BrazilFW WiFi How-To

Version 0.3

17 March 2007

Mac A. Cody

Table of Contents

1 Introduction

This how-to describes the set-up of a computer using Brazil Firewall and Router (BrazilFW)[1] 2.28 to act as a bridged firewall/router with WiFi support. With the availability of WiFi firewall/routers costing less than US$30 these days, it would seem unnecessary to provide such a how-to. Still, there are reasons for making a wireless router yourself rather than buying one off the shelf. In many parts of the world, the benefits of commodity pricing are unavailable. Therefore, inexpensive wireless routers are difficult to obtain. For some people, the learning experience gained and the satisfaction obtained from “I made it myself” inspires them to take on a challenge that would otherwise be unnecessary. Finally, a wireless router of one's own design can possess unique features that would be unavailable in a commercial wireless router.

The following sections present various aspects related to the design and implementation of a wireless firewall/router. Section 2 describes the hardware configuration. Recommended hardware characteristics are provided for the base computer that will become the firewall/router. Recommendations for Ethernet and WiFi network cards are also provided. Section 3 provides a description of the installation of BrazilFW 2.28 a the computer. The installation steps for a basic BrazilFW hard drive install are presented. The general configuration a bridge between networks is also described. Section 4 presents the various packages and configurations required for Wired Equivalent Privacy (WEP) security, WiFi Protected Access (WPA) security, and WiFi Protected Access 2 (WPA2) security. Section 5 is an appendix of supporting information for this how-to.

2 Hardware Configuration

This section describes the computer I assembled to run the firewall/router software. For this how-to, it shall be considered an example implementation, but not a manditory design. The main goal for this implementation is economy. Where possible, components that were on hand were used. Depending upon desired performance and features, a more powerful computer may be needed.

2.1 Base Computer Characteristics

One of the nice aspects of implementing a firewall/router using a Linux-based distribution is that relatively humble hardware reqirements are required to obtain a suitable implementation. It was once commonplace to use an otherwise antiquated 386 or 486-class computer as a firewall/router and still

1 The BrazilFW distribution and supporting information can be obtained at http://www.brazilfw.com.br/

enjoy good performance. It is certain that many are still being used for this purpose. The advent of high-bandwidth Internet services, VPN tunneling, sophisticated packet filtering schemes, and wireless encryption encourages the use of a computer with greater processing capabilities. A more contemporary computer also stands a greater likelihood of supporting a wider variety of data storage and networking options.

2.1.1 Motherboard

For the wireless firewall/router presented here, it is recommended to use a motherboard that supports at least a Pentium-class processor. Nowadays, Pentium and Pentium-II class computers can be obtained at near throw-away prices. The computer I use was cobbled together mostly from components that were lying around and had been obtained at no cost.

The motherboard must have a floppy diskette interface to allow installation of the BrazilFW software. An interface for mass storage, such as an IDE interface or USB interface needs to be present as well. Note that if the mass storage is handled through a USB interface, the motherboard's BIOS must have support for booting off of the USB device.

A sufficient quantity of open bus slots must be available that will support the video and network cards used. Either ISA or PCI bus slots can be used, though PCI bus network cards will support higher data rates. Note that if high-speed network cards will be used, pay close attention to the PCI bus standard supported by the motherboard. Some cards comply with the PCI 2.2 bus standard, which is not supported by many older motherboards.

The motherboard BIOS should allow the computer to operate without a keyboard. Once the wireless firewall/router is completed, a keyboard will be rarely, if ever, used. It will end up just being in the way and cluttering up the physical layout of the network equipment. Most firewall/router maintenance can be peformed though secure shell (i.e. SSH) or a web-based interface. A BIOS that supports computer operation without a video card will be needed if it is desired to remove the video card after installation is completed.

Optional motherboard features are the availability of a parallel port and a serial port. Parallel and serial ports are commonly used to interface to a embedded LCD device that is used for syslog display. A parallel port can also be useful if the firewall/router will also provide print server duties. Serial ports are also used for TTY login to the router as an alternate form of access for maintenance (especially in situations where the computer's video card has been removed).

The motherboard I used for my computer is a Spacewalker HOT-557 motherboard with a Pentium 133 MHz processor (non-MMX). It has floppy and IDE interfaces, three 16-bit ISA bus slots, and four 32-bit PCI 2.2 bus slots. While the HOT-557 has a USB interface, it only supports USB 1.0 standard, which is not usable. The Award BIOS allows for ignoring all errors, ignoring only keyboard errors, or ignoring no errors.

2.1.2 Memory

A BrazilFW installation requires that at least 16 MB of RAM be available on the motherboard. Of course, as features are added to the firewall/router, more system memory will be required. Remember that the RAM will not only hold the running programs, but a RAM disk as well. The RAM disk will contain all executable and support files used by the firewell/router during its operation. Sixty-four megabytes of RAM was installed in my firewall/router computer, which is probably overkill. It will allow plenty of room for a large RAM disk during operation and little concern for running out of RAM space for running programs.

2.1.3 Video

A VGA-compatible video graphics card is needed for the install. The speed of the video card is not at all critical for this application. Neither is graphic support, as the video card will always be run in text mode. The graphics card can be removed afterward, if it desired to run the firewall/router completely “headless”. Remember that the motherboard's BIOS must allow the computer to run if a video card is not present. My firewall/router computer uses an old Cirrus CL-GD5422 ISA VGA card.

2.1.4 Disk Storage

For installation of the BrazilFW software, a floppy diskette drive that supports at least 1.44 MB diskettes is required. Diskette drives of this type are commonly available. They can sometimes be problematic when booting DOS floppies formatted under Linux, so it may be necessary to test and replace as appropriate. Many diskette drives are capable of supporting greater storage capacities via custom formatting procedures. While this can be useful, this capability is not required for installing BrazilFW on a hard disk.

The installation of BrazilFW for the wireless firewall/router will far exceed the storage capacity of a floppy diskette, Consequently, some form of mass storage must be used. Several avenues can be taken to supply mass storage for the computer.

First, a hard disk with at least 32 MB capacity can be used. A hard disk has the advantage of being inexpensive and easy to obtain. The disadvantage is that a hard disk is a mechanical devce that can wear out. This problem can be mitigated by selecting a hard disk that is capable of being put to sleep with the `hdparm` command.

A second choice is a Compact Flash (CF) drive with at least 8 MB of capacity. One advantage of a CF drive is that it looks like an IDE drive to the computer, but there are no mechanical parts to wear out. Another advantage is that the read access time of a CF drive is faster than a hard disk. One disadvantage of a CF drive is that a bootable CF-IDE adapter must be available to interface the CF card with the IDE bus. This adds to the cost of the firewall/router computer if an adapter isn't already available.

A third choice is an Iomega Zip drive or Mitsubishi LS120 SuperDisk drive with an IDE interface. Like a hard disk, Zip drives and SuperDisk drives are mechanical devices that can wear out. The motherboard will also require a BIOS that supports booting of these devices. This choice should be taken only if you already have one of these drives and their corresponding media on hand.

Finally, a USB flash drive with at least 8 MB of capacity can be used. Like a CF drive, a USB flash drive has no moving parts and will boot faster than a hard drive. In order to use a USB flash drive, the motherboard must have a USB interface and the motherboard BIOS must support booting of the USB flash drive.

My firewall/router computer uses a Segate ST3123A hard disk drive. It has a capacity of 105 MB, which is more than enough for a firewall/router application. It can also be put to sleep using the `hdparm` utility.

2.2 Ethernet Cards

Two Ethernet cards are required for this wireless router configuration. One card is for the Wide Area Network (WAN), which is the Internet. Its supported data rate is not critical, as it is communicating with the cable or DSL modem. A 10 Mbps Ethernet card will be sufficient. My firewall/router computer uses a 3COM EtherLink III 3c509B-C.

The other Ethernet card is used for the wired portion of the Local Area Network (LAN). For a router/firewall that only supports a wired LAN, the data rate of this card is usually not critical. In this application, though, the router is also acting as a bridge to a wireless network. If a high-speed wireless network (802.11a/g) is being deployed, then the Ethernet card communicating with the wired side of the LAN needs to support the same data rate as well. Otherwise, a data bottleneck between the wired and wireless LANs will occur at the firewall/router. An Ethernet card that supports at least a 100 Mbps data rate is recommended. With 250 Mbps wireless networking now becoming available, a Gigabit Ethernet card might be the proper choice. My computer uses a 3COM Fast EtherLink XL PCI 3C905B-TX, which will handle 100 Mbps Ethernet.

2.3 Wireless Card

The easiest way to determine which WiFi card to use for the wireless firewall/router is to pick one that is already supported by BrazilFW 2.28. There are currently two approaches to this selection: native Linux driver packages or packages with MS Windows NDIS drivers augmented with the **ndiswrapper** package.

Regardless of the approach chosen, it is recommended that a WiFi card based on either the Atheros chipset or the Prisim chipset should be selected. Both of these WiFi chipsets support WPA security via `hostapd`. If WPA is not needed for the wireless router, then another WiFi card can be chosen. Precompiled drivers for Atheros, Prism, and other WiFi cards can be downloaded from the BrazilFW website in the Download Database.

It is also possible to create a driver package for a WiFi cards not currently supported by BrazilFW. If the package will contain a native Linux driver for the WiFi card, then the BrazilFW 2.27 build tree and the sources for the Linux driver are required. Alternately, a driver package using an NDIS driver for the WiFi card could be created. A discussion of the steps required to create either of these types of WiFi card driver packages is outside the scope of this how-to.

My firewall/router computer uses an AirLink 101 AWLH4030 Super G™ Wireless PCI Adapter. It uses the Atheros chipset and the MadWiFi[2] driver. It will support 802.11a/b/g data rates.

3 BrazilFW Installation and Setup

This section describes the steps taken to perform the BrazilFW firewall/router installation. The initial steps will be covered generally because it will be a regular install. Later steps will be described in more detail, as they are what add the wireless capabilities.

The installation assumes that a Linux workstation is used to create the installation floppies. The approach is similar for the BrazilFW Windows Floppy Creator. Note that BrazilFW package names are specified in bold face with the gzipped-tar extension (.tgz) dropped. For example, the **bridge** package has the file name bridge.tgz.

3.1 Basic BrazilFW Hard Drive Installation

First, let it be stressed that a mass-storage device (hard disk, CF drive, ZIP disk, etc) is required to implement the wireless firewall/router described in this how-to. It is required because of the disk storage requirements for the basic BrazilFW installation, along with the packages needed for the bridge

2 See http://madwifi.org for further information on the MadWiFi drivers.

software, the wireless network card driver, and the wireless security software. See Section 2.1.4 for additional guidance for selecting a mass-storage device.

The following is a brief description of the steps taken to perform a basic BrazilFW “hard drive” installation:

First, create the BrazilFW hard drive installation boot diskette using the `makeinstaller.sh` command. This is a straightforward task; just run the script at the command line and follow the prompts. A blank 1.44 MB floppy diskette is needed. It is recommended that a new diskette be used. It is more difficult to guarantee a successful boot from a worn diskette.

Next, create the BrazilFW floppy boot diskette using the `makefloppy.sh` shell script. The configuration options chosen must result in a boot floppy that will fit on a 1.44 MB floppy diskette. This is due to a limitation of the BrazilFW hard drive installation boot diskette. It expects the BrazilFW floppy boot diskette to be formatted to 1.44 MB capacity[3].

At a minimum, the BrazilFW floppy boot diskette should contain the appropriate WAN client software (DHCP client, DSL client, modem client), the WAN Ethernet card driver, the LAN Ethernet card driver, and the DHCP server. Any other packages that will fit can also be placed of the floppy boot diskette. Admittedly, not too many additional packages will fit on a 1.44 MB floppy diskette. A sample output of the `makefloppy.sh` script is listed in Section 5.1 of the Appendix.

Now, boot the computer using the BrazilFW hard drive installation boot diskette. Follow the directions and prompts provided by the installation scripts. Allocate a sufficiently large amount of the hard disk to the boot partition to at least hold all packages that will be installed. A second partition for proxy storage can be created, but it is optional. If a hard disk is being used, also install the **hdparm** package.

When prompted, insert the BrazilFW floppy boot diskette. The packages installed on this floppy and copied onto the mass-storage device. Once this is completed, remove the floppy boot diskette and reboot the computer. After rebooting, log into the computer and configure and check out the basic installation. Basic BrazilFW configuration is not covered in this how-to.

3.2 Including Additional Packages

With the basic installation completed, other BrazilFW packages can be installed on the computer. There are many packages that come with the

3 It would be nice if the installation scripts on the BrazilFW hard drive installation boot diskette could be changed to provide a prompt to allow the user to select the appropriate device type (/dev/fd0u1440, /dev/fd0u1680, /dev/fd0u1720) for the floppy boot diskette.

BrazilFW distribution and others are also avialable on the BrazilFW website. It is up to the user to determine which packages should be installed. This topic is outside the scope of this How-To.

All packages can be installed via floppy diskette transfers. This can be done by using another computer to copy the packages onto a floppy diskette. It is necessary to create a mount point for the diskette drive using the `mkdir` command.

```
mkdir /floppy
```

Next, mount the floppy diskette, which has a vfat file system, to the mount point using the `mount` command

```
mount -v vfat /dev/fd0 /floppy
```

Then, mount the mass-storage device (e.g. hard drive) to the existing mount point `/mnt` using the command `mt`. Now copy any files from the diskette to the mass-storage device. Afterwards, unmount the diskette using the `umount` command

```
umount /floppy
```

before removing the diskette from the computer. The mass-storage device using the command `umt`.

Packages can also be installed using either ftp or scp, once one of these packages have been installed by diskette and configured. After all packages have been copied to the mass-storage device, reboot the computer and perform any necessary configurations. Don't forget to perform a back up of the BrazilFW packages after configurations are completed so they are not lost at the next reboot.

3.3 Bridge Configuration

In this how-to, the wireless firewall/router will have both a wired LAN and a wireless LAN. In order to enable these two LANs to talk to each other and, in turn, talk to the WAN (Internet), it is necessary to implement an Ethernet bridge. This section provides a general description of how this is done.

A bridge connects two or more networks operating through two or more network cards such that they appear to be one, seamless network. The BrazilFW **bridge** package can be employed on the wireless firewall/router to create a bridge. The BrazilFW **bridge** package uses the `brctl` command to create Ethernet bridge configurations on the Linux kernel.

For the purposes of this how-to, only two brctl commands will be used in the creation of a bridge on the BrazilFW router. The command

```
brctl addbr <name>
```

creates a new instance of the ethernet bridge called `<name>`. The command

```
brctl addif <brname> <ifname>
```

will make the interface `<ifname>` a port of the bridge `<brname>`. These commands, along with the `ifconfig` command are used to configure the physical interfaces `eth0` (wired Ethernet) and `ath0` (wireless Ethernet) as ports on the bridge `br0`.

```
# Create a new instance of a bridge
brctl addbr br0
# Add interface eth0 as a port to the bridge
brctl addif br0 eth0
# Configure interface eth0 to inherit IP address from bridge
ifconfig eth0 0.0.0.0
# Activate interface eth0
ifconfig eth0 up
# Add interface ath0 as a port to the bridge
brctl addif br0 ath0
# Configure interface ath0 to inherit IP address from bridge
ifconfig ath0 0.0.0.0
# Activate interface ath0
ifconfig ath0 up
```

Additional information on the `brctl` and `ifconfig` commands can be found in their respective man pages.

This direct approach to bridge configuration will not work on BrazilFW. This is due to the script execution mechanism used by `/etc/rc.d/rc.modules`. The scripts in the `/etc/rc.d/pkgs` directory are executed in alphabetical order. The configuration script `/etc/rc.d/pkgs/mod.bridge` is executed before the configuration script `/etc/rc.d/pkgs/mod.wireless`. Consequently, the bridge script attempts to add the wireless interface (`ath0` in my computer) to the bridge before it has been created. An approach for configuring the bridge given the constraints of the BrazilFW package system is described in Section 4.2.

4 Wireless Network Installation and Setup

Wireless network cards can be configured to act in several different capacities. On client computers, the wireless card can be configured to operate in “ad hoc” mode, if an informal network between computers is set up, or in “station” mode, if the computers will be communicating through an Access Point (AP). For this wireless firewall/router computer, the WiFi component will act as an Access Point (AP). The following sections will describe how to add wireless capability and properly configure the Ethernet bridge on the BrazilFW installation. Since wireless capability without security is risky nowadays, instructions for setting up wireless security are also provided. Don't forget to perform a back up of the BrazilFW packages after configurations are completed so they are not lost at the next reboot.

4.1 Adding Wireless Network Card Support

The addition and configuration of wireless capability in BrazilFW is a fairly straightforward process. First, make sure that the appropriate wireless package for the WiFi card being used is installed on the computer. Do not forget to also install the **nidswrapper** package, if required. For my computer the **ath** package is used. Next, reboot the computer so that the wireless package is extracted onto the RAM disk. The configuration for the wireless card is found in the shell script file `/etc/rc.d/pkgs/mod.wireless`. At the beginning of the script, the following modules[4] are loaded for my computer's wireless card, an Atheros-based Airlink 101 AWLH4030:

```
# Load the md5 checksum module
insmod /lib/modules/md5.o
# Load the wireless LAN module
insmod /lib/modules/wlan.o
# Load the wireless LAN ACL module
insmod /lib/modules/wlan_acl.o
# Load the the CCMP module
insmod /lib/modules/wlan_ccmp.o
# Load the WEP encryption module
insmod /lib/modules/wlan_wep.o
# Load the TKIP encryption module
insmod /lib/modules/wlan_tkip.o
# Load the X authorization module
insmod /lib/modules/wlan_xauth.o
# Load the hardware access layer
insmod /lib/modules/ath_hal.o
# Load the Atheros rate sampling module
insmod /lib/modules/ath_rate_sample.o
# Load the wireless lan ap scanning module
insmod /lib/modules/wlan_scan_ap.o
# Load the wireless lan station scanning module module
insmod /lib/modules/wlan_scan_sta.o
# Make the default VAP that is created an access point
insmod ath_pci.o autocreate=ap
```

For other wireless cards, the modules that are loaded will be different. Next in the shell script file, the basic wireless interface is configured:

```
# Set up the Network Name (ESSID) of the Access Point
iwconfig ath0 essid Your_ESSID_here
# Set up the channel the Access Point operates on (default is channel 6)
iwconfig ath0 channel 5
```

This completes the basic configuration for the BrazilFW wireless firewall/router. Configuration of wireless client computers is similar, but outside the scope of this how-to. Note that this basic configuration has no

4 The list of modules loaded is subject to change. All of these modules may not be needed to support an Access Point.

wireless security whatsoever. In most situations, this is unsatisfactory, as it leaves the network wide open for access by anyone that is within range with a computer equiped with a wireless network card. Section 4.3 describes how to set up WEP encryption. Section 4.4 describes how to set up WPA security, which is much more secure than WEP. Section 4.5 describes how to set up WPA2 security, which is more secure than WPA.

4.2 Bridge Configuration for BrazilFW

The bridge configuration described in Section 3.3 will not work correctly with a BrazilFW installation. This is due to the script execution mechanism used by the shell script `/etc/rc.d/rc.modules`. During startup, the shell scripts found in the directory `/etc/rc.d/pkgs` are executed in alphabetical order. The shell script `/etc/rc.d/pkgs/mod.bridge` is executed before the shell script `/etc/rc.d/pkgs/mod.wireless`. Consequently, the bridge script attempts to add the wireless interface (`ath0` in my computer) to the bridge before it has been created. A workable bridge configuration approach that works with the BrazilFW package concept is described below.

In the **bridge** package's shell script file `/etc/rc.d/pkgs/mod.bridge`, the following commands are placed at the end:

```
# Create a new instance of an bridge
brctl addbr br0
# Add interface eth0 as a port to the bridge
brctl addif br0 eth0
# Configure interface eth0 to inherit IP address from bridge
ifconfig eth0 0.0.0.0
# Activate interface eth0
ifconfig eth0 up
```

The remainder of the configuration of the bridge is performed in the wireless module shell script file `/etc/rc.d/pkgs/mod.wireless`. For my computer, this is found in the **ath** package. The following commands are placed at the end of the file:

```
# Add interface ath0 as a port to the bridge
brctl addif br0 ath0
# Configure interface ath0 to inherit IP address from bridge
ifconfig ath0 0.0.0.0
# Activate interface ath0
ifconfig ath0 up
```

In the main configuration file `/etc/coyote/coyote.conf` add the following line:

```
IF_LOCAL=br0
```

This causes the `rc.inet1` script to assign the Ethernet bridge the IP address of the LAN gateway (e.g. 192.168.0.1).

4.3 WEP Security Configuration

Wired Equivalent Privacy (WEP) provides limited security to a wireless network. It provides protection from casual association with the AP by a neighbor with a computer containing a wireless network card. WEP is not secure against a determined war driver.

WiFi cards that support WEP encryption usually provide both 40-bit and 104-bit encryption keys. It has been determined, though, that it is not exponentially more secure to use a 104-bit WEP key than a 40-bit WEP key. This is due to the fact that the initialization vector (IV) is the same length (24-bits) for both key types. Nevertheless, it is better to use WEP to provide some form of wireless security than none at all and a 104-bit WEP key is better than a 40-bit WEP key.

Configuring WEP security is done by adding the command

```
iwconfig <ifname> key <WEPkey>
```

(where `ifname` is the wireless card interface and `WEPkey` is the WEP encryption key) into the shell script file `/etc/rc.d/pkgs/mod.wireless`. The command is placed immediately after the basic wireless interface configuration performed in Section 4.1. If an ASCII string is used for the WEP encryption key, the command is specified as follows:

```
iwconfig ath0 key “s:abcdefghijklm”
```

The 13-character string (after the s:) represents a 104-bit WEP key. If the WEP encryption key is specified as a hexidecimal character sequence, then the command is specified as follows:

```
iwconfig ath0 key 01234abcde
```

for a WEP 40-bit key or

```
iwconfig ath0 key 0123456789afcdef0123456789
```

for a WEP 104-bit key. See the `iwconfig` man page for additional details on WEP encryption keys.

4.4 WPA Security Configuration

WiFi Protected Access (WPA) is much more secure than WEP. In order to use WPA security on BrazilFW, the **hostapd** package needs to be installed on the computer. The package uses hostapd[5] 0.4.9. Currently, the WiFi card used must be based on either the atheros or prism chipsets. Once the **hostapd** package is installed, the computer needs to be rebooted so that hostapd can be configured.

5 See the “Host AP driver for Intersil Prism2/2.5/3, hostapd, and WPA Supplicant” website at http://hostap.epitest.fi/

The configuration file for hostapd, `/etc/hostapd.conf`, is very large. This is because it is very heavily documented. A complete listing of the file is presented in Section 5.2 of the Appendix. The example below is the configuration to support the Atheros-based WiFi card used in my wireless firewall router.

```
# An additional configuration parameter, bridge,
# must be used to notify hostapd if the interface is included in a bridge.

# Enable this for standard bridging, leave disabled for netfilter firewalls
bridge=br0

interface=ath0
driver=madwifi
logger_syslog=-1
logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2
debug=0
ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
macaddr_acl=0
accept_mac_file=/etc/hostapd.accept
deny_mac_file=/etc/hostapd.deny
auth_algs=1
eap_server=0
dump_file=/tmp/hostapd.dump
ssid=XXXXXXXXXXXXXXXX
wpa=1
wpa_psk=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
```

The files `/etc/hostapd.accept` and `/etc/hostapd.deny` contain, respectively, lists of MAC addresses belonging to WiFi cards that are either authorized or unauthorized to operate on the wireless network. The `ssid` entry is the same network name set up in Section 4.1. The `wpa_psk` entry is a passkey with up to 64 hexidecimal characters (256-bits). If the passkey is surrounded by double quotes, then up to 32 ASCII characters are needed. The `wpa_key_mgmt` entry specifies that passkey security (WPA Personal) is being used on the networks. The `wpa_pairwise` entry specifies that only Temporal Key Integrity Protocol (TKIP) will be allowed on the network[6]. Once all of the configuration parameters have been set, don't forget to back up the package files to the mass storage device.

If a client wireless computer is running Linux, then wpa_supplicant[7] will probably be used to provide WPA support. While this is not actually part of

6 It is uncertain whether this restriction is needed. This may be revised in the future.

7 See the “Linux WPA/WPA2/IEEE 802.1X Supplicant” web page at http://hostap.epitest.fi/wpa_supplicant/

the BrazilFW wireless firewall/router configuration, an example client configuration file `/etc/wpa_supplicant.conf` is provided below for completeness:

```
# See /usr/doc/wpa_supplicant-0.4.7/wpa_supplicant.conf.sample
# for many more options that you can use in this file.

# This line enables the use of wpa_cli which is used by rc.wireless
# if possible (to check for successful association)
ctrl_interface=/var/run/wpa_supplicant
# By default, only root (group 0) may use wpa_cli
ctrl_interface_group=0
ap_scan=1

# WPA protected network, supply your own ESSID and WPAPSK here:
network={
scan_ssid=1
ssid="XXXXXXXXXXXXXXXXX"
proto=WPA
key_mgmt=WPA-PSK
pairwise=TKIP
psk=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
}

# Plaintext connection (no WPA, no IEEE 802.1X),
# nice for hotel/airport types of WiFi network.
# You'll need a recent version of wireless-tools for this!
#network={
# ssid="any"
# key_mgmt=NONE
# priority=2
#}
```

4.5 WPA2 Security Configuration

This section will be generated at a later time.

4.6 Web-Based Wireless Configuration

This section will be generated at a later time.

5 Appendix

The Appendix contains supporting information for this how-to.

5.1 BrazilFW Floppy Generation

Below is an example output of `makefloppy.sh` script used for the Linux-based creation of a BrazilFW boot floppy. Note that the floppy generated is used solely as part of the BrazilFW hard drive installation procedure. Personal information, such as the DHCP hostname, has been obscured.

```
bash-2.05b# ./makefloppy.sh


BrazilFW floppy builder script v2.9

Please choose the desired capacity for the created floppy:

1) 1.44Mb (Safest and most reliable but may lack space needed for
          some options)
2) 1.68Mb (Good reliability with extra space) - recommended
3) 1.72Mb (Most space but may not work on all systems or with all
          diskettes)

Enter selection: 1

Please select the type of Internet connection that your system uses.

1) Standard Ethernet Connection
2) PPP over Ethernet Connection
3) PPP Dialup Connection

Enter Selection: 1

Configuring system for Ethernet based Internet connection.


By default, BrazilFW uses the following settings for the local network
interface:

IP Address: 192.168.0.1
Netmask:    255.255.255.0
Broadcast:  192.168.0.255
Network:    192.168.0.0

Would you like to change these settings? [Y/N]: n

Does your Internet connection get its IP via DHCP? [y/n]: y

Some areas require you to specify a hostname when obtaining a DHCP address.
This hostname would have been given to you when you originally had
your Internet access installed.  If you have been supplied with a hostname
for your computer, enter it here. If not, simply press enter.

Enter your DHCP hostname: xxxxxxxxxx

Do you want to enable the coyote DHCP server? [y/n]: y
Enter DHCP range starting IP [192.168.0.100]:
Enter DHCP range ending IP [192.168.0.200]:

If you don't know what a DMZ is, just answer NO
You you like to configure a De-Militarized Zone? [Y/N]: n

You now need to specify the module name and parameters for your network cards.
```

```
If you are using PCI or EISA cards, leave the IO and IRQ lines blank.

Enter the module name for you local network card: 3c59x
Enter IO address (Leave blank for PCI cards):
Enter IRQ (Leave blank for PCI cards):

Enter the module name for your Internet network card: 3c509
Enter IO address (Leave blank for PCI cards):
Enter IRQ (Leave blank for PCI cards):
Checking module deps for (3c59x,3c509)...
Module 3c509 dep =
Module 3c59x dep =
Copying module: drivers/3c59x.o
Copying module: drivers/3c509.o

The default language of the BrazilFW Web Administrator is English
Do you like to configure a different language ? [Y/N]: n

If you have a syslog server on your LAN you want BrazilFW to send its syslog
data to, you can specify the address here. If unsure or you do not have a
syslog server, leave this entry blank.

Syslog server address:

##### ATTENTION! #####
You will be presented to some extra packages now.
You should not to install all of them because they might not fit
to the floppy. Do not install what you don't need


*** L7-Filter
L7-filter is a nice way to block Peer-to-pear programs like
kazaa, e-mule and bittorrent, it has a extense list of protocols
you can use to match your traffic. If you're unsure, answer YES.
Do you add want to add the L7-Filter package? [y/n]: n

*** Wireless Tools
These are the utilities you need to add a Wi-fi Network Interface
Interface to your system. This package is not the NIC Driver, you
need to add the driver package according on your card chipset also.
If you're unsure, answer YES only if you planning to have wireless.
Do you add want to add the Wireless Tools package? [y/n]: n

*** Advanced Router
This package add some extra feature to your system, like
802.1Q Vlan Support, CBQ Based QOS, and some aditional matches
to iptables. Usually, this is used by professionals, if you're
unsure, answer NO.
Do you add want to add the Advanced Router package? [y/n]: n

*** Bridged Firewalling (ebtables)
This package allows you to add a bridge to your system
Usually, this is used by professionals, if you're unsure, answer NO.
Do you add want to add the Bridge package? [y/n]: y
```

```
*** PPTPD - Point to Point Tunneling Protocol Server
PPTPD uses a client-server model for establishing VPN connections
Do you add want to add the PPTP package? [y/n]: n
Creating Bridge Package
Building package: bridge
Building package: etc
Building package: modules
Building package: root
Building package: dhcpd
Building package: webadmin

Make sure that you have a floppy in the first floppy drive
in this system and press enter to continue...

Formatting /dev/fd0u1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
bin/mkdosfs 2.2 (06 Jul 1999)
Installing boot loader...
Copying files...
`floppy/SYSLINUX.DPY' -> `mnt/SYSLINUX.DPY'
`floppy/bridge.tgz' -> `mnt/bridge.tgz'
cp: omitting directory `floppy/config'
`floppy/dhcpd.tgz' -> `mnt/dhcpd.tgz'
`floppy/etc.tgz' -> `mnt/etc.tgz'
`floppy/linux' -> `mnt/linux'
`floppy/modules.tgz' -> `mnt/modules.tgz'
`floppy/root.tgz' -> `mnt/root.tgz'
`floppy/syslinux.cfg' -> `mnt/syslinux.cfg'
`floppy/webadmin.tgz' -> `mnt/webadmin.tgz'
`floppy/config/coyote.cfg' -> `mnt/config/coyote.cfg'
`floppy/config/cron.cfg' -> `mnt/config/cron.cfg'
`floppy/config/fireloc.cfg' -> `mnt/config/fireloc.cfg'
`floppy/config/firewall.cfg' -> `mnt/config/firewall.cfg'
`floppy/config/hosts.dns' -> `mnt/config/hosts.dns'
`floppy/config/portfw.cfg' -> `mnt/config/portfw.cfg'
`floppy/config/qosclass.cfg' -> `mnt/config/qosclass.cfg'
`floppy/config/qosfilt.cfg' -> `mnt/config/qosfilt.cfg'
`floppy/config/rclocal.cfg' -> `mnt/config/rclocal.cfg'
`floppy/config/reserve.cfg' -> `mnt/config/reserve.cfg'
`floppy/config/subnet.cfg' -> `mnt/config/subnet.cfg'

Would you like to create another copy of this disk [y/n]? n
Done.
bash-2.05b#
```

5.2 Configuration File for hostapd

Below is the hostapd.conf file that comes with hostapd 0.4.9. It has many detailed comments explaining the various configuration options.

```
##### hostapd configuration file ##############################################
```

```
# Empty lines and lines starting with # are ignored

# AP netdevice name (without 'ap' postfix, i.e., wlan0 uses wlan0ap for
# management frames); ath0 for madwifi
interface=wlan0

# In case of madwifi driver, an additional configuration parameter, bridge,
# must be used to notify hostapd if the interface is included in a bridge.
# This parameter is not used with Host AP driver.
#bridge=br0

# Driver interface type (hostap/wired/madwifi/prism54; default: hostap)
# driver=hostap

# hostapd event logger configuration
#
# Two output method: syslog and stdout (only usable if not forking to
# background).
#
# Module bitfield (ORed bitfield of modules that will be logged; -1 = all
# modules):
# bit 0 (1) = IEEE 802.11
# bit 1 (2) = IEEE 802.1X
# bit 2 (4) = RADIUS
# bit 3 (8) = WPA
# bit 4 (16) = driver interface
# bit 5 (32) = IAPP
#
# Levels (minimum value for logged events):
#  0 = verbose debugging
#  1 = debugging
#  2 = informational messages
#  3 = notification
#  4 = warning
#
logger_syslog=-1
logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2

# Debugging: 0 = no, 1 = minimal, 2 = verbose, 3 = msg dumps, 4 = excessive
debug=0

# Dump file for state information (on SIGUSR1)
dump_file=/tmp/hostapd.dump

# Interface for separate control program. If this is specified, hostapd
# will create this directory and a UNIX domain socket for listening to
requests
# from external programs (CLI/GUI, etc.) for status information and
# configuration. The socket file will be named based on the interface name, so
# multiple hostapd processes/interfaces can be run at the same time if more
# than one interface is used.
# /var/run/hostapd is the recommended directory for sockets and by default,
# hostapd_cli will use it when trying to connect with hostapd.
ctrl_interface=/var/run/hostapd

# Access control for the control interface can be configured by setting the
# directory to allow only members of a group to use sockets. This way, it is
# possible to run hostapd as root (since it needs to change network
# configuration and open raw sockets) and still allow GUI/CLI components to be
# run as non-root users. However, since the control interface can be used to
# change the network configuration, this access needs to be protected in many
# cases. By default, hostapd is configured to use gid 0 (root). If you
# want to allow non-root users to use the contron interface, add a new group
# and change this value to match with that group. Add users that should have
# control interface access to this group.
#
# This variable can be a group name or gid.
#ctrl_interface_group=wheel
ctrl_interface_group=0


##### IEEE 802.11 related configuration ######################################

# SSID to be used in IEEE 802.11 management frames
ssid=test

# Station MAC address -based authentication
# 0 = accept unless in deny list
# 1 = deny unless in accept list
# 2 = use external RADIUS server (accept/deny lists are searched first)
macaddr_acl=0

# Accept/deny lists are read from separate files (containing list of
# MAC addresses, one per line). Use absolute path name to make sure that the
# files can be read on SIGHUP configuration reloads.
#accept_mac_file=/etc/hostapd.accept
#deny_mac_file=/etc/hostapd.deny

# IEEE 802.11 specifies two authentication algorithms. hostapd can be
# configured to allow both of these or only one. Open system authentication
# should be used with IEEE 802.1X.
# Bit fields of allowed authentication algorithms:
# bit 0 = Open System Authentication
# bit 1 = Shared Key Authentication (requires WEP)
auth_algs=3

# Associate as a station to another AP while still acting as an AP on the same
# channel.
#assoc_ap_addr=00:12:34:56:78:9a


##### IEEE 802.1X-2004 related configuration
##################################

# Require IEEE 802.1X authorization
#ieee8021x=1
```

```
# IEEE 802.1X/EAPOL version
# hostapd is implemented based on IEEE Std 802.1X-2004 which defines EAPOL
# version 2. However, there are many client implementations that do not handle
# the new version number correctly (they seem to drop the frames completely).
# In order to make hostapd interoperate with these clients, the version number
# can be set to the older version (1) with this configuration value.
#eapol_version=2

# Optional displayable message sent with EAP Request-Identity. The first \0
# in this string will be converted to ASCII-0 (nul). This can be used to
# separate network info (comma separated list of attribute=value pairs); see,
# e.g., draft-adrangi-eap-network-discovery-07.txt.
#eap_message=hello
#eap_message=hello\0networkid=netw,nasid=foo,portid=0,NAIRealms=example.com

# WEP rekeying (disabled if key lengths are not set or are set to 0)
# Key lengths for default/broadcast and individual/unicast keys:
# 5 = 40-bit WEP (also known as 64-bit WEP with 40 secret bits)
# 13 = 104-bit WEP (also known as 128-bit WEP with 104 secret bits)
#wep_key_len_broadcast=5
#wep_key_len_unicast=5
# Rekeying period in seconds. 0 = do not rekey (i.e., set keys only once)
#wep_rekey_period=300

# EAPOL-Key index workaround (set bit7) for WinXP Supplicant (needed only if
# only broadcast keys are used)
eapol_key_index_workaround=0

# EAP reauthentication period in seconds (default: 3600 seconds; 0 = disable
# reauthentication).
#eap_reauth_period=3600

# Use PAE group address (01:80:c2:00:00:03) instead of individual target
# address when sending EAPOL frames with driver=wired. This is the most common
# mechanism used in wired authentication, but it also requires that the port
# is only used by one station.
#use_pae_group_addr=1

##### Integrated EAP server ###################################################

# Optionally, hostapd can be configured to use an integrated EAP server
# to process EAP authentication locally without need for an external RADIUS
# server. This functionality can be used both as a local authentication server
# for IEEE 802.1X/EAPOL and as a RADIUS server for other devices.

# Use integrated EAP server instead of external RADIUS authentication
# server. This is also needed if hostapd is configured to act as a RADIUS
# authentication server.
eap_server=0

# Path for EAP server user database
#eap_user_file=/etc/hostapd.eap_user
```

```
# CA certificate (PEM or DER file) for EAP-TLS/PEAP/TTLS
#ca_cert=/etc/hostapd.ca.pem

# Server certificate (PEM or DER file) for EAP-TLS/PEAP/TTLS
#server_cert=/etc/hostapd.server.pem

# Private key matching with the server certificate for EAP-TLS/PEAP/TTLS
# This may point to the same file as server_cert if both certificate and key
# are included in a single file. PKCS#12 (PFX) file (.p12/.pfx) can also be
# used by commenting out server_cert and specifying the PFX file as the
# private_key.
#private_key=/etc/hostapd.server.prv

# Passphrase for private key
#private_key_passwd=secret passphrase

# Enable CRL verification.
# Note: hostapd does not yet support CRL downloading based on CDP. Thus, a
# valid CRL signed by the CA is required to be included in the ca_cert file.
# This can be done by using PEM format for CA certificate and CRL and
# concatenating these into one file. Whenever CRL changes, hostapd needs to be
# restarted to take the new CRL into use.
# 0 = do not verify CRLs (default)
# 1 = check the CRL of the user certificate
# 2 = check all CRLs in the certificate path
#check_crl=1

# Configuration data for EAP-SIM database/authentication gateway interface.
# This is a text string in implementation specific format. The example
# implementation in eap_sim_db.c uses this as the file name for the GSM
# authentication triplets.
#eap_sim_db=/etc/hostapd.sim_db


##### IEEE 802.11f - Inter-Access Point Protocol (IAPP)
#######################

# Interface to be used for IAPP broadcast packets
#iapp_interface=eth0


##### RADIUS client configuration
#############################################
# for IEEE 802.1X with external Authentication Server, IEEE 802.11
# authentication with external ACL for MAC addresses, and accounting

# The own IP address of the access point (used as NAS-IP-Address)
own_ip_addr=127.0.0.1

# Optional NAS-Identifier string for RADIUS messages. When used, this should
# be a unique to the NAS within the scope of the RADIUS server. For example, a
# fully qualified domain name can be used here.
#nas_identifier=ap.example.com
```

```
# RADIUS authentication server
#auth_server_addr=127.0.0.1
#auth_server_port=1812
#auth_server_shared_secret=secret

# RADIUS accounting server
#acct_server_addr=127.0.0.1
#acct_server_port=1813
#acct_server_shared_secret=secret

# Secondary RADIUS servers; to be used if primary one does not reply to
# RADIUS packets. These are optional and there can be more than one secondary
# server listed.
#auth_server_addr=127.0.0.2
#auth_server_port=1812
#auth_server_shared_secret=secret2
#
#acct_server_addr=127.0.0.2
#acct_server_port=1813
#acct_server_shared_secret=secret2

# Retry interval for trying to return to the primary RADIUS server (in
# seconds). RADIUS client code will automatically try to use the next server
# when the current server is not replying to requests. If this interval is
# set, primary server will be retried after configured amount of time even if
# the currently used secondary server is still working.
#radius_retry_primary_interval=600


# Interim accounting update interval
# If this is set (larger than 0) and acct_server is configured, hostapd will
# send interim accounting updates every N seconds. Note: if set, this
# overrides possible Acct-Interim-Interval attribute in Access-Accept
# message. Thus, this value should not be configured in hostapd.conf, if
# RADIUS server is used to control the interim interval.
# This value should not be less 600 (10 minutes) and must not be less than
# 60 (1 minute).
#radius_acct_interim_interval=600


##### RADIUS authentication server configuration #############################

# hostapd can be used as a RADIUS authentication server for other hosts. This
# requires that the integrated EAP authenticator is also enabled and both
# authentication services are sharing the same configuration.

# File name of the RADIUS clients configuration for the RADIUS server. If this
# commented out, RADIUS server is disabled.
#radius_server_clients=/etc/hostapd.radius_clients

# The UDP port number for the RADIUS authentication server
#radius_server_auth_port=1812
```

```
# Use IPv6 with RADIUS server (IPv4 will also be supported using IPv6 API)
#radius_server_ipv6=1


##### WPA/IEEE 802.11i configuration ##########################################

# Enable WPA. Setting this variable configures the AP to require WPA (either
# WPA-PSK or WPA-RADIUS/EAP based on other configuration). For WPA-PSK, either
# wpa_psk or wpa_passphrase must be set and wpa_key_mgmt must include WPA-PSK.
# For WPA-RADIUS/EAP, ieee8021x must be set (but without dynamic WEP keys),
# RADIUS authentication server must be configured, and WPA-EAP must be
# included in wpa_key_mgmt.
# This field is a bit field that can be used to enable WPA (IEEE 802.11i/D3.0)
# and/or WPA2 (full IEEE 802.11i/RSN):
# bit0 = WPA
# bit1 = IEEE 802.11i/RSN (WPA2) (dot11RSNAEnabled)
#wpa=1

# WPA pre-shared keys for WPA-PSK. This can be either entered as a 256-bit
# secret in hex format (64 hex digits), wpa_psk, or as an ASCII passphrase
# (8..63 characters) that will be converted to PSK. This conversion uses SSID
# so the PSK changes when ASCII passphrase is used and the SSID is changed.
# wpa_psk (dot11RSNAConfigPSKValue)
# wpa_passphrase (dot11RSNAConfigPSKPassPhrase)
#wpa_psk=0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
#wpa_passphrase=secret passphrase

# Optionally, WPA PSKs can be read from a separate text file (containing list
# of (PSK,MAC address) pairs. This allows more than one PSK to be configured.
# Use absolute path name to make sure that the files can be read on SIGHUP
# configuration reloads.
#wpa_psk_file=/etc/hostapd.wpa_psk

# Set of accepted key management algorithms (WPA-PSK, WPA-EAP, or both). The
# entries are separated with a space.
# (dot11RSNAConfigAuthenticationSuitesTable)
#wpa_key_mgmt=WPA-PSK WPA-EAP

# Set of accepted cipher suites (encryption algorithms) for pairwise keys
# (unicast packets). This is a space separated list of algorithms:
# CCMP = AES in Counter mode with CBC-MAC [RFC 3610, IEEE 802.11i/D7.0]
# TKIP = Temporal Key Integrity Protocol [IEEE 802.11i/D7.0]
# Group cipher suite (encryption algorithm for broadcast and multicast frames)
# is automatically selected based on this configuration. If only CCMP is
# allowed as the pairwise cipher, group cipher will also be CCMP. Otherwise,
# TKIP will be used as the group cipher.
# (dot11RSNAConfigPairwiseCiphersTable)
#wpa_pairwise=TKIP CCMP

# Time interval for rekeying GTK (broadcast/multicast encryption keys) in
# seconds. (dot11RSNAConfigGroupRekeyTime)
#wpa_group_rekey=600

# Rekey GTK when any STA that possesses the current GTK is leaving the BSS.
# (dot11RSNAConfigGroupRekeyStrict)
#wpa_strict_rekey=1

# Time interval for rekeying GMK (master key used internally to generate GTKs
# (in seconds).
#wpa_gmk_rekey=86400

# Enable IEEE 802.11i/RSN/WPA2 pre-authentication. This is used to speed up
# roaming be pre-authenticating IEEE 802.1X/EAP part of the full RSN
# authentication and key handshake before actually associating with a new AP.
# (dot11RSNAPreauthenticationEnabled)
#rsn_preauth=1
#
# Space separated list of interfaces from which pre-authentication frames are
# accepted (e.g., 'eth0' or 'eth0 wlan0wds0'. This list should include all
# interface that are used for connections to other APs. This could include
# wired interfaces and WDS links. The normal wireless data interface towards
# associated stations (e.g., wlan0) should not be added, since
# pre-authentication is only used with APs other than the currently associated
# one.
#rsn_preauth_interfaces=eth0
```

5.3 Additional Materials

Additional materials will be added as needed.