

WLAN-Karten, die nativ supportet werden

3Com: 3Com 3CRPAG175

AZTech: Aztech WL830PC

BayStack: BayStack 650, 660

Cisco: Aironet 802.11b wireless adapters

D-Link: DWL-A520, DWL-A650, DWL-AB650, DWL-AG520, DWL-AG650, DWL-G520B, DWL-G650B

Elecom: LD-WL54, LD-WL54AG

Farallon: SkyLINE

Fujitsu: E5454, E5454, FMV-JW481

Hewlett-Packard: P NC4000

I/O Data: WN-A54, WN-AB, WN-AG

Icom: SL-200

Linksys: WMP55AG, WPC51AB, WPC55AG

Melco: WLI-PCM

NEC: PA-WL/54AG

NEL: SSMagic

Netgear: WAB501, WAG311, WAG511, WG311, WG311T, WG511T

NetWave: AirSurfer, AirSurfer Plus, AirSurfer Pro

Nokia: C020 WLAN

Orinoco: 8470WD, 8480

Proxim: Skyline 4030, Skyline 4032

Raytheon: Raylink 2.4GHz wireless adapters

Samsung: SWL-5200N

SMC: SMC2735W

Sony: PCWA-C300S, PCWA-C500, PCWA-C700

Xircom: CreditCard Netwave

Lucent Technologies: WaveLAN/IEEE 802.11b wireless network adapters and workalikes using the Lucent Hermes, Intersil PRISM-II, Intersil PRISM-2.5, Intersil Prism-3, und Symbol Spectrum24 Chips

NCR / AT&T / Lucent Technologies: WaveLan T1-speed ISA radio LAN cards

Sonstige Informationen

Die beste Unterstützung seitens FreeBSD für Wireless-LAN bietet im Moment der Atheros-Treiber. Es handelt sich dabei um die Chipsätze AR5210, AR5211 und AR5212.

Bitte lesen Sie auch die manual page des Atheros-Treibers „ath“, die im Internet über die Homepage www.FreeBSD.org zu erreichen ist, aufmerksam durch. Beachten Sie weiterhin, dass die Hersteller teilweise willkürlich die verwendeten Chipsätze ohne Vorankündigung ändern.

Wenn Sie beim Kauf einer WLAN-Karte sicher gehen wollen, dass ein Atheros-Chipsatz integriert ist, dann empfehlen wir einen Blick auf die Homepage von Atheros:

<http://customerproducts.atheros.com/customerproducts/>

Dort kann man durch Eingabe der Produktdaten prüfen lassen, ob ein Atheros-Chipsatz verwendet wird.

Weiterführende Links

FreeBSD-Homepage: www.FreeBSD.org

BSD-Forum (deutsch): www.BSDForen.de

Originalartikel (englisch):

www.pingwales.co.uk/tutorials/project-evil.html

Hinweis:

Es sei nochmals ausdrücklich darauf hingewiesen, das WLAN unbedingt ausreichend abzusichern. Ein Eindringling kann erheblichen Schaden technischer aber auch finanzieller Art anrichten. Im FreeBSD Handbook gibt es hierfür ein eigenes Kapitel mit dem Titel „Wireless Networks“.

FreeBSD

Wlan mit Project Evil



Project Evil: Windows-Treiber in FreeBSD

Verfasser: David Chisnall, July 2005, Übersetzer: Jürgen Dankoweit, Oktober 2005

Vorstellung „Project Evil“

Eines der Probleme, welche die freie Software-Gemeinde plagen, ist die Verfügbarkeit von Gerätetreibern. Solange ein Betriebssystem keine deutliche Marktdurchdringung hat, macht es keinen ökonomischen Sinn, Gerätetreiber dafür zu entwickeln.

Viele Hersteller stellen noch nicht einmal eine Dokumentation zur Verfügung, die es erlaubt Treiber zu schreiben, weil sie meinen, dass dies eine Aufdeckung geistigen Eigentums bedeuten würde.

Im Fall von WiFi-Karten kann dies zu einem Problem werden. Es ist sehr schwierig zu sagen, welcher Chipsatz zukünftig für eine Karte verwendet wird, manche Hersteller ändern die Hardware ohne die Produktbezeichnung zu modifizieren. So kann die Suche nach einer WiFi-Karte für das bevorzugte Betriebssystem schwierig werden.

Die FreeBSD-Gemeinde hat daher das „Project Evil“ ins Leben gerufen. Das ist eine teilweise Einbindung des Windows-APIs, welche es erlaubt, Windows-Treiber für Netzwerkkarten zu verwenden.

Wie funktioniert „Project Evil“?

„Project Evil“ stellt Basisfunktionen zur Verfügung, die von Windows Netzwerkkarten-Treibern verwendet werden. Diese Funktionen werden intern in das FreeBSD-Treibermodell umgesetzt. Für den Treiber sieht es so aus, als ob er in einer Windows-Umgebung laufen würde. Aus Sicht des Betriebssystems ist es ein natives FreeBSD-Kernelmodul.

Unter Windows besteht ein WiFi-Treiber aus drei Komponenten. Der Treiber selber, der meist die Erweiterung .sys hat. Es gibt auch eine .inf -Datei, die Informationen zum Treiber enthält. Schließlich gibt es noch eine Datei mit einer Kopie der Karten-Firmware.

Traditionell wird die Firmware einer Netzwerkkarte in einem ROM gebrannt. Später wurde erkannt, dass eine Möglichkeit einer Aktualisierung gegeben sein muss, und so wurde die Firmware in einem Flash-ROM gespeichert. In modernen Billigkarten wird kein Flash- ROM mehr verwendet und so wird die Firmware im RAM hinterlegt. Das bedeutet, dass der Treiber diese erst laden muss, bevor die Karte benutzt werden kann.

Um die Sache noch komplizierter zu machen, haben einige Treiber separate Firmware für den Ethernet-Controller und für den Wlan-Teil. Firmware-Dateien haben meist die Endung .bin.

Erstellen der Kernelmodule

Es wird eine Kopie des Windows-Treibers benötigt. Dieser befindet sich auf der Treiber-CD des Herstellers oder auf dessen Internet-Seite. Nun alle Dateien mit der Erweiterung .sys , .inf und .bin nach

```
/sys/modules/if_ndis
```

kopieren.

Für dieses Tutorial verwendet der Übersetzer eine Wifi-Karte *WIFI1234* , so dass man man die Treibernamen durch die eigenen ersetzen muss.

Diese Treiber werden mitgeliefert:

```
wifi1234.bin:      Firmware der Netzwerkkarte
wifi1234funk.bin: Firmware des Funkteils
WIFI1234.INF:    Treiberinformation
wifi1234.sys:    Der Treiber selber
```

Die Art und Weise, wie „Project Evil“-Kernelmodule erstellt werden, hat sich von FreeBSD 5.3 zu FreeBSD 5.4 geändert und leider enthält die Dokumentation zu FreeBSD 5.4 noch immer die Beschreibung der Vorgehensweise unter FreeBSD 5.3.

Dieses Tutorial beschreibt aber nur den Weg, der ab FreeBSD 5.4 gültig ist!

Wichtiger Hinweis:

Es ist sinnvoll nach -STABLE zu aktualisieren, da Project Evil ständig weiterentwickelt wird.

Vorgehensweise

Ab FreeBSD 5.4 werden für „Project Evil“ keine Kernel-Quelltexte mehr benötigt. Das ndis- und if_ndis -Modul sollten bereit installiert sein. Es muss nur noch ein Modul generiert werden, welches den Treiber und die Firmware enthält. Dies wird vom Assistent *ndisgen* gehandhabt.

```
# ndisgen
```

Das Programm fragt nach dem Verzeichnis, in dem Treiber und Firmware abgelegt sind. Bitte beachten, dass zwischen Groß- und Kleinschreibung unterschieden wird und es muss auch der volle Verzeichnisname angegeben werden. Als Ergebnis erhält man ein einzelnes Modul (.ko). In diesem Fall ist es *wifi1234_sys.ko*. Das Modul ist nur noch nach */boot/kernel* zu kopieren.

```
# cp wifi1234_sys.ko /boot/kernel/
```

```
# kldload ndis
```

```
# kldload if_ndis
```

```
# kldload wifi1234_sys
```

Auch hier gilt es wieder zu beachten, dass die Reihenfolge der *kldload*-Kommandos sehr wichtig ist! Bei Missachtung droht ein „kernel panic“.

Wie vorherigen Abschnitt beschrieben sollte man auch wieder Eintragungen in */boot/loader.conf* vornehmen:

```
ndis_load="YES"
```

```
if_ndis_load="YES"
```

```
wifi1234_sys_load="YES"
```

Nun kann das System neu gestartet werden. Wie bereits beschrieben, wird mit */stand/sysinstall* die Netzwerkkarte konfiguriert.

Bemerkung::

„Project Evil“ funktioniert sowohl auf Systemen mit 32-bit oder 64-bit Prozessoren von AMD oder Intel. Voraussetzung ist allerdings, dass die Treiber entweder 32-bit oder 64-bit sind.

Wichtige Informationen

Hinweis: Ab FreeBSD 6.0 ist nur noch der Assistent *ndisgen* zu verwenden.

Kernel kompilieren (i386!!!)

Natürlich kann man den ndis-Treiber auch in den Kernel einkompilieren:

```
# cd /usr/src/sys/i386/conf
```

```
# cp GENERIC NDISKERNEL
```

Um den FreeBSD-Kernel NDIS-fähig zu machen, müssen zwei Einträge gemacht bzw. auskommentiert werden:

```
options NDISAPI
```

```
device ndis
```

```
device wlan # nur, wenn WLAN verwendet wird!
```

```
Speichern der Konfiguration
```

```
# cd /usr/src
```

```
# make buildkernel KERNCONF=NDISKERNEL
```

```
# make installkernel KERNCONF=NDISKERNEL
```

NDISKERNEL ist nur ein Platzhalter für den Namen des eigenen Kernels. Beim Kompilieren und Installieren ist unbedingt auf Fehlermeldungen zu achten! Durch das Einkompilieren des Treibers in den Kernel erspart man sich in der Datei */boot/loader.conf* die Einträge *if_ndis_load="YES"*.

Vergessen Sie vor dem Erstellen eines neuen Kernels nicht, sich ein separates Verzeichnis anzulegen und den alten Kernel und seine Module dorthin zu sichern (z.B. */root/kernels/NDISKERNEL*).