WPA supplicant

wpa_supplicant (http://hostap.epitest.fi/wpa_supplicant/) is a cross-platform **supplicant** with support for WEP, WPA and WPA2 (**IEEE 802.11i** / RSN (Robust Secure Network)). It is suitable for desktops, laptops and embedded systems.

wpa_supplicant is the IEEE 802.1X/WPA component that is used in the client stations. It implements key negotiation with a WPA authenticator and it controls the roaming and IEEE 802.11 authentication/association of the wireless driver.

Related articles

Network configuration

Wireless network configuration

WPA2 Enterprise

Contents

Installation

Install the `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` package, which includes the main program *wpa_supplicant*, the passphrase tool *wpa_passphrase*, and the text front-end *wpa_cli*.

Optionally also install `wpa_supplicant_gui (https://aur.archlinux.org/packages/wpa_supplicant_gui/)`[AUR], which provides *wpa_gui*, a graphical front-end for *wpa_supplicant*.

Overview

The first step to connect to an encrypted wireless network is having *wpa_supplicant* obtain authentication from a WPA authenticator. In order to do this, *wpa_supplicant* must be configured so that it will be able to submit the correct credentials to the authenticator.

Once the authentication is successful, it will be possible to connect to the network by obtaining an IP address in the usual way. For example, the **iproute2** suite of utilities can be used for temporary configuration during initial testing of the network interface. **dhcpcd** and **systemd-networkd** can obtain an IP address automatically via DHCP. See **Network configuration#Dynamic IP address** and subsequent for more details. The **Wireless network configuration#Wireless management** section may help as well.

Connecting with wpa_cli

This connection method allows scanning for available networks, making use of *wpa_cli*, a command line tool which can be used to configure *wpa_supplicant*. See **wpa_cli(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/wpa_cli.8)** for details.

In order to use *wpa_cli*, a control interface must be specified for *wpa_supplicant*, and it must be given the rights to update the configuration. Do this by creating a minimal configuration file:

/etc/wpa_supplicant/wpa_supplicant.conf

```
ctrl_interface=/run/wpa_supplicant
update_config=1
```

Now start *wpa_supplicant* with:

```
# wpa_supplicant -B -i interface -c /etc/wpa_supplicant/wpa_supplicant.conf
```

Tip: To discover your wireless network interface name, see **Network configuration#Network interfaces**.

At this point run:

```
# wpa_cli
```

This will present an interactive prompt (`>`), which has tab completion and descriptions of completed commands.

Tip:

- The default location of the control socket is `/var/run/wpa_supplicant/` . A custom path can be set manually with the `-p` option to match the *wpa_supplicant* configuration.
- It is possible to specify the interface to be configured with the `-i` option; otherwise, the first found wireless interface managed by *wpa_supplicant* will be used.

Use the `scan` and `scan_results` commands to see the available networks:

```
> scan
OK
<3>CTRL-EVENT-SCAN-RESULTS
> scan_results
bssid / frequency / signal level / flags / ssid
00:00:00:00:00:00 2462 -49 [WPA2-PSK-CCMP][ESS] MYSSID
11:11:11:11:11:11 2437 -64 [WPA2-PSK-CCMP][ESS] ANOTHERSSID
```

To associate with `MYSSID`, add the network, set the credentials and enable it:

```
> add_network
0
> set_network 0 ssid "MYSSID"
> set_network 0 psk "passphrase"
> enable_network 0
<2>CTRL-EVENT-CONNECTED - Connection to 00:00:00:00:00:00 completed (reauth) [id=0 id_str=]
```

If the SSID does not have password authentication, you must explicitly configure the network as keyless by replacing the command `set_network 0 psk "passphrase"` with `set_network 0 key_mgmt NONE`.

Note:

- Each network is indexed numerically, so the first network will have index 0.
- The **PSK** is computed from the *quoted* "passphrase" string, as also shown by the **wpa_passphrase** command. Nonetheless, you can enter the PSK directly by passing it to `psk` *without* quotes.

Finally save this network in the configuration file:

```
> save_config
OK
```

Once association is complete, you must obtain an IP address, for example, using **dhcpcd**.

Connecting with wpa_passphrase

This connection method allows quickly connecting to a network whose SSID is already known, making use of *wpa_passphrase*, a command line tool which generates the minimal configuration needed by *wpa_supplicant*. For example:

```
$ wpa_passphrase MYSSID passphrase
```

```
network={
    ssid="MYSSID"
    #psk="passphrase"
    psk=59e0d07fa4c7741797a4e394f38a5c321e3bed51d54ad5fcbd3f84bc7415d73d
}
```

This means that *wpa_supplicant* can be associated with *wpa_passphrase* and started with:

```
# wpa_supplicant -B -i interface -c <(wpa_passphrase MYSSID passphrase)
```

Note: Because of the process substitution, you **cannot** run this command with **sudo** - you will need a root shell. Just pre-pending *sudo* will lead to the following error:

```
Successfully initialized wpa_supplicant
Failed to open config file '/dev/fd/63', error: No such file or directory
Failed to read or parse configuration '/dev/fd/63'
```

See also **Help:Reading#Regular user or root**.

Tip:

- Use quotes, if the input contains spaces. For example: `"secret passphrase"`.
- To discover your wireless network interface name, see **Network configuration#Get current interface names**.
- Some unusually complex passphrases may require input from a file, e.g. `wpa_passphrase MYSSID < passphrase.txt`, or here strings, e.g. `wpa_passphrase MYSSID <<< "passphrase"`.

Finally, you should obtain an IP address (e.g., using **Dhcpcd#Running**).

Advanced usage

For networks of varying complexity, possibly employing extensive use of **EAP**, it will be useful to maintain a customised configuration file. For an overview of the configuration with examples, refer to **`wpa_supplicant.conf(5)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/wpa_supplicant.conf.5)**; for details on all the supported configuration parameters, refer to the example file `/usr/share/doc/wpa_supplicant/wpa_supplicant.conf`.**[1] (https://w1.fi/cgit/hostap/plain/wpa_supplicant/wpa_supplicant.conf)**

Configuration

As is clear after reading **#Connecting with wpa_passphrase**, a basic configuration file can be generated with:

```
# wpa_passphrase MYSSID passphrase > /etc/wpa_supplicant/example.conf
```

This will only create a `network` section. A configuration file with some more common options may look like:

/etc/wpa_supplicant/example.conf

```
ctrl_interface=/run/wpa_supplicant
ctrl_interface_group=wheel
update_config=1
ap_scan=1

network={
    ssid="MYSSID"
    psk=59e0d07fa4c7741797a4e394f38a5c321e3bed51d54ad5fcbd3f84bc7415d73d
}
```

The passphrase can alternatively be defined in clear text by enclosing it in quotes, if the resulting security problems are not of concern:

```
network={
    ssid="MYSSID"
    psk="passphrase"
}
```

If the network does not have a passphrase, e.g. a public Wi-Fi:

```
network={
    ssid="MYSSID"
    key_mgmt=NONE
}
```

Further `network` blocks may be added manually, or using *wpa_cli* as illustrated in **#Connecting with wpa_cli**. In order to use *wpa_cli*, a control interface must be set with the `ctrl_interface` option. Setting `ctrl_interface_group=wheel` allows users belonging to such group to execute *wpa_cli*. This setting can be used to enable users without root access (or equivalent via sudo etc) to connect to wireless networks. Also add `update_config=1` so that changes made with *wpa_cli* to `example.conf` can be saved. Note that any user that is a member of the `ctrl_interface_group` group will be able to make changes to the file if this is turned on.

`fast_reauth=1` and `ap_scan=1` are the *wpa_supplicant* options active globally at the time of writing. Whether you need them, or other global options too for that matter, depends on the type of network to connect to. If you need other global options, simply copy them over to the file from `/usr/share/doc/wpa_supplicant/wpa_supplicant.conf`.

Alternatively, `wpa_cli set` can be used to see options' status or set new ones. Multiple network blocks may be appended to this configuration: the supplicant will handle association to and roaming between all of them. The strongest signal defined with a network block usually is connected to by default, one may define `priority=` to influence behaviour.

Once you have finished the configuration file, you can optionally use it as a system-wide or per-interface default configuration by naming it according to the paths listed in **#At boot (systemd)**. This also applies if you use additional network manager tools, which may rely on the paths (for example **Dhcpcd#10-wpa_supplicant**).

Tip: To configure a network block to a hidden wireless *SSID*, which by definition will not turn up in a regular scan, the option `scan_ssid=1` has to be defined in the network block.

Connection

Manual

First start *wpa_supplicant* command, whose most commonly used arguments are:

- `-B` - Fork into background.
- `-c filename` - Path to configuration file.
- `-i interface` - Interface to listen on.
- `-D driver` - Optionally specify the driver to be used. For a list of supported drivers see the output of `wpa_supplicant -h`.
  - `nl80211` is the current standard, but not all wireless chip's modules support it.
  - `wext` is currently deprecated, but still widely supported.

See `wpa_supplicant(8)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/wpa_supplicant.8) for the full argument list. For example:

```
# wpa_supplicant -B -i interface -c /etc/wpa_supplicant/example.conf
```

followed by a method to obtain an ip address manually as indicated in the **#Overview**, for example:

```
# dhcpcd interface
```

Tip:

- *dhcpcd* has a hook that can launch *wpa_supplicant* implicitly, see **dhcpcd#10-wpa_supplicant**.
- While testing arguments/configuration it may be helpful to launch *wpa_supplicant* in the foreground (i.e. *without* the `-B` option) for better debugging messages.

At boot (systemd)

The *wpa_supplicant* package provides multiple **systemd** service files:

- `wpa_supplicant.service` - uses **D-Bus**, recommended for **NetworkManager** users.

- `wpa_supplicant@interface.service` - accepts the interface name as an argument and starts the *wpa_supplicant* daemon for this interface. It reads a `/etc/wpa_supplicant/wpa_supplicant-interface.conf` configuration file.
- `wpa_supplicant-nl80211@interface.service` - also interface specific, but explicitly forces the `nl80211` driver (see below). The configuration file path is `/etc/wpa_supplicant/wpa_supplicant-nl80211-interface.conf`.
- `wpa_supplicant-wired@interface.service` - also interface specific, uses the `wired` driver. The configuration file path is `/etc/wpa_supplicant/wpa_supplicant-wired-interface.conf`.

To enable wireless at boot, enable an instance of one of the above services on a particular wireless interface. For example, **enable** the `wpa_supplicant@interface` systemd unit.

Now choose and **enable** an instance of a service to obtain an ip address for the particular *interface* as indicated in the **#Overview**. For example, **enable** the `dhcpcd@interface` systemd unit.

Tip: *dhcpcd* has a hook that can launch *wpa_supplicant* implicitly, see **dhcpcd#10-wpa_supplicant**.

802.1x/radius

To connect a wired adapter using 802.1x/radius you will need to specify some configurations and enable the necessary service for the adapter. This is useful for headless servers using *networkd*.

Replace `adapter` with the wired adapter you wish to connect, and adapt the settings to match your 802.1x/radius requirements.

/etc/wpa_supplicant/wpa_supplicant-wired-*adapter*.conf

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=0
network={
  key_mgmt=IEEE8021X
  eap=PEAP
  identity="user_name"
  password="user_password"
  phase2="autheap=MSCHAPV2"
}
```

Tip: The same configuration, but for a wireless adapter, would require changing `IEEE8021X` to `WPA-EAP` and removing the `ap_scan=0` line

Since this file is storing a plaintext password, **chown** it to `root:root` and **chmod** it to `600`.

Before running the `wpa_supplicant-wired@adapter.service` service, make sure to set the device down:

```
# ip link set adapter down
```

Tip: This setup can be used during system installation as well, though you may want to run using `dhcpcd@adapter.service` to solicit an address.

wpa_cli action script

wpa_cli can run in daemon mode and execute a specified script based on events from *wpa_supplicant*. Two events are supported: `CONNECTED` and `DISCONNECTED`. Some **environment variables** are available to the script, see **wpa_cli(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/wpa_cli.8)** for details.

The following example will use **desktop notifications** to notify the user about the events:

```
#!/bin/bash

case "$2" in
    CONNECTED)
        notify-send "WPA supplicant: connection established";
        ;;
    DISCONNECTED)
        notify-send "WPA supplicant: connection lost";
        ;;
esac
```

Remember to make the script executable, then use the `-a` flag to pass the script path to *wpa_cli*:

```
$ wpa_cli -a /path/to/script
```

Troubleshooting

Note: Make sure that you do not have remnant configuration files based on the full documentation example `/usr/share/doc/wpa_supplicant/wpa_supplicant.conf`. It is filled with uncommented network examples that may lead random errors in practice (**FS#40661 (https://bugs.archlinux.org/task/40661)**).

nl80211 driver not supported on some hardware

On some (especially old) hardware, *wpa_supplicant* may fail with the following error:

```
Successfully initialized wpa_supplicant
nl80211: Driver does not support authentication/association or connect commands
wlan0: Failed to initialize driver interface
```

This indicates that the standard `nl80211` driver does not support the given hardware. The deprecated `wext` driver might still support the device:

```
# wpa_supplicant -B -i wlan0 -D wext -c /etc/wpa_supplicant/example.conf
```

If the command works to connect, and the user wishes to use **systemd** to manage the wireless connection, it is necessary to **edit** the `wpa_supplicant@.service` unit provided by the package and modify the `ExecStart` line accordingly:

```
/etc/systemd/system/wpa_supplicant@.service.d/wext.conf
```

```
[Service]
ExecStart=
ExecStart=/usr/bin/wpa_supplicant -c/etc/wpa_supplicant/wpa_supplicant-%I.conf -i%I -Dnl80211,wext
```

Note: Multiple comma separated driver wrappers in option `-Dnl80211,wext` makes *wpa_supplicant* use the first driver wrapper that is able to initialize the interface (see **wpa_supplicant(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/wpa_supplicant.8)**). This is useful when using mutiple or removable (e.g. USB) wireless devices which use different drivers.

Problem with mounted network shares (cifs) and shutdown

When you use wireless to connect to network shares you might have the problem that the shutdown takes a very long time. That is because systemd runs against a 3 minute timeout. The reason is that WPA supplicant is shut down too early, i.e. before systemd tries to unmount the share(s). A **bug report (https://github.com/systemd/systemd/issues/1435)** suggests a work-around by **editing** the `wpa_supplicant@.service` as follows:

```
/etc/systemd/system/wpa_supplicant.service.d/override.conf
```

```
[Unit]
After=dbus.service
```

Password-related problems

`wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` may not work properly if directly passed via stdin particularly long or complex passphrases which include special characters. This may lead to errors such as `failed 4-way WPA handshake, PSK may be wrong` when launching `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)`.

In order to solve this try using here strings `wpa_passphrase <MYSSID> <<< "<passphrase>"` or passing a file to the `-c` flag instead:

```
# wpa_supplicant -i <interface> -c /etc/wpa_supplicant/example.conf
```

In some instances it was found that storing the passphrase cleartext in the `psk` key of the `wpa_supplicant.conf` `network` block gave positive results (see **[2] (http://www.linuxquestions.org/questions/linux-wireless-networking-41/wpa-4-way-handshake-failed-843394/)**). However, this approach is rather insecure. Using `wpa_cli` to create this file instead of manually writing it gives the best results most of the time and therefore is the recommended way to proceed.

Problems with eduroam and other MSCHAPv2 connections

Ensure that your config uses

```
phase2="auth=MSCHAPV2"
```

with a capital "v" (see **FS#51358 (https://bugs.archlinux.org/task/51358)**). You could even omit this setting entirely, since MSCHAPV2 is the default.

See also

- **wpa_supplicant home (https://w1.fi/wpa_supplicant/)**
- **wpa_supplicant README (http://w1.fi/cgit/hostap/plain/wpa_supplicant/README)** - contains full documentation of project, including *wpa_cli* commands not listed in manpage.
- **wpa_cli usage examples (https://gist.github.com/buhman/7162560)**
- **`wpa_supplicant(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/wpa_supplicant.8)`**
- **`wpa_supplicant.conf(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/wpa_supplicant.conf.5)`**
- **`wpa_cli(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/wpa_cli.8)`**
- **Kernel.org wpa_supplicant documentation (http://wireless.kernel.org/en/users/Documentation/wpa_supplicant)**

Retrieved from "https://wiki.archlinux.org/index.php?title=WPA_supplicant&oldid=507451"

- This page was last edited on 13 January 2018, at 17:50.