Comment out any other sections in the file that refer to wlan0 or wlan1 to make sure that they don't interfere with our setup.

This syntax for setting up bridges via the **interfaces** file is specific to Debian-based distributions, and the details of actually setting up the bridge are handled by a couple of scripts: **/etc/network/if-pre-up.d/bridge** and **/etc/network/if-post-down.d/bridge**. The documentation for these scripts is found in **/usr/share/doc/bridge-utils/**.

If those scripts don't exist on your distribution (such as Fedora Core), here is an alternative setup for **/etc/network/interfaces** which will achieve the same thing with only marginally more hassle:

```
iface br0 inet static
  pre-up ifconfig wlan 0 0.0.0.0 up
  pre-up ifconfig wlan1 0.0.0.0 up
  pre-up iwconfig wlan0 essid “office” mode Managed
  pre-up iwconfig wlan1 essid “repeater” mode Master
  pre-up brctl addbr br0
  pre-up brctl addif br0 wlan0
  pre-up brctl addif br0 wlan1
  post-down ifconfig wlan1 down
  post-down ifconfig wlan0 down
  post-down brctl delif br0 wlan0
  post-down brctl delif br0 wlan1
  post-down brctl delbr br0
```

Starting the bridge

Once the bridge is defined as an interface, starting the bridge is as simple as typing:

```
# ifup -v br0
```

The “-v” means verbose output and will give you information to what is going on.

On Fedora Core (i.e. non-debian distributions) you still need to give your bridge interface an ip address and add a default route to the rest of the network:

```
#ifconfig br0 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255
#route add default gw 192.168.1.1
```

You should now be able to connect a wireless laptop to this new access point, and connect to the Internet (or at least to the rest of your network) through this box.

Use the **brctl** command to see what your bridge is doing:

```
# brctl show br0
```

Scenario 1 & 2 the easy way

Instead of setting up your computer as an access point from scratch, you may wish to use a dedicated Linux distribution that is specially tailored for this purpose. These distributions can make the job as simple as booting from a particular CD on a computer with a wireless interface. See the following section, "Wireless-friendly operating systems" for more information.

As you can see, it is straightforward to provide access point services from a standard Linux router. Using Linux gives you significantly more control over how packets are routed through your network, and allows for features that simply aren't possible on consumer grade access point hardware.

For example, you could start with either of the above two examples and implement a private wireless network where users are authenticated using a standard web browser. Using a captive portal such as Chillispot, wireless users can be checked against credentials in an existing database (say, a Windows domain server accessible via RADIUS). This arrangement could allow for preferential access to users in the database, while providing a very limited level of access for the general public.

Another popular application is the prepaid commercial model. In this model, users must purchase a ticket before accessing the network. This ticket provides a password that is valid for a limited amount of time (typically one day). When the ticket expires, the user must purchase another. This ticketing feature is only available on relatively expensive commercial networking equipment, but can be implemented using free software such as Chillispot and phpMyPrePaid. We will see more about captive portal technology and ticketing systems in the **Authentication** section in **Chapter 6**.

Wireless-friendly operating systems

There are a number of open source operating system that provide useful tools for working with wireless networks. These are intended to be used on repurposed PCs or other networking hardware (rather than on a laptop or server) and are fine-tuned for building wireless networks. Some of these projects include:

- **Freifunk**. Based on the OpenWRT project (*http://openwrt.org/*), the Freifunk firmware brings easy OLSR support to MIPS-based consumer access points, such as the Linksys WRT54G / WRT54GS / WAP54G, Siemens SE505, and others. By simply flashing one of these APs with the Freifunk firmware, you can rapidly build a self-forming OLSR mesh. Freifunk is not currently available for x86 architecture machines. It is maintained by Sven Ola of the Freifunk wireless group in Berlin. You can download the firmware from *http://www.freifunk.net/wiki/FreifunkFirmware* .

- **Pyramid Linux**. Pyramid is a Linux distribution for use on embedded platforms that evolved out of the venerable Pebble Linux platform. It supports several different wireless cards, and has a simple web interface for configuring networking interfaces, port forwarding, WifiDog, and OLSR. Pyramid is distributed and maintained by Metrix Communication LLC, and is available at *http://pyramid.metrix.net/*.
- **m0n0wall**. Based on FreeBSD, m0n0wall is a very tiny but complete firewall package that provides AP services. It is configured from a web interface and the entire system configuration is stored in a single XML file. Its tiny size (less than 6MB) makes it attractive for use in very small embedded systems. Its goal is to provide a secure firewall, and as such does not include userspace tools (it is not even possible to log into the machine over the network). Despite this limitation, it is a popular choice for wireless networkers, particularly those with a background in FreeBSD. You can download m0n0wall from *http://www.m0n0.ch/ .*

All of these distributions are designed to fit in machines with limited storage. If you are using a very large flash disk or hard drive, you can certainly install a more complete OS (such as Ubuntu or Debian) and use the machine as a router or access point. It will likely take a fair amount of development time to be sure all needed tools are included, without installing unnecessary packages. By using one of these projects as a starting point for building a wireless node, you will save yourself considerable time and effort.

The Linksys WRT54G

One of the most popular consumer access points currently on the market is the Linksys WRT54G. This access point features two external RP-TNC antenna connectors, a four port Ethernet switch, and an 802.11b/g radio. It is configured through a simple web interface. While it is not designed as an outdoor solution, it can be installed in a large sprinkler box or plastic tub for relatively little cost. As of this writing, the WRT54G sells for about $60.

Back in 2003, network hackers realized that the firmware that shipped with the WRT54G was actually a version of Linux. This led to a tremendous interest in building custom firmware that extended the capabilities of the router significantly. Some of these new features include client radio mode support, captive portals, and mesh networking. Some popular alternative firmware packages for the WRT54G are DD-Wrt (*http://www.dd-wrt.com/*), OpenWRT (*http://openwrt.org/*), Tomato (*http://www.polarcloud.com/tomato*) and Freifunk (*http://www.freifunk.net/*).

Unfortunately, in the fall of 2005, Linksys released version 5 of the WRT54G. This hardware revision eliminated some RAM and flash storage on the motherboard, making it very difficult to run Linux (it ships with VxWorks, a much

smaller operating system that does not allow easy customization). Linksys also released the WRT54GL, which is essentially the WRT54G v4 (which runs Linux) with a slightly bigger price tag.

A number of other Linksys access points also run Linux, including the WRT54GS and WAP54G. While these also have relatively low price tags, the hardware specifications may change at any time. It is difficult to know which hardware revision is used without opening the packaging, making it risky to purchase them at a retail store and practically impossible to order online. While the WRT54GL is guaranteed to run Linux, Linksys has made it known that it does not expect to sell this model in large volume, and it is unclear how long it will be offered for sale.

Fortunately, wireless hackers have now been able to install custom firmware on the notoriously difficult WRT54G version 5 and 6, and the latest revisions as well(v7 and v8). For details on getting alternate firmware installed on a v5 or v6 access point see: *http://www.scorpiontek.org/portal/content/view/27/36/*

For more information about the current state of Linksys wireless router hacking, see *http:///linksysinfo.org/*

DD-WRT

One popular alternate firmware for the Linksys family of access point hardware is DD-WRT (*http://www.dd-wrt.com/*). It includes several useful features, including radio client mode, adjustable transmission power, various captive portals, QoS support, and much more. It uses an intuitive web-based configuration tool (unencrypted or via HTTPS), and also provides SSH and telnet access.

Several versions of the firmware are available from the DD-WRT website. The general procedure for upgrading is to download the version of the firmware appropriate for your hardware, and upload it via the router's "firmware update" feature. Specific installation details vary according to the hardware version of your router. In addition to Linksys hardware, DD-WRT will run on Buffalo, ASUS, the La Fonera, and other access points.

For specific instructions for your hardware, see the installation guide on the DD-WRT wiki at *http://www.dd-wrt.com/wiki/index.php/Installation.* The default login for a fresh DD-WRT installation is **root** with the password **admin**.

Figure 5.3: The DD-WRT (v23) control panel

6
Security & Monitoring

In a traditional wired network, access control is very straightforward: If a person has physical access to a computer or network hub, they can use (or abuse) the network resources. While software mechanisms are an important component of network security, limiting physical access to the network devices is the ultimate access control mechanism. Simply put, if all terminals and network components are only accessible to trusted individuals, the network can likely be trusted.

The rules change significantly with wireless networks. While the apparent range of your access point may seem to be just a few hundred meters, a user with a high gain antenna may be able to make use of the network from several blocks away. Should an unauthorized user be detected, is impossible to simply "trace the cable" back to the user's location. Without transmitting a single packet, a nefarious user can even log all network data to disk. This data can later be used to launch a more sophisticated attack against the network. Never assume that radio waves simply "stop" at the edge of your property line.

It is usually unreasonable to completely trust all users of the network, even on wired networks. Disgruntled employees, uneducated network users, and simple mistakes on the part of honest users can cause significant harm to network operations. As the network architect, your goal is to facilitate private communication between legitimate users of the network. While a certain amount of access control and authentication is necessary in any network, you have failed in your job if legitimate users find it difficult to use the network to communicate.

There's an old saying that the only way to completely secure a computer is to unplug it, lock it in a safe, destroy the key, and bury the whole thing in con-

crete. While such a system might be completely “secure”, it is useless for communication. When you make security decisions for your network, remember that above all else, the network exists so that its users can communicate with each other. Security considerations are important, but should not get in the way of the network’s users.

Physical security

When installing a network, you are building an infrastructure that people depend on. Security measures exist to ensure that the network is reliable. For many installations, outages often occur due to human tampering, whether accidental or not. Networks have physical components, such as wires and boxes, which are easily disturbed. In many installations, people will not understand the purpose of the installed equipment, or curiosity may lead them to experiment. They may not realize the importance of a cable connected to a port. Someone may unplug an Ethernet cable so that they can connect their laptop for 5 minutes, or move a switch because it is in their way. A plug might be removed from a power bar because someone needs that receptacle. Assuring the physical security of an installation is paramount. Signs and labels will only be useful to those who can read your language. Putting things out of the way and limiting access is the best means to assure that accidents and tinkering do not occur.

In less developed economies, proper fasteners, ties, or boxes will not be as easy to find. You should be able to find electrical supplies that will work just as well. Custom enclosures are also easy to manufacture and should be considered essential to any installation. It is often economical to pay a mason to make holes and install conduit. Where this would be an expensive option in the developed world, this type of labour intensive activity can be affordable in Southern countries. PVC can be embedded in cement walls for passing cable from room to room. This avoids the need to smash new holes every time a cable needs to be passed. Plastic bags can be stuffed into the conduit around the cables for insulation.

Small equipment should be mounted on the wall and larger equipment should be put in a closet or in a cabinet.

Switches

Switches, hubs or interior access points can be screwed directly onto a wall with a wall plug. It is best to put this equipment as high as possible to reduce the chance that someone will touch the device or its cables.

Cables

At the very least, cables should be hidden and fastened. It is possible to find plastic cable conduit that can be used in buildings. If you cannot find it, simple cable attachments can be nailed into the wall to secure the cable. This will make sure that the cable doesn't hang where it can be snagged, pinched or cut.

It is preferable to bury cables, rather than to leave them hanging across a yard. Hanging wires might be used for drying clothes, or be snagged by a ladder, etc. To avoid vermin and insects, use plastic electrical conduit. The marginal expense will be well worth the trouble. The conduit should be buried about 30 cm deep, or below the frost level in cold climates. It is worth the extra investment of buying larger conduit than is presently required, so that future cables can be run through the same tubing. Consider labeling buried cable with a "call before you dig" sign to avoid future accidental outages.

Power

It is best to have power bars locked in a cabinet. If that is not possible, mount the power bar under a desk, or on the wall and use duct tape (or gaffer tape, a strong adhesive tape) to secure the plug into the receptacle. On the UPS and power bar, do not leave any empty receptacles. Tape them if necessary. People will have the tendency to use the easiest receptacle, so make these critical ones difficult to use. If you do not, you might find a fan or light plugged into your UPS; though it is nice to have light, it is nicer to keep your server running!

Water

Protect your equipment from water and moisture. In all cases make sure that your equipment, including your UPS is at least 30 cm from the ground, to avoid damage from flooding. Also try to have a roof over your equipment, so that water and moisture will not fall onto it. In moist climates, it is important that the equipment has proper ventilation to assure that moisture can be exhausted. Small closets need to have ventilation, or moisture and heat can degrade or destroy your gear.

Masts

Equipment installed on a mast is often safe from thieves. Nevertheless, to deter thieves and to keep your equipment safe from winds it is good to overengineer mounts. Painting equipment a dull white or grey color reflects the sun and makes it look plain and uninteresting. Panel antennas are often preferred because they are much more subtle and less interesting than dishes. Any installation on walls should be high enough to require a ladder to reach. Try choosing well-lit but not prominent places to put equipment. Also avoid anten-

nae that resemble television antennae, as those are items that will attract interest by thieves, where a wifi antenna will be useless to the average thief.

Threats to the network

One critical difference between Ethernet and wireless is that wireless networks are built on a ***shared medium***. They more closely resemble the old network hubs than modern switches, in that every computer connected to the network can "see" the traffic of every other user. To monitor all network traffic on an access point, one can simply tune to the channel being used, put the network card into monitor mode, and log every frame. This data might be directly valuable to an eavesdropper (including data such as email, voice data, or online chat logs). It may also provide passwords and other sensitive data, making it possible to compromise the network even further. As we'll see later in this chapter, this problem can be mitigated by the use of encryption.

Another serious problem with wireless networks is that its users are relatively ***anonymous***. While it is true that every wireless device includes a unique MAC address that is supplied by the manufacturer, these addresses can often be changed with software. Even when the MAC address is known, it can be very difficult to judge where a wireless user is physically located. Multipath effects, high-gain antennas, and widely varying radio transmitter characteristics can make it impossible to determine if a malicious wireless user is sitting in the next room or is in an apartment building a mile away.

While unlicensed spectrum provides a huge cost savings to the user, it has the unfortunate side effect that ***denial of service*** (***DoS***) attacks are trivially simple. By simply turning on a high powered access point, cordless phone, video transmitter, or other 2.4 GHz device, a malicious person could cause significant problems on the network. Many network devices are vulnerable to other forms of denial of service attacks as well, such as disassociation flooding and ARP table overflows.

Here are several categories of individuals who may cause problems on a wireless network:

- ***Unintentional users***. As more wireless networks are installed in densely populated areas, it is common for laptop users to accidentally associate to the wrong network. Most wireless clients will simply choose any available wireless network when their preferred network is unavailable. The user may then make use of this network as usual, completely unaware that they may be transmitting sensitive data on someone else's network. Malicious people may even take advantage of this by setting up access points in strategic locations, to try to attract unwitting users and capture their data.

The first step in avoiding this problem is educating your users, and stressing the importance of connecting only to known and trusted networks. Many wireless clients can be configured to only connect to trusted networks, or to ask permission before joining a new network. As we will see later in this chapter, users can safely connect to open public networks by using strong encryption.

- ***War drivers***. The "war driving" phenomenon draws its name from the popular 1983 hacker film, "War Games". War drivers are interested in finding the physical location of wireless networks. They typically drive around with a laptop, GPS, and omnidirectional antenna, logging the name and location of any networks they find. These logs are then combined with logs from other war drivers, and are turned into graphical maps depicting the wireless "footprint" of a particular city.

  The vast majority of war drivers likely pose no direct threat to networks, but the data they collect might be of interest to a network cracker. For example, it might be obvious that an unprotected access point detected by a war driver is located inside a sensitive building, such as a government or corporate office. A malicious person could use this information to illegally access the network there. Arguably, such an AP should never have been set up in the first place, but war driving makes the problem all the more urgent. As we will see later in this chapter, war drivers who use the popular program NetStumbler can be detected with programs such as Kismet. For more information about war driving, see sites such as *http://www.wifimaps.com/, http://www.nodedb.com/,* or *http://www.netstumbler.com/ .*

- ***Rogue access points***. There are two general classes of rogue access points: those incorrectly installed by legitimate users, and those installed by malicious people who intend to collect data or do harm to the network. In the simplest case, a legitimate network user may want better wireless coverage in their office, or they might find security restrictions on the corporate wireless network too difficult to comply with. By installing an inexpensive consumer access point without permission, the user opens the entire network up to potential attacks from the inside. While it is possible to scan for unauthorized access points on your wired network, setting a clear policy that prohibits them is very important.

  The second class of rogue access point can be very difficult to deal with. By installing a high powered AP that uses the same ESSID as an existing network, a malicious person can trick people into using their equipment, and log or even manipulate all data that passes through it. Again, if your users are trained to use strong encryption, this problem is significantly reduced.

- ***Eavesdroppers***. As mentioned earlier, eavesdropping is a very difficult problem to deal with on wireless networks. By using a passive monitoring tool (such as Kismet), an eavesdropper can log all network data from a great distance away, without ever making their presence known. Poorly

encrypted data can simply be logged and cracked later, while unencrypted data can be easily read in real time.

If you have difficulty convincing others of this problem, you might want to demonstrate tools such as Etherpeg (*http://www.etherpeg.org/*) or Driftnet (*http://www.ex-parrot.com/~chris/driftnet/*). These tools watch a wireless network for graphical data, such as GIF and JPEG files. While other users are browsing the Internet, these tools simply display all graphics found in a graphical collage. I often use tools such as this as a demonstration when lecturing on wireless security. While you can tell a user that their email is vulnerable without encryption, nothing drives the message home like showing them the pictures they are looking at in their web browser.

Again, while it cannot be completely prevented, proper application of strong encryption will discourage eavesdropping.

This introduction is intended to give you an idea of the problems you are up against when designing a wireless network. Later in this chapter, we will look at tools and techniques that will help you to mitigate these problems.

Authentication

Before being granted access to network resources, users should first be ***authenticated***. In an ideal world, every wireless user would have an identifier that is unique, unchangeable, and cannot be impersonated by other users. This turns out to be a very difficult problem to solve in the real world.

The closest feature we have to a unique identifier is the MAC address. This is the 48-bit number assigned by the manufacturer to every wireless and Ethernet device. By employing ***mac filtering*** on our access points, we can authenticate users based on their MAC address. With this feature, the access point keeps an internal table of approved MAC addresses. When a wireless user tries to associate to the access point, the MAC address of the client must be on the approved list, or the association will be denied. Alternately, the AP may keep a table of known "bad" MAC addresses, and permit all devices that are not on the list.

Unfortunately, this is not an ideal security mechanism. Maintaining MAC tables on every device can be cumbersome, requiring all client devices to have their MAC addresses recorded and uploaded to the APs. Even worse, MAC addresses can often be changed in software. By observing MAC addresses in use on a wireless network, a determined attacker can ***spoof*** (impersonate) an approved MAC address and successfully associate to the AP. While MAC filtering will prevent unintentional users and even most curious individuals from accessing the network, MAC filtering alone cannot prevent attacks from determined attackers.

MAC filters are useful for temporarily limiting access from misbehaving clients. For example, if a laptop has a virus that sends large amounts of spam or other traffic, its MAC address can be added to the filter table to stop the traffic immediately. This will buy you time to track down the user and fix the problem.

Another popular authentication feature of wireless the so-called ***closed network***. In a typical network, APs will broadcast their ESSID many times per second, allowing wireless clients (as well as tools such as NetStumbler) to find the network and display its presence to the user. In a closed network, the AP does not beacon the ESSID, and users must know the full name of the network before the AP will allow association. This prevents casual users from discovering the network and selecting it in their wireless client.

There are a number of drawbacks to this feature. Forcing users to type in the full ESSID before connecting to the network is error prone and often leads to support calls and complaints. Since the network isn't obviously present in site survey tools like NetStumbler, this can prevent your networks from showing up on war driving maps. But it also means that other network builders cannot easily find your network either, and specifically won't know that you are already using a given channel. A conscientious neighbor may perform a site survey, see no nearby networks, and install their own network on the same channel you are using. This will cause interference problems for both you and your neighbor.

Finally, using closed networks ultimately adds little to your overall networks security. By using passive monitoring tools (such as Kismet), a skilled user can detect frames sent from your legitimate clients to the AP. These frames necessarily contain the network name. A malicious user can then use this name to associate to the access point, just like a normal user would.

Encryption is probably the best tool we have for authenticating wireless users. Through strong encryption, we can uniquely identify a user in a manner that is very difficult to spoof, and use that identity to determine further network access. Encryption also has the benefit of adding a layer of privacy by preventing eavesdroppers from easily watching network traffic.

The most widely employed encryption method on wireless networks is ***WEP encryption***. WEP stands for ***wired equivalent privacy***, and is supported by virtually all 802.11a/b/g equipment. WEP uses a shared 40-bit key to encrypt data between the access point and client. The key must be entered on the APs as well as on each of the clients. With WEP enabled, wireless clients cannot associate with the AP until they use the correct key. An eavesdropper listening to a WEP-enabled network will still see traffic and MAC addresses, but the data payload of each packet is encrypted. This provides a fairly good authentication mechanism while also adding a bit of privacy to the network.

WEP is definitely not the strongest encryption solution available. For one thing, the WEP key is shared between all users. If the key is compromised (say, if one user tells a friend what the password is, or if an employee is let go) then changing the password can be prohibitively difficult, since all APs and client devices need to be changed. This also means that legitimate users of the network can still eavesdrop on each others' traffic, since they all know the shared key.

The key itself is often poorly chosen, making offline cracking attempts feasible. Even worse, the implementation of WEP itself is broken in many access points, making it even easier to crack some networks. While manufacturers have implemented a number of extensions to WEP (such as longer keys and fast rotation schemes), these extensions are not part of the standard, and generally will not interoperate between equipment from different manufacturers. By upgrading to the most recent firmware for all of your wireless devices, you can prevent some of the early attacks found in WEP.

WEP can still be a useful authentication tool. Assuming your users can be trusted not to give away the password, you can be fairly sure that your wireless clients are legitimate. While WEP cracking is possible, it is beyond the skill of most users. WEP is quite useful for securing long distance point-to-point links, even on generally open networks. By using WEP on such a link, you will discourage others from associating to the link, and they will likely use other available APs instead. Think of WEP as a handy "keep out" sign for your network. Anyone who detects the network will see that a key is required, making it clear that they are not welcome to use it.

WEPs greatest strength is its interoperability. In order to comply with the 802.11 standards, all wireless devices support basic WEP. While it isn't the strongest method available, it is certainly the most commonly implemented encryption feature. We will look at other more advanced encryption techniques later in this chapter.

For more details about the state of WEP encryption, see these papers:

- http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html
- http://www.cs.umd.edu/~waa/wireless.pdf
- http://www.crypto.com/papers/others/rc4_ksaproc.ps

Another data-link layer authentication protocol is ***Wi-Fi Protected Access***, or ***WPA***. WPA was created specifically to deal with the known problems with WEP mentioned earlier. It provides a significantly stronger encryption scheme, and can use a shared private key, unique keys assigned to each user, or even SSL certificates to authenticate both the client and the access point. Authentication credentials are checked using the 802.1X protocol,

which can consult a third party database such as RADIUS. Through the use of ***Temporal Key Integrity Protocol*** (***TKIP***), keys can be rotated quickly over time, further reducing the likelihood that a particular session can be cracked. Overall, WPA provides significantly better authentication and privacy than standard WEP.

WPA requires fairly recent access point hardware and up-to-date firmware on all wireless clients, as well as a substantial amount of configuration. If you are installing a network in a setting where you control the entire hardware platform, WPA can be ideal. By authenticating both clients and APs, it solves the rogue access point problem and provides many significant advantages over WEP. But in most network settings where the vintage of hardware is mixed and the knowledge of wireless users is limited, WPA can be a nightmare to install. It is for this reason that most sites continue to use WEP, if encryption is used at all.

Captive portals

One common authentication tool used on wireless networks is the ***captive portal***. A captive portal uses a standard web browser to give a wireless user the opportunity to present login credentials. It can also be used to present information (such as an Acceptable Use Policy) to the user before granting further access. By using a web browser instead of a custom program for authentication, captive portals work with virtually all laptops and operating systems. Captive portals are typically used on open networks with no other authentication methods (such as WEP or MAC filters).

To begin, a wireless user opens their laptop and selects the network. Their computer requests a DHCP lease, which is granted. They then use their web browser to go to any site on the Internet.

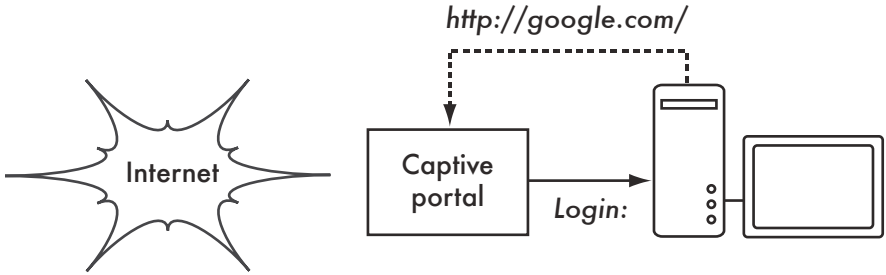

Figure 6.1: The user requests a web page and is redirected.

Instead of receiving the requested page, the user is presented with a login screen. This page can require the user to enter a user name and password, simply click a “login” button, type in numbers from a pre-paid ticket, or enter any other credentials that the network administrators require. The user then

enters their credentials, which are checked by the access point or another server on the network. All other network access is blocked until these credentials are verified.

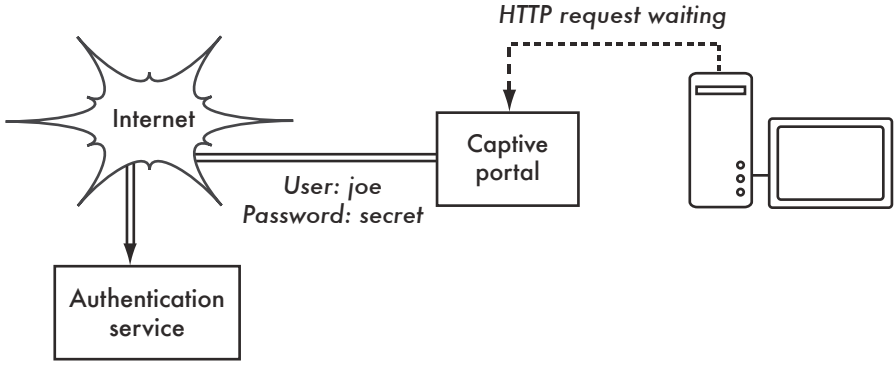

Figure 6.2: The user's credentials are verified before further network access is granted. The authentication server can be the access point itself, another machine on the local network, or a server anywhere on the Internet.

Once authenticated, the user is permitted to access network resources, and is typically redirected to the site they originally requested.

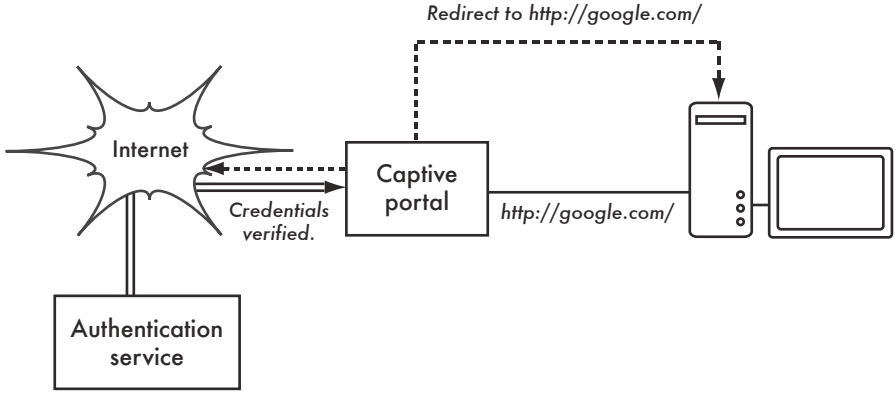

Figure 6.3: After authenticating, the user is permitted to access the rest of the network.

Captive portals provide no encryption for the wireless users, instead relying on the MAC and IP address of the client as a unique identifier. Since this is not necessarily very secure, many implementations will require the user to re-authenticate periodically. This can often be automatically done by minimizing a special pop-up browser window when the user first logs in.

Since they do not provide strong encryption, captive portals are not a very good choice for networks that need to be locked down to only allow access

from trusted users. They are much more suited to cafes, hotels, and other public access locations where casual network users are expected.

In public or semi-public network settings, encryption techniques such as WEP and WPA are effectively useless. There is simply no way to distribute public or shared keys to members of the general public without compromising the security of those keys. In these settings, a simple application such as a captive portal provides a level of service somewhere between completely open and completely closed.

Popular hotspot projects

- Chillispot (*http://www.chillispot.info/*). Chillispot is a captive portal designed to authenticate against an existing user credentials database, such as RADUIS. Combined with the application phpMyPrePaid, pre-paid ticket based authentication can be implemented very easily You can download phpMyPrePaid from *http://sourceforge.net/projects/phpmyprepaid/*.
- WiFi Dog (*http://www.wifidog.org/*). WiFi Dog provides a very complete captive portal authentication package in very little space (typically under 30kb). From a user's perspective, it requires no pop-up or javascript support, allowing it to work on a wider variety of wireless devices.
- m0n0wall (*http://m0n0.ch/wall/*). m0n0wall is a complete embedded operating system based on FreeBSD. It includes a captive portal with RADIUS support, as well as a PHP web server.
- NoCatSplash (*http://nocat.net/download/NoCatSplash/*) provides a customizable splash page to your users, requiring them to click a "login" button before using the network. This is useful for identifying the operators of the network and displaying rules for network access. It provides a very easy solution in situations where you need to provide users of an open network with information and an acceptable use policy.

Privacy

Most users are blissfully unaware that their private email, chat conversations, and even passwords are often sent "in the clear" over dozens of untrusted networks before arriving at their ultimate destination on the Internet. However mistaken they may be, users still typically have some expectation of privacy when using computer networks.

Privacy can be achieved, even on untrusted networks such as public access points and the Internet. The only proven effective method for protecting privacy is the use of strong ***end-to-end encryption***.

Encryption techniques such as WEP and WPA attempt to address the privacy issue at layer two, the data-link layer. This does protect against eavesdroppers listening in on the wireless connection, but this protection ends at the access point. If the wireless client uses insecure protocols (such as POP or simple SMTP for receiving and sending email), then users beyond the AP can still log the session and see the sensitive data. As mentioned earlier, WEP also suffers from the fact that it uses a shared private key. This means that legitimate wireless users can eavesdrop on each other, since they all know the private key.

By using encryption to the remote end of the connection, users can neatly sidestep the entire problem. These techniques work well even on untrusted public networks, where eavesdroppers are listening and possibly even manipulating data coming from the access point.

To ensure data privacy, good end-to-end encryption should provide the following features:

- ***Verified authentication of the remote end***. The user should be able to know without a doubt that the remote end is who it claims to be. Without authentication, a user could give sensitive data to anyone claiming to be the legitimate service.
- ***Strong encryption methods***. The encryption algorithm should stand up to public scrutiny, and not be easily decrypted by a third party. There is no security in obscurity, and strong encryption is even stronger when the algorithm is widely known and subject to peer review. A good algorithm with a suitably large and protected key can provide encryption that is unlikely to be broken by any effort in our lifetimes using current technology.
- ***Public key cryptography***. While not an absolute requirement for end-to-end encryption, the use of public key cryptography instead of a shared key can ensure that an individual's data remains private, even if the key of another user of the service is compromised. It also solves certain problems with distributing keys to users over untrusted networks.
- ***Data encapsulation***. A good end-to-end encryption mechanism protects as much data as possible. This can range from encrypting a single email transaction to encapsulation of all IP traffic, including DNS lookups and other supporting protocols. Some encryption tools simply provide a secure channel that other applications can use. This allows users to run any program they like and still have the protection of strong encryption, even if the programs themselves don't support it.

Be aware that laws regarding the use of encryption vary widely from place to place. Some countries treat encryption as munitions, and may require a permit, escrow of private keys, or even prohibit its use altogether. Before

implementing any solution that involves encryption, be sure to verify that use of this technology is permitted in your local area.

In the following sections, we'll take a look at some specific tools that can provide good protection for your users' data.

SSL

The most widely available end-to-end encryption technology is ***Secure Sockets Layer***, known simply as ***SSL***. Built into virtually all web browsers, SSL uses public key cryptography and a trusted ***public key infrastructure*** (***PKI***) to secure data communications on the web. Whenever you visit a web URL that starts with https, you are using SSL.

The SSL implementation built into web browsers includes a collection of certificates from trusted sources, called ***certificate authorities*** (***CA***). These certificates are cryptographic keys that are used to verify the authenticity of websites. When you browse to a website that uses SSL, the browser and the server first exchange certificates. The browser then verifies that the certificate provided by the server matches its DNS host name, that it has not expired, and that it is signed by a trusted certificate authority. The server optionally verifies the identity of the browser's certificate. If the certificates are approved, the browser and server then negotiate a master session key using the previously exchanged certificates to protect it. That key is then used to encrypt all communications until the browser disconnects. This kind of data encapsulation is known as a ***tunnel***.

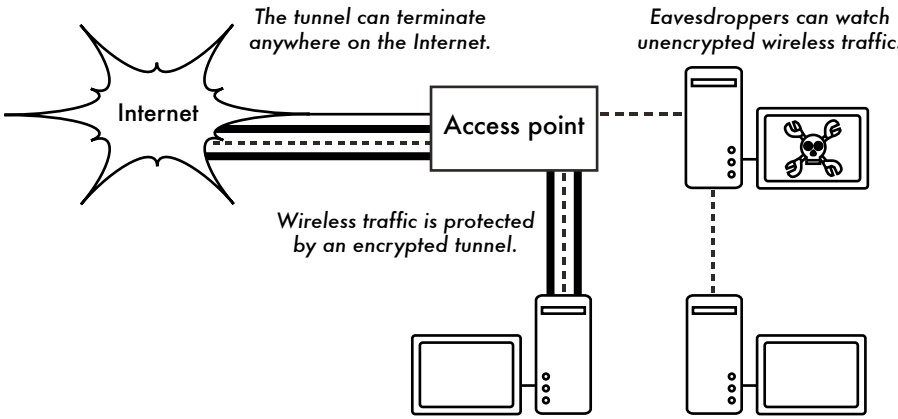

Figure 6.4: Eavesdroppers must break strong encryption to monitor traffic over an encrypted tunnel. The conversation inside the tunnel is identical to any other unencrypted conversation.

The use of certificates with a PKI not only protects the communication from eavesdroppers, but also prevents so-called ***man-in-the-middle*** (***MITM***) at-

tacks. In a man-in-the-middle attack, a malicious user intercepts all communication between the browser and the server. By presenting counterfeit certificates to both the browser and the server, the malicious user could carry on two simultaneous encrypted sessions. Since the malicious user knows the secret on both connections, it is trivial to observe and manipulate data passing between the server and the browser.

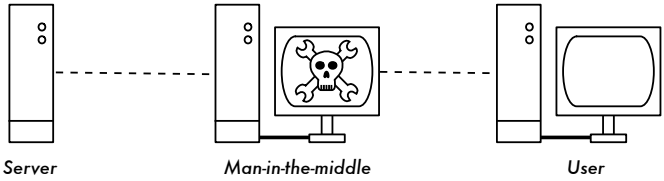

Figure 6.5: The man-in-the-middle effectively controls everything the user sees, and can record and manipulate all traffic. Without a public key infrastructure to verify the authenticity of keys, strong encryption alone cannot protect against this kind of attack.

Use of a good PKI prevents this kind of attack. In order to be successful, the malicious user would have to present a certificate to the client that is signed by a trusted certificate authority. Unless a CA has been compromised (very unlikely) or the user is tricked into accepting the forged certificate, then such an attack is not possible. This is why it is vitally important that users understand that ignoring warnings about expired or improper certificates is very dangerous, especially when using wireless networks. By clicking the "ignore" button when prompted by their browser, users open themselves up to many potential attacks.

SSL is not only used for web browsing. Insecure email protocols such as IMAP, POP, and SMTP can be secured by wrapping them in an SSL tunnel. Most modern email clients support IMAPS and POPS (secure IMAP and POP) as well as SSL/TLS protected SMTP. If your email server does not provide SSL support, you can still secure it with SSL using a package like Stunnel (*http://www.stunnel.org/*). SSL can be used to effectively secure just about any service that runs over TCP.

SSH

Most people think of SSH as a secure replacement for **`telnet`**, just as **`scp`** and **`sftp`** are the secure counterparts of **`rcp`** and **`ftp`**. But SSH is much more than encrypted remote shell. Like SSL, it uses strong public key cryptography to verify the remote server and encrypt data. Instead of a PKI, it uses a key fingerprint cache that is checked before a connection is permitted. It can use passwords, public keys, or other methods for user authentication.

Many people do not know that SSH can also act as a general purpose encrypting tunnel, or even an encrypting web proxy. By first establishing an