Installation Guide for Linux

Contents

Ubuntu 16.04 LTS

1. Development Environment

The development environment in Ubuntu is required as follows:

Development Environment	
OS	Ubuntu 16.04 LTS
Kernel version	4.13.0-36-generic
Gcc version	5.4.0

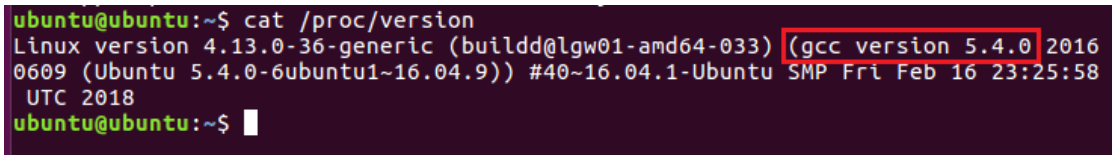
2. Compile the Driver

2.1. Compilation tool and kernel sources

Before you compile the driver, please make sure you have the correct compile tool and kernel sources. In Ubuntu 16.04 LTS, we can install compile tool gcc by command “apt-get install gcc”

```
$ apt-get install gcc
```

Note: We recommend you install the same version tool to compile the driver.
For example:



According to the command “*cat /proc/version*”, you could see your Ubuntu 16.04 LTS system is compiled by gcc5.4.0. By default, gcc5.4.0 is already installed in Ubuntu 16.04 LTS, you could use gcc5.4.0 to compile the driver directly.
Generally, compatible kernel headers are already built in Ubuntu 16.04 LTS, so you don’t need to separately download and compile the kernel sources. However, if no related kernel headers are integrated in your system, please install the kernel sources first.

2.2. Compile the Driver

Use Terminal to go to the driver directory and run the following commands to compile the driver.

```
$ make clean
$ make
```

After compiling, you can see a *name of the chip.ko* file is stored in the directory of the driver.

3. Load the Driver

Here we show the 88x2bu.ko wireless driver loading process as an example. Run the following command to load the driver.

```
$ sudo cp 88x2bu.ko /lib/modules/[kernel version]/kernel/drivers/net/wireless/  #[kernel version] is the directory name of the system kernel version
$ sudo depmod –a
$ sudo modprobe 88x2bu
```

Or directly use insmod to load the driver.

```
$ sudo insmod 88x2bu.ko
```

After loading the driver, run the following command to check if the driver is successfully loaded.

```
$ lsmod
```

Mint 18.03

1. Development Environment

The development environment in Mint is required as follows：

Development Environment	
OS	Mint 18.03
Kernel version	4.10.0-38.generic
Gcc version	5.4.0

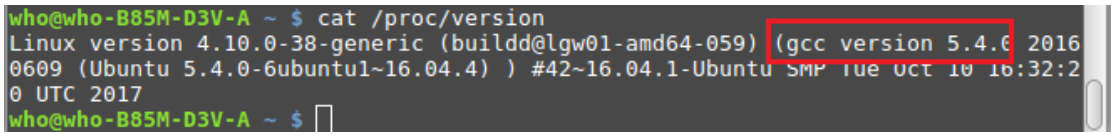
2. Compile the Driver

2.1. Compilation tool and kernel sources

Before you compile the driver, please make sure you have the correct compile tool and kernel sources. In Mint, we can install compile tool gcc by command “apt-get install gcc”

```
$ apt-get install gcc
```

Note: We recommend you install the same version tool to compile the driver.
For example:



According to the command "*cat /proc/version*", you could see your Mint system is compiled by gcc5.4.0, so we should use gcc5.4.0 to compile the driver.
Generally, compatible kernel headers are already built in Mint, so you don't need to separately download and compile the kernel sources. However, if no related kernel headers are integrated in your system, please install the kernel sources first.

2.2. Compile the Driver

Use Terminal to go to the driver directory and run the following commands to compile the driver.

```
$ make clean
$ make
```

After compiling, you can see a *name of the chip.ko* file is stored in the directory of the driver.

3. Load the Driver

Here we show the 88x2bu.ko wireless driver loading process as an example. Run the following command to load the driver.

```
$ sudo cp 88x2bu.ko /lib/modules/[kernel version]/kernel/drivers/net/wireless/
$ sudo depmod -a
$ sudo modprobe 88x2bu
```

Or directly use insmod to load the driver.

```
$ sudo insmod 88x2bu.ko
```

After loading the driver, run the following command to check if the driver is successfully loaded.

```
$ lsmod
```

Raspberry Pi3

1. Development Environment

The development environment in Raspberry Pi 3 is required as follows

Development Environment	
OS	6.3.0-18-rpi-deb9u1
Kernel version	4.9.80-v7
Hardware version of development board	Raspberry Pi 3 Model B

2. Compile the Driver

Before you compile the driver, please make sure you have the correct compile tool and kernel sources.

2.1. Compile Kernel source

Here we illustrate the instructions for local building to compile the driver for Linux.

2.1.1. Download and Install Tools

Note: Before local building, make sure your raspberrypi system is connected to the internet.
Install Git, bc and other related tools.

```
$ sudo apt-get install git bc
```

2.1.2. Get Kernel source

Click the following links to download raspberrypi kernel source and other related tools.

```
https://github.com/raspberrypi/linux
https://github.com/raspberrypi/tools
```

Before local building, make sure if you need to update the kernel. If your adapter supports the current kernel version, you don't need to update the kernel, and just download the kernel sources of this version. If you have to update the kernel, choose the kernel sources of the desired version. Here we download the version 4.9 kernel sources.
Create Linux-src directory in the local user's root directory to store kernel sources. If you have installed Git, you can use Git to obtain Linux kernel sources from Github; if you directly download the .zip file, use the following jar command to decompress this file.

```
$ sudo jar –xf XXX.zip
```

Note: It is recommended not to use the *unzip* software to decompress the .zip file.

2.1.3. Modify Kernel

Run the following commands to modify Linux kernel. You can also modify the kernel according to your demands.

```
$ cd linux                    /* go the directory of kernel sources */
$ KERNEL=kernel7
$ make bcm2709_defconfig
```

Note: The instructions for Raspberry Pi3 and other versions of Raspberry are slightly different, for details of other versions, please refer to the instructions on Raspberry official website.

2.1.4. Compile the Kernel

Run the following commands to compile and install the kernel and related device tree. It

may take a few minutes.

```
$ make –j4 zImage modules dtbs
$ sudo make modules_install
$ sudo cp arch /arm/boot/dts/*.dtb /boot/
$ sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
$ sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/
$ sudo cp arch/arm/boot/zImage /boot/$KERNEL.img
```

Note: “-j4” refers to using *Raspberry Pi3 and 4* to compile to accelerate the compilation process.

Power off the development board of Raspberry Pi3 and then run the following command to confirm the kernel version.

```
$ uname –a
or
$ cat /proc/version
```

2.2. Compile the Driver Source

Go to the driver’s directory, open the Makefile file to support Raspberry Pi3. By default, the CONFIG_PLATFORM_I386_PC macro is enabled. Set the value for CONFIG_PLATFORM_BCM2709 to y, and set the value for CONFIG_PLATFORM_I386_PC to n.

```
CONFIG_PLATFORM_BCM2709 = y
CONFIG_PLATFORM_I386_PC = n
CONFIG_PLATFORM_ANDROID_X86 = n
```

After setting the parameters, use Terminal to go to the directory in which the driver source file is stored. Run the following commands to compile the driver.

```
$ make clean
$ make
```

3. Load the Driver

Here we show the 8192eu.ko wireless driver loading process as an example. Run the following command to load the driver.

```
$ sudo cp 8192eu.ko /lib/modules/[kernel version]/kernel/drivers/net/wireless/
$ sudo depmod -a
$ sudo modprobe 8192eu
```

Or directly use insmod to load the driver.

```
$ sudo insmod 8192eu.ko
```

After loading the driver, run the following command to check if the driver is successfully loaded.

```
$ lsmod
```

Kali 2018.1

1. Development Environment

The development environment in Kali 2018.1 is required as follows.

Development Environment	
OS	Kali 2018.1
Kernel Source Version	4.14.0-kali3-amd64

2. Compile the Driver

2.1. Install the Kernel Header File

Before compiling the driver in Kali 2018, make sure you have installed and compiled the right Linux header file. Follow the instructions to install and compile the Linux header file.

2.1.1. Update the Software Source

Run the following commands to update the software source and related tools.

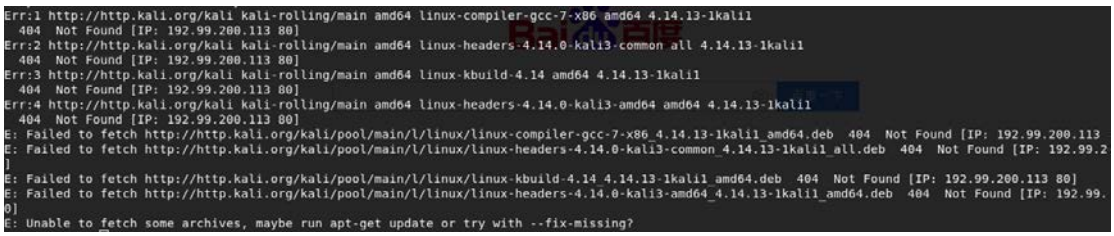
```
$ sudo apt-get clean
$ sudo apt-get update
$ sudo apt-get upgrade
```

2.1.2. Install the Kernel Header File

1) Method 1: Run the following command to install the kernel header file.

```
$ sudo apt-get install linux-headers-$(uname -r)
```

After running this command, the system will automatically find the matched kernel header file to download and install it. If the Kali server is updated, you may not find the specific file, in this case, you can manually download and install the header file.



2) Method 2: Manually Download and Compile to Install

Find the matched kernel header file in the download source of your Kali software.

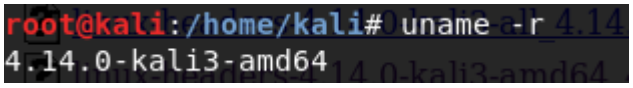
Click the following link to go to the official website to download Linux header file and related tools.

http://http.kali.org/kali/pool/main/l/linux

- Check the system version of Kali

```
$ uname -r
```

The system version we used here is shown as below.



- Download and compile linux-kbuild

In the Kali download links, find the linux-kbuild file of your system. Here we choose linux-kbuild-4.14_4.14.17.-1kali_amd64.deb as an example.

After downloading the file, use Terminal to go to the directory and run the following command to install the file.

```
$ sudo dpkg -i linux-kbuild-4.14_4.14.17-1kali_amd64.deb
```

- Download and compile linux-header-common

In the Kali download links, find the linux-header-common file of your system. Here we choose linux-header-4.14.0-kali3-common_4.14.17-1kali_all.deb as an example.

After downloading the file, use Terminal to go to the directory and run the following command to install the file.

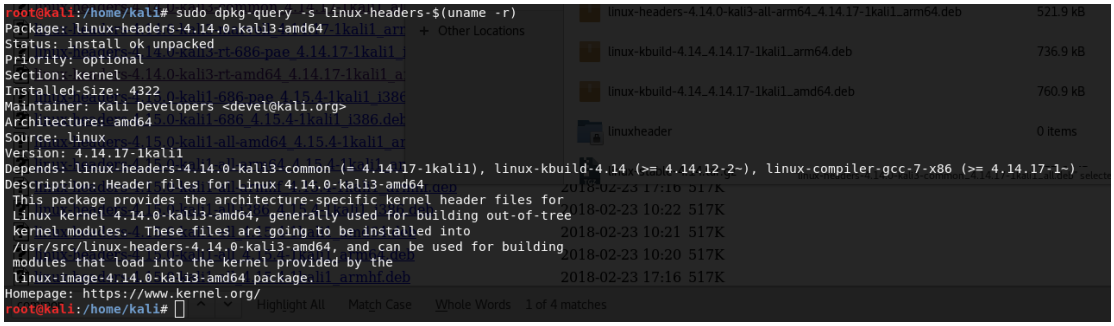
```
$ sudo dpkg -i linux-header-4.14.0-kali3-amd64_4.14.17-1kali_amd64.deb
```

Run the following command to check if the kernel header file is successfully installed.

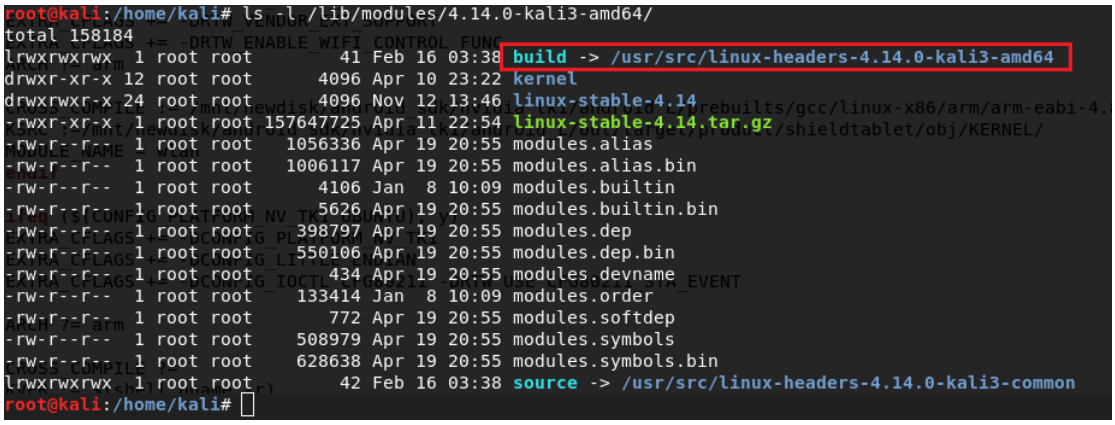
```
$ dpkg-query -s linux-headers-$(uname -r)
```

After the linux-header is successfully installed, run the following command to display the

detailed linux-header information.



Check the /lib/modules/<kernel-version>/ directory and you will see a *build* link file.



2.1.3. Compile Driver Source

Use Terminal to go to the driver directory. Run the following commands to compile the driver.

```
$ make clean
$ make
```

After compiling, you can see a *name of the chip.ko* file is stored in the directory of the driver.

3. Load the Driver

Here we show the 88x2bu.ko wireless driver loading process as an example. Run the following command to load the driver.

```
$ sudo cp 88x2bu.ko /lib/modules/[kernel version]/kernel/drivers/net/wireless/
$ sudo depmod -a
$ sudo modprobe 88x2bu
```

Or directly use insmod to load the driver.

```
$ sudo insmod 88x2bu.ko
```

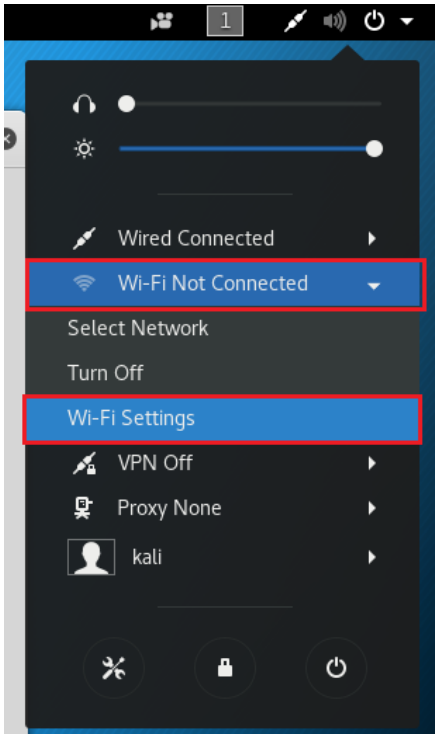
After loading the driver, run the following command to check if the driver is successfully loaded.

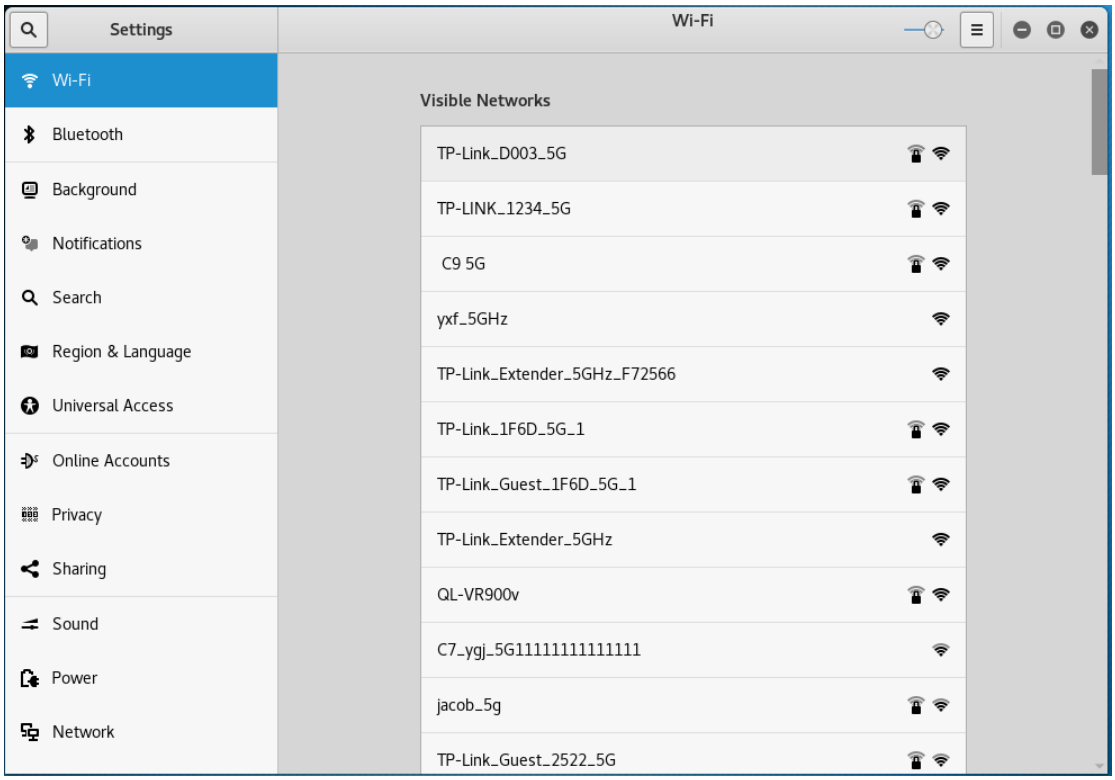
```
$ lsmod
```

Use the Graphical Interface

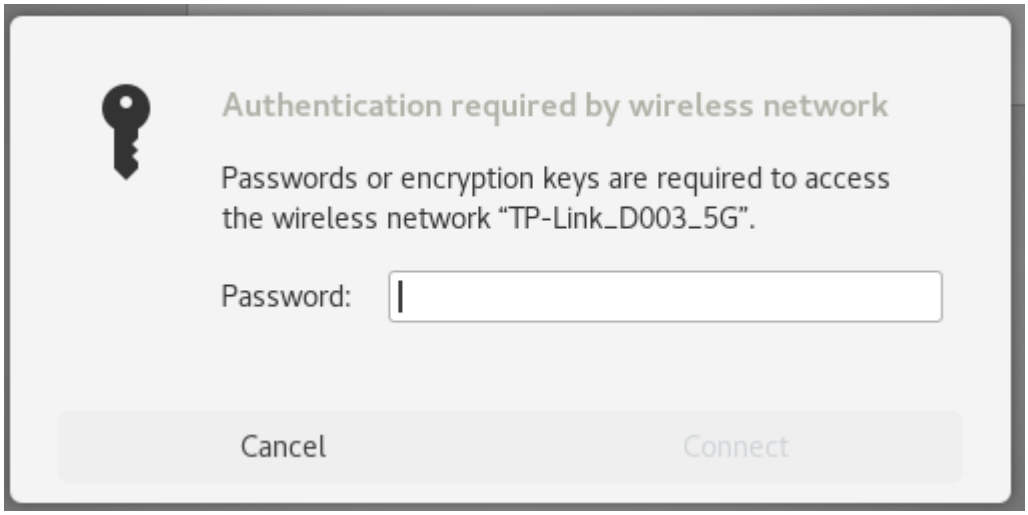
Ubuntu, Mint, Raspberry Pi and Kali all provide friendly graphical interface. After the adapter driver is successfully installed, you can use the graphical interface to manage your wireless settings. The interfaces for different system version are slightly different and here we use the interfaces for Kali 2018.1 as an example for illustration.

1. After successfully loading the driver, you will see a network connection icon in the task bar. Choose Wi-Fi Not Connected > Wi-Fi Settings to display the available wireless networks.

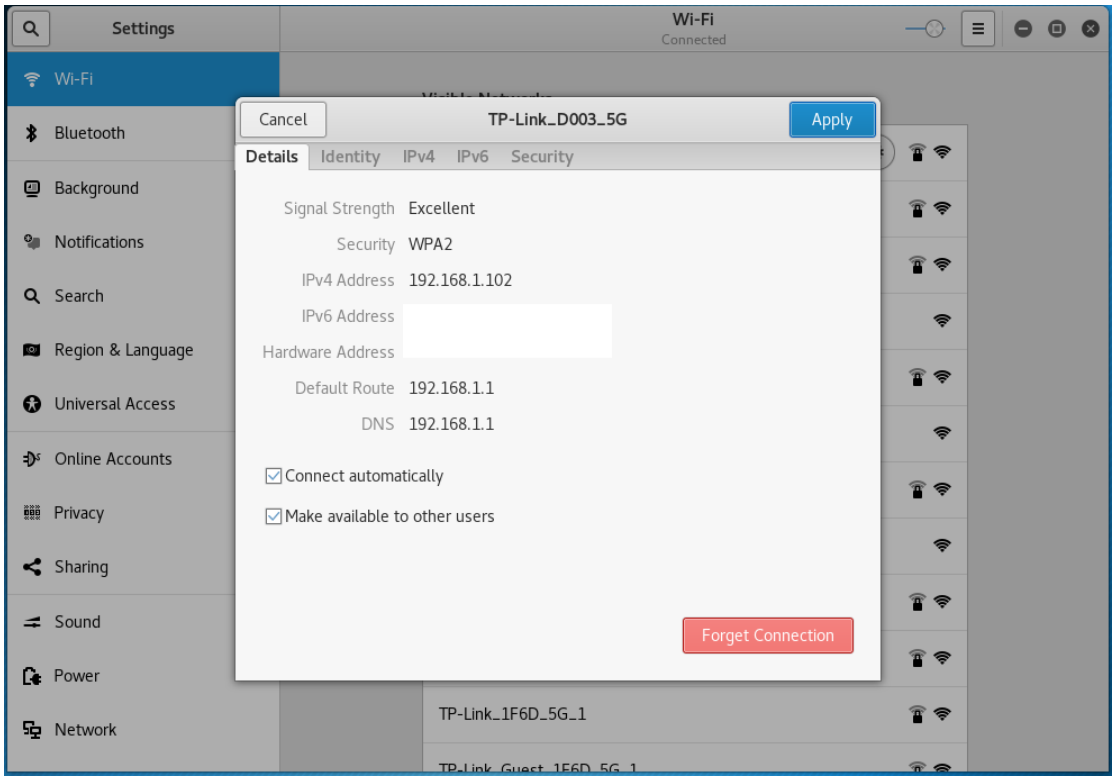




2. Select TP-Link_D003_5G and enter its password to connect to this network.



3. After connecting to this network, you can check its detailed wireless settings.


Settings
Wi-Fi
Connected
Wi-Fi
Bluetooth
Background
Notifications
Search
Region & Language
Universal Access
Online Accounts
Privacy
Sharing
Sound
Power
Network
Cancel
TP-Link_D003_5G
Apply
Details
Identity
IPv4
IPv6
Security
Signal Strength Excellent
Security WPA2
IPv4 Address 192.168.1.102
IPv6 Address
Hardware Address
Default Route 192.168.1.1
DNS 192.168.1.1
Connect automatically
Make available to other users
Forget Connection
TP-Link_1F6D_5G_1

Use the Command Line

You can use commands to manage your wireless setting in Linux. Here we use the interfaces for Kali 2018.1 as an example for illustration.

1. Identify the Device

Inert the USB wireless adapter, and run the following command to check if the adapter is identified.

```
$ lsusb
```

2. Create the Interface

Run the following command to check if the wireless network interface is created.

```
$ ifconfig
```

3. Change the Interface Status to Up

Check if the WLAN interface is *up*. If not, run the following command. Here we use *wlan1* as an example.

```
$ ifconfig wlan1 up
```

If it failed to change to *up*, run the following command to set the state again.

```
$ rfkill unblock wifi
$ ifconfig wlan1 up
```

4. Start wpa_supplicant in the background

Run the following command:

```
$ wpa_supplicant –Dnl80211 –iwlan1 –c ./ wpa_0_8.conf -B
```

Note: wpa_0_8.conf is a file in the current driver directory, go to the driver directory when running the command.

If the command above is not effective, run the following command to end the wpa_supplicant procedure and then run the above command again.

```
$ killall wpa_supplicant
```

If your Linux kernel does not support 802.11, run the following command.

```
$ wpa_supplicant -Dwext -iwlan0 -c ./wpa_0_8.conf -B
```

4.1. Scan Wireless Networks (SSID)

Run the following commands.

```
$ wpa_cli –p /var/run/wpa_supplicant scan
$ wpa_cli –p /var/run/wpa_supplicant scan_results
```

4.2. Connect to the AP

1） Open

Run the following commands

```
$ wpa_cli –p /var/run/wpa_supplicant remove_network 0
$ wpa_cli –p /var/run/wpa_supplicant ap_scan 1
$ wpa_cli –p /var/run/wpa_supplicant add_network
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 ssid '"tplink"' //tplink is the SSID of the desired AP. The SSID is in double quotation marks and then as a whole enclosed by single quotation marks.
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 key_mgmt NONE
$ wpa_cli –p /var/run/wpa_supplicant select_network 0
```

2） WEP40 with open system

```
$ wpa_cli –p /var/run/wpa_supplicant remove_network 0
$ wpa_cli –p /var/run/wpa_supplicant ap_scan 1
$ wpa_cli –p /var/run/wpa_supplicant add_network
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 ssid '"tplink"'
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 key_mgmt NONE
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 wep_key0 1234567890
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 wep_tx_keyidx 0
$ wpa_cli –p /var/run/wpa_supplicant select_network 0
```

3） WEP40 with shared key

```
$ wpa_cli –p /var/run/wpa_supplicant remove_network 0
$ wpa_cli –p /var/run/wpa_supplicant ap_scan 1
$ wpa_cli –p /var/run/wpa_supplicant add_network
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 ssid '"tplink"'
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 key_mgmt NONE
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 wep_key0 1234567890
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 wep_tx_keyidx 0
```

```
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 auth_alg SHARED
$ wpa_cli –p /var/run/wpa_supplicant select_network 0
```

4） WEP 104 with open system

```
$ wpa_cli –p /var/run/wpa_supplicant remove_network 0
$ wpa_cli -p /var/run/wpa_supplicant ap_scan 1
$ wpa_cli -p /var/run/wpa_supplicant add_network
$ wpa_cli -p /var/run/wpa_supplicant set_network 0 ssid '"tplink"'
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 key_mgmt NONE
$ wpa_cli -p /var/run/wpa_supplicant set_network 0 wep_key0
12345678901234567890123456
$ wpa_cli -p /var/run/wpa_supplicant set_network 0 wep_tx_keyidx 0
$ wpa_cli -p /var/run/wpa_supplicant select_network 0
```

5）WEP 104 with open system

```
$ wpa_cli –p /var/run/wpa_supplicant remove_network 0
$ wpa_cli –p /var/run/wpa_supplicant ap_scan 1
$ wpa_cli –p /var/run/wpa_supplicant add_network
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 ssid '"tplink"'
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 key_mgmt NONE
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 wep_key0
12345678901234567890123456
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 wep_tx_keyidx 0
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 auth_alg SHARED
$ wpa_cli –p /var/run/wpa_supplicant select_network 0
```

Note:
If the WEP key is ASCII, run the following command:
#WEP40: wpa_cli -p/var/run/wpa_supplicant set_network 0 wep_key0 '"12345"'
#WEP104: wpa_cli -p/var/run/wpa_supplicant set_network 0 wep_key0
'"1234567890123"'
If the index for WEP key is 0-3, run the following command
#wpa_cli -p/var/run/wpa_supplicant set_network 0 wep_keyX
12345678901234567890123456
#wpa_cli -p/var/run/wpa_supplicant set_network 0 wep_tx_keyidx X

6）TIKP/AES

```
$ wpa_cli –p /var/run/wpa_supplicant remove_network 0
$ wpa_cli –p /var/run/wpa_supplicant ap_scan 1
```

```
$ wpa_cli –p /var/run/wpa_supplicant add_network
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 ssid '"tplink"'
$ wpa_cli –p /var/run/wpa_supplicant set_network 0 psk '"12345678"'
$ wpa_cli –p /var/run/wpa_supplicant select_network 0
```

4.3. Enable DHCP client

Run the following command

```
$ dhclient wlan1
```

After running the command, the adapter will get an IP assigned by the AP. Then you can run the ping command to check if the wireless connection is successful.

Note:

1. Run the commands under the root account.
2. If you use *ifconfig* command to confirm you have obtained the IP address and use *ping* command to confirm you wireless connection is successful, but the internet is still unavailable, you can run the following commands to change the default system

gateway to the router's LAN IP.

```
$ route del default wlan0                  //Delete the default gateway of wlan0
$ route add default gw 192.168.1.1         //Add the router's LAN IP as the default gateway.
```