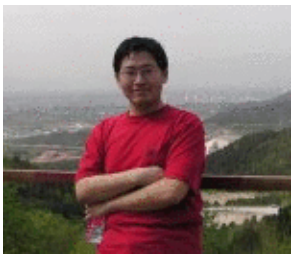
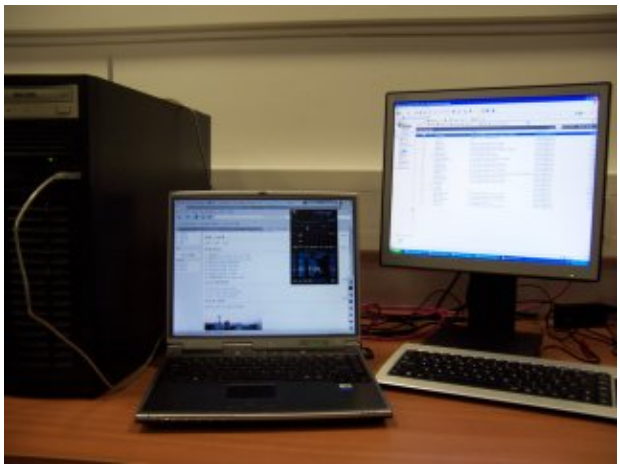


von Wang Xu
<wangxu[at]linuxfocus.org>

Über den Autor:

Wang Xu ist PhD Student an der Beijinger Universität für Post und Telekommunikation in China und beschäftigt sich mit drahtloser Kommunikation. Er wurde 1999 als Collegestudent zum Linuxliebhaber. Außer Linux mag er auch TeX, C/C++, Perl, etc.

Wireless LAN unter Linux



Zusammenfassung:

In diesem Artikel spreche ich über einige Treiber für gewöhnliche Wireless LAN Adapter und einige verwandte Themen wie die Autentifizierung basierend auf 802.1x.

Übersetzt ins Deutsche von:
Katja Socher
<katja[at]linuxfocus.org>

Einleitung

WLAN (IEEE 802.11b/a/g) wird immer populärer, da WLAN Karten immer billiger werden und mehr Organisationen ihren Angestallten oder der Öffentlichkeit Zugang zu WLAN ermöglichen. Fast jeder neue Laptop−Computer hat eine integrierte WLAN Karte und ältere können PCMCIA WLAN Karten benutzen, sogar Desktop−Computer können USB WLAN Karten haben oder sogar eingebaute WLAN Karten. Andererseits werden WLAN Access Points (AP) an Universitäten, in Bürogebäuden, Hotels, Wohnhäusern etc. eingerichtet. Das WLAN erleichtert das Einrichten lokaler Netzwerke und unterstützt mobiles/normadisches Computern, was eine weitere Revolution in unserem Arbeits− und täglichen Leben darstellt.

Von daher ist es auch für die Linuxwelt essentiell, WLAN Zugang zu unterstützen. Der Rest dieses Artikels ist wie folgt unterteilt: laßt uns überlegen, wie man die Treiber für die Karten zum Laufen kriegt; dann versuchen wir, zu Netzwerken, die Autentifizierung benutzen, Zugang zu bekommen; und es gibt eine kurze Einführung in die Werkzeuge für die WLAN Interface Konfiguration; schließlich gibt es eine Schlußbetrachtung.

WLAN Kartentreiber zum Laufen kriegen

Wenn du einmal eine Wireless LAN Karte in deinem Computer installiert hast, ist der erste Schritt, den Treiber zu installieren und zum Laufen zu kriegen. Eine WLAN Karte implementiert die Funktionen der physikalischen Schicht (PHY) und des media access control sublayer (MAC), wie es in mindestens einem der IEEE 802.11 Serien von Protokollen spezifiziert wurde, während der Treiber die Karte steuert, eine Netzwerkschnittstelle identisch zu einer Ethernetschnittstelle sowie andere WLAN spezifische Managementschnittstellen zur Verfügung stellt.

Es gibt keine einheitliche Methode zum Installieren von Treibern bei der großen Zahl von Verkäufern und Karten. Die meisten Treiber können jedoch durch eine der folgeden drei Methoden zum Laufen gebracht werden:

- Benutze die native Linux®kernelunterstützung für die Karte,
- Kompiliere und installiere ein Treibermodul für die spezifizierte Karte ,
- Benutze den NDIS wrapper [1], um Karten zu betreiben, die MS Windows ® Treiber benutzen.

In den folgenden Abschnitten illustriere ich diese Methoden mit Beispielen.

Anmerkung, auch wenn du die letzten beiden Methoden benutzt, mußt du sicherstellen, dass der wireless LAN support in deiner Kernelkonfiguration gesetzt ist:

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

Wenn das nicht der Fall ist, solltest du den Kernel noch mal mit der aktivierten Option "Wireless LAN (non−hamradio) Drivers and Wireless Extensions" konfigurieren.

Kernelunterstützung für WLAN

Die Treiber, die ausgereift genug sind und keine Lizenzprobleme machen, werden zum Linux Kernel hinzugefügt. Von daher ist die Liste der unterstützten WLAN Karten im Kernel abhängig von der Version des Kernels und es ist deshalb klug, zu überprüfen, ob ein neuer Kernel eine bessere Unterstützung für deine Karte bietet, bevor du den Treiber installierst.

In diesem Abschnitt illustriere ich, wie man den Treiber für WLAN Karten basierend auf Intersil Prism chipsets (ISL38xx) benutzt. Die Liste mit den unterstützten Karten findet man unter http://prism54.org [2].

Um eine auf Prism basierende Karte benutzen zu können, braucht man den neuesten 2.6 Kernel, und aktiviert die "Intersil Prism GT/Duette/Indigo PCI/Cardbus" Option im Wireless LAN Treiber Abschnitt der Kernelkonfiguration, und baut den Kernel erneut.

Wenn du den Hilfetext des Moduls sorgfältig durchliest, wenn du den Kernel konfigurierst, wirst du vielleicht feststellen, daß du eine firmware von der prism54.org project Webseite [2] brauchst. Dies ist so, weil die Karte keinen EPROM hat, um die firmware zu speichern. Deshalb muß man die firmware herunterladen, wenn der Treiber die Karte initialisiert. Die firmware kann wegen ihrer Lizenz nicht Teil des Kernels werden. Nach dem Holen der firmware and schieben nach "/usr/lib/hotplug/firmware/", reboote deinen Computer, und du wirst feststellen, daß du eine zusätzliche Ethernetschnittstelle hast, die von der WLAN Karte zur Verfügung gestellt wird.

Unabhängige Module für spezielle WLAN Karten

Viele neue Karten haben, wie viele andere neue Hardware, keine GPL−kompatiblen Treiber der Hersteller oder die Treiber, die von der Open Source Community entwickelt wurden, sind noch nicht ausgereift genug, um in den Kernel integriert zu werden. Deshalb werden diese Treiber als Module bereitgestellt und einige von ihnen werden schließlich in zukünftigen Kernelversionen hinzugefügt werden, wenn sie vollständig sind.

Einer der bekanntesten Treiber ist der ipw2100 [3] für die Intel Pro/Wireless 2100 Karte, die ein Teil der Intel Centrino® Technologie ist und auf vielen Laptops installiert ist. In diesem Abschnitt führe ich in die Installation des ipw2100 Treibers ein.

Zuerst muß man das Quellpaket des Treibers sowie die firmware von der Projektwebseite http://ipw2100.sourceforge.net herunterladen. Nachdem man sichergestellt hat, daß sein Kernel neu genug ist und mit Unterstützung für die Module hotplug firmware, and wireless LAN wie eben gesagt gebaut wurde, dekomprimiert man das Quellpaket:

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

Gehe dann zum Bilden und Installieren in das Quellverzeichnis:

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

Nach dem Installieren der Module, muß man nun die firmware installieren:

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

Genau so, wie die Meldung es sagt: dekomprimiere die firmware in das hotplug Verzeichnis, dann ist die Installationsprozedur beendet. Jetzt kannst du durch das folgende das ipw2100 Modul aktivieren:

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

Außerdem fügt man hier vielleicht noch ein paar Parameter für verschiedene Konfigurationen hinzu. Zum Beispiel kann der ifname Parameter den Schnittstellennamen spezifizieren:

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

D.h. die Schnittstelle wird wlan0 genannt. Für andere Parameter kann man die Dokumente im Quellpaket des ipw2100 Treibers lesen.

Treiber für andere Karten

Leider gibt es für einige Karten überhaupt keine Treiber für Linux oder der Treiber läuft aus bestimmten Gründen nicht. Dies bedeutet jedoch nicht, daß wir sie unter Linux nicht benutzen können. Zumindest gibt es NDIS wrapper [1].

Die meisten WLAN Karten für Desktop− und Laptopcomputer unterstützen Windows 2000/XP, das die WLAN Unterstützung durch eine Standardschnittstelle namens NDIS handhabt. Deshalb unterstützen die Treiber für solche Karten normalerweise NDIS. Deshalb können wir solch einen Treiber einwickeln (wrap) und unter Linux zum Laufen bringen, als wenn es Windows 2000/XP wäre, was uns zum ndiswrapper Projekt führt.

In diesem Abschnitt installiere ich den ndiswrapper für eine Netgear 121 WLAN Karte als ein Beispiel. Als erstes sollte man den ndiswrapper von der Projektseite http://ndiswrapper.sourceforge.net herunterladen und den NDIS Treiber für Windows vorbereiten. Der ndiswrapper besteht aus einem Kernelmodul und einem Satz von Werkzeugen. Du solltest es bauen und installieren:

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

Dann kannst du den Windowstreiber im Wrapper laden

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

wobei die `.inf` Datei der NDIS Treiber für Windows ist. Nach der Installation des NDIS Treibers solltest du das folgende sehen

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

Cheers! Die Installationsprozedur ist beendet.

Autentifizierung

Wenn du in einem öffentlichen Bereich auf ein WLAN zugreifst, verlangt das WLAN eventuell einige Autentifizierungsmethoden aus Sicherheitsgründen. Die meisten verfügbaren Autentifizierungsmethoden für WLAN basieren auf IEEE 802.1x (EAP) und IEEE 802.11i, und auf EAP basierende Methoden sind momentan die beliebtesten.

Es gibt viele auf EAP basierende Autentifizierungsmethoden, z.B. EAP−MD5, EAP−TLS, EAP−TTLS, EAP−SIM, LEAP, etc. Linuxbenutzer können *xsupplicant* benutzen, das von Open1x project [4] zur Verfügung gestellt wird, um auf ein Netzwerk zuzugreifen, das auf 802.1x basierende Autentifizierung erfordert. In diesem Abschnitt benutze ich das LEAP protocol, das von Cisco corp. vorgeschlagen wurde, als eine Illustration. Beachte: ob das Protokol unterstützt wird, hängt von der Karte und dem Treiber ab, z.B. selbst wenn dein xsupplicant richtig installiert und konfiguriert wurde, kann es dir trotzdem unmöglich sein, auf das Netzwerk zuzugreifen, weil deine Karte oder dein Treiber es nicht unterstützen.

Du solltest *xsupplicant* von der Projektseite http://open1x.sourceforge.net herunterladen und installieren. Dann ändere die Konfigurationsdatei, /etc/xsupplicant/xsupplicant.conf. Hier ist ein Beispiel für 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 ist eine einfache Autentifizierungsmethode, und es gibt viele Einstellungen für andere Methoden, bitte lies die Beispiele und Dokumente von xsupplicant.

Utilities für WLAN

Wie du weißt, stellt WLAN eine Netzwerkschnittstelle identisch zu Ethernet zur Verfügung, und du kannst sie als eine weitere Ethernetschnittstelle benutzen. Auf der anderen Seite hat eine WLAN Karte viel mehr Features als eine Ethernetkarte wegen des drahtlosen Mediums. Deshalb gibt es eine Menge Werkzeuge, um WLAN Karten zu konfigurieren und Informationen über den Status zu bekommen. Du findest die wireless tools auf http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html [5], die von Jean Tourrilhes beigesteuert wurde.

Das nützlichste Werkzeug ist *iwconfig*, das ähnlich wie ifconfig benutzt werden kann. Der iwconfig Befehl ohne Parameter, aber mit dem Schnittstellennamen druckt den Arbeitsstatus der Karte:

```
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
```

Mit dem "mode" Parameter kannst du den WLAN Karten Arbeitsmode ändern:

```
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
```

Hier ändern wir die WLAN Karte in den "Ad Hoc" mode. Wir können auch den Netzwerknamen durch den "essid" Parameter ändern:

```
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
```

Du hast vielleicht bemerkt, dass wir jetzt vernünftigere Statusinformationen haben als vorher, weil wir eine gültige ESSID haben. Für andere Parameter von iwconfig lies bitte iwconfig(8).

Ein weiteres mächtiges utility ist *iwlist*, mit dem wir eine Liste der verfügbaren Ressourcen bekommen können. Mit dem "scanning" Parameter können wir eine Liste verfügbarer Zugriffspunkte (access points) bekommen:

```
gnawux@APTITUDE:~$ /sbin/iwlist wlan0 scanning
wlan0     Scan completed :
          Cell 01 - Address: 00:0D:BD:6F:B4:48
                    ESSID:""
                    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
```

Und mit dem "Frequenz" Parameter (`freq`), können wir eine Frequenzliste bekommen:

```
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
```

Und du kannst iwlist(8) für weitere Parameter lesen.

Außer den beiden obengenannten gibt es auch noch andere utilities, z.B. iwevent, iwgetid, iwpriv, iwspy, um den Arbeitsstatus deiner WLAN Karte zu erhalten und zu verwalten.

Schlußbemerkung

Ich habe die Treiberinstallation für verschiedene WLAN Karten gezeigt und illustriert, wie man eine Autentifizierung durchführt. Außerdem wurden mächtige Konfigurations−utilities für WLAN in diesem Artikel angesprochen.

Dank der Beteiligung der Open Source Community können wir nicht nur auf Wireless LAN unter Linux zugreifen, sondern auch Spaß damit haben!

Referenzen

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

Der LinuxFocus Redaktion schreiben © Wang Xu "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Autoren und Übersetzer: cn −−> −− : Wang Xu <wangxu[at]linuxfocus.org> cn −−> en: Wang Xu <wangxu[at]linuxfocus.org> en −−> de: Katja Socher <katja[at]linuxfocus.org>

2005−01−20, generated by lfparser_pdf version 2.51