Implement strong WiFi encryption the easy way with hostapd

Secure and protect your open WiFi access with WPA2 encryption

Carla Schroder
Linux administrator and author
Tux Computing

10 April 2012

Keep wireless security simple. hostapd, the Host Access Point daemon provides solid WiFi encryption that meets enterprise standards without all the overhead of running FreeRADIUS. Learn more about this tool and how to incorporate it into your environment.

Introduction

hostapd, the Host Access Point daemon, provides strong WPA2 encryption and authentication on Linux-based wireless access points. It is fairly simple to configure, supports WPA2-Personal and Enterprise, and also provides a unique modification to WPA2-Personal that makes it both strong and simple to administer.

The gap between WPA2-Personal and WPA2-Enterprise is rather large. WPA2-Enterprise is the strongest wireless security, but it is complex to administer because it requires a public key infrastructure (PKI) with server and client certificates, and a certificate authority. Most shops use FreeRADIUS servers to manage all this, which is overkill when you have a small number of access points to manage, or want to set one up for a temporary event or project.

WPA2-Personal is supposed to be both easy and strong for small shops because it uses a single shared key for all users, which is easy to roll out but presents ongoing security and administration hassles. There is no good way to remove a single user, because any time the key is changed the new key has to be distributed to all users, and you can't keep unwanted users out because they only need one friend on the inside to get the new key. This is better suited for a semi-public hotspot; for example, you want to provide free WiFi for visitors to your office, but not to every freeloader in the neighborhood.

hostapd gives you a great middle ground, a way to use WPA2-Personal with individual keys for each user rather than a single shared key for everyone. These keys are just passwords in the `hostapd` configuration file and on the clients, so a PKI or a separate authentication server is unnecessary.

Remember that traffic is encrypted only between the client and the wireless router, to prevent eavesdropping on the wireless link; it does not provide end-to-end encryption. That is a job for something like OpenVPN or an SSH tunnel.

Prerequisites

You need a wireless access point that either includes `hostapd` or lets you install it, the `iw` command, and you'll need `wpa_supplicant` on a Linux PC for testing. Your access point should support `hostapd` 0.6.8 or newer. (The current `hostapd` release is 0.7.3.) With version 0.6.8 `hostapd` implemented the nl80211 driver. No special drivers are needed for any wireless interface card (WIC) supported by the mac80211 framework in the Linux kernel; it is built-in native support.

The nl80211 driver moves encryption, authentication, key rotation, and other access point functions into userspace. If you typically use the `iwconfig` command, start using the `iw` command because `iwconfig` does not work on 0.6.8.

DD-WRT and OpenWRT are two excellent open source firmware replacements for consumer-level wireless routers like the Linksys WRT54G-type devices (see Resources), and they include `hostapd`. Both have extensive databases of supported devices. I prefer to create my own WAPs using stripped-down Linuxes on Soekris, PC Engines, MicroTik single-board computers (see Resources). These little boards are durable, and I get complete control and flexibility.

If you like to build your own WAP, the most important component is a WIC with native Linux kernel support, and that supports the all-important AP mode. This is also called Access Point, Master, and Infrastructure mode, and it is required for a wireless access point. Many wireless network interfaces do not support AP mode, but are only client devices stripped down to a minimal functionality. I stick with Atheros wireless interfaces because they are fully-featured, and well-supported with both their legacy Madwifi drivers and the newer mac80211 drivers.

Avoid ndiswrapper on your access point. It's a nice hack for making a WIC work when you have no other options, but it is still a hack that hides a multitude of problems. Stick with good wireless interfaces with native kernel support.

Consult the Linux Wireless.org device databases to find supported interfaces, and also lots of information on wireless drivers and userspace commands (see Resources). The Linux Wireless project has done a great job of cleaning up and harmonizing the Linux wireless stack.

It's easier on the client side as nearly any WiFi-compliant WIC with native Linux kernel support can connect to your access point with strong WPA2 security. Mac and Windows® clients can also use your nice Linux-based access point.

Probing WICs

How do you know what functions your WIC supports? `iw` tells you. Look for the "Supported interface modes" section to see if it supports AP mode. Listing 1 shows an example.

Listing 1. iw listing

```
$ iw list

[...]

Supported interface modes:
*IBSS
*managed
*monitor
*AP
*AP/VLAN
```

This example shows a WIC that supports AP mode and wireless VLANs. IBSS is ad-hoc mode. Monitor mode is for sniffing wireless networks. All WICs support managed mode, which is a client of an access point.

On Atheros interfaces that use the Madwifi drivers, try `wlanconfig`. See Listing 2.

Listing 2. Example of wlanconfig

```
# wlanconfig ath0 list caps
ath0=7782e40f<WEP,TKIP,AES,AES_CCM,HOSTAP,TXPMGT,SHSLOT,SHPREAMBLE,TKIPMIC,WPA1,WPA2,BURST
     ,WME>
```

This shows that it supports AP mode, as well as WPA2 and the strong AES-CCMP cipher.

Another good way to probe your wireless hardware is with the extremely useful `hwinfo` command. It has a special option for wireless interfaces, and produces all kinds of great information as the snippet in Listing 3 shows:

Listing 3. Example of hwinfo data for WIC

```
$ hwinfo --wlan
27: PCI 500.0: 0282 WLAN controller
Model: "Intel WLAN controller"
Driver: "iwlagn"
Driver Modules: "iwlagn"
WLAN encryption modes: WEP40
WEP104 TKIP CCMP
WLAN authentication modes: open sharedkey wpa-psk wpa-eap
Status: iwlagn is active
Driver Activation Cmd: "modprobe iwlagn"
```

`hwinfo` names the driver, tells what encryption it supports, the name of the device, and lots more. You also might try `lspci` for PCI network interfaces, and `lsusb` for USB interfaces. This WIC will not work as an access point because the iwlagn driver is not supported by `hostapd`, and in any case it does not support AP mode. (It is part of a low-budget integrated Centrino chip.)

Configuring hostapd

Installation depends on which Linux distribution you use, so I shall leave it as homework for you to address on your own. First configure `hostapd` on the access point, and then wpa_supplicant on the client PC for testing the key exchange.

If you do not have a /etc/hostapd.conf file on your access point then create a new one. If your installation provides one, make a backup copy of it for reference and start with a clean new one. The example in Listing 4 has everything you need for our WPA2-Personal setup:

Listing 4. Example /etc/hostapd.conf

```
interface=ath0
bridge=br0
driver=nl80211
ssid=alracnet
auth_algs=1
wpa=1
wpa_psk_file=/etc/hostapd-psk
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP TKIP
rsn_pairwise=CCMP
```

You might need to replace some of these parameters, such as the interface, driver, ssid, with your own values. When you list more than one option, separate them with spaces, like on the wpa_pairwise line. Here are notes on this example.

- Atheros interfaces are always named athx, and all others are wlanx.
- Leave out the bridge line if your access point does not have an Ethernet bridge.
- The driver is nl80211 if you're using `hostapd` 0.6.8 or later and a WIC with mac80211 support. The only supported legacy drivers are HostAP, madwifi, and prism54. Pre-0.6.8 `hostapd` releases support the hostap, wired, madwifi, test, nl80211, and bsd drivers.
- ssid is whatever you want your ssid, or access point name, to be.
- auth_algs=1 allows only WPA2 authentication algorithms. 2 is WEP. Never ever use WEP (wired equivalent privacy) because it has been thoroughly broken for years, and is trivially easy to crack. 3 allows both.
- wpa=2 allows only WPA2. 1 is WPA1, and 3 allows both.
- wpa_psk_file points to the file containing the shared keys.
- wpa_key_mgmt specifies the encryption key algorithms you want to allow. Your choices are WPA-PSK, WPA-EAP, or both. PSK is pre-shared key. EAP is Extensible Authentication Protocol, which is a framework that supports a number of different authentication methods. You do not need it for your little pre-shared key setup.
- wpa_pairwise and rsn_pairwise control which ciphers are allowed for encrypting your data, and you can use CCMP, TKIP, or both. CCMP is much stronger than TKIP, so you could try allowing only CCMP. Windows clients are notorious for being finicky and troublesome with strong security, so you might have to allow TKIP for them.

Everyone should use WPA2; WEP (Wired Equivalent Privacy) is so weak it's useless, and WPA is almost as weak as WEP. WPA2 support has been mandatory in WiFi certified devices since 2006, and is supported in all modern operating systems, including Windows XP SP3. If you have to replace some WICs it's a lot cheaper than cleaning up after an intrusion.

Next, create a /etc/hostapd-psk file containing a wildcard MAC address and a simple plain-text test password up to 63 characters long:

```
00:00:00:00:00:00 testpassword
```

Now go to your Linux client PC and create a simple configuration file for wpa_supplicant, wpa_supplicant.conf similar to the example in Listing 5.

Listing 5. Sample wpa_supplicant.conf

```
ctrl_interface=/var/run/supplicant
network={
  ssid="alracnet"
  psk="testpassword"
  priority=5
}
```

ctrl_interface allows you to interact with wpa_supplicant on the command line. Use your own ssid and test plain-text password, and enclose both in quotation marks. The higher the priority number, the faster the connection is made to the access point. Now go back to your access point and fire up `hostapd` in debugging mode:

```
# hostapd -d /etc/hostapd.conf
```

If there are configuration errors, it will report them and not run. Otherwise it will emit many lines of output. Press CTRL+c to stop it. When you've worked the bugs out you can configure it to start automatically, which you'll do presently.

Then on the client, stop your wireless connection if it is running and run wpa_supplicant as root:

```
# wpa_supplicant -i wlan0 -D wext -c wpa_supplicant.conf -d
```

-i specifies the wireless interface, -D wext is the generic wpa_supplicant driver, -c points to the configuration file, and -d means debug mode. You'll see lots of output on both the access point and the client. When the key exchange is successful it completes quickly, and you'll see messages like the ones in Listing 6 on the client.

Listing 6. Sample messages from wpa_supplicant

```
EAPOL: SUPP_BE entering state IDLE
EAPOL authentication completed successfully
RTM_NEWLINK: operstate=1 ifi_flags=0x11043 ([UP][RUNNING][LOWER_UP])
RTM_NEWLINK, IFLA_IFNAME: Interface 'wlan0' added
```

Hurrah, it works. Press CTRL+c to end your wpa_supplicant session. The final step is to create individual user keys. First create these on the access point, and then copy them to your clients using your favorite network configuration utilities. All graphical configurators are pretty much the same: enter the SSID, select WPA2 Personal authentication, and copy in the key.

Adding Users, Stronger Keys

Now it's time to tighten things up a bit and add users. A plain-text password is computationally expensive because it must be encrypted, so wpa_supplicant comes with a nice command for generating 256-bit encrypted keys out of plaintext passwords, wpa_passphrase. Use it as shown in Listing 7, along with the SSID:

Listing 7. Creating users with wpa_passphrase

```
$ wpa_passphrase "alracnet" "greatbiglongpasswordbecauselongerisbetter"
network={
 ssid="alracnet"
 #psk="greatbiglongpasswordbecauselongerisbetter"
 psk=a8ed05e96eed9df63bdc4edc77b965770d802e5b4389641cda22d0ecbbdcc71c
}
```

Back in /etc/hostapd-psk you can start to add users. Each encrypted pre-shared key must be matched with the MAC address of the user. Listing 8 shows an example.

Listing 8. Example /etc/hostapd-psk

```
11:22:33:44:55:66     a8ed05e96eed9df63bdc4edc77b965770d802e5b4389641cda22d0ecbbdcc71c
22:33:44:55:66:77     eac8f79f06e167352c18c266ef56cc26982513dbb25ffa485923b07bed95757a
33:44:55:66:77:aa     550a613348ffe64698438a7e7bc319fc3f1f55f6f3facf43c15e11aaa954caf6
44:55:66:77:aa:bb     ad328e5f2b16bdd9b44987793ed7e09e6d7cca3131bc2417d99e48720b4de58c
```

When you are satisfied that everything is working, you probably want `hostapd` to run automatically. There are several ways to do this: create a startup script so it starts at boot, or start it when the wireless network interface comes up. There are so many ways to configure this in the various Linux distributions I shall leave this as your homework as well. You'll probably want to use the -B option, which forks it into the background, rather than the -d option for debugging.

That wraps up your introduction to the excellent `hostapd` daemon. See Resources for more information.

Resources

Learn

- Review an example hostapd configuration file that describes configuration file format and lists available options.
- Review an example wpa_supplicant configuration file that describes configuration file format and lists the available options.
- Dig into the Devices - Linux Wireless databases to find supported interfaces, plus info on wireless drivers and userspace commands.
- Find more information in these Linux man pages:
  - man hostapd
  - wpa_supplicant
  - iw
  - iwconfig
  - wlanconfig
  - lshw
  - hwinfo
  - lsusb
  - lspci
- Visit Linux Wireless, home of the Linux wireless stack. Find information for users, vendors, and developers, including supported device databases.
- In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth of other resources for Linux developers and administrators.
- The Open Source developerWorks zone provides a wealth of information on open source tools and using open source technologies.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools, as well as IT industry trends.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Follow developerWorks on Twitter, or subscribe to a feed of Linux tweets on developerWorks.

Get products and technologies

- Try DD-WRT and OpenWRT, two open source firmware replacements that include hostapd.
- Check out Soekris, PC Engines, or MicroTik single-board computers.
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Check out developerWorks blogs and get involved in the developerWorks community.

- Get involved in the developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Carla Schroder

Carla Schroder is a freelance PC tamer, administering Linux and Windows systems for small businesses, and writes how-tos for real people. Loves computers and high tech, thinks Linux/Open Source/Free Software is the best playground in the world. Carla discovered computers and high-tech in 1994; her first PC was an Apple II. She progressed through DOS/Windows, from 3.1 to XP. Discovered Linux in 1998. Carla is living proof that self-taught middle-aged persons can be fine computer gurus.