

users

- Support
- Drivers
- Devices
- Download
- Documentation

vendors


developers

Contents

1. About hostapd
2. Getting hostapd
 1. Using your distributions hostapd
 2. Download and compile hostapd
3. Configuring hostapd
 1. Establishing Baseline for Configuration
 2. Common Options
 3. Wireless Interface
 4. Wireless Environment
 5. Authentication and Encryption
 6. Dynamic VLAN tagging
 7. IEEE 802.11i/RSN/WPA2 pre-authentication
 8. Admission Control Mandatory settings
 9. Automatic channel selection

hostapd Linux documentation page

About hostapd

Homepage:  <http://w1.fi/hostapd/>

hostapd is an *IEEE 802.11 AP and IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator*.

This page is dedicated to the Linux documentation of it's implementation and use. Please refer to the hostapd home page for information for other Operating Systems.

As far a Linux is concerned, out of the **old** drivers you can only use these 3 drivers with hostapd:

- HostAP
- madwifi
- prism54

All **new** mac80211 based drivers that implement AP functionality are supported with hostapd's **nl80211** driver.

The mac80211 subsystem moves all aspects of master mode into user space. It depends on hostapd to handle authenticating clients, setting encryption keys, establishing key rotation policy, and other aspects of the wireless infrastructure. Due to this, the old method of issuing 'iwconfig <wireless interface> mode master' no longer works. Userspace programs like hostapd now use netlink (the nl80211 driver) to create a master mode interface for your traffic and a monitor mode interface for receiving and transmitting management frames.

Getting hostapd

Using your distributions hostapd

It is advisable to try your distributions version of hostapd before taking the time to compile and install your own copy. This will make future maintenance easier as you'll be able to use the init scripts shipped by the distro and hostapd will be updated by it as well. If your distribution ships 0.6.8 or later, you can test with this bare minimum config by creating the file hostapd-minimal.conf:

```
#change wlan0 to your wireless device
interface=wlan0
driver=nl80211
ssid=test
channel=1
```

If that config errors out with something like:

```
hostapd $ sudo hostapd ./hostapd-minimal.conf
Configuration file: ./hostapd-minimal.conf
Line 2: invalid/unknown driver 'nl80211'
1 errors found in configuration file './hostapd-minimal.conf'
```

that means that your distro is not shipping hostapd with nl80211 driver support and you'll need to follow the building instructions that follow. If it works, you can skip down to the configuring hostapd section. If not, continue on.

Download and compile hostapd


Using hostapd with nl80211 requires you to have at least libnl-1.0 pre8 as this release introduced genl, Generic Netlink, which nl80211 relies on. Most distributions are shipping this or a later release by now. To compile on fedora or other distributions that separate out the headers from the binaries, you need the libnl-devel package.

Throughout this section, versions will be referred to by: x.y.z

ex: hostapd-0.6.8.tar.gz would be referred to as hostapd-x.y.z.tar.gz

You can get the latest development version of hostapd from the git repository with:

```
git clone git://w1.fi/srv/git/hostap.git
cd hostap/hostapd
```

Or you can get a stable release (0.6.8 or later recommended) by downloading the tarball from  <http://w1.fi/hostapd/>.

```
wget http://w1.fi/releases/hostapd-x.y.z.tar.gz
tar xzvf hostapd-x.y.z.tar.gz
cd hostapd-x.y.z/hostapd
```

Next, we need to configure the hostapd build to enable nl80211 driver support. Copy defconfig to .config, and open it in your preferred text editor. Also, there are other options that you may want to enable, like 802.11n support if your hardware can do it. Most of the other encryption types and features aren't needed for most applications, so if you're questioning if you need to enable it, you probably don't need to.

```
cp defconfig .config
vi .config
```

Now find this line:

```
#CONFIG_DRIVER_NL80211=y
```

and uncomment it by removing the '#' sign. Repeat for other settings that you may be interested in. The basic configuration, with only this line uncommented is enough to get hostapd up and running with WPA/WPA2 authentication and encryption.

Next, compile hostapd:

```
make
```

if this fails with errors like:

```
driver_nl80211.c:21:31: warning: netlink/genl/genl.h: No such file or
directory
driver_nl80211.c:22:33: warning: netlink/genl/family.h: No such file or
directory
driver_nl80211.c:23:31: warning: netlink/genl/ctrl.h: No such file or
directory
driver_nl80211.c:24:25: warning: netlink/msg.h: No such file or directory
driver_nl80211.c:25:26: warning: netlink/attr.h: No such file or
directory
```

you need to install/update libnl-1.0pre8 (or later). If all goes well and the compilation finishes, try the minimal hostapd again, see the section Using your distributions hostapd above for that.

```
hostapd # ./hostapd ./hostapd-minimal.conf
```

```
Configuration file: ./hostapd-minimal.conf
Using interface wlan1 with hwaddr 00:0d:0b:cf:04:40 and ssid 'test'
```

If that starts as the example here shows, you can move on to configuring hostapd. If it fails to start with errors about the driver not being found, review the steps listed above for compiling hostapd again. If it gets the error messages:

```
Hardware does not support configured mode
wlan0: IEEE 802.11 Hardware does not support configured mode (2)
Could not select hw_mode and channel. (-2)
wlan0: Unable to setup interface.
rmdir[ctrl_interface]: No such file or directory
```

then it means the `hw_mode` (a, b or g) in the config file is set to a value not supported by the hardware.

Configuring hostapd

Establishing Baseline for Configuration

Before configuring hostapd, you need to know the capabilities of the clients that will be using it. Not all clients will support all of the methods you may want to implement, so a baseline configuration needs to be established. You will also want to do a survey of your area to find the channel that has the fewest other APs on it. When choosing which channel to use, it is important to remember that the channels overlap with any channels that are within 20MHz.

Examples of the baseline you might establish:

```
Encryption: wpa-psk + tkip
Wireless Mode: g
Normal for an environment that has to support semi legacy devices, that
don't support ccmp or wpa2
```

```
Encryption: wpa2-psk + ccmp
Wireless Mode: g+n
Normal for an environment that has only up to date hardware and software
```

```
Encryption: wep
Wireless Mode: b
This is the works case scenario, as wep is broken and can be trivially
cracked. Don't consider this as anything more than keeping casual free
loaders out.
```

Once you've found your baseline, it's time to edit `hostapd.conf`. The configuration options will be broken into 3 sections:

```
Common Options: options that you will probably want to set
Additional Options: options that are likely useful to at least know you
```

```
have
```

```
Extra Options: options that you aren't likely to need for most setups
```

Common Options

The most basic set of options for using hostapd with the nl80211 driver have already been provided as the hostapd-minimal.conf. That is all you need if you don't care about consistently being on the same channel, don't need/want encryption, and don't need a flashy name. However, that is not a realistic idea in the real world.

First, we'll setup the wireless interface settings, then the wireless environment settings, and finally the authentication and encryption.

Wireless Interface

Setting Summary:

- **interface:** Tells hostapd what wireless interface to use
- **bridge:** Set to a bridge if the wireless interface in use is part of a network bridge interface
- **driver:** For our purposes, always nl80211

If you only have 1 wireless interface, and it's going to be bridged with a wired interface, a good example setup would be:

```
interface=wlan0  
bridge=br0  
driver=nl80211
```

Wireless Environment

Setting Summary:

- **ssid:** Sets the name (SSID = service set identifier) of the network, wireless extensions/iwconfig incorrectly calls this "*essid*".
- **hw_mode:** Sets the operating mode of the interface, and the allowed channels. Valid values depend on hardware, but are always a subset of a, b, g
- **channel:** Sets the channel for hostapd to operate on. Must be a channel supported by the mode set in hw_mode, as well as allowed by your countries Wireless Regulatory rules.

The ssid is just for ease of configuration. It is what shows up in scan results, and can help in configuring your clients. Check the scan results for your area and choose a name.

hw_mode needs to be something that all of your hardware supports. Setting this to 'g' is probably the most common setup, and also enables backwards compatibility with 802.11b devices. Note, this is not where you enable 802.11n support, as 802.11n operates on top of 802.11a or 802.11g's functionality.

channel should be chosen so that it has the minimum overlap with other APs or other networks in your area. 802.11 channels are 20mhz (4 channels) wide in total, or 10mhz (2 channels) wide on each side. This means that an access point on channel 3 will interfere with an access point on channel 1 or channel 5. Use this to pick a channel. Most consumer APs default to channel 6, so you can use channel 1 or channel 11 in most cases for the best results. Also note that the channels available to you depends heavily entirely on the local regulatory rules.

An example of a good normal setup is:

```
ssid=MyNetwork
hw_mode=g
channel=1
```

802.11n Setting Summary

802.11n builds on the settings above, and adds additional functionality. If your hardware doesn't support 802.11n, or you don't plan on using it, you can ignore these.

- `ieee80211n`: Set to 1 to enable 802.11n support, 0 to disable it
- `ht_capab`: A list of the 802.11n features supported by your device

The explanation of these settings in the sample config file are quite helpful, so I'll suggest reading those. You can use the command 'iw list' to find a short list of the capabilities of your device.

Example settings:

```
wme_enabled=1
ieee80211n=1
ht_capab=[HT40+] [SHORT-GI-40] [DSSS_CCK-40]
```

Authentication and Encryption

There is a lot to the authentication and encryption options in hostapd. This section will cover the basics as far as wep/wpa/wpa2 goes, as well as some of the other commonly used options.

Settings Summary:

- `macaddr_acl`: This controls mac address filtering. Mac addresses are easily spoofed, so only consider the use of this to be augmenting other security measures you have in place.
- `auth_algs`: This is a bit field where the first bit (1) is for open auth, the second bit (2) is for Shared key auth (wep) and both (3) is both.
- `ignore_broadcast_ssid`: This enables/disables broadcasting the ssid.
- `wpa`: This is a bitfield like `auth_algs`. The first bit enables wpa1 (1), the second bit enables wpa2 (2), and both enables both (3)
- `wpa_psk/wpa_passphrase`: These establish what the pre-shared key will be for wpa authentication.

- `wpa_key_mgmt`: This controls what key management algorithms a client can authenticate with.
- `wpa_pairwise`: This controls wpa's data encryption
- `rsn_pairwise`: This controls wpa2's data encryption

First, scratch `macaddr_acl` and `ignore_broadcast_ssid` from your priorities as they only enhance security (and even then, only slightly). Also, WEP has been effectively broken now, so unless you HAVE to support wep, scratch that from your list. This just leaves wpa/wpa2. Per the draft standard, wpa2 is required for 802.11n, and as there are known attacks on wpa now, wpa2 is the recommended authentication and encryption suite to use. Fortunately, you can have both enabled at once. If Windows clients are going to be connecting, you should leave ccmp encryption out of the `wpa_pairwise` option, as some windows drivers have problems with systems that enable it.

A good starting point for a wpa & wpa2 enabled access point is:

```
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=3
wpa_passphrase=YourPassPhrase
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

If, alternately, you just want to support wpa2, you could use something like:

```
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=YourPassPhrase
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

That should be all of the settings that you'll need to change for a basic, secure, access point using hostapd with an AP enabled mac80211 driver.

Dynamic VLAN tagging

hostapd can be configured to move STAs into separate VLANs based on RADIUS tunnel attributes (as specified in RFC3580, <http://tools.ietf.org/html/rfc3580#section-3.31>):

```
Tunnel-Type=VLAN (13)
Tunnel-Medium-Type=802
Tunnel-Private-Group-ID=VLANID
```

To enable dynamic VLAN tagging the following options in `hostapd.conf` need to be set:

```
dynamic_vlan=1
vlan_file=/etc/hostapd.vlan
```

A value of 0 disables dynamic VLAN tagging, a value of 1 allows dynamic VLAN tagging and a value of 2 will reject the authentication if the RADIUS server does not provide the appropriate tunnel attributes.


Furthermore, hostapd needs to know how the VLAN interfaces should be named, this is done through an additional config file as specified in `vlan_file`.

Example `/etc/hostapd.vlan`:

```
1      wlan0.1
*      wlan0.#
```

This will create a `wlan0.1` interface on top of `wlan0` and move all STAs with the RADIUS supplied `vlangid 1` to that interface. The second entry is used to dynamically create VLAN interfaces on top of `wlan0`, hostapd will create an interface `wlan0.vlangid` for each different `vlangid` as supplied by the RADIUS server. For example, if a STA associates and the RADIUS server attributes contain the `vlangid 100` hostapd will create a `wlan0.100` interface and map the STA to this new interface.

IEEE 802.11i/RSN/WPA2 pre-authentication

IEEE 802.11 roaming experience can be enhanced by pre-authenticating the IEEE 802.1X/EAP part of the full RSN authentication and key handshake before actually associating with a new AP. To enable RSN pre-authentication you will need hostapd enabled RSN pre-authentication and a STA supplicant that also supports and enables RSN pre-authentication. In this section we'll provide a brief on how to enable `rsn_preauthentication`, what this does, and also provide references on how to  enable RSN preauthentication on OpenWrt and enabling RSN preauthentication on `wpa_supplicant` in order to test it.

First and foremost you need to ensure all your APs will be using the same SSID.

In order to ensure your APs and RADIUS are on the same network / switch and can talk to each other you can ping each other and review the **arp -a** output, you should see the MAC address of each other's AP's bridge interface on the **HW address** column, as well as the RADIUS server's MAC address.

For example, say we have two APs and one RADIUS server:

- ap136: 192.168.4.120
- db120: 192.168.4.139
- Radius server: 192.168.4.149

On ap136:


```

root@ap136 ~ # arp -a
IP address      HW type    Flags      HW address    Mask
Device
192.168.4.139   0x1       0x2       00:03:7f:11:20:00  *
br-lan
192.168.4.1     0x1       0x2       68:7f:74:3b:b1:0d  *
br-lan
192.168.4.149   0x1       0x2       c8:60:00:da:57:a7  *
br-lan
192.168.4.109   0x1       0x2       00:27:10:49:c6:44  *
br-lan

```

On db120:

```

root@db120 ~ # arp -a
IP address      HW type    Flags      HW address    Mask
Device
192.168.4.109   0x1       0x2       00:27:10:49:c6:44  *
br-lan
192.168.4.149   0x1       0x2       c8:60:00:da:57:a7  *
br-lan
192.168.4.1     0x1       0x2       68:7f:74:3b:b1:0d  *
br-lan
192.168.4.120   0x1       0x2       a2:69:db:89:44:88  *
br-lan

```



Assuming in this example the RADIUS server is on 192.168.4.149 on hostapd.conf you'll need to enable:

```

auth_server_addr=192.168.4.149
auth_server_port=1812
auth_server_shared_secret=testing123
wpa_key_mgmt=WPA-EAP
disable_pmksa_caching=1
okc=0
nas_identifier=
eapol_key_index_workaround=1
ieee8021x=1
wpa_key_mgmt=WPA-EAP
wpa_group_rekey=2000
auth_algs=1
wpa=2
wpa_pairwise=CCMP
wpa_group_rekey=2000
ssid=mcgrof-ap136-01
bridge=br-lan
rsn_preauth=1
rsn_preauth_interfaces=br-lan

```

Full example conf files generated by OpenWrt AA releases:

-  ap136 hostapd-phy0.conf
-  db120 hostapd-phy0.conf

Note: the same **SSID** must be used for RSN pre-authentication.

If using OpenWrt, simply enabling `rsn_preauth` is sufficient, `openwrt` will automatically add the `rsn_preauth_interfaces` for you, this is typically your bridge interface. You'll need two instances of `hostapd` running on two separate devices on the same network / switch. You'll also need a Radius server installed on a server on the same network / switch. FreeRADIUS is an example RADIUS solution to install, go [here](#) read how to install and configure FreeRADIUS for authenticating 802.11 users.

Be sure to test the Radius server, you can do this by using the `eapol_test` program, part of `hostapd` code.

```
cd wpa_supplicant/  
cp defconfig .config  
make eapol_test
```

Then edit a file called `eapol-config`, only to be used for this simple test of the RADIUS server:

```
network={  
  eap=TTLS  
  eapol_flags=0  
  key_mgmt=IEEE8021X  
  identity="testuser"  
  password="testpassword"  
  ca_cert="/home/mcgrof/server.pem"  
  phase2="auth=TTLS"  
}
```

You should now be able to test this user as follows:

```
./eapol_test -c eapol-config -a 192.168.x.x -p 1812 -s testing123 -r1
```

Provided you have two APs properly configured as describe with `rsn_preauth` as described above and on the same network you should now be able to configure a client for RSN pre-authentication. Details on this are available on the `wpa_supplicant` RSN preauthentication documentation section.

On the RADIUS server you want to see something like this, the second authentication from the STA to the second AP would go through the already established network on the first AP, after it associated with it. To be precise when a STA decides to try to preauthenticate against another AP is left up to each implementation to decide. This depends on the driver behavior and how it reports scan results or PMKSA candidates. Today `wpa_supplicant` does this after association and after a first scan completion.

```
root@radius:~# radsniff| grep ^Access  
Access-Request Id 0      192.168.4.120:51442 -> 192.168.4.149:1812  
+0.000  
Access-Challenge Id 0    192.168.4.149:1812 -> 192.168.4.120:51442  
+0.000  
Access-Request Id 1      192.168.4.120:51442 -> 192.168.4.149:1812  
+0.009  
Access-Challenge Id 1    192.168.4.149:1812 -> 192.168.4.120:51442  
+0.012
```

```

Access-Request Id 2      192.168.4.120:51442 -> 192.168.4.149:1812
+0.017
Access-Challenge Id 2    192.168.4.149:1812 -> 192.168.4.120:51442
+0.018
Access-Request Id 3      192.168.4.120:51442 -> 192.168.4.149:1812
+0.065
Access-Challenge Id 3    192.168.4.149:1812 -> 192.168.4.120:51442
+0.066
Access-Request Id 4      192.168.4.120:51442 -> 192.168.4.149:1812
+0.077
Access-Challenge Id 4    192.168.4.149:1812 -> 192.168.4.120:51442
+0.078
Access-Request Id 5      192.168.4.120:51442 -> 192.168.4.149:1812
+0.083
Access-Accept Id 5       192.168.4.149:1812 -> 192.168.4.120:51442
+0.083
Access-Request Id 0      192.168.4.139:35038 -> 192.168.4.149:1812
+2.162
Access-Challenge Id 0    192.168.4.149:1812 -> 192.168.4.139:35038
+2.162
Access-Request Id 1      192.168.4.139:35038 -> 192.168.4.149:1812
+2.168
Access-Challenge Id 1    192.168.4.149:1812 -> 192.168.4.139:35038
+2.171
Access-Request Id 2      192.168.4.139:35038 -> 192.168.4.149:1812
+2.174
Access-Challenge Id 2    192.168.4.149:1812 -> 192.168.4.139:35038
+2.175
Access-Request Id 3      192.168.4.139:35038 -> 192.168.4.149:1812
+2.216
Access-Challenge Id 3    192.168.4.149:1812 -> 192.168.4.139:35038
+2.217
Access-Request Id 4      192.168.4.139:35038 -> 192.168.4.149:1812
+2.222
Access-Challenge Id 4    192.168.4.149:1812 -> 192.168.4.139:35038
+2.223
Access-Request Id 5      192.168.4.139:35038 -> 192.168.4.149:1812
+2.225
Access-Accept Id 5       192.168.4.149:1812 -> 192.168.4.139:35038
+2.225

```

These logs reveal the STA choose to authenticate first with ap136 and then db120 once connected on the network with ap136.

Assuming you can log in to the STA you should be able to see **two** PMKSA entries:

```

root@android:/data/local # wpa_cli -i wlan0
pmksa

Index / AA / PMKID / expiration (in seconds) / opportunistic
1 00:03:7f:47:20:a5 eb25d3d579742c0384230fa66748f857 43042 0
2 00:03:7f:42:10:09 a99081d41e18f4632994b59b50bb2447 43044 0

```

The first one should correspond to the BSSID / MAC address of the wlan interface of ap136, so for example for ap136 this would be the MAC address of wlan1. The second one corresponds to the BSSID / MAC address of the wlan interface of db120, in this case wlan0.

To test PMKSA caching you can roam from the STA from one AP to another and verify that upon *reassociation* to the first AP the STA still has present the same PMKID (third field above, or sniff it from the network).

For testing purposes of RSN preauthentication you want to disable Opportunistic Key Caching as otherwise the PMKSA that the AP derived may have come from Opportunistic Key Caching instead of RSN preauthentication. Opportunistic Key Caching enables the PMKSA entries to be shared between configured interfaces and BSSes (i.e., all configurations within a single hostapd process).

Admission Control Mandatory settings

Admission Control Mandatory (ACM) can be used to limit access to higher priority ACs for traffic control **if** admission control were implemented but it is not implemented in hostapd today. ACM can be enabled for the 4 ACs but should be disabled by default in hostapd.conf


```
wmm_ac_bk_acm=0  
wmm_ac_be_acm=0  
wmm_ac_vi_acm=0  
wmm_ac_vo_acm=0
```

ACM should be disabled by default. In mac80211 we have a work around to deal with **strange** access points that have been configured all ACs to require admission control to transmit frames using AC_BK. Setting all ACs to require admission control would be very strange configuration and that should never be used.

There is no point in setting wmm_ac_[option]_acm=1 with any mac80211-based driver since they do not support admission control anyway. The only use for this with a driver that does not support admission control is for testing purposes.

Automatic channel selection

This is a work in progress. The patches are yet to be included in mainline hostapd.

See the  ACS sub-page for more details.

- [Print View](#)
- [Login](#)
- [About this site](#)