



# Linux & 802.11 Wireless

Mike Kershaw  
dragorn@kismetwireless.net

April 2, 2004



- 802.11 (aka WiFi) is a set of standards for wireless networking
- IEEE defined 802.11 in 1997 but it didn't become affordable or popular until around 2001
- Three 802.11 standards in use:
  - 802.11b at 2.4GHz, 11Mbit maximum speed. Most common.
  - 802.11g at 2.4GHz, 54Mbit maximum speed. New standard, but growing rapidly. Backwards compatible with 802.11b
  - 802.11a at 5.8GHz, 54MBit maximum speed. Less popular, shorter range.
- **NOT** the same as cellular data (gprs, gsm, etc) or Bluetooth
- Used for local networks (houses), corporate networks (office environments), and neighborhood networks (central point or distributed mesh)



**Access Point** The Access Point (or AP) is the center of your wireless network. Access points may function as simple bridges, or they may contain routing, NAT, and other more complex functionality.

**Wireless NIC** The client card is (obviously) what allows your computer to connect to the wireless network. Almost all operating systems, including Linux, treat this as a standard Ethernet network device. Wireless NICs come in every form factor - PCMCIA, Cardbus, PCI, ISA, compactflash, USB, etc.

**Antennas** Antennas don't add to the transmission power, but they shape the power available. Most of the time you will be interested in omnidirectional antennas, which disperse the signal equally in all directions.



The 10,000 foot view of how 802.11 networks work and what happens when you plug in an AP:

- Each 802.11 network is defined by the SSID, or **S**ervice **S**et **I**dentifier. This is essentially the name of the network.
- 802.11b and 802.11g use channels 1-11 in the US, and 802.11a uses channels between 36 and 216.
- Multiple networks can be on the same channel but this will significantly reduce the bandwidth. 802.11b/802.11g networks should be at least 3 channels away from other networks in the area.
- When clients are looking to join a network or get a weak signal, they will hop through the channels looking for the strongest network with the desired SSID.
- Networks operate in Infrastructure mode (BSS) and coordinate through an AP, or in Ad-hoc mode (IBSS) which does not require an AP



**Infrastructure** Infra mode requires an AP. There are generally fewer problems than with Ad-hoc - not all vendors implement ad-hoc in the same way. The AP usually also functions as a bridge and may function as a router, though there is no reason it has to be.

**Ad-hoc** Cheaper if you only need to link 2 or 3 computers. All computers must be able to see each other, and there is no central coordinator.



Wireless support in Linux is relatively young (compared to long-established network drivers like ethernet), and isn't always an easy path to follow.

1. Identify the chipset your card uses
2. Download and install drivers
3. Configure PCMCIA-CS if necessary
4. Manually configure your card for testing
5. Configure your card to work automatically with your distribution



Many people are eager to move to the new 2.6.x kernel series. Depending on the wireless card you chose, there may not be support yet in 2.6. Nearly all of the wireless drivers are maintained separately from the kernel tree, and have not yet been updated. It may be possible to get your card working in 2.6, but don't be surprised if you need to use 2.4 a while longer.



More important than the brand of card you chose is the chipset inside of it. Most companies like Linksys, DLink, Netgear, Belkin, etc, do not make their own wireless cards - they license the chipset from another company and put a new label on it. This is good, because the hundreds of models and brands break down to about 9 actual types, but bad because it can be hard to know what you're getting.

The best idea is to get a card which is definitely known to work. The linux-wlan project has compiled a very complete list of all the card models:

[http://www.linux-wlan.org/docs/wlan\\_adapters.html.gz](http://www.linux-wlan.org/docs/wlan_adapters.html.gz)



Just like Ethernet, different wireless manufacturers use different chipsets, which means Linux needs different drivers. Some work very well, and others ... don't. The chipsets best supported in Linux are:

**Intersil Prism2** Intersil has been extremely cooperative with the Linux community. Older Linksys, Netgear, Dlink, as well as new high-power Senao and Demarctech cards use Prism2

**Orinoco** Older Orinoco cards have more support than brand new ones, but both will work well in most cases. Orinoco A/B/G combo cards are also supported as Atheros cards.

**PrismGT** Intersils 802.11G chipset, also well supported. Found in the SMC 802.11B/G cards, and others

**Atheros** 802.11A/B/G combo chipset, well supported with assistance from Atheros. Found in some DLink, Linksys, on-board.



Some cards have some support, but not drivers that work well - many drivers have unfortunate flaws that lead to reliability and stability problems. If you can avoid getting one of these cards, you'll probably have an easier time.

**Cisco** While there have been Cisco drivers for quite some time, they are plagued with stability and throughput problems.

**ACX100** Marketed by several companies (notably DLink) ACX100 uses a proprietary method to get 22mbit under "802.11b+".

**ADMTek** Used in some some cards by SMC, Belkin, others. Drivers are not mature.

**Atmel** Used in many USB devices. Drivers seem to be unstable - some have reported success but many have not gotten it working.



Some companies will not release specifications or help develop drivers. If you're unlucky enough to have one of these cards, you may just want to get another one:

**Intel Centrino** One of the biggest dissapointments has been the poor support from Intel for their Centrino 802.11b chipset. As of writing this, beta drivers with extremely limited support have finally been released, but the chipset is far from usefully supported.

**Broadcom** Found in many PCI cards for 802.11b and 802.11g, as well as some Linksys PCMCIA cards. Broadcom refuses to even acknowledge Linux as an operating system. There are kluges to get these working by loading Windows drivers into the kernel and emulating NDIS (Linuxant and NDISWrapper) with very mixed results.



It is relatively easy, if you have an unknown card, to figure out what chipset it uses. First consult the Linux-Wlan chipset list at [http://www.linux-wlan.org/docs/wlan\\_adapters.html.gz](http://www.linux-wlan.org/docs/wlan_adapters.html.gz) and try to find your brand and model. If that doesn't work,

- For PCI, MiniPCI, and Cardbus cards:
  1. Install (PCI) or insert (Cardbus) the card
  2. Run `lspci` or `cat /proc/pci` to get the model number
- For PCMCIA:
  1. Install or insert the card. Chances are you'll get the high-low failure beep.
  2. Run `cardctl ident` to get the model information.
- For USB:
  1. Plug the device in.
  2. Run `dmesg` or look at your syslog files to get the manufacturer identification, which will 2 sets of numbers, XXXX/XXXX.



Armed with your recently acquired manufacturer info, recheck the Linux-WLAN list and, of course, <http://www.google.com>.



Very few chipsets are supported directly in the Linux kernel, and several which DO have drivers in the kernel are better supported by third-party drivers.

Generally speaking, third-party drivers come in three varieties:

1. Fully standalone drivers which compile from their own directory with a standard configure script and makefile (built with `./configure` and `make` respectively)
2. Drivers which patch or install over the `pcmcia-cs` package. These drivers require the source to the PCMCIA-CS package, either get this manually from the site or get the source package for your distribution.
3. Binary packages for your distribution. Often these do not work as well as compiling the drivers from source yourself,



and may not be supported by the driver authors if you experience problems.

There are too many subtle variations to go into, but most drivers will be compiled with `make` or `make config` and `make all`. The drivers you download should contain a `README` file that explains how to compile them.

Some distro vendors provide packages, but it is usually a better idea to download the latest versions of the driver source and compile it yourself - many vendors do not correctly package the wireless drivers, or package older versions which do not work correctly.



Just like `ifconfig` configures network interfaces, `iwconfig` configures the wireless part of a wireless network interface. `iwconfig` with no options lists all available interfaces and any wireless information present on each.

```
lo      no wireless extensions.
eth0    no wireless extensions.
eth1    IEEE 802.11b  ESSID:"nerv"  Nickname:"Linux"
        Mode:Managed  Frequency:2.437GHz  Access Point: 00:0C:41:9C:C4
        Bit Rate=11.5343Mb/s  Tx-Power:off  Sensitivity:1/0
        RTS thr:off
        Power Management:off
        Link Quality:39/0  Signal level:-48 dBm  Noise level:-87 dBm
        Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
        Tx excessive retries:0  Invalid misc:0  Missed beacon:0
```



**ESSID** ESSID is the current effective SSID, in this case.

**Nickname** Nickname of the card, this is for organization only and holds no real meaning.

**Mode** Mode the card is in. “Managed” is standard infrastructure mode, connected to an AP.

**Frequency** Channel frequency the card is on.

**Access Point** MAC address of the AP. If the card is not associated with an AP, this will be 00:00:00:00:00:00, FF:FF:FF:FF:FF:FF or sometimes 44:44:44:44:44:44.

**Bit rate** Current bitrate of the link. This will drop as your link quality degrades.

**Tx-Power, Sensitivity, RTS** Most drivers don't support these controls. For those that do, they control transmit power, radio sensitivity. Generally these should be left alone.



**Power Management** 802.11 power management status. Support for power management varies widely, generally it makes sense to leave it alone.

**Link Quality** Different drivers have different ideas about what the link quality is defined as. Most of them make no sense.

**Signal & Noise Levels** Some drivers report these in arbitrary numbers with marginal relevance, while some report them in decibels (dB).

**Packet counts** Different drivers track various packet counts for transmission failures, collisions, and cross-network talk.



`iwconfig` can set values as well as display them. The syntax is fairly straightforward: `iwconfig device option settings`.

- Setting the SSID

```
iwconfig eth1 essid "ILikeMonkeys"
```

This sets the SSID to “ILikeMonkeys”. Capitalization is important, as are spaces. Put the SSID in quotes if there are spaces in the name.

Setting the SSID to "" will tell the card to join any network which allows it in.

- Setting the encryption

```
iwconfig eth1 key FEEDFACEDEADBEEF0123456789
```

The key must be given in hex, which will vary in length depending on the type of encryption used on your network.

```
iwconfig eth1 key off
```

While not always necessary, it's a good idea to explicitly turn



off encryption when it is not needed, especially if you are changing between networks.

- Setting the mode

```
iwconfig eth1 mode managed
```

Will set a card into normal infrastructure mode, to connect to an AP.

```
iwconfig eth1 mode ad-hoc
```

Will set a card into ad-hoc mode. All cards on an ad-hoc network must be in ad-hoc mode.

```
iwconfig eth1 mode master
```

Some drivers (not many) support AP mode by setting the card to master. This lets a normal card function as an AP. Drivers like HostAP include a suite of management tools and functions to control it, other drivers provide only minimal support and can't control the finer points of AP operation.



```
iwconfig eth1 mode monitor
```

Raw sniffing mode is also called monitor mode. If you want to capture 802.11 frames, use this. Tools like Kismet will put the card in monitor mode automatically.

- Setting the channel

Setting the channel is actually unnecessary, and many drivers will return an error if you try to force it. 802.11 cards roam between networks by automatically scanning channels for the strongest AP responding to the SSID.

- Stacking commands

```
iwconfig eth1 essid "ScaryMonkeyShow" key off
```

Commands can be stacked to configure all the attributes of a card at once instead of in multiple commands.



`iwlist` can query a number of stats from the wireless card. Not all the options are supported by all drivers, but the most useful are:

- Fetch a list of supported channels:  
`iwlist eth1 channel`
- Actively scan for access points:  
`iwlist eth1 scan`

Other `iwlist` commands can be found with `iwlist -h`.



- Cannot associate

Check that you are loading the right drivers. A lot of distributions like to bind prism2 cards to the Orinoco drivers. This is wrong.

Check that SSID is correct

Check that the WEP key is correct - always use the hex key! There is no standard for transforming an ascii string ("crappy password") to the hex string ("00112233"), so each vendor does it differently.

- Intermittent Failure

Check that no other networks in the area are using the same channel (iwlist scan or Kismet)

Check that no other networks in the area have the same SSID (layer2 roaming will make you wander to them, check with iwlist scan or Kismet)



Look for other 2.4ghz devices - microwaves in use, wireless speakers, cordless phones, etc. Try changing channels on them or changing channels on your AP



There are several extremely important things to remember about network security on wireless networks. Confusion about security is compounded by vendors often failed attempts to secure their APs.

1. **Wireless goes places you might not expect.** It isn't going to stop at the edge of your house or business.
2. **WEP encryption is not secure.** There are a number of major flaws with the encryption methods used in WEP. It's fine for most home users, but if your business is interesting enough to be worth breaking into, WEP is not secure enough.
3. **SSID hiding is not a security measure.** Most vendors now offer the ability to cloak the SSID of a network by not announcing it. This will prevent casual snoopers from getting on your network, but will not prevent someone from discovering the network name, and will not provide any real security.



4. **Non-beaconing is not a security measure.** Some vendors offer the ability to hide the network by not sending beacon frames announcing its presence. However, as soon as data is transferred, the network will be visible, and the SSID can be extracted.
5. **You can't trust the network.** Spoofing an AP is trivial. You can't trust that what you're talking to is what you think you're talking to. Man In The Middle is easy.
6. **Wireless is not reliable.** It may be convenient, but it can be disrupted easily. Don't trust wireless for mission critical links. **There is no way to prevent a dedicated attacker from killing a wireless network. Period.**



In an effort to secure wireless, several standards have been added which enhance (but don't always solve) security on 802.11:

**TKIP & 802.11i** Temporal key exchange system. TKIP is designed to automatically provide and rotate encryption keys once a client has authenticated with the AP. By limiting the amount of time a particular key is used, the possibility of a brute force attack or stolen key is decreased.

**\*EAP** The \*EAP systems are a method for mutual authentication. For a long time Cisco has hyped LEAP in their products, however LEAP is flawed and is vulnerable to dictionary attacks by programs like Asleap. Cisco will be moving to a new authentication system in the future, but it may be vulnerable to similar flaws. PEAP (Cisco, Microsoft, others) provides mutual authentication and a secure transport layer, similar to VPN.



**WPA** Wifi-Protected Access, includes TKIP key rotation, stronger authentication of users, and AES encryption. AES is far more secure than the RC4 encryption used in WEP today. All hardware manufactured after August 2003 is required to support WPA, but drivers for Linux still have to catch up.



Sniffing 802.11 is not as simple as sniffing Ethernet networks. Sniffing Ethernet happens in promiscuous mode - the MAC filter of the card is turned off so that all frames are passed to the host computer.

Promisc mode is possible with some drivers, but does not have the same meaning for 802.11. Sniffing in promisc mode on 802.11 will yield only the data packets for the network you are associated with.

True 802.11 sniffing requires rf-monitor mode. In rfmon mode, a wireless card is not associated with any network, and returns full 802.11 frames, including the management frames which define the network.



**Scanning** involves a wireless card actively sending out probe requests looking for a network that will let it join. Most drivers and cards will do this normally when building a list of networks that the user can join. Scanning mode can't sniff data, and it won't usually report networks that are set to hide their SSID, and it requires that the network be in range of your card. Scanning mode can be detected, as well. The NetStumbler-style programs use scanning mode to find networks.

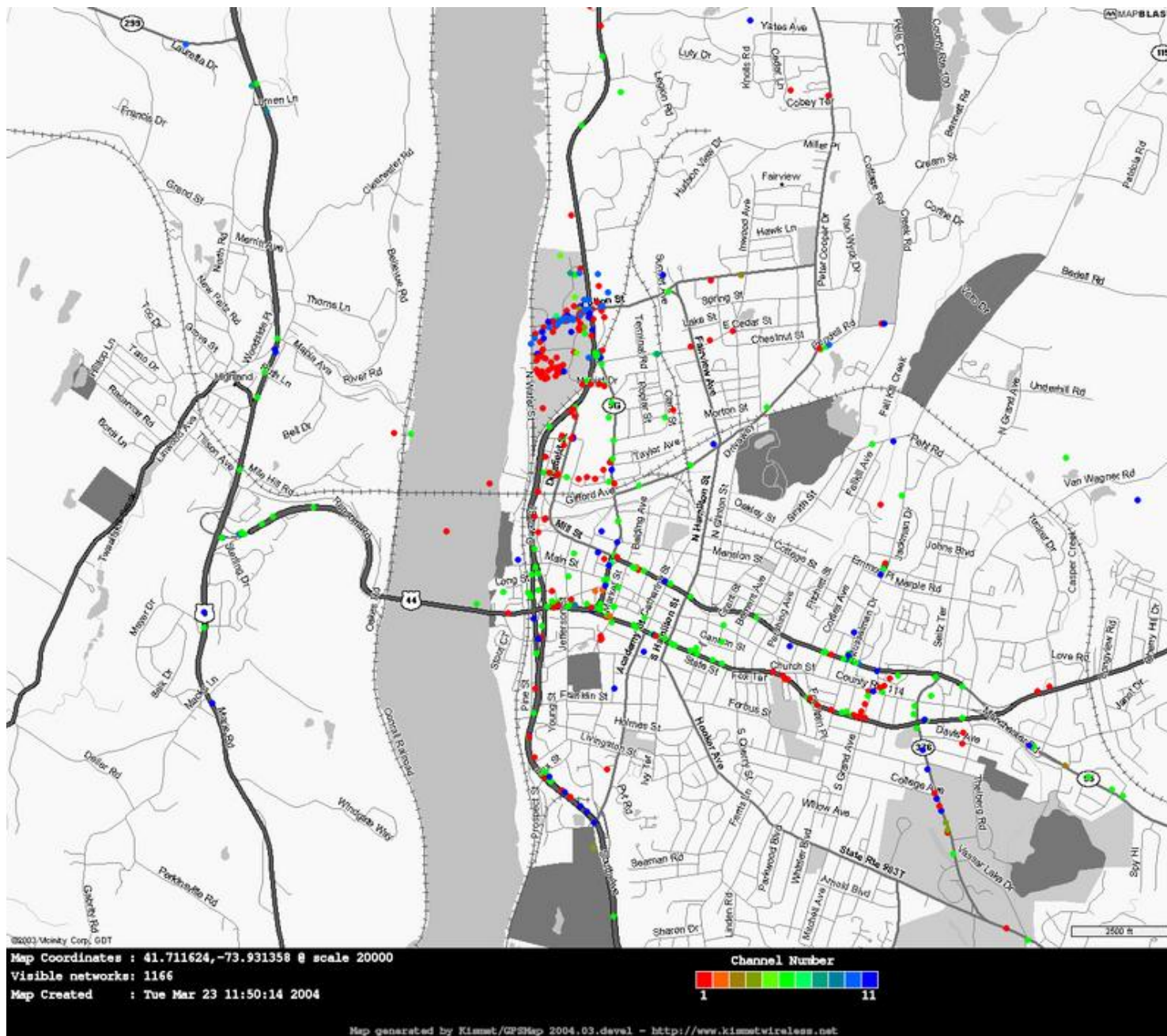
**Sniffing** mode uses the rfmon abilities found in most (but not all) wireless cards. This turns the card into a completely passive listener, like a car radio. Everything on the current (and adjacent) channels is reported to the host, including data and control frames. Kismet, Ethereal, KisMac, and other full sniffers use sniffing mode.



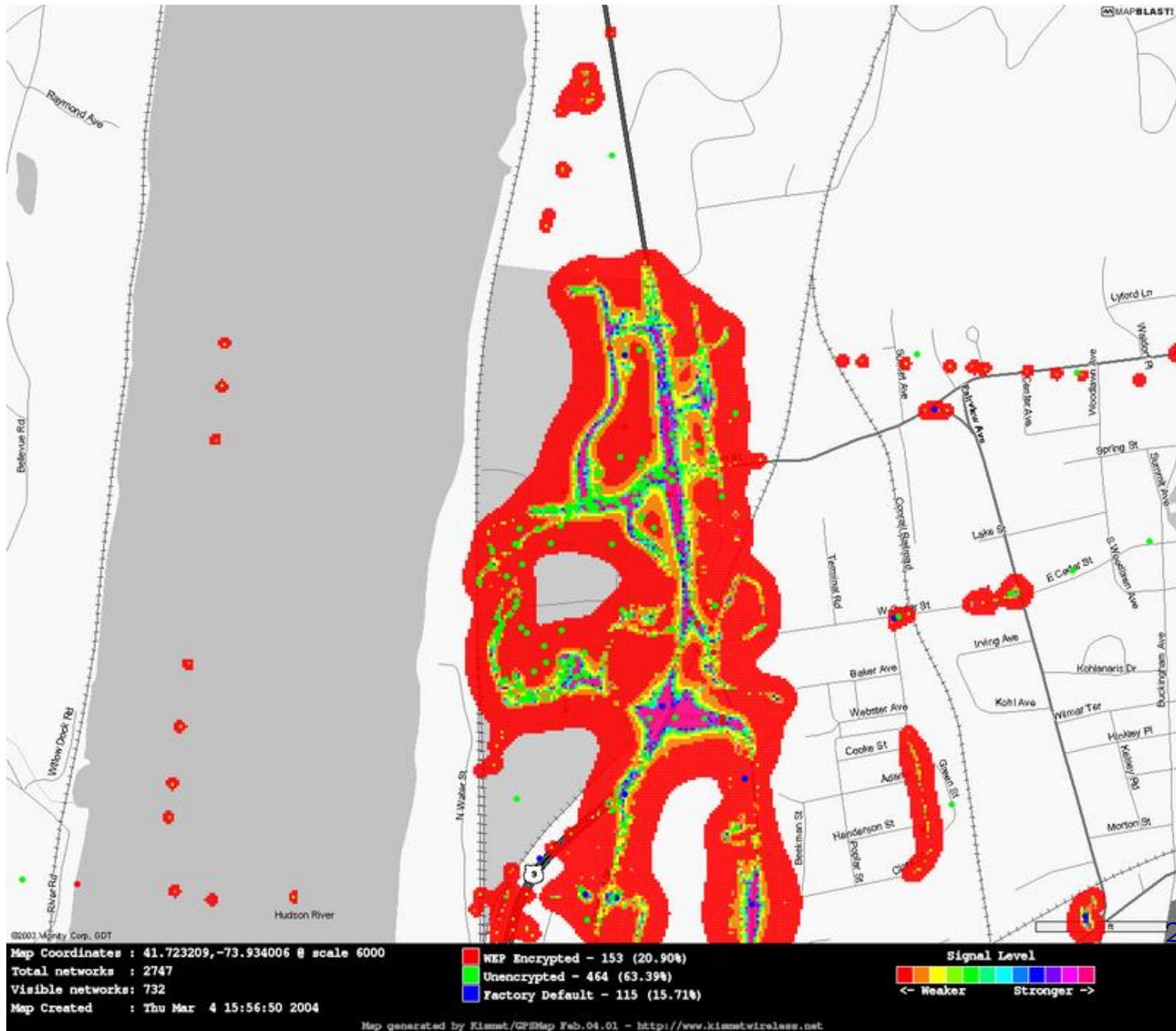
(Shameless plug) Kismet is a packet sniffer designed to track wireless networks. Among other things, it:

- Has builtin channel hopping
- Network IP detection
- Detection of hidden networks
- Uncloaking hidden SSIDs
- Flagging default configurations
- Mapping of network locations

# APs In Poughkeepsie



# Signal Levels In Poughkeepsie





Setting up a secure home wireless network is relatively easy:

1. If at all possible, create a second network segment for your wireless. This will prevent your normal house traffic from being needlessly replicated into the wireless.
2. Use WEP. WEP isn't very good, but for most home users it's sufficient.
3. Rotate WEP keys once in a while. Once a month or so, change your WEP key.
4. Consider using a VPN. Various VPN solutions (FreeSWAN, IPSec, PPP-over-SSH) are fairly easy to set up.
5. You can turn off SSID broadcasting if it really makes you happy



Setting up a secure wireless network in a business environment is trickier:

1. Do not use wireless for any critical systems - cash registers, servers, kiosks handling customer data that must be kept secure, office desktop systems, etc. Wireless is a convenience.
2. Treat wireless as a hostile external network. This means the wireless network should be outside your firewall, with all the authentication implications.
3. Don't trust WEP. Home users can get away with it because they're not interesting enough to spend a lot of time cracking. Businesses are.
4. Firewall heavily. I suggest firewalling the wireless so that it can talk only to the VPN port on the VPN server, nothing else.



5. Audit your building on a regular basis. Users are often smart enough to bring in an AP and plug it in for their own use, but this doesn't mean they're smart enough to have done it securely. All the policies in the world won't matter if Wobbly Headed Bob drops an AP behind your firewall with no WEP.