



tp-link Linux Installation Guide

[Home](#) » [tp-link](#) » tp-link Linux Installation Guide 

tp-link Linux Installation Guide



Contents

- 1 Development Environment
- 2 Compile the Driver
 - 2.1 Compilation tool and kernel sources
 - 2.2 Compile the Driver
- 3 Load the Driver
- 4 Development Environment
- 5 Compile the Driver
 - 5.1 Compilation tool and kernel sources
 - 5.2 Compile the Driver
- 6 Load the Driver
- 7 Development Environment
- 8 Compile the Drive
 - 8.1 Compile Kernel source
 - 8.2 Compile the Driver Source
- 9 Load the Driver
- 10 Development Environment
- 11 Compile the Driver
 - 11.1 Install the Kernel Header File
- 12 Load the Driver
- 13 Use the Graphical Interface
- 14 Identify the Device
- 15 Create the Interface
 - 15.1 Change the Interface Status to Up
- 16 Start wpa_supplicant in the background
 - 16.1 Scan Wireless Networks (SSID)
 - 16.2 Connect to the AP
 - 16.3 Enable DHCP client
- 17 Documents / Resources
 - 17.1 References
- 18 Related Posts

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

Compile the Driver

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:



```
ubuntu@ubuntu:~$ cat /proc/version
Linux version 4.15.0-38-generic (buildd@lgw01-amd64-033) (gcc version 5.4.0 20160808 (ubuntu 5.4.0-6ubuntu1-16.04.9)) #40-16.04.1-Ubuntu SMP Fri Feb 16 23:25:58 UTC 2018
ubuntu@ubuntu:~$
```

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.

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.

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.ko

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

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

Compile the Driver

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:

```

ubuntu-885H-33V-A ~$ cat /proc/version
Linux version 4.10.0-38-generic (build@lgw01-amd64-039) (gcc version 5.4.0 2016
0609 (Ubuntu 5.4.0-6ubuntu1-16.04.4)) #42-16.04.1-Ubuntu SMP Tue Oct 18 16:32:1
8 UTC 2017
ubuntu-885H-33V-A ~$

```

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.

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.

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

Development Environment

The development environment in Raspberry Pi 3 is required as follows

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

Compile the Drive

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

Compile Kernel source

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

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

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.

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 .

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`

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

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.

Kali 2018.1

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

Compile the Driver

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.

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

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

```
root@kali:~# sudo apt-get install linux-headers-$(uname -r)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
linux-headers-4.14.0-kali3-amd64 is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

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

```
root@kali:~# uname -r
4.14.0-kali3-amd64
```

- 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.

 linux-kbuild-4.14-dbgym_4.14.17-1kali_amd64.deb	2018-02-16 12:48 609K
 linux-kbuild-4.14-dbgym_4.14.17-1kali_arm64.deb	2018-02-16 12:16 627K
 linux-kbuild-4.14-dbgym_4.14.17-1kali_armhf.deb	2018-02-16 17:40 599K
 linux-kbuild-4.14-dbgym_4.14.17-1kali_i386.deb	2018-02-16 18:54 593K
 linux-kbuild-4.14-dbgym_4.14.17-1kali_ppc64.deb	2018-02-16 12:54 562K
 linux-kbuild-4.14_4.14.17-1kali_amd64.deb	2018-02-16 12:48 743K
 linux-kbuild-4.14_4.14.17-1kali_arm64.deb	2018-02-16 12:16 720K
 linux-kbuild-4.14_4.14.17-1kali_armhf.deb	2018-02-16 17:40 722K
 linux-kbuild-4.14_4.14.17-1kali_i386.deb	2018-02-16 18:54 724K

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.

linux-headers-4.14.0-kali3-common_4.14.17-1kali1_all.deb	2018-02-16 12:47	5.7M
linux-headers-4.14.0-kali3-common_4.14.17-1kali1_all.deb	2018-02-16 12:47	7.5M
linux-headers-4.14.0-kali3-marvell_4.14.17-1kali1_armel.deb	2018-02-16 17:40	345K
linux-headers-4.14.0-kali3-rt-686-pae_4.14.17-1kali1_i386.deb	2018-02-16 12:53	450K
linux-headers-4.14.0-kali3-rt-amd64_4.14.17-1kali1_amd64.deb	2018-02-16 13:47	453K
linux-headers-4.15.0-kali3-686-pae_4.15.4-1kali1_i386.deb	2018-02-23 10:22	450K
linux-headers-4.15.0-kali3-686_4.15.4-1kali1_i386.deb	2018-02-23 10:22	450K

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)`

detailed linux-header information.

[illegible]

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

```
root@kali:~/Downloads# ls -l /etc/passwd
-rw-r--r-- 1 root:root 41 Feb 18 03:18 /etc/passwd
root@kali:~/Downloads# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
cron:x:4:4:cron:/var/spool/cron/root:/bin/bash
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
sshd:x:65536:65536:ssh:/usr/sbin/nologin
root@kali:~/Downloads# cat /etc/passwd | grep root
root:x:0:0:root:/root:/bin/bash
cron:x:4:4:cron:/var/spool/cron/root:/bin/bash
root@kali:~/Downloads# cat /etc/passwd | grep root | grep /bin
root:x:0:0:root:/root:/bin/bash
cron:x:4:4:cron:/var/spool/cron/root:/bin/bash
```

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.

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.

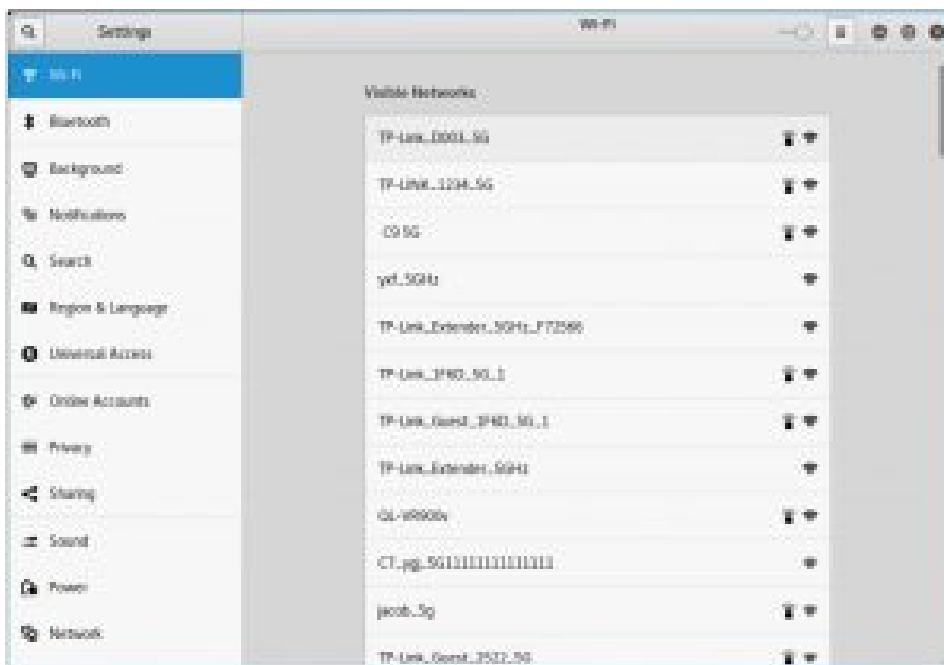
- `$ lsmod`

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.

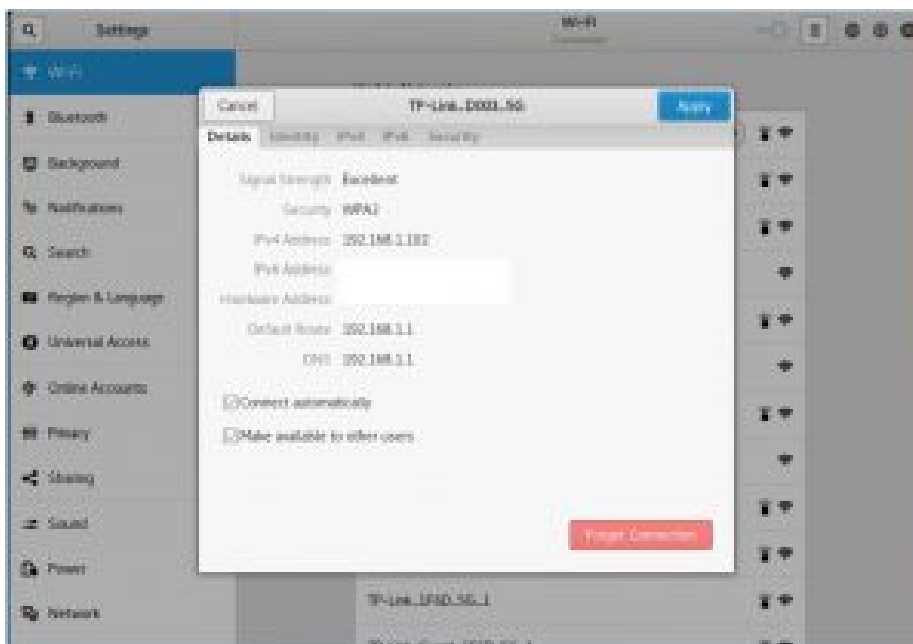




2. Select TP-Link_D003_5G and enter its password to connect to this network



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



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.

Identify the Device

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

- `$ lsusb`

Create the Interface

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

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`

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`

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`

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`

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.

```

root@kali: /home/kali/Documents/wpa_supplicant_hostapd# ifconfig
eth0: flags=4096<UP,BROADCAST,MULTICAST> mtu 1500
    ether 48:8d:5c:1b:34:38 txqueuelen 1000 (Ethernet)
    RX packets 9950 bytes 5063140 (5.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7420 bytes 676787 (660.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 474 bytes 38286 (37.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 474 bytes 38286 (37.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.113 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::c9cc:8e6c:6977:cf24 prefixlen 64 scopeid 0x20<link>
    ether 58:1e:aa:44:85:51 txqueuelen 1000 (Ethernet)
    RX packets 118 bytes 14574 (14.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 181 bytes 11253 (10.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali: /home/kali/Documents/wpa_supplicant_hostapd# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56(84) bytes of data:
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.24 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=1.45 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=1.08 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=1.08 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=7.88 ms
^C
--- 192.168.1.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/ndev = 1.081/2.536/7.867/2.672 ms
root@kali: /home/kali/Documents/wpa_supplicant_hostapd# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default Archer.lan 0.0.0.0 UG 0 0 0 wlan0
192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 wlan0
root@kali: /home/kali/Documents/wpa_supplicant_hostapd#

```


Note:

1. Run the commands under the root account.
2. If you use if config 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.

Documents / Resources

 <p>Installation Guide for Linux</p>	<p>tp-link Linux [pdf] Installation Guide Linux</p>
---	---

References

-  [Index of /kali/pool/main/l/linux](#)
-  [GitHub - raspberrypi/linux: Kernel source tree for Raspberry Pi Foundation-provided kernel builds.](#)
[Issues unrelated to the linux kernel should be posted on the community forum at <https://www.raspberrypi.org/forum>](#)
-  [GitHub - raspberrypi/tools](#)

Manuals+.