;login:
THE MAGAZINE OF USENIX & SAGE
August 2002 volume 27 • number 4

inside:

SECURITY
Foust: Identifying and Tracking Unauthorized 802.11 Cards and Access Points

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

identifying and tracking unauthorized 802.11 cards and access points

by Robert Foust

Robert Foust has been experimenting with Linux since version 0.95. He is currently employed as a system administrator where he enjoys working in a heavily mixed hardware and software environment.

rfoust@interlinknetworks.com

A Practical Approach

Introduction

The object of this paper is simple: to determine whether an individual, using inexpensive off-the-shelf components and free software, could detect when an unauthorized 802.11 card or access point was powered up and began broadcasting within range of a local WLAN. Furthermore, could the event be tracked, activity monitored, and the offending card or access point physically located?

Tracking unauthorized external accesses, is considered by many to be the biggest concern. Midnight parking-lot attacks are a real, tangible threat to many people, and especially frightening because the attacker could do almost anything. It's the unknown and uncontrollable risk that frightens many security professionals.

Realistically, though, the biggest threat in most organizations may come from inside. Through maliciousness or plain old carelessness, an internal employee has the ability to inflict major damage on a local network or open up large, gaping holes in the local network.

For example, many users would think nothing of starting up Microsoft's Internet Information Server (IIS) on their Windows laptop, but most security professionals would shudder at the thought of an untracked, unpatched IIS server running open to the world. The user may bring their machine home at night, connect it to their ISP, get infected with a worm or virus, and the next day, bring that machine on the local network to wreak all kinds of havoc.

The 802.11 specification opens up a more interesting and far more dangerous possibility: a power user could simply bring an access point to work because they want the convenience of a wireless network without the bother of the IT department's delays in deployment. Being a power user, they know that they can simply assign the access point an address via DHCP, plug their own wireless card into their laptop, and then walk around the office with their laptop. With proxying and NAT software, this kind of activity might even go totally unnoticed by security personnel or automated intrusion detection systems. Little does this user know that the IT department's concerns are well founded and that the user has unwittingly opened a gaping hole in the local network, such that any drive-by attacker could simply hop on the local network and do anything they wish. To address these concerns, we assembled the readily available necessary pieces into a usable detection and tracking tool.

Hardware and Software Used

All software used is freely available and can be downloaded and used by anyone.

- Generic x86-based PC laptop.
- Cisco Aironet 340 802.11 card.
- Cisco Aironet 340 access point, used as an experimental rogue access point.
- Lucent Orinoco "Silver" 802.11 card, used as an experimental unauthorized wireless card.

- SuSE Linux 7.1.
- Development version of libpcap (libpcap current-cvs-2001.07.20), a sniffing library used under Linux. The latest development version was used because it understands the 802.11 frame format (obtained from *http://www.tcpdump.org/*).
- Development version of ethereal (v0.8.19), a very complete sniffer and protocol analysis tool. This was used because it was one of the only tools available which could dissect 802.11 frames in a human-readable format (obtained from *http://www.ethereal.com/*).
- Linux kernel 2.4.9. This was used because it had an updated driver for the Cisco Aironet 340 wireless Ethernet card. There had been numerous changes to the driver since 2.4.0 (one of the standard SuSE 7.1 kernels), and we wanted to be sure the driver supported the 802.11 frame format, offered complete reporting, supported RF Monitor mode (which allowed "promiscuous" recording of 802.11 packets), and supported the iwspy ioctls, which were necessary to gather signal strength statistics on arbitrary wireless clients and access points. The Aironet driver included with the 2.4.9 kernel supported all of these.
- Octave v2.0.16, a Matlab-like mathematics package. This was used to solve some relatively complex simultaneous equations encountered while trying to track down the physical location of unauthorized cards/access points. This package was included with *SuSE 7.1 but can also be downloaded from Octave's Web site, *http://www.octave.org/*.
- Gnuplot, latest CVS snapshot as of September 7, 2001. Gnuplot was used to find a best-fit curve instead of Octave. Octave did not have any pre-canned best-fit functions, while Gnuplot did, and a best-fit curve ended up yielding better results than finding the simultaneous solution of three equations with three data points (the first attempt).
- The GIMP, for modifying and manipulating various graphics.
- Emacs, Xfig, TeX, LaTeX, Ghostview, xv.

Note: We would *very* much liked to have used snort, a free IDS. This tool is quite remarkable and extensible, but we couldn't find any easy way within the scope of this project to define or act on events that happened at the 802.11 frame level, which is essential for detecting unauthorized cards or access points. Perhaps in future versions snort will offer a combination of robust high-level intrusion detection along with the complete protocol analysis tools offered by ethereal. For this project, the command-line version of ethereal was used to track unauthorized wireless clients.

Background

The 802.11 specification refers collectively to a family of IEEE standards for a new wireless networking protocol designed to be compatible with previous 802 specifications; it covers everything from the physical medium (microwave radio and IR) and modulation techniques to the link-layer protocol. The 802.11 specification can be obtained from the IEEE Web site, *http://www.ieee.org/*. As implemented by most vendors, 802.11 operates over microwave frequencies. In North America, it operates in the 2412 to 2462MHz range over 11 channels. When used over radio links, 802.11 makes use of a technology called "spread spectrum," summarized in an article by Schilling, Pickholtz , and Milstein:

> Spread-spectrum radio communications, long a favorite technology of the military because it resists jamming and is hard for an enemy to intercept, is now on the verge of potentially explosive commercial development. The reason: spread-spec-

trum signals, which are distributed over a wide range of frequencies and then collected onto their original frequency at the receiver, are so inconspicuous as to be "transparent." Just as they are unlikely to be intercepted by a military opponent, so are they unlikely to interfere with other signals intended for business and consumer users – even ones transmitted on the same frequencies. Such an advantage opens up crowded frequency spectra to vastly expanded use.

In a nutshell, spread spectrum makes it difficult to distinguish legitimate signals from normal background noise and makes 802.11 less susceptible to noise from the environment. This is especially important considering that the microwave frequencies used by 802.11 (~2.4GHz) are unlicensed, and so are susceptible to interference from many sources, most notably microwave ovens.

Unfortunately, spread spectrum turns out not to offer much in the way of security of communications, since the spreading algorithm used by 802.11 is well known and implemented in every 802.11-capable card. However, spread spectrum still complicates efforts to locate the source of unauthorized wireless cards and access points, since normal fixed-frequency receivers may not be able to easily distinguish 802.11 transmissions from normal background noise. Using a traditional microwave receiver and directional antenna to locate the source of unauthorized transmissions would probably be difficult.

Detection

The first goal is detection. Can we tell when someone powers on a card within range of the local network? This can be done with off-the-shelf components and free software. The Cisco Aironet driver included with the more recent Linux kernels supports "RF Monitor" mode, which permits promiscuous monitoring of 802.11 packets – specifically, monitoring raw 802.11 frames to detect if there are any telltale frames broadcast by a rogue access point or card.

As outlined in the original 802.11 specification, there are three classes of 802.11 frames. With the goal of detecting rogue access points and unauthorized wireless Ethernet cards, we are primarily interested in class 1 and 2 frames. Class 1 frames are the only frames allowed in state 1, the unauthenticated state, and are largely management frames used for authentication, beacons, and probe requests. Class 2 frames are allowed in both states 1 and 2 and are used for association and re-association. From access points, we would expect to see a large number of beacon frames (class 1). From unassociated ad hoc clients scanning in active mode, we would expect to see a large number of probe requests (also class 1). To test this hypothesis, a method of monitoring all 802.11 management frames is needed, which the Cisco card and Linux driver are capable of in RF Monitor mode.

SETUP

To put the card into RF Monitor mode, use any BSS (use `Mode: r` for plain RF Monitor mode):

```
# echo "Mode: y" >   /proc/driver/aironet/eth0/Config
#
```

Then, start logging packets with tcpdump, saving them to a file for later analysis with ethereal:

```
# tcpdump -i eth0 -s 0 -w capturefile
#
```

UNAUTHORIZED AD HOC NETWORK

The first test was to confirm the ability to detect a WLAN card being powered on. A Lucent Orinoco card was configured in ad hoc mode on a Win2K laptop, and turned on to find out if there were any characteristic frames sent out by the Orinoco card when it was put into ad hoc mode.

After the card initialized, tcpdump was stopped, ethereal started, and the capture file opened. A large number of probe requests from the Orinoco card were found, confirming that it was indeed possible to detect when someone within close range had powered up a wireless Ethernet card in ad hoc mode.

The dissected frame was as follows:

```
IEEE 802.11
    Type/Subtype: Probe Request (4)
    Frame Control: 0x0040
      Version: 0
      Type: Management frame (0)
      Subtype: 4
      Flags: 0x0
        DS status: Not leaving DS or network is operating in AD HOC mode(To DS: 0 F...)
        .... .0.. = Fragments: No fragments
        .... 0... = Retry: Frame is not being retransmitted
        ...0 .... = PWR MGT: STA will stay up
        ..0. .... = More Data: No data buffered
        .0.. .... = WEP flag: WEP is disabled
        0... .... = Order flag: Not strictly ordered
    Duration: 0
    Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source address: 00:02:2d:1b:51:ca (Agere_1b:51:ca)
    BSS Id: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Fragment number: 0
    Sequence number: 118
IEEE 802.11 wireless LAN management frame
    Tagged parameters (19 bytes)
      Tag Number: 0 (SSID parameter set)
      Tag length: 15
      Tag interpretation: roguepeertopeer
      Tag Number: 1 (Supported Rates)
      Tag length: 4
      Tag interpretation: Supported rates: 1.0 2.0 5.5 11.0 [Mbit/sec]
0000  40 00 00 00 ff ff ff ff ff ff 00 02 2d 1b 51 ca @...........-.Q.
0010  ff ff ff ff ff ff 60 07 00 0f 72 6f 67 75 65 70 ......`...roguep
0020  65 65 72 74 6f 70 65 65 72 01 04 02 04 0b 16    eertopeer......
```

Indeed, it is possible to tell if someone starts an actively scanning card in ad hoc mode, and quite a bit of useful information can be gleaned from a single frame. Most relevant are the SSID and the MAC address, since they can be used to track down a particular card and/or person.

UNAUTHORIZED ACCESS POINT

The next test was to confirm the possibility of detecting a rogue access point. A tcpdump session was started, and then a Cisco Aironet 340 access point was turned on. After the access point had finished booting, the dump was examined with ethereal, and a large number of beacon frames sent out by the access point were found. The following is one such frame, again dissected by ethereal:

```
IEEE 802.11
    Type/Subtype: Beacon frame (8)
    Frame Control: 0x0080
      Version: 0
      Type: Management frame (0)
      Subtype: 8
      Flags: 0x0
        DS status: Not leaving DS or network is operating in AD HOC mode (To DS: 0  From DS: 0) (0x00)
        .... .0..   = Fragments: No fragments
        .... 0...   = Retry: Frame is not being retransmitted
        ...0 ....   = PWR MGT: STA will stay up
        ..0. ....   = More Data: No data buffered
        .0.. .... = WEP flag: WEP is disabled
        0... ....   = Order flag: Not strictly ordered
    Duration: 0
    Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source address: 00:40:96:36:88:23 (Telesyst_36:88:23)
    BSS Id: 00:40:96:36:88:23 (Telesyst_36:88:23)
    Fragment number: 0
    Sequence number: 0
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
      Timestamp: 0x0000000000019274
      Beacon Interval: 0.102400 [Seconds]
      Capability Information: 0x0021
        .... ...1   = ESS capabilities: Transmitter is an AP
        .... ..0.   = IBSS status: Transmitter belongs to a BSS
        ...0 ....   = Privacy: AP/STA cannot support WEP
        ..1. ....   = Short Preamble: Short preamble allowed
        .0.. ....   = PBCC: PBCC modulation not allowed
        0... ....   = Channel Agility: Channel agility not in use
        CFP participation capabilities: No point coordinator at AP (0x0000)
    Tagged parameters (31 bytes)
      Tag Number: 0 (SSID parameter set)
      Tag length: 18
      Tag interpretation:
      Tag Number: 1 (Supported Rates)
      Tag length: 4
      Tag interpretation: Supported rates: 1.0(B) 2.0(B) 5.5 11.0 [Mbit/sec]
      Tag Number: 3 (DS Parameter set)
      Tag length: 1
      Tag interpretation: Current Channel: 11
      Tag Number: 5 ((TIM) Traffic Indication Map)
      Tag length: 4
      Tag interpretation: DTIM count 1, DTIM period 2, Bitmap control 0x0,
                          (Bitmap suppressed)
0000  80 00 00 00 ff ff ff ff ff ff 00 40 96 36 88 23   ...........@.6.#
0010  00 40 96 36 88 23 00 00 74 92 01 00 00 00 00 00   .@.6.#..t.......
0020  64 00 21 00 00 12 00 00 00 00 00 00 00 00 00 00   d.!.............
0030  00 00 00 00 00 00 00 00 01 04 82 84 0b 16 03 01   ................
0040  0b 05 04 01 02 00 00                              .......
```

UNAUTHORIZED CLIENT

The final condition tested for was unauthorized clients. The first scenario considered (the more likely scenario) is that someone brings a foreign card and powers it up with the wrong SSID. If the card was actively scanning, probe requests would be seen from this card as it attempted to find an access point. The second scenario is that someone brings a foreign card and powers it up with the correct SSID. This one turns out to be a little more problematic to detect, in that there will be only a few 802.11 management frames to trigger an alert, and then more "normal" traffic. This is problematic primarily because of the way RFMON_ANYBSS mode on the Cisco card works – despite its name, the card cannot receive packets simultaneously from all BSSes in range, espe-

cially if those BSSes use different frequencies. The consequence is that it takes some manual intervention to sniff traffic from a particular BSS – see "Problems and Complications," below, for more details on this problem and how to work around it. This problem was ignored and instead the focus was on the few 802.11 management frames that do show up readily in the sniffer; both scenarios turned out to produce similar probe requests, so both scenarios are treated as identical.

The dissected probe request sent out by this card:

```
IEEE 802.11
    Type/Subtype: Probe Request (4)
    Frame Control: 0x0040
        Version: 0
        Type: Management frame (0)
        Subtype: 4
        Flags: 0x0
            DS status: Not leaving DS or network is operating in AD HOC mode (To DS: 0  From DS: 0)
(0x00)
            .... .0.. = Fragments: No fragments
            .... 0... = Retry: Frame is not being retransmitted
            ...0 .... = PWR MGT: STA will stay up
            ..0. .... = More Data: No data buffered
            .0.. .... = WEP flag: WEP is disabled
            0... .... = Order flag: Not strictly ordered
    Duration: 0
    Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source address: 00:02:2d:1b:51:ca (Agere_1b:51:ca)
    BSS Id: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Fragment number: 0
    Sequence number: 1
IEEE 802.11 wireless LAN management frame
    Tagged parameters (13 bytes)
        Tag Number: 0 (SSID parameter set)
        Tag length: 9
        Tag interpretation: roguehost
        Tag Number: 1 (Supported Rates)
        Tag length: 4
        Tag interpretation: Supported rates: 1.0 2.0 5.5 11.0 [Mbit/sec]
0000  40 00 00 00 ff ff ff ff ff ff 00 02 2d 1b 51 ca   @...........-.Q.
0010  ff ff ff ff ff ff 10 00 00 09 72 6f 67 75 65 68   ..........rogueh
0020  6f 73 74 01 04 02 04 0b 16                       ost......
```

PROBLEMS AND COMPLICATIONS

A few problems came to light with the Cisco card and driver that need to be mentioned.

The first problem is that the Cisco card, by default, even in RFMON and RFMON_ANYBSS modes, does not actively scan for traffic on all channels at all times. The following are the conditions under which it will rescan for BSSes:

- When the card is first inserted
- When the interface enters or leaves promiscuous mode
- When synchronization with the current BSS is lost (due to interference, moving out of range, or anything else that would cause the loss of a few beacon frames)
- When the /proc entry /proc/driver/aironet/eth0/BSSList is opened for writing (use touch /proc/driver/aironet/eth0/BSSList)

All of these conditions will "kick" the card into rescanning. To build a practical detection device, the card should be kicked at regular intervals, perhaps every minute. A simple script to touch the BSSList file every minute will do the trick.

Second problem: not all the BSSes in the range showed up reliably in the file /proc/driver/aironet/eth0/BSSList.

When the card is put into RFMON mode, transmitting is disabled, so the card cannot scan actively for BSSes by sending out probe requests. Therefore, the card must use passive scanning instead – instead of sending out probe requests, the card listens for beacons. Passive scans use a timer – the card will listen for beacon frames until the timer expires and then will move to another channel. The problem with the Cisco card is that this timer is set too low. The default value is 40ms, which was insufficient on our test network to notice all BSSes, regardless of the range or relative signal strength of the access points. The solution was to add this line to the card initialization routine, setup_card, in airo.c:

```
cfg.beaconListenTimeout = 120
```

Tripling this timeout made BSS detection work reliably. Consequently, all of our access points showed up in BSSList, all the time.

Third problem: despite its name, even putting the card in RFMON_ANYBSS mode did not cause the card to receive traffic from all of our access points, which were all using different frequencies and were probably synchronized differently.

The card itself chose a BSS to synchronize to based on its own algorithm (probably on its assessment of the relative signal strength). The problem with this is that we *want* to see traffic from all BSSes in range, not just those that happen to have the strongest signals. A way could not be found to disable this feature on the Cisco card, but there is a workaround — the Linux driver provides a /proc interface to set a preferred AP. Once the list of BSSes in range of the scanner is found (/proc/driver/aironet/eth0/BSSList), choose the one to monitor and enter the MAC address in the file /proc/driver/aironet/eth0/APList. This will force the card to synchronize with that BSS and switch to that channel, after which traffic from that BSS can be received and used for signal strength assessments or monitoring for suspicious activity.

CONCLUSIONS

These simple tests confirm that there are 802.11 frames that are characteristic of typical rogue access points and unauthorized ad hoc networks, and that these frames can be detected and analyzed using off-the-shelf components and free software.

Using these concepts along with a database of trusted access points and cards and the fingerprints of suspicious frames, ethereal could be used as a fundamental building block in a full-blown 802.11 intrusion detection system.

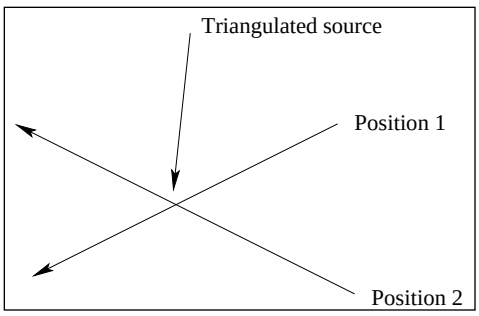

Fig. 1

Tracking

While detection of rogue access points and unauthorized ad hoc networks is certainly useful, physically tracking the offender is more interesting.

THEORY AND TECHNIQUES

Most triangulation done today uses directional techniques. A dish, yagi, or other directional antenna with a narrow range is attached to a receiver, and an area is scanned for the strongest signal. The station is moved some distance away, the area is scanned again, and then the position is found using simple trigonometry.

In practice, there is usually more involved than this. Radio propagation is affected by terrain, obstructions, heat and atmospheric effects, reflections, refraction, antenna

design, and many other factors that could affect the apparent location of the transmitter. The apparent location of the strongest signal may not actually be the location of the transmitter. A full discussion of these effects is beyond the scope of this paper.

Regardless, the basic theory is sound and can be a very effective technique to locate a transmitter.

Another technique, not quite as common today, is to locate the transmitter using only the relative signal strength at various well-placed locations around the area and, if known, the free-space propagation losses and the power of the transmitter.

If we know the output power of the transmitter, the gain of the receiving and transmitting antennas, power drop-off, and the received signal strength, the location of the transmitter can be narrowed down to two possible places with only two samples.

At point A, from the signal strength and power drop-off equation, we'd be able to determine the distance from the transmitter. At point B, we'd also know that, so the location of the transmitter will be found at the point(s) where both d_A and d_B meet:

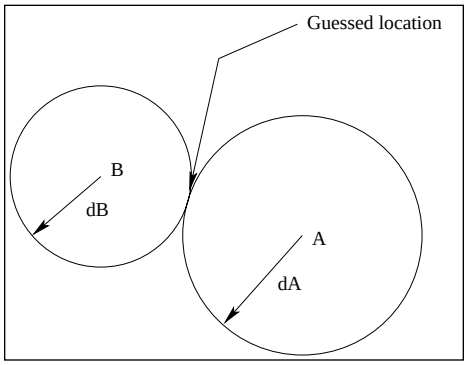

Fig. 2

If we don't know the strength of the transmitter or the gain of any of the antennas involved, we just have to introduce one more unknown, a signal strength ratio; the problem then turns into a simultaneous equation with three unknowns. Thus, a minimum of three data points are needed. If these data points are well placed, there should only be one or two realistic solutions to the equations.

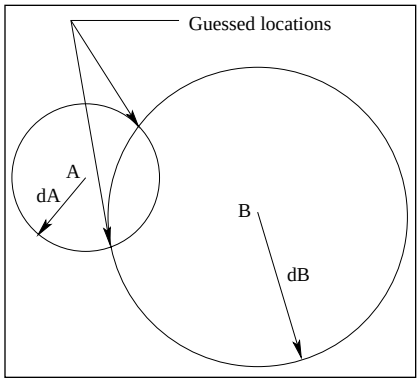

Fig. 3

PRACTICAL APPROACH

The approach taken is a mixture of experimentation and extremely simplified indoor propagation theory. In "A Study of Indoor Radio Propagation," available at *http://www.sss-mag.com/indoor.html*, researchers have come up with an approximation for losses at 2.4GHz inside buildings.

What was needed was a solution that would work well in any office environment. Instead of meticulously accounting for every possible loss or gain, "A Study of Indoor Radio Propagation" provided a generic model for indoor radio propagation that, while not perfect, would work well in most indoor environments.

TEST ENVIRONMENT

Testing was done in a typical office suite, an area of about 30 by 20 meters. There were a large number of offices, and some larger open rooms with no more than six cubicles each. Structurally, the office is largely concrete and steel. This is fairly typical of most offices, and is probably similar to other offices that use wireless networks.

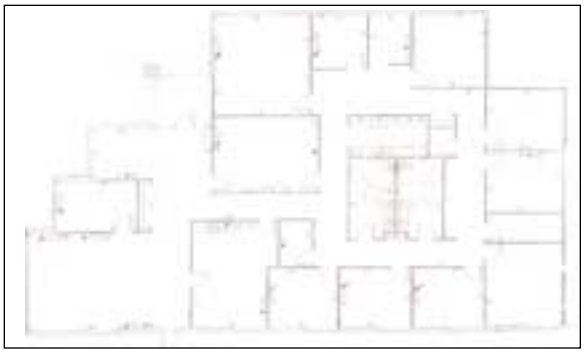
Fig. 4. The office used for testing

Three separate access points were used for testing, placed at random locations around the office. The first two were Cisco AP350s. One was equipped with two antennas. The second did not have any antennas. The third was an Orinoco AP-1000 with a small Lucent antenna.

DATA GATHERING

The first goal was to make some plots of distance vs. relative signal strength as reported by the card's driver, so that the hypothesis that the signal strength would drop off at some predictable rate could be tested. The program iwspy was used to gather signal strength statistics on a particular MAC address:

```
# iwspy eth0 0b:0b:0b:0b:0b:0b
#
```

Then, successive calls to "iwspy eth0" returned the signal strength of packets received from that transmitter.

The tester then simply walked around the office, taking and recording readings at random locations around the office. After some processing, three tab-delimited datafiles containing the distance from the transmitter in the first column and the recorded signal strength in the second were generated.

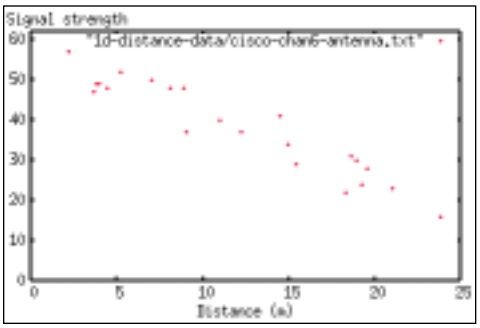

Fig. 5: Cisco access point – antenna

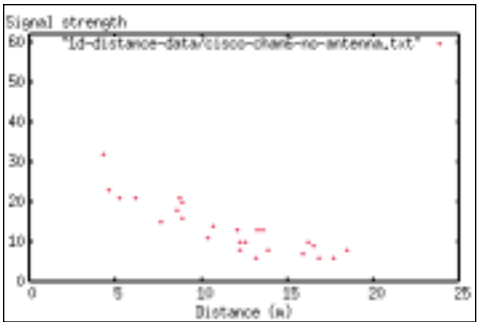

Fig. 6: Cisco access point – no antenna

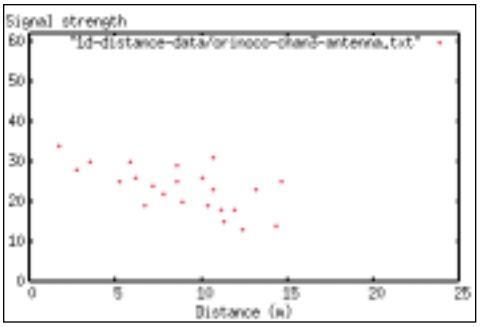

Fig. 7: Orinoco access point – antenna

DISTANCE–SIGNAL STRENGTH TRENDS

The distance–signal strength plots for each of the three access points are shown here.

DISTANCE–SIGNAL STRENGTH CURVE FITTING

The next step was to determine whether or not the data could be fitted to an equation approximating indoor losses at 2.4GHz. A simple least-squares fitting present in Gnuplot was used to verify this. If the curve fit the data "well," one would assume that the equation was a good approximation.

First, signal strength values returned by the iwspy interface (RSSI values, or Received Signal Strength Indicator) needed to be converted to dBm in order to use canned indoor loss equations. dBm is a logarithmic measure commonly used when dealing with low-power radio equipment.

The values returned by iwspy were most likely already logarithmic, to make them easier to read. If they were logarithmic, they should have some linear relationship to the signal strength in dBm.

The relationship between the values returned by iwspy and dBm was found to be linear (determined experimentally using the Windows interface, so this is not exact):

$$P_{dBm} = 1.205\, P_{non\text{-}normalized\ \%} - 101.07$$

$$P_{non\text{-}normalized\ \%} = .83\, P_{dBm} + 83.891$$

If these values are "normalized" (this is an option that can be turned on in the card), the equation becomes:

$$P_{dBm} = .602\, P_{normalized\ \%} - 101.07$$

$$P_{normalized\ \%} = 1.66\, P_{dBm} + 167.782$$

Currently, the iwspy ioctl for the Linux driver returns non-normalized values (incidentally, the Cisco client for Windows returns normalized values), so the first two equations should be used to convert between dBm and iwspy percentages, but this may change at some future date.

Rather than pre-process the datafiles, one of the above conversion functions was included in the curve-fitting function.

For the curve-fitting function, an equation was used from "A Study of Indoor Radio Propagation," which presents a modification to the traditional free-space loss equation to account for indoor attenuation, reflection, refraction, and interference. The researchers experimentally found that the following equation could be used to approximate losses indoors at 2.4GHz in most typical buildings, where D is the distance between the transmitter and receiver in meters:

$$\textit{Path loss (in dB)} = 40 + 35\log_{10}D$$

All of the unknown constants as well as the 40 term were gathered into one constant, C, where C is some unknown constant representing the output power and static cumulative gains and losses due to attenuation, the antenna, and other factors:

$$\textit{Received signal strength (dBm)} = C - 35\log_{10}D$$

Using the equations above to convert to non-normalized signal strength percentage (to match the output from Linux iwspy):

$$\textit{Received signal strength (non-normalized \%)} = .83(C - 35\log_{10}D) + 83.891$$

To the right are the resulting plots. They fit fairly well, so we can assume that the above equation is a suitable approximation for indoor propagation losses.

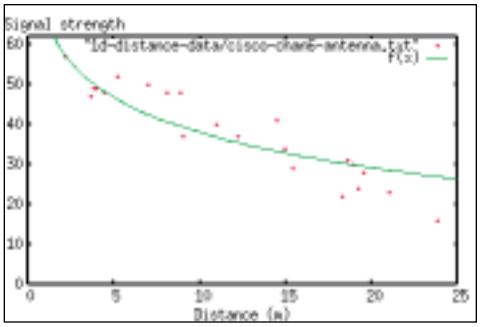

Fig. 8: Cisco access point – antenna

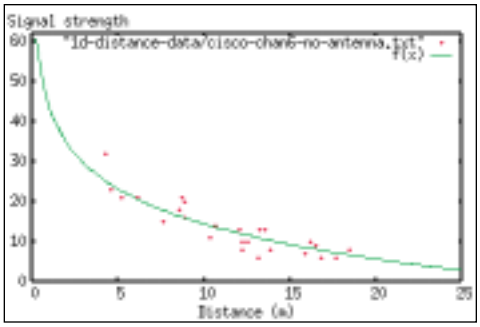

Fig. 9: Cisco access point – no antenna

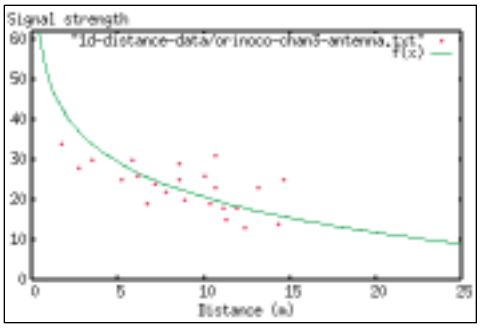

Fig. 10: Orinoco access point – antenna

3-D CURVE FITTING

The final step in tracking was to write the distance-power approximation equation as a function of a position (x,y) and three unknowns, the unknown coordinates (A,B) of the transmitter and an unknown constant C. The resulting equations look like this:

$$\textit{Received signal strength (dBm)} = C - 35\log_{10}\sqrt{(A-x)^2 + (B-y)^2}$$

$$\textit{Received signal strength (non-normalized \%)} = .83(C - 35\log_{10}\sqrt{(A-x)^2 + (B-y)^2}) + 83.891$$

This equation could then be plugged into Gnuplot for fitting. Below is one sample plot. The initial parameter file, used to provide initial guesses to the best-fit algorithm., consists of a = 10, b = –10, and c = –20. It's best to guess physically reasonable values, so coordinates near the center of the office space were chosen, and an initial value of -20 for c. The results are summarized in the next section.

RESULTS

The Gnuplot program, when run, will display a graph containing the parameters of the best fit found and give an estimate of the error. The following is a table of actual values and the best-fit values found from the data sets.

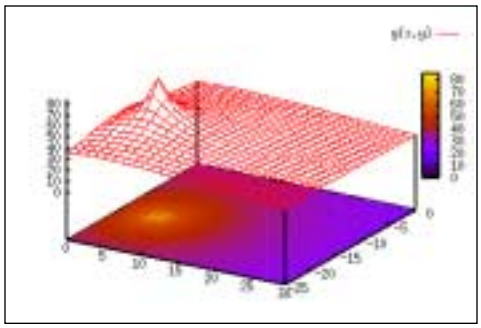
Fig. 11

Two results were very encouraging; one was not. While the first two were placed within a couple meters of the actual location of the transmitter, the results for the Orinoco antenna placed the transmitter about 8 meters south of its actual location.

	Actual location (x)	Actual location (y)	Guessed location (x)	Guessed location (y)	Guessed constant C (related) to the transmitter's output power
Cisco with antenna	5.18	-14.94	5.15986 +/- 0.9969	-14.847 +/- 0.7456	-20.1008 +/- 1.706
Cisco without antenna	12.50	-18.29	11.3324 +/- 0.2842	-16.2561 +/- 0.55	-50.4432 +/- 0.6815
Orinico with antenna	14.17	-13.11	14.6243 +/- 1.3	-21.1688 +/- 2.547	-34.1176 +/- 2.639

The causes for this discrepancy could be many. Among the primary suspected causes:

The location of the samples is of utmost importance, as are the number of data points.

- The distance–signal strength trend was not very strong for the Orinoco antenna. There was a large cluster, but not many points from very near the antenna or very far, essential for getting a proper fit.
- The transmitter was centrally located, so sampling a wide range of distances was not possible.
- The transmitter is located in a room with a lot of metal equipment, probably scattering the transmissions a great deal.
- Some "hot spots" near the southern end of the building pulled the fit too far south. The east-west coordinate was a good fit.

Regardless, the basic theory is sound and more careful data gathering and more data points would probably yield better results.

This has driven home an important lesson, however – the location of the samples is of utmost importance, as are the number of data points. If the sensing stations are stationary, there should be as many as possible and they should be as far away from each other as possible, probably in the furthest corners of the building, as long as reception is still possible.

LIMITATIONS

There are a number of limitations and restrictions to keep in mind when deploying a system like this.

- Using these techniques, it is impossible to trace a card or access point that is not transmitting. It would not be possible to track down the location of a truly passive sniffer using these methods.
- The Cisco driver and card are limited in how fast and how reliable their scans are. Therefore, it is possible for intermittent traffic to be missed completely.
- The Cisco card can only reliably sniff traffic from one channel at a time. If something interesting shows up, all cards should stop scanning and pay attention to the particular channel of interest until enough data is gathered to track down a location.
- If there is more than one tracking station, they should use identical hardware (antenna included) and driver revisions.
- The more data points, the better.
- Tracking stations, if stationary, should be in well-placed locations, so fit errors can be kept to a minimum. Ideally, they should be placed far away from each other, in open locations so path losses and attenuation can be kept to a minimum, but also placed such that all stations can detect traffic from a card or access point anywhere in the building.
- Administrators should also consider an active scanning station. If an unauthorized transmitter is found, one station might be selected to send pings, probe requests, or anything that might prod the transmitter into sending responses, from which signal strength information could be gleaned.
- A high-quality, high-gain antenna is important.
- No effort was made to handle the case of a mobile transmitter.
- The methods used here can occasionally yield results with extremely large errors, especially if the input set is not rational. It's a good idea to pick meaningful initial guesses for the fitting algorithm. It may also be a good idea to reject physically unlikely results — for instance, if the fitted constant C seems to indicate that an unauthorized transmitter is transmitting at 1000 watts, the result can probably be

discarded out-of-hand. Likewise, it's unlikely to receive transmissions from a transmitter miles away. It is possible, but unlikely.

- These methods could not, by default, detect cards or ad hoc networks that happen to be running on channels other than the ones reserved for use in North America.

IDENTIFIED ISSUES AND AREAS FOR IMPROVEMENT

- Make traffic detection more reliable and less dependant on channel scans. Ideally, a truly promiscuous 802.11 sniffer that could monitor traffic on all frequencies and all spread-spectrum synchronizations at the same time would be preferred.
- Be aware that the relationship between dBm and % signal strength from the Cisco card is not exact and may not be an exactly linear relationship.
- Integrate a GPS unit so as to allow for active processing and simplified data gathering.
- Build a working prototype system involving at least three stationary monitoring stations. Test it with mock attacks on the local network.
- Find a more robust IDS than ethereal and some scripting glue. Hopefully, snort's author(s) will integrate ethereal's protocol analysis engine.
- Write a protocol to link monitoring stations so that they can compare data with each other and cooperatively locate rogues – part of the working prototype.
- Consider adding some real path-length and wall-attenuation algorithms to reduce the amount of scatter.
- Consider full-blown physical modeling of the office.
- Gather more data to verify that $35 \log_{10}(D)$ really is a good approximation.
- Build some profiles of expected power outputs of various vendors' access points and cards with and without antennas, in an attempt to minimize the unknowns.
- Experiment with "real" triangulation using directional antennas.
- Gather hardware documentation on the Cisco Aironet cards. RFMON and RFMON_ANYBSS are still confusing, especially with synchronization and channel scanning.
- Get info on new Cisco RIDs and add them to the driver.
- Experiment with monitor mode on other cards and chipsets.

REFERENCES

Donald L. Schilling, Raymond L. Pickholtz, and Laurence B. Milstein, "Spread Spectrum Goes Commercial," *The IEEE Spectrum*, August 1990.

"A Study of Indoor Radio Propagation" – http://www.sss-mag.com/indoor.html. Thanks to the folks at Spread Spectrum Scene for this article, as it provided a good approximation for indoor radio propagation losses. This was essential to feed to the curve-fitting algorithms, and it yielded good results.

"Cisco – Wireless Point-to-Point Quick Reference Sheet"

http://www.cisco.com/warp/public/102/wwan/quick-ref.pdf

"Peter Mikulik's gnuplot page"
http://www.sci.muni.cz/~mikulik/gnuplot.html#pm3d

GNU Octave Documentation
http://www.octave.org/doc/octave_toc.html

Linux Aironet driver – comments and source were invaluable.
http://sourceforge.net/projects/airo-linux/

Ethereal home page – great documentation and source for disassembling all sorts of network protocols.
http://www.ethereal.com/

IEEE – lots of great collected information on 802 standards.
http://www.ieee.org/

I am indebted to V.N. Padmanabhan and P. Bahl for their article "RADAR: An In-Building RF-Based User Location and Tracking System," available at *http://research.microsoft.com/~padmanab/*. This article was not a reference, per se (in fact, it was discovered after this paper was nearly finished), but it was interesting to see that some of my work had been a duplication of the efforts of someone else, and it provided independent confirmation that this method can work in an office environment. My approach was not nearly as elegant or as scientifically rigorous as theirs, but it was encouraging to know that the results were similar.

Special thanks to Richard Stallman for GNU.