Securing Wireless Networks

802.11 Security

O'REILLY®

Bruce Potter & Bob Fleck

802.11 Security

Bruce Potter and Bob Fleck

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Table of Contents

Preface . . . . . **ix**

Part I. 802.11 Security Basics

1. A Wireless World **3**
What Is Wireless? 4
Radio Transmission 5
Inherent Insecurity 7
802.11 8
Structure of 802.11 MAC 12
WEP 13
Problems with WEP 15
Is It Hopeless? 17

2. Attacks and Risks **18**
An Example Network 18
Denial-of-Service Attacks 19
Man-in-the-Middle Attacks 24
Illicit Use 27
Wireless Risks 28
Knowing Is Half the Battle 29

Part II. Station Security

3. Station Security **33**
Client Security Goals 33

Audit Logging 36
Security Updates 37

4. FreeBSD Station Security 38
FreeBSD Client Setup 38

5. Linux Station Security 51
Linux Client Setup 51
Kernel Configuration 51
OS Protection 59
Audit Logging 63
Secure Communication 64

6. OpenBSD Station Security 66
OpenBSD Client Setup 66
Kernel Configuration 66
OS Protection 72
Audit Logging 74

7. Mac OS X Station Security 75
Mac OS X Setup 75
OS Protection 78

8. Windows Station Security 86
Windows Client Setup 86
OS protection 86
Audit Logging 88
Secure Communication 88

Part III. Access Point Security

9. Setting Up an Access Point 91
General Access Point Security 91
Setting Up a Linux Access Point 97
Setting Up a FreeBSD Access Point 102
Setting Up an OpenBSD Access Point 104
Taking It to the Gateway 108

Part IV. Gateway Security

10. Gateway Security 111
Gateway Architecture 111
Secure Installation 112
Firewall Rule Creation 114
Audit Logging 115

11. Building a Linux Gateway 116
Laying Out the Network 116
Building the Gateway 117
Configuring Network Interfaces 119
Building the Firewall Rules 120
MAC Address Filtering 126
DHCP 127
DNS 128
Static ARP 128
Audit Logging 129
Wrapping Up 129

12. Building a FreeBSD Gateway 130
Building the Gateway 130
Building the Firewall Rules 132
Rate Limiting 136
DHCP 136
DNS 137
Static ARP 138
Auditing 138

13. Building an OpenBSD Gateway 139
Building the Gateway 139
Building the Firewall Rules 141
Rate Limiting 144
DHCP 146
DNS 147
Static ARP 147
Auditing 147

14. Authentication and Encryption 149
Portals 149
IPsec VPN 151
802.1x 158

15. Putting It All Together 165
Pieces of a Coherent System 165
User Knowledge 166
Looking Ahead 167

Index 169

CHAPTER 7

Mac OS X Station Security

Apple's Mac OS X operating system has been rapidly gaining in popularity among security professionals. This can most likely be attributed to its excellent GUI, BSD underpinnings, and increased focus on security features. Apple has taken a proactive stance in developing a more secure OS, and is working hand-in-hand with the BSD community to explore secure standards for the BSD family of operating systems.

Mac OS X Setup

The underlying structure of Mac OS X uses many BSD-derived components. Because of this, the configuration, scripts, and firewall are very similar to FreeBSD. File paths are often different, but the concepts remain the same. The examples and walk-throughs in this chapter work on both Mac OS X Versions 10.1 and 10.2.

Kernel Configuration

Mac OS X installs with a pre-compiled kernel that contains support for everything needed to use the OS as a wireless client. There is no need to compile a custom kernel, but if you do want to experiment with different options for the kernel, visit *http://www.opendarwin.org* to get started. The Mac OS X kernel builds are derived from the OpenDarwin kernel but changed somewhat before release by Apple. Building a custom kernel is a path for the more daring, and technical, user.

Card Configuration

Support for the Apple AirPort wireless card is completely integrated into Mac OS X. Configuration is accomplished through the System Preferences dialog boxes. The settings and options are primarily contained in two tabs of the Network section of System Preferences.

Figure 7-1 shows the AirPort configuration tab. The AirPort ID is the MAC address of the wireless card in the computer. The series of options below determine the way the OS will select which wireless network to join at startup or when to come out of a standby mode. The first option joins the network with the strongest signal. The second option will rejoin a recently used network. This option has a checkbox to remember network passwords, which is how Mac OS X refers to WEP keys. The final option restricts the computer to only connecting to a specified network SSID. (The SSID is Wireless in this example.)

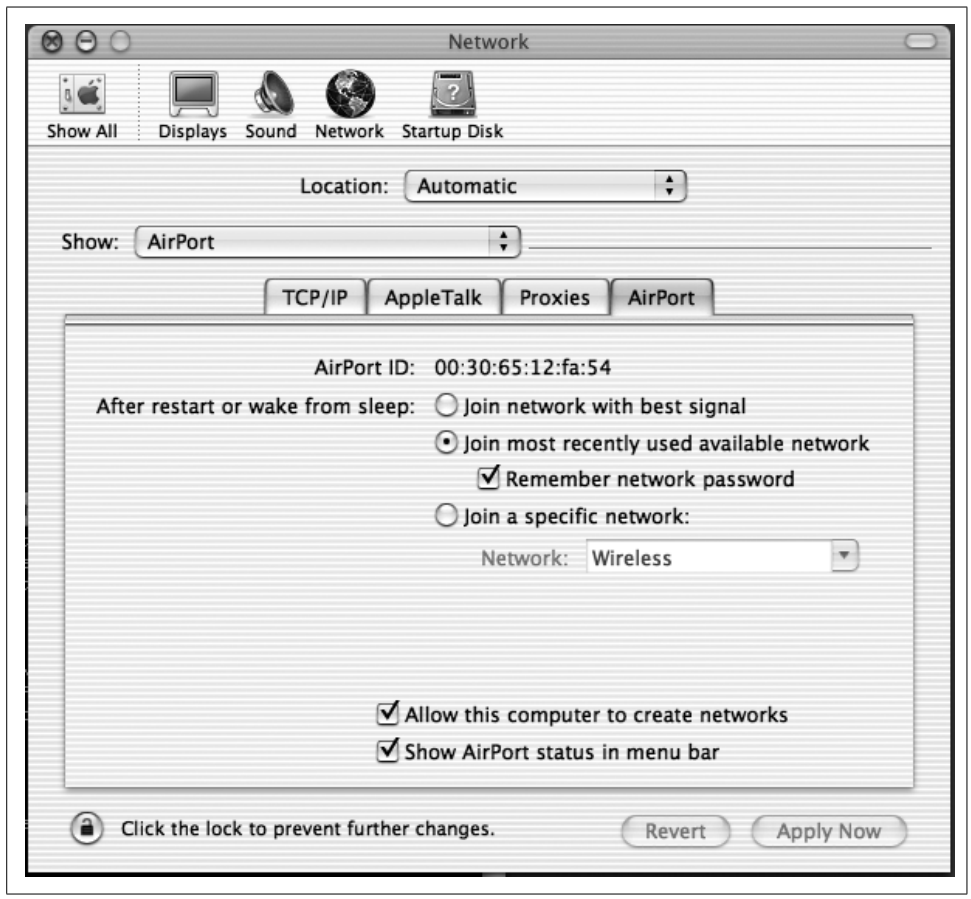

Figure 7-1. AirPort card configuration tab

There are two checkboxes at the bottom of the tab. The first enables the creation of IBSS networks, operating in peer-to-peer mode between workstations. The second adds an icon (Figure 7-2) to the menu bar to display the status of the network connection and provide a small drop-down menu of common actions.

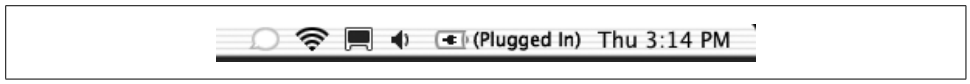

Figure 7-2. AirPort status icon on menu bar (second from left)

Also in the Network section of System Preferences is a tab for the wireless card TCP/IP settings. An example of this tab is shown in Figure 7-3. Make sure you select your wireless card in the Show drop-down list before changing settings. The Configure setting will determine whether DHCP or static address information is to be used. If static settings are used, the rest of the fields in the tab can be used to set the network configuration.

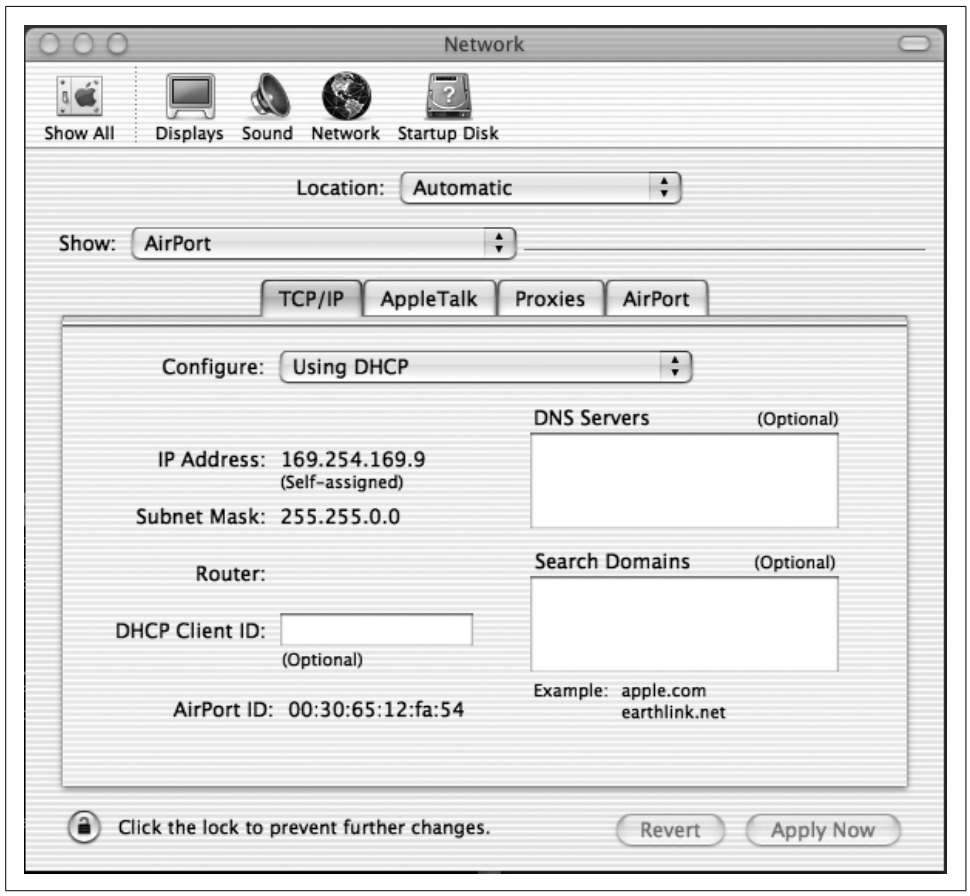

Figure 7-3. TCP/IP networking configuration tab

Besides showing whether the AirPort is connected to a wireless network, the status icon in Figure 7-2 also provides a menu of common actions. All the detected SSIDs of nearby wireless networks are displayed on the menu; clicking on one will cause the

computer to attempt to connect. If the network is closed (requires WEP), a dialog box prompting for the password (WEP key) will appear. The WEP key should be entered in hexadecimal notation, not as regular characters.

There is also a choice titled Create Network. After selecting this, a dialog box will request a network name (SSID), password (WEP key), and channel. The wireless card will create an IBSS network with these settings.

AirPort Access Point Utilities

Mac OS X includes two programs to help in configuring the Apple AirPort AP. If you are using an AirPort AP, the AirPort Admin Utility and AirPort Setup Assistant, which are both located in the Utilities folder, can be used to remotely configure the AP. Since these tools are SNMP based, you may also have (limited) success using them to configure other APs.

OS Protection

In addition to securing the wireless connection itself, it is important to lock down the rest of the services running on the computer. Unneeded services should be disabled, and a firewall should be established to protect the host. It is also worthwhile to set up a static ARP address for the gateway to protect against man-in-the-middle attacks and automate the monitoring of logs.

Disable Unneeded Services

The System Preferences section titled Sharing controls what services will be run on the system. The Services tab, shown in Figure 7-4, controls what file-sharing services will be started on the system. Uncheck any of these you do not need to use. Remote Login is one service you might want to use; it provides a SSH login server to allow remote shell access.

Firewall Configuration

The Firewall tab of the Sharing configuration allows groups of ports to be filtered. Disable access to all services, unless you need to allow access. In the example shown in Figure 7-5, the firewall is off.

You can enable the limited firewalling configuration provided in this GUI setup program by pressing the Start button. If you want to use a more complicated set of filtering rules, it requires creating a script to run at system startup.

If you don't use the GUI firewall configuration, you need to enable the automatic running of a script as a startup item by the *SystemStarter* program. First, create the directory */System/Library/StartupItems/Firewall*. In this directory, place the file

Figure 7-4. Services tab

/System/Library/StartupItems/Firewall/StartupParameters.plist. This file describes the services provided by the startup item, and should contain the following:

```
{
  Description     = "Firewall";
  Provides        = ("Firewall");
  Requires        = ("Network");
  OrderPreference = "Late";
  Messages =
  {
    start = "Starting firewall";
    stop  = "Stopping firewall";
  };
}
```

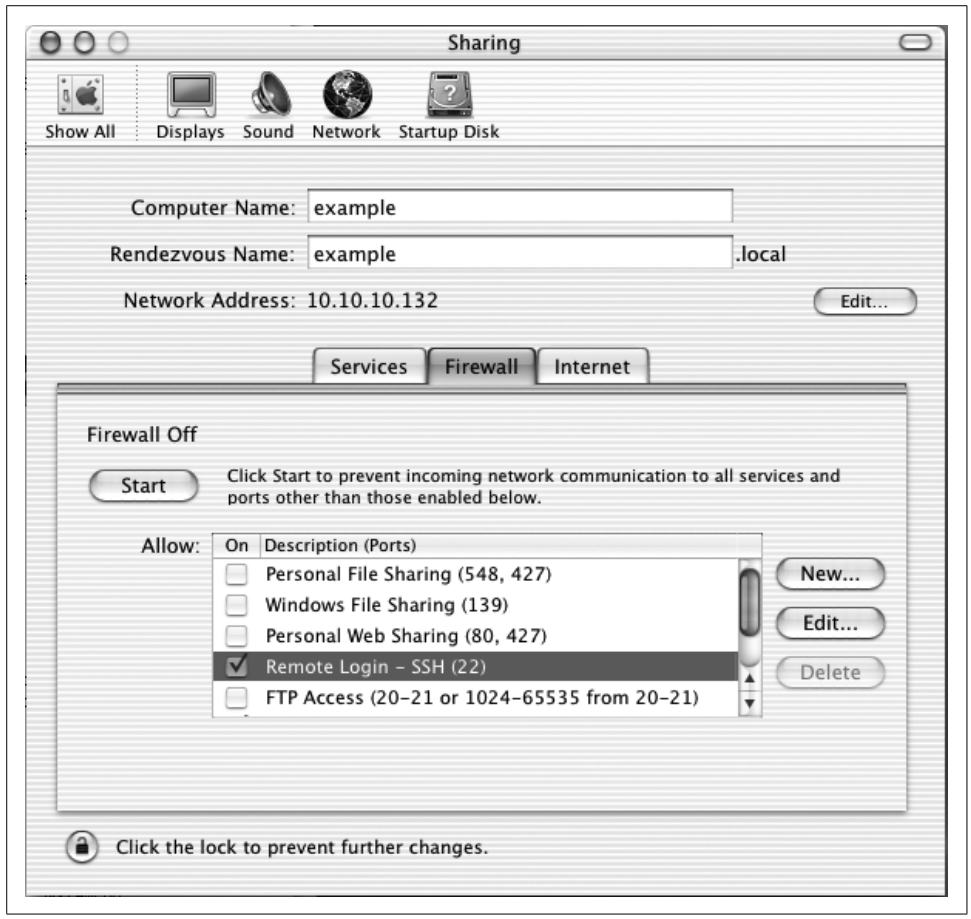

Figure 7-5. Firewall tab

The main script that starts the firewall should have the same name as the directory it lives in. This file will be called */System/Library/StartupItems/Firewall/Firewall*. It writes a log message and then starts a monitoring script. The Firewall script is:

```
#!/bin/sh

##
# Firewall
##

. /etc/rc.common

ConsoleMessage "Starting Firewall"

/usr/local/sbin/fwmon &
echo $! > /var/run/fwmon.pid
```

The monitoring script called by Firewall is */usr/local/sbin/fwmon*. This script will listen for events signaling a change in IP addresses or the restart of network adapters. When one of those events is received, it will call */usr/local/sbin/firewall.sh* to reload the ruleset. Create this file with the following contents:

```
#!/bin/sh

SystemLog( )
{
    local Message="$*"

    logger -it fwmon "${Message}"
}

UpdateFirewall( )
{
    /usr/local/sbin/firewall.sh > /dev/null
    for iface in `ifconfig -lu`; do
     case "${iface}" in
         ppp*)
          ifconfig ${iface} mtu 1448
          ;;
     esac
    done
}

SystemLog "Firewall monitoring started"
UpdateFirewall

while true
do
     /usr/sbin/scutil -p <<-SC_SCRIPT > /dev/null
          open
          n.add State:/Network/Global/IPv4
          n.wait
          close
     SC_SCRIPT

     sleep 3
     SystemLog "IP Address added, removed or changed.  Reconfiguring firewall."
     UpdateFirewall
done

exit 0
```

The final script is */usr/local/sbin/firewall.sh* itself. This file contains the rules that will be assigned to each interface in the machine. It should contain:

```
#!/bin/sh

ipfw=ipfw

AddRule( )
{
     local rule="$*"
```

```
    ${ipfw} add ${rule} via ${iface} > /dev/null
}

LoopbackFirewall( )
{
    AddRule allow all from any to any
}

DefaultFirewall( )
{
    # Prevent spoofing of the loopback network
    AddRule deny log all from any to 127.0.0.0/8

       # Allow DHCP traffic
       AddRule allow udp from any 67 to any 67
       AddRule allow udp from any 67 to any 68
       AddRule allow udp from any 68 to any 67
       AddRule allow udp from any 68 to any 68

    # Prevent bogus addresses
    AddRule deny log all from 0.0.0.0/8 to any in
    AddRule deny log all from 169.254.0.0/16 to any in
    AddRule deny log all from 192.0.2.0/24 to any in
    AddRule deny log all from 224.0.0.0/4 to any in
    AddRule deny log all from 240.0.0.0/4 to any in
    AddRule deny log all from any to 0.0.0.0/8 in
    AddRule deny log all from any to 169.254.0.0/16 in
    AddRule deny log all from any to 192.0.2.0/24 in
    AddRule deny log all from any to 224.0.0.0/4 in
    AddRule deny log all from any to 240.0.0.0/4 in

    # Allow established connections to persist - DANGEROUS, but unavoidable
    # since ipfw sadly does not keep state as of OS X 10.1.
    AddRule allow tcp from any to any established

    # Allow certain ICMP traffic (ping and required stuff)
    AddRule allow icmp from any to any icmptypes 0,3,4,8,11,12

    # Allow DNS traffic in both directions
    AddRule allow udp from any 53 to ${ipaddr} in
    AddRule allow udp from ${ipaddr} to any 53 out

    # Refuse AUTH requests.  Reject the connection rather than deny it so
    # that we don't have to wait for timeouts, etc.
    AddRule reject tcp from any to any 113

    # Allow traffic outgoing from this machine
    AddRule allow tcp from ${ipaddr} to any out
    AddRule allow udp from ${ipaddr} to any out

    # If we're on our home network, allow SMB traffic
    if [ "${domain}" == "yourhomedomainhere.com" ]; then
        AddRule allow tcp from ${ipaddr}:${netmask} 137-139 to any in
        AddRule allow udp from ${ipaddr}:${netmask} 137-139 to any in
```

```
        AddRule allow tcp from ${ipaddr} to ${ipaddr}:${netmask} 137-139 out
        AddRule allow udp from ${ipaddr} to ${ipaddr}:${netmask} 137-139 out
    fi

    # Finally, a default rule to deny
    AddRule deny log ip from any to any
}

# Ensure logging is enabled in the kernel
if [ `/usr/sbin/sysctl -n net.inet.ip.fw.verbose` == 0 ]; then
    /usr/sbin/sysctl -w net.inet.ip.fw.verbose=1 > /dev/null
fi

# Cleanup any mess that may have been left behind from previous invocations
# or manual use of the firewalling interface.
${ipfw} -f flush > /dev/null

# Check the nameserver configuration to find out what our domain is
read junk domain < /etc/resolv.conf

for iface in `ifconfig -lu`; do
    # Get the ip address assigned to the interface.  If there is none, then
    # the interface isn't truly active and we skip it.
    line=`ifconfig ${iface} | grep -E '^[[:space:]]+inet[[:space:]]+.*$'`
    if [ -z "${line}" ]; then
        continue
    fi
    echo "${line}" | read junk ipaddr junk netmask junk

    case "${iface}" in
        lo*)
            LoopbackFirewall
            ;;
        *)
            DefaultFirewall
            ;;
    esac
done

exit 0
```

After creating the scripts, use the following commands to ensure that the files are owned by the proper user and that the three script files are set executable with *chmod*:

```
# chown root:admin /Library/StartupItems/StaticArp/*
# chown root:admin /usr/local/sbin/firewall.sh
# chown root:admin /usr/local/sbin/fwmon
# chmod 755 /Library/StartupItems/StaticArp/StaticArp
# chmod 755 /usr/local/sbin/firewall.sh
# chmod 755 /usr/local/sbin/fwmon
```

The firewall will start upon the next reboot, or it can be started immediately with:

```
# SystemStarter start Firewall
```

Static ARP Entries

To protect against ARP man-in-the-middle attacks, which are described in Chapter 2, set static ARP entries using a startup item script similar to the one described for the firewall.

Create the directory */System/Library/StartupItems/StaticARP*, and place the file */System/Library/StartupItems/StaticARP/StartupParameters.plist* in there with the following contents:

```
{
  Description     = "Static ARP";
  Provides        = ("StaticARP");
  Requires        = ("Network");
  OrderPreference = "Late";
  Messages =
  {
    start = "Starting Static ARP";
    stop  = "Stopping Static ARP";
  };
}
```

The script */System/Library/StartupItems/StaticARP/StaticARP* will add the ARP entry for the gateway when it is started and remove it when stopped. Insert the correct gateway IP and MAC address into the listing below, and place in the file:

```
#!/bin/sh

##
# StaticArp
##

. /etc/rc.common

case "$1" in
    start)
        ConsoleMessage "Adding static ARP entry for gateway"
        /usr/sbin/arp -s <gatewayIP> <gatewayMAC>
    ;;

    stop)
        ConsoleMessage "Deleting static ARP entry for gateway"
        /usr/sbin/arp -d <gatewayIP>
    ;;
esac
```

Set the ownership of these two files, and make the script executable using these commands:

```
# chown root:admin /Library/StartupItems/StaticARP/*
# chmod 755 /Library/StartupItems/StaticARP/StaticARP
```

The static ARP entries will be set upon the next reboot, or they can be set immediately with:

```
# SystemStarter start StaticARP
```

To clear the static ARP entries, use the following command:

```
# SystemStarter stop StaticARP
```

Audit Logging

The messages generated by the firewall and other services are written to the file */var/log/System.log* under Mac OS X. Make sure to review these logs on a regular basis to look for evidence of attacks or compromise. A monitoring tool, such as *swatch*, can help by automating the monitoring.

If you want to use *swatch* as described in Chapter 4, you must start it with the following command:

```
swatch –tail-file=/var/log/System.log –-config-file=swatch.config
```

This will tell *swatch* the path to the *System.log* file, so it knows where to find the logs on Mac OS X. See "Audit Logging" in Chapter 4 for more information on this tool.