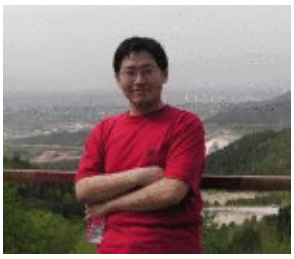
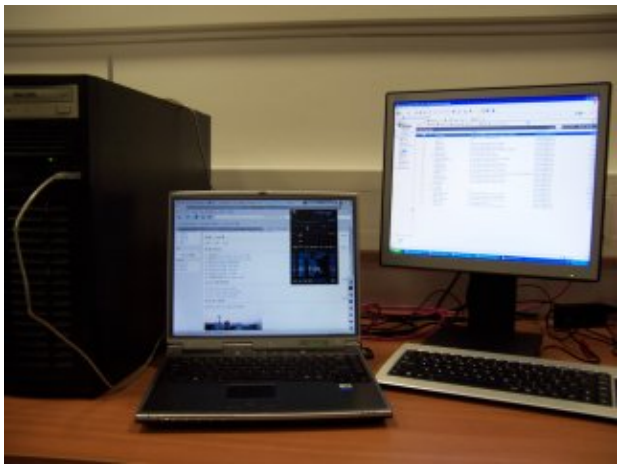


by Wang Xu
<wangxu[at]linuxfocus.org>

About the author:

Wang Xu is a PhD student in Beijing University of Posts and Telecommunications, China, and engages in wireless communication. He became a Linux lover since 1999 when he was a college student. Besides Linux, he also likes TeX, C/C++, Perl, etc.

Wireless LAN under Linux



Abstract:

In this article, the author talks about the drivers for some common Wireless LAN adapters and some related issues such as authentication based on 802.1x.

Translated to English by:
Wang Xu
<wangxu[at]linuxfocus.org>

Introduction

WLAN (IEEE 802.11b/a/g) becomes more and more popular as WLAN devices become cheaper and more organizations provide WLAN access to their staff or to the public. Almost every new laptop computer has an embedded WLAN card and older ones may use a PCMCIA WLAN card; even desktop computers may have USB WLAN cards or even on−board WLAN cards. On the other hand, WLAN Access Points (AP) are deployed on campuses, office buildings, hotels, residences and so on. The WLAN facilitates the deployment of local networks and supports mobile/nomadic computing, which is another revolution to our working and daily life.

Thus, in the Linux world, it is also essential to support WLAN access. The rest of this article is organized as follows: let's consider how to drive the cards first; then try to access networks using authentication; and give a brief introduction of the tools for WLAN interface configuration; in the end, we will draw a conclusion.

Driving WLAN Cards

Once you get a Wireless LAN card installed in your computer, the first step to do is to install the driver and make it work. A WLAN card implements the functions of the physical layer (PHY) and media access control sublayer (MAC) as specified in at least one of the IEEE 802.11 series of protocols, while the driver controls the card, provides a network interface identical to an ethernet interface, and provides other WLAN specific management interfaces.

There is no unified method to install drivers for the diverse number of vendors and cards. However, most of them can be driven through three methods:

- Use the Linux® kernel's native support for the card,
- To compile and install a driver module for the specified card,
- To employ the NDIS wrapper [1] to drive cards using drivers for MS Windows ®.

In the following sections, the author will illustrate these methods with examples.

Note, even if you use one of the latter two methods, you must make sure that the wireless LAN support has been set in the kernel configuration:

```
gnawux@APTITUDE:~$ grep CONFIG_NET_RADIO /boot/config-`uname -r`
CONFIG_NET_RADIO=y
```

If not, you should reconfigure the kernel and enable the option "Wireless LAN (non−hamradio) Drivers and Wireless Extensions".

Kernel support for WLAN

The drivers that are mature enough and that don't have license issues are introduced into the Linux Kernel. Therefore, the supported WLAN cards list in kernel is dependent on the version of kernel, and it is smart to check whether the new kernel has better support for your card before you install the driver.

In this section, the author will illustrate how to use the driver for WLAN cards based on the Intersil Prism chipsets (ISL38xx). The supported cards list can be found at http://prism54.org [2].

To use a Prism based card, you need the latest 2.6 kernel, and enable the "Intersil Prism GT/Duette/Indigo PCI/Cardbus" option in the Wireless LAN driver section of the kernel configuration, and re−build the kernel.

If you read the help text of the module carefully when you configure the kernel, you may find that you need to get a firmware from the prism54.org project website [2]. This is because the card does not have an EPROM to store its firmware. Therefore, it needs to download the firmware when the driver is initializing the card. The firmware can not be part of the kernel because of a license issue. After getting the firmware and putting it into "/usr/lib/hotplug/firmware/", reboot your computer, and you will find that you have an additional ethernet interface provided by the WLAN card.

Independent Modules for Specific WLAN Cards

Many new cards, just like other new hardware, have no GPL−compatible drivers provided by the vendors, or the drivers developed by the open source community have not become mature enough to be included in the kernel. Thus, these drivers are provided as modules, and some of them will eventually be added into future

kernels when they are completed.

One of the most famous drivers is ipw2100 [3] for the Intel Pro/Wireless 2100 card, which is a part of the Intel Centrino® technology and installed in many laptop computers. In this section, the author will introduce the installation of the ipw2100 driver.

Firstly, you need to download the source package of the driver as well as the firmware from the project web site, http://ipw2100.sourceforge.net. After making sure your kernel is recent enough and built with support for modules, hotplug firmware, and wireless LAN as said above, uncompress the source package:

```
APTITUDE:/usr/src# tar -zxvf ipw2100-1.0.1.tgz
```

Then go into the source directory to build and install it:

```
APTITUDE:/usr/src/ipw2100-1.0.1# make
APTITUDE:/usr/src/ipw2100-1.0.1# make install
```

Having installed the modules, it indicates that you need to install the firmware:

```
Don't forget to copy firmware to /usr/lib/hotplug/firmware/ and have the
hotplug tools in place.
```

Just as the message says: uncompress the firmware into the hotplug directory, then the installation procedure is completed. Now you can enable the ipw2100 module by doing:

```
APTITUDE:/usr/src/ipw2100-1.0.1# modprobe ipw2100
```

And you may add some parameters here for different configurations. For instance, the ifname parameter can specify the interface name:

```
APTITUDE:/usr/src/ipw2100-1.0.1# modprobe ipw2100 ifname=wlan0
```

Thus the interface will be named as wlan0. For other parameters, you can read the documents in the source package of the ipw2100 driver.

Driving other cards

Unfortunately, some cards have no drivers for Linux at all or the driver does not work for a certain reason. However, this does not mean that we cannot use it under Linux. At least, we have NDIS wrapper [1].

Most of the WLAN cards for desktop or laptop computers support Windows 2000/XP, which handles the WLAN support by a standard interface called NDIS. Therefore, the drivers for such cards usually support NDIS. Thus we can wrap such a driver and let it work under Linux as if it was Windows 2000/XP, which leads us to the ndiswrapper project.

In this section, the author will install the ndiswrapper for a Netgear 121 WLAN card as an example. Firstly, you should download the ndiswrapper from the project site, http://ndiswrapper.sourceforge.net and prepare the NDIS driver for Windows. The ndiswrapper consists of a kernel module and a set of tools. You should build and install it:

```
APTITUDE:/usr/src/ndiswrapper-0.11# make install
```

Then you can load the windows driver in the wrapper

```
APTITUDE:/usr/src/ndiswrapper-0.11# ndiswrapper -i ../wg121/WG121V200/ndis5/netwg121.inf
```

where the `.inf` file is the NDIS driver for Windows. Having installed the NDIS driver, you may see

```
APTITUDE:/usr/src/ndiswrapper-0.11# ndiswrapper -l
Installed ndis drivers:
netwg121        driver present
```

Cheers! The installation procedure is completed.

Authentication

If you access a WLAN in some public environment, the WLAN may require some authentication methods for security reasons. Most of the available authentication methods for WLAN are based on IEEE 802.1x (EAP) and IEEE 802.11i, and EAP based methods are currently the most popular.

There are many EAP based authentication methods, e.g. EAP−MD5, EAP−TLS, EAP−TTLS, EAP−SIM, LEAP, etc. Linux users can use *xsupplicant* provided by the Open1x project [4] to access a network that requires 802.1x based authentication. In this section, the author will use the LEAP protocol, which is proposed by Cisco corp., as an illustration. Note: whether the protocol can be supported is dependent on the card and the driver, i.e. even if your xsupplicant is correctly installed and configured, you may still be unable to access the network because of the card or driver not supporting it.

You should download *xsupplicant* from the project site, http://open1x.sourceforge.net, and install it. Then modify the configuration file, /etc/xsupplicant/xsupplicant.conf. Here is an example for LEAP.

```
#example of /etc/xsupplicant/xsupplicant.conf
#for LEAP protocol

network_list = all
#the list of networks to access

default_netname = default
#the default access network

first_auth_command = <BEGIN_COMMAND>dhclient %i<END_COMMAND>
#The command before authentication, which is usually used to get some info from
#the network

logfile = /var/log/xsupplicant.log
#log file

myssid #here is your network id, may be listed in the network list
{
  type = wireless
  ssid = <BEGIN_SSID>myssid<END_SSID>
  allow_types = all
  identity = <BEGIN_ID>aptitude<END_ID>
  eap-leap {
      username = <BEGIN_UNAME>aptitude<END_UNAME>
      password = <BEGIN_PASS>passwd<END_PASS>
  }#setup for leap
}
```

LEAP is a simple authentication method, and there are many other settings for other methods, please refer to the examples and documents from xsupplicant.

Utilities for WLAN

As you know, WLAN provides an network interface identical to ethernet, and you can use it as another ethernet interface. On the other hand, a WLAN card has much more features than a ethernet card because of the wireless media. Thus there is a set of tools to configure WLAN and obtain information about the status of it. You can find the wireless tools from http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html [5], which is contributed by Jean Tourrilhes.

The most useful tool is *iwconfig*, which can be used similar to ifconfig. The iwconfig command without any parameters but the interface name will print the working status of the card:

```
gnawux@APTITUDE:~$ /sbin/iwconfig wlan0
wlan0     unassociated  ESSID:off/any  Nickname:"ipw2100"
          Mode:Managed  Channel=0  Access Point: 00:00:00:00:00:00
          Bit Rate=0 kb/s   Tx-Power:off
          Retry:on   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

With the "mode" parameter, you can change the WLAN card working mode:

```
APTITUDE:/home/gnawux# iwconfig wlan0 mode 1
APTITUDE:/home/gnawux# iwconfig wlan0
wlan0     unassociated  ESSID:off/any  Nickname:"ipw2100"
          Mode:Ad-Hoc  Channel=0  Cell: 00:00:00:00:00:00
          Bit Rate=0 kb/s   Tx-Power:off
          Retry:on   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Here we change the WLAN card into "Ad Hoc" mode. We can also change the network name through the "essid" parameter:

```
APTITUDE:/home/gnawux# iwconfig wlan0 essid gnawux
APTITUDE:/home/gnawux# iwconfig wlan0
wlan0     IEEE 802.11b  ESSID:"gnawux"  Nickname:"ipw2100"
          Mode:Ad-Hoc  Frequency:2.412 GHz  Cell: 02:0C:F1:0F:11:2A
          Bit Rate=0 kb/s   Tx-Power:off
          Retry:on   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=60/100  Signal level=-83 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

You maybe have noticed that we have now more reasonable status info than before because we have a valid ESSID. For other parameters of iwconfig, please read iwconfig(8).

Another powerful utility is *iwlist*, with which we can get the list of available resources. With the "scanning" parameter, we can obtain a list of the available access points:

```
gnawux@APTITUDE:~$ /sbin/iwlist wlan0 scanning
wlan0     Scan completed :
          Cell 01 - Address: 00:0D:BD:6F:B4:48
                    ESSID:"<hidden>"
                    Protocol:IEEE 802.11b
                    Mode:Master
                    Channel:6
                    Encryption key:on
                    Bit Rate:11 Mb/s
                    Extra: Rates (Mb/s): 1 2 5.5 11
                    Extra: Signal: -70  dBm
                    Extra: Last beacon: 59ms ago
          Cell 02 - Address: 86:CF:C1:34:12:06
                    ESSID:"gnawux"
                    Protocol:IEEE 802.11b
                    Mode:Ad-Hoc
                    Channel:11
                    Encryption key:off
                    Bit Rate:11 Mb/s
                    Extra: Rates (Mb/s): 1 2 5.5 11
                    Extra: Signal: -37  dBm
                    Extra: Last beacon: 2ms ago

</hidden>
```

And with the "frequency" parameter (`freq`), we can get the frequency list:

```
gnawux@APTITUDE:~$ /sbin/iwlist wlan0 freq
wlan0     14 channels in total; available frequencies :
          Channel 01 : 2.412 GHz
          Channel 02 : 2.417 GHz
          Channel 03 : 2.422 GHz
          Channel 04 : 2.427 GHz
          Channel 05 : 2.432 GHz
          Channel 06 : 2.437 GHz
          Channel 07 : 2.442 GHz
          Channel 08 : 2.447 GHz
          Channel 09 : 2.452 GHz
          Channel 10 : 2.457 GHz
          Channel 11 : 2.462 GHz
          Channel 12 : 2.467 GHz
          Channel 13 : 2.472 GHz
          Channel 14 : 2.484 GHz
          Current Channel=1
```

And you can refer to iwlist(8) for other parameters.

Besides the above two, there are other utilities, e.g. iwevent, iwgetid, iwpriv, iwspy, for you to obtain the working status of your WLAN card and manage it.

Conclusion

The author has shown the driver installation for different WLAN cards, and illustrated how to perform

authentication. Powerful configuration utilities for WLAN are also introduced in the article.

Thanks to the contribution from the open source community, we can not only access Wireless LAN under Linux, but also enjoy it!

References

1. NDIS wrapper project, http://ndiswrapper.sourceforge.net;
2. Prism54 project, http://prism54.org;
3. IPW2100 project, http://ipw2100.sourceforge.net;
4. Open1x project, http://open1x.sourceforge.net;
5. Jean Tourrilhes, wireless tools, http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html.
