Enable Wireless networks in Debian Initramfs

Dec 7, 2017 · 5 min · Marc Fargas

▼ Table of Contents

> **Initramfs is** a very tiny environment in which your Linux system boots in order to do a lot of initialisation magic *before loading your system*. The most common use case is to mount the root filesystem, like when it's encrypted and you need to type a passphrase to mount it. Or if fsck needs to run on an unclean root filesystem. Basically anything that goes *before* mounting the root filesystem and, after that, before launching INIT (you can read more in the Debian Wiki).

There are scenarios in which you need the network available at this very preliminary state of the system boot. Doing that over wired ethernet is easy (as long as you don't need authentication on the wire) with the `ip=` kernel parameter (see here the documentation of the ip= parameter). Wireless connectivity is a whole different issue, WPA needs a helper program (`wpa_supplicant`), network configuration and, most likely, firmware files.

So, lets find out how can we get a wireless connection up and running on initramfs to do something useful there (in a later post: unattended boot with an encrypted rootfs).

A few warnings:

- I assume you know how to move around on the console, really.
- If you screw up you might end up on an emergency shell or event with an unbootable system.
- This solution kills `wpa_supplicant` at the last stage of init so the underlying system can take over the wireless connection (ie: NetworkManager) so, be sure that whatever you attempt to achieve can live with that network interruption.
- Make sure you have a keyboard and display (or a serial console). **Do not** attempt to do this on a headless system.

This has been tested on:

- Debian Stretch
- Intel Corporation Wireless 7260 AC
- WPA2 (AP is a Time Capsule)

initramfs-tools

The Debian system utilises what is known as `initramfs-tools` to build the initramfs for the installed kernel images. A command comes with it: `update-initramfs` which updates/creates suchs images. This tools is highly extendable (in fact, lots of packages extend it) by the end-user in the folder `/etc/initramfs-tools/` where hooks and scripts can be placed in order to **customize the image build and/or the boot processes**. A very good place to start would be `man initramfs-tools` for our purpose today: the SCRIPTS section.

What we need initramfs-tools to do for us

- Add our wireless modules and firmware to the initramfs
- Copy our dependencies to the image (`wpa_supplicant` , `wpa_cli` , `wpa_supplicant.conf` and its libraries)
- Start `wpa_supplicant` to connect to the network
- Setup networking (ip, etc)
- Hold the boot process until the network is up or a timeout occurs
- Kill `wpa_supplicant` before booting the underlying system to not conflict with whatever is there expecting full control of the network interface (aka: NetworkManager)

That sounds easy.

Custom hook

We will use a hook to do the first two steps, note that for the modules part you can just type the module name in `/etc/initramfs-tools/modules` and it should work. We do it in the hook just to keep everything together.

This goes into `/etc/initramfs-tools/hooks/enable-wireless` , make sure to put the right modules on the `manual_add_modules` line.

```
# !/bin/sh
set -e
PREREQ=""
prereqs()
{
    echo "${PREREQ}"
}
case "${1}" in
    prereqs)
        prereqs
        exit 0
        ;;
esac

. /usr/share/initramfs-tools/hook-functions

# CHANGE HERE for your correct modules.
manual_add_modules iwlwifi iwlmvm
copy_exec /sbin/wpa_supplicant
copy_exec /sbin/wpa_cli
copy_file config /etc/initramfs-tools/wpa_supplicant.conf /etc/wpa_supplicant.conf
```

Pretty straighforward, the upper half is boilerplate as required by initramfs-tools, see the manpage for more. The rest is quite readable: add the modules (it will also add the firmwares), copy wpa* stuff, copy the configuration.

Now, `wpa_supplicant.conf` is unique to you, as always, `man wpa_supplicant.conf` is your friend, and here is an example:

```
# Sample /etc/initramfs-tools/wpa_supplicant.conf
# note that this is independent of the system /etc/wpa_supplicant.conf (if any)
# only add the network you need at boot time. **And keep the ctrl_interface** !!
ctrl_interface=/tmp/wpa_supplicant

network={
    ssid="MyNetwork"
    scan_ssid=1
    psk="network passphrase"
    key_mgmt=WPA-PSK
}
```

Connection script (init-premount)

Now, we need the system to startup the supplicant, connect and go on. This can't be done at the init-top stage because not even the kernel modules are available by then, to init-premount looks fine. Problem? Whatever the reason you are reading this, most likely it also happens in init-premount (mandos-client, cryptsetup, …) and initramfs-tools comes with this warning on the manpage:

> No guarantees are made as to the order in which the different scripts are executed unless the prereqs are setup in the script.

So… dirty hack is to assume alphabetical order of execution and put "a_" in front of the script. It works, for now.

This goes into ` `/etc/initramfs-tools/scripts/init-premount/a_enable_wireless` , you need to change the `INTERFACE=` and, maybe, the `AUTH_LIMIT` one (the timeout):

```
#!/bin/sh
PREREQ=""
prereqs()
{
    echo "$PREREQ"
}

case $1 in
prereqs)
    prereqs
    exit 0
    ;;
esac

. /scripts/functions

AUTH_LIMIT=30
INTERFACE="wlp5s0"
alias WPACLI="/sbin/wpa_cli -p/tmp/wpa_supplicant -i$INTERFACE "

log_begin_msg "Starting WLAN connection"
/sbin/wpa_supplicant  -i$INTERFACE -c/etc/wpa_supplicant.conf -P/run/initram-wpa_supp

# Wait for AUTH_LIMIT seconds, then check the status
limit=${AUTH_LIMIT}

echo -n "Waiting for connection (max ${AUTH_LIMIT} seconds)"
while [ $limit -ge 0 -a `WPACLI status | grep wpa_state` != "wpa_state=COMPLETED" ]
do
    sleep 1
    echo -n "."
    limit=`expr $limit - 1`
done
echo ""

if [ `WPACLI status | grep wpa_state` != "wpa_state=COMPLETED" ]; then
  ONLINE=0
  log_failure_msg "WLAN offline after timeout"
  panic
else
  ONLINE=1
  log_success_msg "WLAN online"
fi

configure_networking
```

Kill when done

Last, but not least, ` `/etc/initramfs-tools/scripts/local-bottom/kill_wireless` should contain:

```
#!/bin/sh
PREREQ=""
prereqs()
{
    echo "$PREREQ"
}

case $1 in
prereqs)
```

```
    prereqs
    exit 0
    ;;
esac

echo "Killing wpa_supplicant so the system takes over later."
kill `cat /run/initram-wpa_supplicant.pid`
```

Final touches

You may have noticed we use the provided `configure_networking` function, it relies on you passing the proper `ip=` kernel parameter, so better supply it, for GRUB just setup the `GRUB_CMDLINE_LINUX` in `/etc/default/grub` like: `GRUB_CMDLINE_LINUX="ip=:::::wlp5s0:on panic=10"` (see here the documentation of the ip= parameter). The `panic=10` makes the system reboot if something goes wrong (like network failure), when testing you might prefer `break=premount` or some other options, see the initramfs-tools manpage5 or the Debian wiki InitramfsDebug page

Make the scripts executable, and, finally, rebuild initramfs and update-grub:

```
chmod +x /etc/initramfs-tools/scripts/local-bottom/kill_wireless
chmod +x /etc/initramfs-tools/scripts/init-premount/a_enable_wireless
chmod +x /etc/initramfs-tools/hooks/enable-wireless

update-initramfs -k all -u
update-grub
```

And… reboot. I suggest you boot with `break=bottom` so you can check things work as expected (i.e with `ip link` and `ip addr`).

network linux

Related content

- Ip Tunnel Over Ssh With Tun
- Moved to Google Apps
- Outgoing connections from Linux not working

© 2021 | Follow on Twitter | PaperMod theme & Hugo ♥ Powered by Hugo & PaperMod