clients wander out of a given AP's range and then back in.

If the network you are monitoring has blocked the Broadcast Probe Responses or removed the SSID from beacon frames, you may need to wait until a client tries to reassociate to obtain the SSID. You can help this process along with the essid_jack tool from the Air-Jack toolkit (http://802.11ninja.net/). Essid_jack will send a deauthentication frame to the broadcast address that is spoofed to look like it's coming from the access point. This kicks all the active clients for the given channel off and causes them to try and reconnect to the WLAN. The client Probe Requests and AP Responses will contain the "hidden" SSID.

To use essid_jack, supply the BSSID address and channel of the wireless network you are trying to enumerate. By default, it will send the packet to the broadcast address affecting all active clients, but you can specify a single client MAC to target with the `-d` switch, as shown here:

```
[root@localhost tools]# ./essid_jack -b
00:40:96:54:1c:0b -d 00:02:2D:07:E2:E1 -c 11 -i aj0
Got it, the essid is (escape characters are c style):
"tsunami"
```

MAC Access Control

Although not defined in the 802.11 specification, most vendors have implemented MAC-level access controls to help beef up the inherently insecure nature of 802.11. When using MAC access control, the admin will define a list of "approved" client MAC addresses that are allowed to connect to the access point. While this may be feasible on small networks, it does require the administrator to track the MAC addresses of all wireless clients and can become a burden in larger installations. Besides the administrative overhead, the MAC address does not provide a good security mechanism because it is both easily observable and reproducible. Any of the station MACs can be observed with a wireless sniffer, and the attacker's MAC address can be changed easily in most cases. Therefore, the attacker simply needs to monitor the network, note the clients that are connecting successfully to the access point, and then change their MAC address to match one of the working clients.

Since it's not defined in the 802.11 spec, there is no packet flag that says "I'm using MAC ACLs," but you can usually figure this via deduction. If you have a correct SSID and WEP key but they still aren't able to associate, they may be using MAC filtering (or another scheme, such as 802.1x). AiroPeek NX has an easy way to see the relationships of systems on the wireless network. Its Peer Map, shown in Figure 10-8, will show each system and the other stations each system is in communication with. As shown, all the nodes are talking to the 00:07:0E:B9:94:32 station, so it is most likely the access point.

WEP

Most war-driving tools will indicate whether or not a network is using WEP encryption. NetStumbler will show a small padlock in the networks icon and indicate "WEP" under the encryption column when WEP encryption is found. Kismet will show a "Y" under the W (for WEP) column when it finds encrypted networks.

Wireless sniffers will show WEP status as well. Tcpdump will use the "PRIVACY" flag when WEP is in use and show the IV for each packet, when collected, as shown here:

```
00:30:36.943042 Beacon (Aironet_350) [1.0 2.0 5.5 11.0 Mbit] ESS CH: 6 , PRIVACY
00:30:36.948759 Data IV:1aa7f6 Pad 0 KeyID 0
00:30:36.949722 Data IV:1ba7f6 Pad 0 KeyID 0
00:30:36.958387 Data IV:1ba7f6 Pad 0 KeyID 0
00:30:36.959349 Data IV:1ca7f6 Pad 0 KeyID 0
00:30:36.968942 Data IV:1ca7f6 Pad 0 KeyID 0
00:30:36.970242 Data IV:1da7f6 Pad 0 KeyID 0
00:30:36.978462 Data IV:1da7f6 Pad 0 KeyID 0
00:30:36.979718 Data IV:1ea7f6 Pad 0 KeyID 0
00:30:36.988863 Data IV:1ea7f6 Pad 0 KeyID 0
00:30:36.990004 Data IV:1fa7f6 Pad 0 KeyID 0
00:30:36.998934 Data IV:1fa7f6 Pad 0 KeyID 0
00:30:37.000148 Data IV:20a7f6 Pad 0 KeyID 0
00:30:37.008549 Data IV:20a7f6 Pad 0 KeyID 0
00:30:37.009741 Data IV:21a7f6 Pad 0 KeyID 0
```

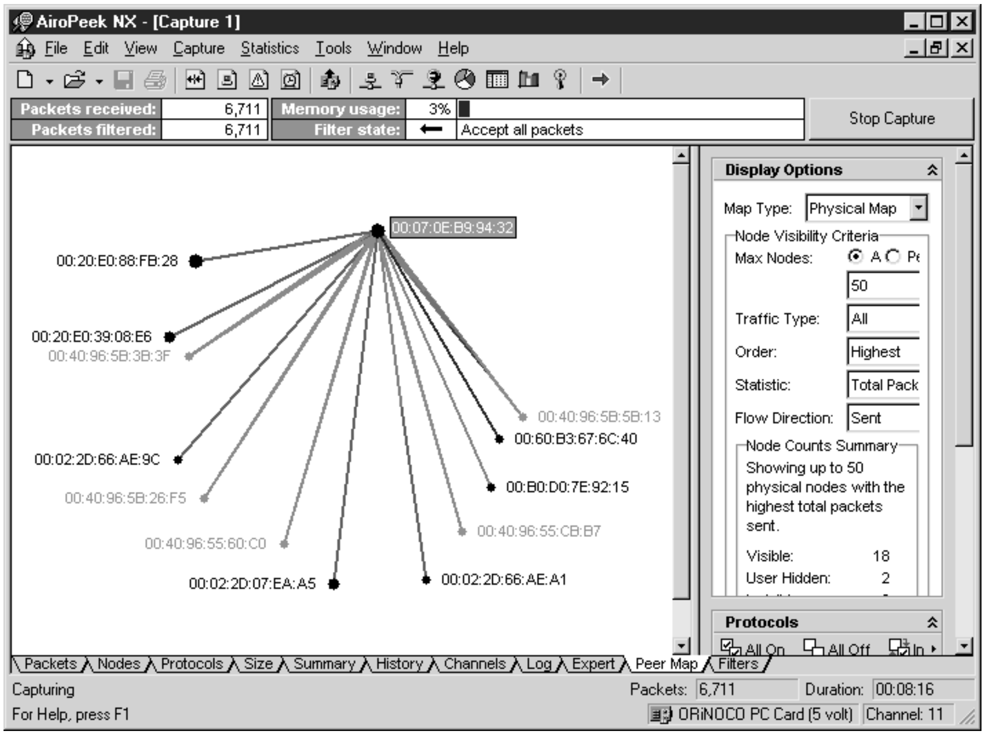


Figure 10-8. The Peer Map tab

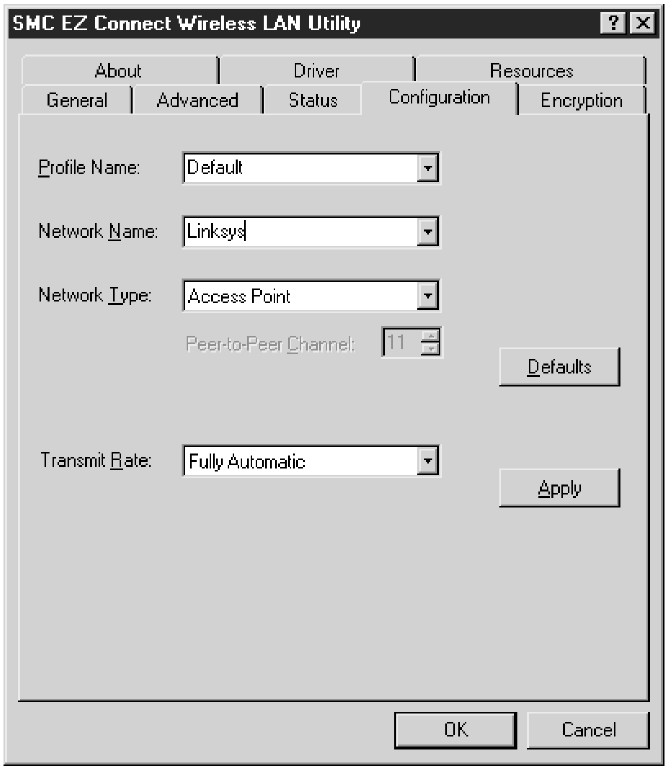
GAINING ACCESS (HACKING 802.11)

Following the proven *Hacking Exposed* attack methodology, "gaining access" is the stage of the assessment in which the attacker or auditor, depending on the situation, leverages the information gathered during the initial phases of the assessment. The goal for just about all system assessments or acquired targets is to gain administrator or root-level access to the system. However, for this to occur, the attacker must know certain types of detailed system, application, and configuration information.

In the realm of wireless and 802.11, gaining system access is significantly different when compared to "wired" systems. In most cases, this is due to a lack of strong WEP-enforced encryption, thereby allowing the attacker to crack weak keys and pertinent transmitted data. If the attacker has gained access to the AP's WEP key, the WLAN is all but penetrated. The small amount of communication information that is still required to effectively gain access should be considered ridiculously elementary when compared to the skill-set required to configure and utilize a wireless-cracking-capable system. As you will notice, a variety of methods is available to gain access to systems, covering a wide range of effort levels.

SSID

Once you have the SSID, you'll need to reconfigure your wireless interface to use it. On Windows operating systems, the card vendor will usually provide a utility to reconfigure card settings or an interface in the driver itself to reconfigure the SSID. Shown next is the configuration screen for an SMC wireless card and its driver settings. The network name has been changed to "Linksys," the SSID of the network we wish to connect to.



Profile Name: Default
Network Name: Linksys
Network Type: Access Point
Peer-to-Peer Channel: 11
Transmit Rate: Fully Automatic

For Linux, most drivers will support the iwconfig interface. Iwconfig is a wireless version of the ifconfig command used to configure basic 802.11 network parameters such as the SSID. To change the SSID with iwconfig, use the following command, where "tsunami" is the network name and "eth1" is the wireless interface:

```
[root@localhost root]# iwconfig eth1 essid tsunami
```

BSD systems such as OpenBSD and FreeBSD use the `wicontrol` command, which is used to change parameters of cards that use the wi (Wavelan) driver and handle the 802.11-specific network configuration parameters. To change the SSID using `wicontrol`, use the following example, where the interface we want to change is "wi0" and the target network name is "Lucent":

```
# wicontrol –I wi0 –n Lucent
```

MAC Access Control

Once you've gathered a list of usable MAC addresses, you will need to reconfigure your system to use a new MAC. For Windows systems, this may be driver dependent. Some older drivers allow you to reconfigure the MAC address in the interface properties, but many vendors have since disabled this capability. A few utilities are available to help with this problem—one of them is Bwmachak created by BlackWave. Bwmachak will change the MAC address of an Orinoco wireless card to one you specify. To use Bwmachak, remove the card first, then run Bwmachak as shown next (00:09:E8:B4CB:E8 is the MAC we want to use):

```
E:\>BWMACHAK.exe 0009E8B4CBE8
```

After the command has run, insert your card and run an `ipconfig /all` to verify the MAC address has changed.

Linux systems can use the `ifconfig` command to change the MAC. You'll need to bring down the interface first, then issue the new hardware Ethernet address, and finally bring the interface back up and check the results. Here is a sample command sequence to use. As you can see, the wireless interface is eth1, and the MAC we wish to use is 00:02:2D:07:E1:FF.

```
[root@localhost root]# ifconfig eth1 down
[root@localhost root]# ifconfig eth1 hw ether 00:02:2D:07:E1:FF
[root@localhost root]# ifconfig eth1 up
[root@localhost root]# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:02:2D:07:E1:FF
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:15 errors:2388 dropped:0 overruns:0 frame:2388
TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:720 (720.0 b)  TX bytes:3300 (3.2 Kb)
Interrupt:3 Base address:0x100
```

FreeBSD systems use the `ifconfig` command as well, but with a slightly different context. Bring down the interface before applying changes, just as in Linux, but omit the "hw" and colons in the address itself:

```
# ifconfig fxp0 ether 00022d07e1ff
```

Then bring the interface up and check it to make sure the changes have taken effect.

OpenBSD users can use the sea utility to change the MAC address because the supplied version of ifconfig does not support that capability. Sea does not have an official download location, so the easiest way to find it is with a Google search for "openbsd" and "sea.c". Sea's operation is very straightforward and works in the following manner. In this example, wi0 is the wireless interface and 00:02:2D:07:E1:FF is the MAC address we want to use:

```
# sea -v wi0 00:02:2D:07:E1:FF
```

WEP

Wired Equivalent Privacy (WEP) is a standard derived by the IEEE to provide an OSI Layer 2 protection schema for 802.11 wireless networks. The goal of WEP is not to completely secure the network but rather to protect the data from others passively and unknowingly eavesdropping on the WLAN. Many people mistake the WEP algorithm for a security solution that encompasses secure authentication and encryption, a goal that the 802.11 standard did not intend to address.

The WEP algorithm relies on a secret key that is shared between the AP and the client node, most commonly a wireless card on a laptop. WEP then uses that shared secret to encrypt all data between the nodes. The common misconception is that WEP provides network authentication via the use of a shared secret. If a WLAN is enforcing WEP, then any party that does not obtain that shared secret may not join that network. Therefore, the network is thought to be secure. The WEP algorithm does not encrypt the 802.11 header, nor does it encrypt the Initialization Vector (IV) or ID portions of the packet (see Figure 10-9).

RC4, a stream cipher encryption algorithm created by RSA, constantly encrypts the data between two nodes, thereby creating a fully encrypted virtual tunnel. In relation to its common use within the wireless arena, RC4 may utilize either a 64-bit or 128-bit shared secret key as the seed for the RC4 streams. One of the issues with the shared secret key is that 24 of the bits are directly derived from the unencrypted IV. As detailed hereafter, multiple attacks leverage the unencrypted IV field. The packet data is then encrypted with the secret key and appended with a packet checksum.

Attacks Against the WEP Algorithm

Several attacks on the WEP algorithm surfaced just shortly after its commercial introduction and implementation in wireless APs and client cards. The attacks range from passive to active, from dictionary based to key length, and one-to-one to man-in-the-middle. However, in general, most of the attacks work via brute-force techniques. Such techniques allow an attacker to test entire keysets, all the possibilities, looking for the single correct instance. The other category for attacking WEP is based on analysis of the IVs in correlation to the first RC4 output byte.

As mentioned previously, brute-force attacks are commonly used to exploit some of the key weaknesses within the WEP algorithm, particularly in determining the shared secret key. Passive attacks—that is, attacks that do not require you to send any

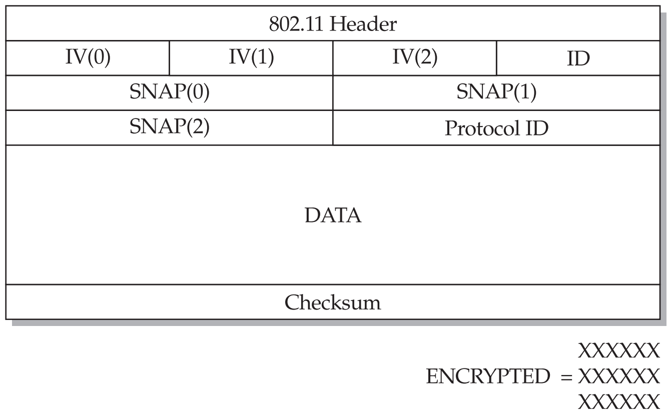


Figure 10-9. IEEE 802.11 packet structure

packets—allow you to sniff 802.11 packets and perform computations on those packets locally. The goal for this type of attack is not to knock other systems off the net or to forge packets to systems but rather to gather information about the network clients, the implemented security features, and the AP configuration, in addition to potentially cracking the WEP key. Through traffic analysis, you can potentially determine the services running, the encryption and authentication methods, whether a MAC-based authentication schema is implemented, and what the size of the key is in bits.

The only passive attacks that target the WEP algorithm are key and packet cracking. The attack starts by sniffing a large number of packets from potentially numerous clients (the more packets, the more likely the attack will be successful). Because the IV is in cleartext, you can do packet analysis based on client and corresponding IV. Once you have two packets that use the same IV, you can XOR the packets and obtain the one XOR of the packets. This can be used to infer information about the packets and further eliminate possibilities within the keyspace for brute-force attacks on the message. Once the XOR, encrypted text, and unencrypted text of a packets is determined, it's trivial to determine the shared secret because the shared secret was used to create the XOR.

The other type of attack is simply brute-forcing the shared secret key. You can attempt to decrypt the message in the same fashion that an AP would, verifying success via the checksum. By taking advantage of the IV weaknesses, you can execute dictionary attacks on WEP checks in minutes or sometimes seconds, depending on the wordlist and CPU speed. An entire 40-bit keyspace brute-force attack only takes about a few weeks when running on a single system.

Almost all the active attacks against the WEP algorithm are not focused on determining the shared secret key. The active attacks focus on injecting packets into current 802.11 streams. However, in all cases, you must first know the MAC of the AP and whether WEP is enforced, as well as the bit-strength and key if it is implemented. Now that you understand what you need, if WEP is disabled, the effort to use a packet-injection technique is insignificant. In either case, you would just forge the packet you want to write to the "wire" and send it off. The tools that use some of these techniques include Air-Jack and Libradiate (discussed later).

Securing WEP

Multiple vendors, including Cisco, Orinoco, and Intel, have developed more secure implementations of the WEP algorithm, key scheduling, or product firmware. WEP-Plus was developed as a firmware upgrade for APs and wireless cards to modify the current IV-creation algorithm within WEP. WEP-Plus uses a more secure algorithm for determining and masking the IV field. PKI vendors such as Baltimore Technologies and Entrust have leveraged their PKI and VPN technologies to be compatible within the realm of wireless. In this case, wireless clients would have to authenticate to the network via a certificate server. If authentication succeeds, the user is allowed to join the network via an encrypted VPN tunnel. This type of security is not much different from a few of the smaller vendors pushing their wireless VPN solutions utilizing an SSL or IPSec tunnel on top of WEP. Granted, the data may be secured because of the additional application- and transport-layer encryption, but the actual wireless sessions are still insecure. The IVs are still in cleartext and not properly randomized with this solution.

WEP has inherent security issues within the protocol, implementation, and overall vendor and consumer usage. Unfortunately, 802.11 offers great functionality because it allows people to work without wires, so wireless technology will never go away. The defensive solution is to layer security with multiple encryption and authentication schemas and to only use vendors that have addressed the IV and weak KSA WEP issue.

TOOLS THAT EXPLOIT WEP WEAKNESSES

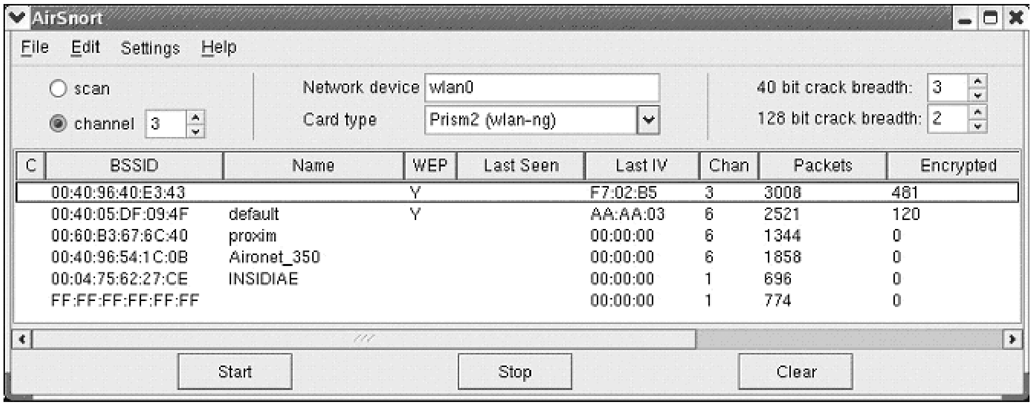
There are a few tools available that automate or aid in the automation of exploiting WEP weaknesses. In most cases the tools use a combination of packet capturing and packet cracking techniques to leverage these weaknesses.

AirSnort

Popularity:	*8*
Simplicity:	*7*
Impact:	*9*
Risk Rating:	*8*

The AirSnort (http://airsnort.shmoo.com/) tool is a collection of the scripts and programs derived from the research conducted by Tim Newsham, the University of Maryland, and the University of California at Berkley. It is by far the most popular and best-known Linux tool in the industry specifically used for wireless packet cracking. Originally, it was a command-line Linux-based tool that merely captured 802.11b wireless packets and attempted to crack the packets via the weak IV flaw. It has since evolved to include a GUI, allowing for the quick configuration of the channel to scan and the ability to specify the strength of the WEP key.

To use AirSnort, you must first compile and install the source code. At the time of this release, the common `./configure`, `make`, `make install` worked for AirSnort installation. Then you just execute AirSnort from the command line, and as long as you are in an X-Windows session, you will be able to use the GUI. In this case, you would first want to run AirSnort in a scanning mode to determine what APs are in range and if any traffic is being transmitted over the wire. As you can see in the following illustration, AirSnort has identified six APs, two of which have implemented WEP functionality. Differentiating numbers of packets must be captured for different attacks to work, but the AirSnort GUI simplifies that process by adding the meaningful buttons Start and Stop for your convenience.



○ scan [✓ channel] 3

Network device: wlan0

Card type: Prism2 (wlan-ng)

40 bit crack breadth: 3

128 bit crack breadth: 2

C	BSSID	Name	WEP	Last Seen	Last IV	Chan	Packets	Encrypted
	00:40:96:40:E3:43		Y		F7:02:B5	3	3008	481
	00:40:05:DF:09:4F	default	Y		AA:AA:03	6	2521	120
	00:60:B3:67:6C:40	proxim			00:00:00	6	1344	0
	00:40:96:54:1C:0B	Aironet_350			00:00:00	6	1858	0
	00:04:75:62:27:CE	INSIDIAE			00:00:00	1	696	0
	FF:FF:FF:FF:FF:FF				00:00:00	1	774	0

AirSnort Countermeasures

Currently, the countermeasures for all WLAN packet sniffers and crackers are rather simplistic. First, it is pertinent that you implement WEP on all your APs with the 128-bit key strength. When selecting a WEP key, it is critical that you select a secret key not found in a dictionary, one that contains a mix of numeric, alphabetic, and special characters, if possible. Also, a WEP key over eight characters in length is ideal because it increases the time required by magnitudes to brute-force the keyspace over a six-character passphrase. The SSID for your AP should be changed from the default setting, and if the vendor provides any type of fix for the WEP algorithm, such as WEP-Plus, then it should be implemented. The last recommendation is to change your WEP key as often as possible. Remember that anyone within range has access to your data transmitting through your 802.11 network. Therefore, protecting that data should be a multilayer, constant process.

WLAN-Tools

Popularity:	*10*
Simplicity:	*8*
Impact:	*9*
Risk Rating:	***9***

The WLAN-Tools (or, as it should be named, the Godfather of Wireless Cracking) was created by Tim Newsham (http://www.lava.net/~newsham/wlan/). It was the original posting of coded exploits for utilizing the weaknesses within the WEP algorithm. Programmed to work in the Linux environment, WLAN-Tools, if properly modified, will also work on many flavors of UNIX, including BSD and Solaris. The toolset includes programs for 802.11 packet capturing and WEP-encrypted packet cracking. The toolkit is an excellent resource for learning the coding aspect behind the vulnerabilities, and it also contains patches for the sniffer drivers. We thought it necessary to inform you of this toolset because it was the original exploiter, but due to user interface and program robustness, we believe it to be outdated. Our recommendation is to use the DWEPUtils from Dachb0den Labs, if possible, or AirSnort.

WLAN-Tools Countermeasures

Refer to the AirSnort countermeasures recommendation, earlier in the chapter, for details on mitigating some of the risk associated with your WLAN.

DWEPCrack

Popularity:	*5*
Simplicity:	*4*
Impact:	*9*
Risk Rating:	***6***

DWEPCrack, written by Dachb0den Labs (http://www.dachb0den.com/projects/dweputils.html), is a tool specifically used to crack WEP-encrypted packets via the BSD platform. Dachb0den Labs prides itself as a security coalition dedicated to security and wireless research and is located in Southern California. The Dachb0den toolkit is into specific functions, thereby allowing each one to be used individually or scripted to work together. It is by far the most comprehensive toolkit available for exploiting numerous weaknesses within the WEP algorithm. In addition, the toolkit allows an attacker to exploit other infrastructure-based weaknesses, such as MAC-based access control lists, with a brute-force algorithm that attempts to brute-force the keyspace of the MAC address in aspirations of unauthorized AP association. DWEPCrack allows you to specify a dictionary list for brute-forcing the WEP key, in addition to the option of brute-forcing the entire keyspace until the proper key is found. Realize that if the AP is using a 128-bit WEP key, it is quite possible that the key will be changed before you come across it. If you want detailed information on cracking or encryption, refer to the WEP section or Google.com.

DWEPCrack parses through the log, determining the number of packets, unique IVs, and corresponding cipher keys used to XOR the payload of the packet. When it determines whether the proper prerequisites exist for attempting a WEP attack, it attempts to brute-force and output the WEP key. Here is what you might expect to see when you execute DWEPCrack from the command line when you provide it a WEP-encrypted log of packets:

```
cloud@gabriel ~$ dwepcrack -w ~/sniffed_wlan_log

* dwepcrack v0.4 by h1kari <h1kari@dachb0den.com> *
* Copyright (c) Dachb0den Labs 2002 [ht*p://dachb0den.com] *

reading in captured ivs, snap headers, and samples... done
total packets: 723092

calculating ksa probabilities...
 0: 88/654 keys (!)
 1: 2850/80900 keys (!)
 2: 5079/187230 keys (!)
 3: 5428/130824 keys (!)
 4: 14002/420103 keys (!)

(!) insufficient ivs, must have > 60 for each key (!)
(!) probability of success for each key with (!) < 0.5 (!)

warming up the grinder...
 packet length: 48
 init ventor: 58:f4:24
 default tx key: 0

progress: .....................................

wep keys successfully cracked!
0: XX:XX:XX:XX:XX *
done.

cloud@gabriel ~$
```

DWEPCrack Countermeasures

Refer to the AirSnort countermeasures recommendation, earlier in the chapter, for details on mitigating some of the risks associated with your WLAN.

DENIAL OF SERVICE (DOS) ATTACKS

802.11 wireless networks can face denial of service attacks using the 802.11 protocol itself and from interference in the S-Band ISM frequency range. The ISM (Industrial Scientific and Medical) range is set aside by the FCC for use by unlicensed devices. This means that if you wish to create an RF system that uses an ISM band, you will not have to pay licensing fees to the FCC to use it, although you will still need to register the device. 802.11, 11b, and 11g all use the 2.4–2.5GHz ISM band, which is extremely crowded at the moment. Cordless phones, baby monitors, X10 cameras, and a host of other devices operate in this band and can cause packet loss or outright disruption of service in 802.11 networks.

802.11's other inherent problem is that the management frames that control client-connection operations are completed unauthenticated and subject to trivial spoofing. Essentially, an attacker can forge a packet so that it appears as if it originates at the access point to all the clients on the network. This packet tells these clients to disconnect. There is nothing that can be done to prevent this if someone wants to execute the attack against your network. The wlan_jack tool that implements this attack is included with the Air-Jack suite. To use it, you'll need to specify the access point MAC, channel, and target MAC address to send the attack to. The default destination is the broadcast address, which means it will be sent to all clients. You can, however, selectively kill one client connection by specifying that station's MAC address only. In the following example, the target MAC we wish to deauthenticate (and therefore keep off the network) is 00:09:E8:B4:CB:E8, and the access point's MAC is 00:07:0E:B9:94:32:

```
[cloud@gabriel tools]# ./wlan_jack -b 00:07:0E:B9:94:32 -v 00:09:E8:
B4:CB:E8 -c 6 -i aj0
Wlan-Jack: 802.11 DOS utility

Jacking Wlan...
```

Wlan_jack operates continually until it is spotted, so it could keep the station off the network indefinitely.

AN 802.1X OVERVIEW

The IEEE 802.11a and 802.11b standards have taken a substantial beating from the media, the commercial product sector, and most of all the information security community for their lack of adequate specifications for protocol-based security. Different efforts have been exhausted in the realms of security being layered on top of 802.11 and vendor-applied firmware upgrades, even to the extent that some vendors are now considering migrating to a Bluetooth-based infrastructure for their wireless solutions. In hopes of addressing the security concerns and risks associated with the current 802.11 infrastructure, the IEEE, in coordination with commercial and educational advocates, designed the 802.1x protocol.

The high-level design goals for 802.1x were simple. The specification provides for an expandable infrastructure that consistently allows for and provides additional clients and APs to be added with minimal technological effort. In addition to the infrastructure goals, security goals were addressed, including authentication and encryption. It was noted that some mechanism for continuous node encryption utilizing multiple secret keys beyond the means of WEP should be implemented. Lastly, dual-mode authentication needed to be addressed. Currently, nodes authenticate via a client-to-server handshake, instead of having a client-to-server, server-to-client schema.

In general, two main issues exist within the proposed 802.1x and 802.11 framework integration plans. The current 802.1x specification does not protect against man-in-the-middle attacks, nor does it address attacks on session-based hijacking. Man-in-the-middle attacks focus on redirecting traffic from a client node to the AP, thereby allowing the hijacker to view all data being transmitted to and from that node to the AP. This kind of attack is successful due to a lack of authentication made by the AP to the client, thereby inherently placing an amount of trust in the client-to-server authentication. For example, there is no current method in the 802.1x specification that allows the client to be certain that it is authenticating to the proper AP. The other attack, session-based hijacking, is successful because of the lack of message confidentiality and low-layer authentication. An attacker could disassociate a legitimate user and then spoof that user's identity to continue the communication session without any notice from the AP. Tools such as Air-Jack and Libradiate can aid in attacks of this sort.

Unfortunately, the solution is not a simple one; hence, it cannot be solved with simply an additional authentication schema, nor can it be solved by creating a secure method for continuous key scheduling. The powers that be need to go back to the design table and create a robust and secure protocol for communicating over networks, specifically wireless networks, without losing the desired functionality.

Detailed research and information pertaining to 802.1x and 802.1x security research can be ascertained at the following websites:

- ▼ **www.cs.umd.edu/~waa/1x.pdf** The University of Maryland's publication for its research into the current IEEE 802.1x protocol standard
- ■ **http://grouper.ieee.org/groups/802/11/index.html** The IEEE 802.11 communication protocol specification
- ▲ **http://www.ieee802.org/1/pages/802.1x.html** The IEEE 802.1x communication protocol specification

SUMMARY

Wireless gateways and multilayered encryption schemas have proved to be the best defenses for the plethora of tools currently floating around the Internet for attacking 802.11 WLANs. Ironically, wireless technology appears to be vastly different from other communication mediums; however, the industry model for layering security via multiple authentication and encryption schemas holds true. Here is a selection of excellent Internet-based resources if you choose to do more research into wireless technology:

- **http://standards.ieee.org/getieee802/** The IEEE designs and publishes the standard for 802.11 wireless transceivers, band usage (in cooperation with the FCC), and general protocol specifications.
- **http://bwrc.eecs.berkeley.edu/** The Berkeley Wireless Research Center (BWRC) is an excellent source for additional information on future communication devices and wireless technologies, especially those devices with high-integrated CMOS implementations and low power consumption.
- **www.hyperlinktech.com** Hyperlink distributes wireless equipment from a wide variety of manufacturers, in addition to its own line of 2.4GHz amplifiers that can be used for long-range transmitting or cracking.
- **www.wirelesscentral.net** Wireless Central is a product vendor and distributor with a good reputation in the war-driving community and even offers its own war-driving bundles for purchase.
- **www.drizzle.com/~aboba/IEEE** The Unofficial 802.11 Security page has links to most of the 802.11 security papers as well as many general 802.11 links.
- **www.cs.umd.edu/~waa/wireless.html** The University of Maryland wireless research page is another excellent source for academic research and technology reports. In addition to the university's own research, the site has links to other good papers and research on wireless technology.
- **http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.htm**l Hewlett Packard sponsors this page full of Linux wireless tools and research reports. It is an excellent source for all things Linux.