Using ndiswrapper for unsupported wireless cards

Wireless card support in linux is pretty good nowadays, as more and more vendors realize there is not per definition evil in using and releasing Open Source Software, or in co-operation with Open Source developers. For the old-style 11Mbit PCMCIA cards, drivers have existed for some time now (hostap, linux-wlan) and a few 54 Mbit card drivers have even made it into the kernel (prism54 and ipw2200/ipw2200). Drivers like madwifi (for Atheros chipsets) and rt2x00 (for RaLink chipsets) receive vendor support and are covering a significant part of currently sold wireless cards, while bcm43xx (for Broadcom 43xx chips) and acx100/111 (for Texas Instruments chipsets) are reverse-engineered drivers due to lack of support from the manufacturers, who refuse to publish specifications of their hardware.

Still, there are many other wireless cards for which no driver exists at all. An Open Source project, ndiswrapper, was created to fill this gap. Ndiswrapper implements the needed Windows kernel API calls and NDIS (Network Driver Interface Specification) API calls as a Linux kernel module. This allows ndiswrapper to load a Windows driver for any wireless network card and let this driver run natively. Nearly all wireless cards with a Windows driver can be made functional in Linux this way, although this approach puts an extra load on your CPU because of the extra software layer.
Also, ndiswrapper-driven cards tend to crash the kernel in high-load situations and even though you should run a Linux kernel with at least 8K stack size (which is the default in Slackware), some NDIS drivers seem to need stack sizes as high as 32K to properly function.
The 32K stack size is not something you can configure in an unpatched kernel, and since I have only read about these large kernel stack sizes in non-Slackware mailing lists, I'd say: stick to the default stack size of 8K you find in Slackware kernel, unless you experience strange kernel crashes that you can trace back to the ndiswrapper module…

For now, rest assured that you probably will not need to change anything to your running kernel to get ndiswrapper support for your wireless card. The rest of the page will concern itself with detailing how to obtain the ndiswrapper source code, how to build and install the software, how to install a Windows driver, and how to configure your card. WPA encryption is discussed in the final paragraph.

Obtaining the source

Ndiswrapper releases can be downloaded from SourceForge. For Slackware 10.2 and earlier, you should stick to release 1.15 or earlier (see the next paragraph). For 2.4 kernels, 1.14 is the last one that compiles. Note that you will have to be running a supported kernel, which means at least 2.6.6 or 2.4.26.

Building and installing ndiswrapper

Note that at some point, the developers seem to have dropped support for the older GCC 3.3 compiler. I can not build the 1.16 release on Slackware 10.2 for instance. You can either compile the downloaded source package yourself, by running

```
tar -zxvf ndiswrapper-<release>.tar.gz
cd ndiswrapper-<release>
make distclean
make install
```

or get a Slackware package for your specific kernel if present, from my repository. If a package for your kernel is not found, you can still use my SlackBuild script to create yourself a Slackware package from source. Run the followinf commands to download the necessary sources and build the package. You'll find the result in the /tmp directory. Read the log output from the build carefully to see if everything went OK!

```
mkdir ndiswrapper
cd ndiswrapper
lftp -c "open http://www.slackware.com/~alien/slackbuilds/ndiswrapper;
mirror build"
cd build
./ndiswrapper.SlackBuild
```

Finally, install the package using the command

```
installpkg /tmp/ndiswrapper-<release>_<kernelversion>-*.tgz
```

where you supply the release and your kernelversion - for instance *installpkg /tmp/ndiswrapper-1.15_2.6.13-i486-1.tgz*.

Installing a Windows driver

You can use the Windows driver that came with your card, and I have not had problems doing so yet, but the ndiswrapper Wiki states that it is better to look up your card's PCI-ID in the supported cards list.

- Look up the PCI ID for your card by first running the command

  ```
  lspci
  ```

  and check the line of output for your card. Mine for instance looks like

  ```
  02:07.0 Network controller: RaLink RT2500 802.11g Cardbus/mini-PCI (rev
  01)
  ```

- Next, look up the starting numbers of that line in the output of the command

  ```
  lspci -n
  ```

  For me, that would be

  ```
  02:07.0 Class 0280: 1814:0201 (rev 01)
  ```

The PCI-ID is `1814:0201` shown in bold here: “*02:07.0 Class 0280:* ***1814:0201*** *(rev 01)*”. The abovementioned List Page will have one or more links to the required Windows .INF and .SYS files you need for the next step.

- If the download is a Windows installer, you might need a tool like cabextract or unshield to be able to extract the driver files.
- Once you finally have the extracted Windows NDIS driver in the form of two .INF and .SYS files and possibly .BIN or other files containing firmware, all together in your current directory, proceed with installing them so they can be used by ndiswrapper. Run the ndiswrapper helper application like this:

  ```
  ndiswrapper -i filename.INF
  ```

  This will install the drivers to the directory `/etc/ndiswrapper` and write a configuration file.

* You can check if this was done correctly by running

```
ndiswrapper -l
```

which should list the installed drivers and detected hardware. In my case, the output is

```
Installed drivers:
rt2500          driver installed, hardware present
```

which means the driver that was installed can be used to drive the hardware.

Activating the card on boot

The ndiswrapper kernel module will not load automatically. You can add the following line to the file `/etc/rc.d/rc.modules` to make it load on every boot:

```
/sbin/modprobe ndiswrapper
```

If you want to have the network interface be known as *wlan0* then you can either run

```
ndiswrapper -m
```

which adds a line

```
alias wlan0 ndiswrapper
```

to /etc/modprobe.d/ndiswrapper (2.6.15 kernels and onward) or you can write that line yourself, in `/etc/modules.conf` (for 2.4 kernels) or `/etc/modprobe.conf` (older 2.6 kernels).

Network configuration

Using ndiswrapper as the driver, your wireless network interface will be assigned the name ***wlan0***. Detailed information about how to configure your wlan0 interface can be found in my Wiki page describes Slackware's wireless network configuration in detail. Just change the string '**ath0**' I use in the examples to '**wlan0**'.

WPA encryption for your wireless network

For more information about how to configure WPA using wpa_supplicant, read my WPA chapter in the *Slackware network configuration* page.

With ndiswrapper version 1.12 and later, you should use wpa_supplicant's ***wext*** driver instead of the *ndiswrapper* driver: the argument to wpa_supplicant would become '-Dwext' instead of '-Dndiswrapper'
A typical configuration line in `rc.inet1.conf` would then become (adjust my example index [**1**] to the value you use for your card):

```
WLAN_WPADRIVER[1]="wext"
```