Linux in heterogenen Netzwerken
Reminder zur Schulungsbegleitung

von M. Henne (martin.henne@web.de)

Stand: 14.10.2007

Dieses Dokument dient zur nachträglichen Gedächtnisstütze der Teilnehmer des Kurses „Linux in heterogenen Netzwerken“ der von der IHK Aachen in Zusammenarbeit mit dem „Aachen Programming Club“ durchgeführt wird. Es handelt sich nicht um eine eigenständige Anleitung zu dieser komplexen Thematik. Die Reihenfolge folgt den Veranstaltungen.

Inhaltsverzeichnis

1. Verbindung zu einem TCP/IP-Netzwerk herstellen....3
2. dnsmasq - Namensauflösung mit DNS für kleine Netze....5
3. DHCP-Server: Automatische Netzwerkkonfiguration 6
4. Webserver....7
5. Dateiserver....7
5.1 Proftpd - der FTP-Server....8
5.2 NFS - Das Network Filesystem....9
5.3 SSHFS - Das Secure-Shell Filesystem....10
6. WLAN-Treiber kompilieren....11
6.1 Grundlagen zum Umgang mit Kerneltreibern....11
6.2 Treiber für den Intel WLAN-Chipsatz (ipw2100/ipw2200)....12
6.3 Broadcom Karten (bcm43xx)....12
6.4 Ralink Chipsätze in USB-Sticks (rt73)....13
6.5 Madfiwi-Treiber für Atheros-Karten (ath_pci)....13
7. WLAN-Konfiguration....13
7.1 Konfiguration mit wireless-tools....13
7.2 WPA Supplicant....14
7.3 Ad-hoc Verbindungen....15
8. Routing - statische Netzwerkrouten....16
8.1 Konfiguration der normalen Rechner (nicht Gateway)....17
8.2 Konfiguration des Routers (Gateway)....17
9. Firewallkonfiguration mit „iptables“....18
9.1 Grundlegende iptables-Aufrufe und Beispiele....19
9.2 Beispiel einer Basis-Firewall....20
9.3 Portforwarding....22
9.4 Sichern und Wiederherstellen einer Firewall....22
10. SSH....23
11. SMB/NetBIOS-Server „samba“....25
11.1 Samba als Primary Domain Controller (PDC)....25
Hinzufügen weiterer Freigaben....28
12. VPN - Virtual Private Network....28
12.1 Vorbereitungen....29

12.2 Erstellung der SSL-Zertifikate.....29
12.3 Konfigurationsdateien erstellen.....30
12.4 Erstellung der virtuellen Netzwerkgeräte.....30

1. Verbindung zu einem TCP/IP-Netzwerk herstellen

Die IP-Konfiguration des Linuxrechners erfolgt bei Debian-basierten Systemen (Ubuntu, Debian, Knoppix, Kanotix, ...) über die Datei */etc/network/interfaces*. Diese wird von dem Befehlen ifup und ifdown (siehe dort) ausgewertet.

Zeilen, die mit einer Raute ('#') beginnen, werden ignoriert. Wenn das letzte Zeichen einer Zeile der Backslash ist ('\'), wird die Zeile in der darunterliegenden Zeile fortgesetzt.

Die Datei */etc/network/interfaces* besteht aus einem oder mehr Blöcken, die mit der Bezeichnung „iface“, „mapping“, „auto“ und „allow-“ beginnen. (Siehe Beispiel)

Beispiel 1: „/etc/network/interfaces“

```
auto lo eth0
allow-hotplug eth1

iface lo inet loopback

mapping eth0
      script /usr/local/sbin/map-scheme
      map HOME eth0-home
      map WORK eth0-work

iface eth0-home inet static
      address 192.168.1.1
      netmask 255.255.255.0
      up flush-mail

iface eth0-work inet dhcp

iface eth1 inet dhcp
```

Zeilen, die mit 'auto' beginnen, kennzeichnen die Schnittstellen, die beim Aufruf des Befehls 'ifconfig -a' aktiviert werden. Ein solcher Aufruf erfolgt auch beim Systemstart.

Mit 'allow-' beginnende Zeilen kennzeichnen Schnittstellen, die von verschiedenen Subsystemen aktiviert werden dürfen. So wird es möglich, Schnittstellen zu aktivieren, wenn ein Netzwerkkabel oder z.B. ein USB-Stick angeschlossen wird.

Um eine einfache statische Netzwerkkonfiguration mit einer Ethernetkarte (eth0) vorzugeben, ist folgender Block ausreichend:

Beispiel 2: „interfaces“ - statische Zuweisung der IP-Konfiguration

```
iface eth0 inet static
      address 192.168.0.42
      netmask 255.255.255.0
```

```
      gateway 192.168.0.1
```

Für die automatische Konfiguration durch einen im eigenen Netzwerk befindlichen DHCP-Server (siehe dort) genügt der folgende „Einzeiler“:

Beispiel 3: „interfaces“ - Zuweisung durch einen DHCP-Server

```
iface eth0 inet dhcp
```

Wenn mehrere, unterschiedliche Konfigurationen für die gleiche Netzwerk-Schnittstelle vorhanden sind, muss der Interface-Name (z.B. 'eth0') durch ein Alias ersetzt werden (z.B. 'home' oder 'work'). Ein „mapping“-Block ist dann für jede so konfigurierte Netzwerkkarte erforderlich. Im einfachsten Fall kennzeichnet der Block nur durch eine Zeile, dass ein bestimmtes Interface Aliasnamen verwendet:

Beispiel 4: „interfaces“ - eth0 verwendet Aliasnamen

```
mapping eth0
```

Die Aktivierung des Interfaces erfolgt dann z.B. durch Eingabe des Befehls 'ifup eth0=home', falls ein iface-Block mit dem Aliasnamen 'home' definiert wurde. Die Entscheidung, welcher Aliasname verwendet werden soll, kann auch durch ein Skript getroffen werden, das z.B. nach einem bestimmten Rechner im Netz oder nach einem Access-Point (WLAN) mit einer bestimmten Kennung (ESSID) sucht. Der folgende Eintrag übergibt einem Skript („erkennenetz.sh“) die IP-Adressen zu suchender Rechner. Wenn diese Rechner im Netz gefunden werden, gibt das Skript den entsprechenden Aliasnamen aus (siehe dort).

Beispiel 5: „interfaces“ - Aufruf eines mapping-Skriptes

```
mapping eth0
      script /usr/local/bin/erkennenetz.sh
      map 192.168.0.113 ihk
      map 192.168.0.1 zuhause
```

Die Hinter „map“ stehenden Informationen werden dem Skript als Standardeingabe übergeben. Die einzige Ausgabe des Skriptes ist der Konfigurationsname.

Weitere Informationen zum Aufbau der Datei „/etc/network/interfaces“ findet man in der Manpage („man interfaces“).

Hilfe beim Erstellen komplexer Skripte finden im „Advanced Bash Skripting Guide“, das nach Installation des Pakets „abs-guide“ im lokalen Verzeichnis „/usr/share/doc/abs-guide“ zur Verfügung steht.

Das entsprechende Skript sieht so aus:

Beispiel 6: „erkennenetz.sh“ - Skript erkennt das Netzwerk

```
#!/bin/sh

ETHUP=$(/sbin/ifconfig -s | grep eth0)

if [ ! "$ETHUP" ]; then
  /sbin/ifconfig eth0 up
fi

while read THEIP THECONFIG; do
ARPING=$(/usr/bin/arping -c 1 -D -I eth0 $THEIP | grep "reply from" | \
    cut -d '[' -f 2 | cut -c 1-17 )
  if [ $ARPING ]; then
    echo $THECONFIG
    exit 0
  fi
done

/sbin/ifconfig eth0 down
exit 1
```

Der Befehl 'arping' steht nach Installation des Paketes 'iputils-arping' zur Verfügung.

2. dnsmasq - Namensauflösung mit DNS für kleine Netze

Das Paket 'dnsmasq' macht aus einem Linuxrechner sofort einen einsatzfähigen DNS-Server. Dabei wird die lokale Datei „/etc/hosts“ gelesen und die dort aufgeführten Rechner-IP's dem anfragenden Client mitgeteilt. Jeder dort nicht aufgeführte Rechner wird bei dem Nameserver ermittelt, der in der Datei „/etc/resolv.conf“ des Servers aufgeführt ist.

Beispiel 7: „/etc/hosts“ - Einträge des DNS-Servers

```
127.0.0.1       localhost
192.168.0.1     loki.lan                    loki
192.168.0.2     midgard.lan                 midgard
192.168.0.4     rivendell.lan               rivendell
```

Beispiel 8: „/etc/resolv.conf“ - Einträge des DNS-Servers

```
# Beispiel: DNS-Rechner eines DSL-Providers
nameserver 213.191.92.82
```

Beispiel 9: „/etc/resolv.conf“ - Einträge der sonstigen Rechner im Netz

```
# IP des (dnsmasq-)DNS-Servers im lokalen Netz
nameserver 192.168.0.1
```

Das Bearbeiten der Konfigurationsdatei „/etc/dnsmasq.conf“ ist für die Standardfunktionalität des Servers nicht erforderlich. Allerdings kann dnsmasq auch als DHCP-Server fungieren, dann wäre Änderungen unumgänglich. Im Rahmen der IHK-Schulung wird jedoch für diesen Zweck das Paket „dhcp“ vorgestellt.

Weitere Informationen finden Sie in der Manpage („man dnsmasq“).

3. DHCP-Server: Automatische Netzwerkkonfiguration

Die Installation des Paketes „dhcp“ installiert einen vollwertigen DHCP-Server und nach wenigen Änderungen an der Datei „/etc/dhcpd.conf“ (siehe dort) kann der Server auf Anfrage DHCP-Adressen an Clients vergeben. Die folgende Konfigurationsdatei vergibt Adressen im Bereich von 192.168.0.50 bis 192.168.0.100, wobei darüberhinaus der Rechner, der sich mit seiner MAC-Adresse authentifiziert, immer die Adresse 192.168.0.5 bekommt.

Beispiel 10: „/etc/dhcpd.conf“

```
default-lease-time 6000;
max-lease-time 12000;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
option routers 192.168.0.1;
option domain-name-servers 192.168.0.1;
option domain-name "lan";

subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.50 192.168.0.100;
}

host midgard {
      hardware ethernet 00:52:CB:0E:7C:B0;
      fixed-address 192.168.0.5;
      option host-name "midgard";
}
```

Nach Änderung der Konfigurationsdatei muss der Serverdienst mit dem Befehl „sudo /etc/init.d/dhcp restart“ neu gestartet werden.

Anfragende Clients bekommen hier auch automatisch die Information, dass der Nameserver im Netz die IP 192.168.0.1 besitzt. Diese Adresse wird automatisch in die Datei „/etc/resolv.conf“ eingetragen.

Weitere Informationen zu den vielen Konfigurationsmöglichkeiten des Paketes „dhcp“ finden Sie in der Manpage („man dhcpd.conf“).

4. Webserver

Als Webserver werden meist HTTP-Server bezeichnet, die in einem Netzwerk primär HTML-basierte Dokumente zur Verfügung stellen. Die Installation des Paketes „apache2“ installiert den Webserver und startet ihn in einer Standardkonfiguration auf Port 80.

Ist eine SSL-Verschlüsselung gewünscht, so führt die Installation des Pakets „apache-ssl“ auch hier zum gleichen Ergebnis. Der Server lauscht in diesem Fall auf Port 443.

Die zentrale Konfigurationsdatei des Webserves Apache ist für Apache2 „/etc/apache2/apache2.conf“ bzw. für Apache-ssl „/etc/apache-ssl/httpd.conf“.

In der Grundeinstellung senden beide Server die HTML-Dokumente, die im Ordner „/var/www“ liegen. Das ist nicht sinnvoll, denn verschlüsselte Dokumente sollen nicht unverschlüsselt auf Port 80 gesendet werden können.

Um die zu verschlüsselnden Dokumente im Ordner „/var/www-ssl“ abzulegen, sind folgende Schritte erforderlich:

1. Anlegen des Ordners

```
$> sudo mkdir /var/www-ssl
```

2. Änderung der Datei „/etc/apache-ssl/httpd.conf“

Beispiel 10.5: Änderung an der apache-ssl Konfiguration

```
#DocumentRoot /var/www
DocumentRoot /var/www-ssl
```

3. Neustart des Servers

```
$> sudo /etc/init.d/apache-ssl restart
```

Weitere Informationen sind nach Installation der Pakete „apache-doc“ bzw. „apache2-doc“ verfügbar.

5. Dateiserver

Die Aufgabe von Dateiservern (Fileserver) besteht primär im zur Verfügungstellen von Netzlaufwerken oder -Verzeichnissen um Dateien zu laden oder zu speichern.

Die Verbindungsart, die Authentifizierung und die Möglichkeiten der Rollen- und Rechteverteilung auf die Benutzer sind Unterscheidungsmerkmale für die verschiedenen Programme.

5.1 Proftpd - der FTP-Server

Im Verlaufe der Installation des Paketes „proftpd“ stellt ein Skript die Frage, ob der Server permanent im Hintergrund laufen soll oder über den sogenannten „inetd“ gestartet werden soll. Das bedeutet, dass proftpd automatisch gestartet wird, wenn eine Anfrage an den FTP-Server kommt (Port 21). Wird die Verbindung abgebaut, so beendet sich proftpd wieder.

Wenn sich ein FTP-Client beim Server authentifiziert, werden sowohl der Loginname als auch das Passwort im Klartext übertragen. Da bei proftpd in der Standardeinstellung solche User einloggen dürfen, die auf dem System bereits als normale Benutzer eingerichtet sind, stellt dies ein Sicherheitsproblem dar.

Mit Hilfe des Programms „ftpasswd“ und kleinen Anpassungen in der Konfigurationsdatei „/etc/proftpd/proftpd.conf“ können FTP-Accounts unabhängig von Rechneraccounts verwaltet werden.

5.1.1 Spezielle FTP-Accounts anlegen

Der folgende Aufruf des Programms „ftpasswd“ für dazu, dass der FTP-Benutzer „hans“ angelegt wird. Das von „hans“ benutzte Home-Verzeichnis (in dem er sich befindet, wenn er mit FTP eingeloggt hat) soll „/home/ftp/hans“ sein. Als Benutzernummer (uid) wird „1001“ verwendet.

Beispiel 11: FTP-User anlegen mit „ftpasswd“

```
$> ftpasswd --passwd --name hans --home /home/ftp/hans \
     --shell /bin/false --uid 1001
```

Die Datei „ftp.passwd“ befindet sich anschliessend im aktuellen Verzeichnis. Nun wird noch eine Datei benötigt, die die Zuordnung zu einer (virtuellen) Gruppe regelt.

Beispiel 12: FTP-User einer Gruppe zuordnen

```
$> ftpasswd --group -gid 1001 --name hans
```

Nun wurde die Datei „ftp.group“ im aktuellen Verzeichnis angelegt. Für die weitere Konfiguration spielt es technisch gesehen keine Rolle, wo sich diese Dateien befinden. Wir gehen für den Augenblick davon aus, dass sie sich im Verzeichnis „/etc/proftpd/“ befinden. In diesem Verzeichnis muss nun die Datei „proftpd.conf“ folgende Zeilen (zusätzlich zu den vorhandenen Einstellungen) erhalten:

Beispiel 13: Ergänzungen an „/etc/proftpd/proftpd.conf“

```
AuthUserFile /etc/proftpd/ftpd.passwd
AuthGroupFile /etc/proftpd/ftpd.group
RequireValidShell off
AuthOrder mod_auth_file.c
```

Die ersten beiden Zeilen verweisen auf die eben erstellten Dateien. Die Zeile „RequireValidShell off“ ermöglicht die Verwendung von „/bin/false“ als Shell, was eine zusätzliche Sicherheit bietet.

Die „AuthOrder“-Zeile verhindert schließlich das Einloggen durch normale, nicht extra angelegte Benutzer.

Nach einem Neustart von proftpd durch die Eingabe von „/etc/init.d/proftpd restart“ wird diese Konfiguration nun aktiv.

Weitere Informationen stehen nach Installation des Paketes „proftpd-doc“ im Verzeichnis „/usr/share/doc/proftpd-doc/“ zur Verfügung.

5.2 NFS - Das Network Filesystem

Mit dem NFS ist es möglich, Verzeichnisse auf einem Server für NFS-Clients freizugeben. Im folgenden ist ein schneller Weg zur Umsetzung solcher Verbindungen beschrieben.

Auf dem Server ist das Paket „nfs-user-server“ (alternativ: nfs-kernel-server) zu installieren. Das Paket „nfs-common“ wird automatisch mitinstalliert. Diese Paket ist auf den Clients zur Verbindungsaufnahme ausreichend.

5.2.1 Konfiguration des Servers

Die maßgebliche Konfigurationsdatei ist „/etc/exports“. Der folgende Eintrag gibt das Verzeichnis „/mnt/shares“ für den Rechner 192.168.0.12 frei.

Beispiel 14: Freigabe auf dem Server in /etc/exports

```
/mnt/shares 192.168.0.12(rw)
```

Die Option „rw“ erlaubt neben rein lesenden Zugriffen („ro“) auch schreibende Zugriffe auf das freigegebene Verzeichnis. Der NFS-Server muss nach Änderungen der „/etc/exports“ neu gestartet werden. Das Kommando dafür lautet „/etc/init.d/nfs-user-server restart“ (in der Variante mit nfs-kernel-server analog).

Weitere Informationen sind in der manpage zur exports-Datei zu finden (man exports).

5.2.2 Konfiguration des Clients

Auf einem NFS-Client kann nach Installation von „nfs-common“ der normale mount-Befehl zum Einbinden des Netzlaufwerkes genutzt werden. Für das o.a. Beispiel kann der Befehl z.B. wie folgt lauten:

Beispiel 15: Mounten auf der Kommandozeile

```
$> sudo mount -t nfs server.lan:/mnt/shared /mnt/tmp -o rw
```

Ein Eintrag kann natürlich auch in der Datei „/etc/fstab“ angefügt werden:

Beispiel 15: Mount-Eintrag in der Filesystem-Table

```
server.lan:/mnt/shares /mnt/netzlaufwerk nfs rw,noauto,user 0 0
```

Nun kann auf dem Client der Befehl „mount /mnt/netzlaufwerk“ verwendet werden, um das Netzlaufwerk einzubinden. Die Einstellungen sind der Manpage von fstab und nfs zu entnehmen („man fstab“ bzw. „man nfs“).

5.2.3 Berechtigungen auf dem Netzlaufwerk

Der Server sendet bei Zugriffen auf das Netzlaufwerk die Benutzer-ID's verbatim an den Client. Das bedeutet, die User-ID's müssen auf Client und Server identisch sein (siehe „ls -ln“ und „cat /etc/passwd“). Der root-Zugriff bildet eine Ausnahme. Nähere Informationen sind der Manpage zur Datei exports entnehmbar (siehe „no-root-squash“).

5.3 SSHFS - Das Secure-Shell Filesystem

Das SSHFS bietet vollständig verschlüsselten Zugriff bei minimalem Konfigurationsaufwand.

5.3.1 Konfiguration des Servers

Auf dem Server ist lediglich das Paket „ssh“ bzw. „openssh-server“ erforderlich. Weitere Konfigurationen müssen nicht vorgenommen werden.

5.3.2 Konfiguration der Clients

Auf den Clients ist das Kernelmodule „fuse“ einzubinden („modprobe fuse“). Ein entsprechender Eintrag in der Datei „/etc/modules“ automatisiert dies nach dem nächsten Neustart.

Schließlich müssen alle Benutzer, die SSHFS nutzen dürfen, der Gruppe „fuse“ hinzugefügt werden.

```
$> sudo addgroup username fuse
```

Nachdem der User neu eingeloggt hat (vorher ist er nicht Mitglied der Gruppe fuse), kann er ein Netzlaufwerk mit einem einfachen Befehl im Verzeichnis „localdirectory“ einbinden.

Beispiel 16: mounten durch den User

```
$> sshfs john@server.lan:/mnt/shares localdirectory
```

Das Verzeichnis „localdirectory“ muss dem Benutzer, der das Laufwerk mit dem o.a. Befehl einbindet, gehören. Es sollte leer sein. Die Option „-o reconnect“ kann verwendet werden, damit eine evtl. unterbrochene Verbindung automatisch wieder hergestellt wird.

Zum Unmounten wird folgender Befehl benutzt:

Beispiel 17: umount des SSHFS-Verzeichnisses

```
$> fusermount -u localdirectory
```

6. WLAN-Treiber kompilieren

Um eine WLAN-Karte anzusprechen, werden i.d.R. ein oder mehrere Kernelmodule (Treiber) sowie für manche Modelle eine Firmwaredatei (manchmal auch „Microcode“ genannt) benötigt. Die Konfiguration wird dann im nachfolgenden Kapitel erklärt.

6.1 Grundlagen zum Umgang mit Kerneltreibern

Kerneltreiber sind entweder fest im Kernel eingebunden, oder in Form sogenannter Module bereitgestellt. Diese Kernelmodule werden dann bei Bedarf ins System nachgeladen.

Kernelmodule sind Dateien mit Endung „.ko“ und sie befinden sich, verteilt auf verschiedene Unterverzeichnisse, im Ordner „/lib/modules/<kernelversion>“. Die Kernelversion kann man mit dem Befehl „uname -r“ herausfinden. Bei der Installation eines WLAN-Treibers geht es darum, ein solches Modul aus dem Quellcode zu kompilieren, falls es nicht schon vorhanden ist, und die Firmware an die richtige Stelle zu kopieren.

Um Module kompilieren zu können, muss das System vorbereitet werden. Alle benötigten Dateien und Programme werden automatisch installiert, wenn der folgende Befehl eingegeben wird.

```
$> module-assistant prepare
```

Die weiteren Schritte bestehen im allgemeinen darin, den Quellcode für den Treiber aus dem Internet zu laden, ihn auszupacken und dann mit dem Befehl „make“ das Kernelmodul zu kompilieren. Meist führt die Eingabe von „make install“ anschliessend dazu, dass der Treiber installiert wird.

Ist das nicht der Fall, muss die *.ko-Datei in das Verzeichnis „/lib/modules/$(uname -r)“ kopiert werden. Der Befehl „depmod -a“ schließlich veranlaßt den Kernel dazu, die Liste der verfügbaren Treiber zu aktualisieren.

Sollte eine Firmware benötigt werden, so wird diese einfach in das Verzeichnis „/lib/firmware“ kopiert.

Von dieser allgemeinen Vorgehensweise weichen die Treiberautoren gerne geringfügig ab, daher sind ein paar Hinweise zu bestimmten Chipsätzen ebenso unumgänglich, wie das lesen der mit dem Treiber mitgelieferten Dokumentation (meist existiert eine Datei namens README oder INSTALL).

Nach erfolgreicher Installation der Treiber kann das Netzwerkgerät (dessen Namen variieren, s.u.) z.B. mit dem ifconfig-Befehl aktiviert werden. Am Beispiel von Atheros-Treibern (madwifi, Netzwerkgerät heißt 'ath0') z.B. so:

```
$> sudo ifconfig ath0 up
```

6.2 Treiber für den Intel WLAN-Chipsatz (ipw2100/ipw2200)

Auf den Websiten http://ipw2200.sf.net und http://ipw2100.sf.net werden die Treiber für diese beiden Karten zum Download angeboten. Darüberhinaus muss noch die Firmware geladen werden. Vorher wird das Kernelmodulpaket ieee80211 benötigt, welches auf der Website http://ieee80211.sf.net zur Verfügung steht.

Die folgenden Schritte sind chronologisch vorzunehmen:

1. Löschen der alten Treibermodule (siehe mitgelieferte Dokumentation)
2. Kompilieren und Installieren des ieee80211 Subsystems.
3. Kompilieren und Installieren des Treibermoduls (z.B. ipw2100.ko)
4. Kopieren der Firmware nach „/lib/firmware“
5. Eingabe von „depmod -a“

Das Netzwerkgerät heißt eth<X>.

6.3 Broadcom Karten (bcm43xx)

Der Treiber bcm43xx ist bereits im Kernel enthalten. Es ist lediglich erforderlich, die Firmware ins Verzeichnis „/lib/firmware“ zu kopieren. Das Programm bcm43xx-fwcutter kann verwendet werden, um die Firmware z.B. vom Windowstreiber zu extrahieren oder aus dem Internet zu laden.

Bei bestehender Internetverbindung wird das bei der Installation von bcm43xx-fwcutter sofort automatisch erledigt.

Weitere Informationen siehe „man bcm43xx-fwcutter“.

6.4 Ralink Chipsätze in USB-Sticks (rt73)

Für Ralink-Chipsätze gibt es Herstellertreiber und ein freies Treiberprojekt auf http://rt2x00.serialmonkey.com . Nach dem Download und Auspacken der passenden Treiber (z.B. rt73-cvs-daily.tar.gz) wird das Modul rt73.ko durch Eingabe von 'make' im ausgepackten Verzeichnis unter „Module/“ einfach kompiliert. Das Netzwerkgerät heißt „wlan0“. Beim Herstellertreiber heißt es „rausb0“.

6.5 Madfiwi-Treiber für Atheros-Karten (ath_pci)

Die madwifi-Treiber können mit Hilfe des Programms „module-assistant“ geladen, kompiliert und installiert werden. Nach dem Start vom module-assistant kann man die entsprechende Auswahl menügesteuert vornehmen. Das Netzwerkgerät heißt „ath0“

7. WLAN-Konfiguration

Wenn die WLAN-Treiber erfolgreich installiert wurden, muss die Konfiguration erfolgen. Dabei muss unterschieden werden, ob eine unverschlüsselte Verbindung, oder eine WEP, WPA oder WPA2-Verschlüsselung gewünscht ist. Es soll hier nur jeweils eine unverschlüsselte und eine mit WPA verschlüsselte Konfiguration aufgezeigt werden.

Im folgenden soll davon ausgegangen werden, dass der Accesspoint (AP) die ESSID „welahn“ verwendet und das Netzwerkgerät „ath0“ heißt.

7.1 Konfiguration mit wireless-tools

Das Paket „wireless-tools“ stellt die Konsolenbefehle „iwconfig“, „iwpriv“ und „iwlist“ zur Verfügung. Diese ermöglichen den Aufbau einer Verbindung zum AP mit einfachen Befehlen.

Eine unverschlüsselte Verbindung kann wie folgt aufgebaut werden:

```
$> ifconfig ath0 up

$> iwconfig ath0 mode managed

$> iwconfig ath0 essid welahn

$> dhclient ath0
```

Für WPA-Verschüsselung, ist z.B. folgende Eingabe erforderlich:

```
$> ifconfig ath0 up

$> iwconfig ath0 mode managed

$> iwconfig ath0 essid welahn

$> iwpriv ath0 set AuthMode=WPAPSK

$> iwpriv ath0 set WPAPSK=<KEY>

$> iwpriv ath0 set EncrypType=TKIP

$> dhclient ath0
```

Der Key muss nicht, sollte aber 63 Zeichen lang sein und in keinem Wörtebuch stehen. Einen guten Key erzeugt man am besten mit dem folgenden Befehl:

```
$> wpa_passphrase <essid> <passwort>
```

Hinweis: Auch das Programm 'pwgen' eignet sich zum erzeugen sicherer Passwörter. Da man mit wpa_passphrase jedoch den Umweg über ein leicht zu merkendes einfache Passwort und die essid gehen kann, ist dieser Weg hier geeigneter.

Weitere Informationen und Beispielkonfigurationen sowie Hinweise zur Einbindung der Datei „/etc/network/interfaces“ sind im Ordner „/usr/share/doc/wireless-tools“ zu finden.

7.2 WPA Supplicant

Das Paket „wpasupplicant“ stellt ein Programm bereit („/sbin/wpa_supplicant“) das nach dem Start die Verschlüsselung und Authentifizierung handhabt. Es wird nicht von allen Treibern unterstützt, so dass ggf. auf die wireless-tools zurückgegriffen werden muss.

Nach der Installation muss im Verzeichnis „/etc/wpa_supplicant/“ eine Konfigurationsdatei erstellt werden. Der Name dieser Datei spielt zunächst keine Rolle. Wir wollen sie „wlan.config“ nennen.

Für eine unverschlüsselte Verbindung wäre z.B. folgender Inhalt sinnvoll:

Beispiel 18: /etc/wpa_supplicant/wlan.config

```
ctrl_interface=/var/run/wpa_supplicant

network={
  ssid="welahn"
  key_mgmt=NONE
}
```

Eine WPA-Verschlüsselung kann mit folgender Konfiguration hergestellt werden:

Beispiel 19: /etc/wpa_supplicant/wlan-wpa.config

```
# WPA-PSK/TKIP

ctrl_interface=/var/run/wpa_supplicant

network={
        ssid="welahn"
        key_mgmt=WPA-PSK
        proto=WPA
        pairwise=TKIP
        group=TKIP
        psk="supergeheimespasswortdaskeinerweiss"
}
```

Der WPA-Supplicant muss gestartet werden, damit die Netzwerkkarte arbeiten kann:

```
$> wpa_supplicant -D ath0 -c /etc/wpa_supplicant/wlan.config -B -d
```

Dieser Befehl ist am Besten neben einer 'pre-up'-Anweisung in der Datei /etc/network/interfaces zu platzieren.

Weitere Informationen und Konfigurationsbeispiele befinden sich im Verzeichnis „/usr/share/doc/wpasupplicant“.

7.3 Ad-hoc Verbindungen

Eine Ad-hoc Verbindung, also eine Direktverbindung ohne Accesspoint, wird am besten unter Verwendung der Wireless-Tools hergestellt:

```
$> ifconfig ath0 up

$> iwconfig ath0 mode ad-hoc

$> iwconfig ath0 essid welahn

$> ifconfig ath0 192.168.20.15 netmask 255.255.255.0 up
```

Zu beachten ist, dass die am Ad-hoc-Netz teilnehmenden Rechner die gleiche ESSID verwenden müssen. Natürlich muss sich die IP-Adresse unterscheiden, jedoch im selben Subnetz liegen.

8. Routing - statische Netzwerkrouten

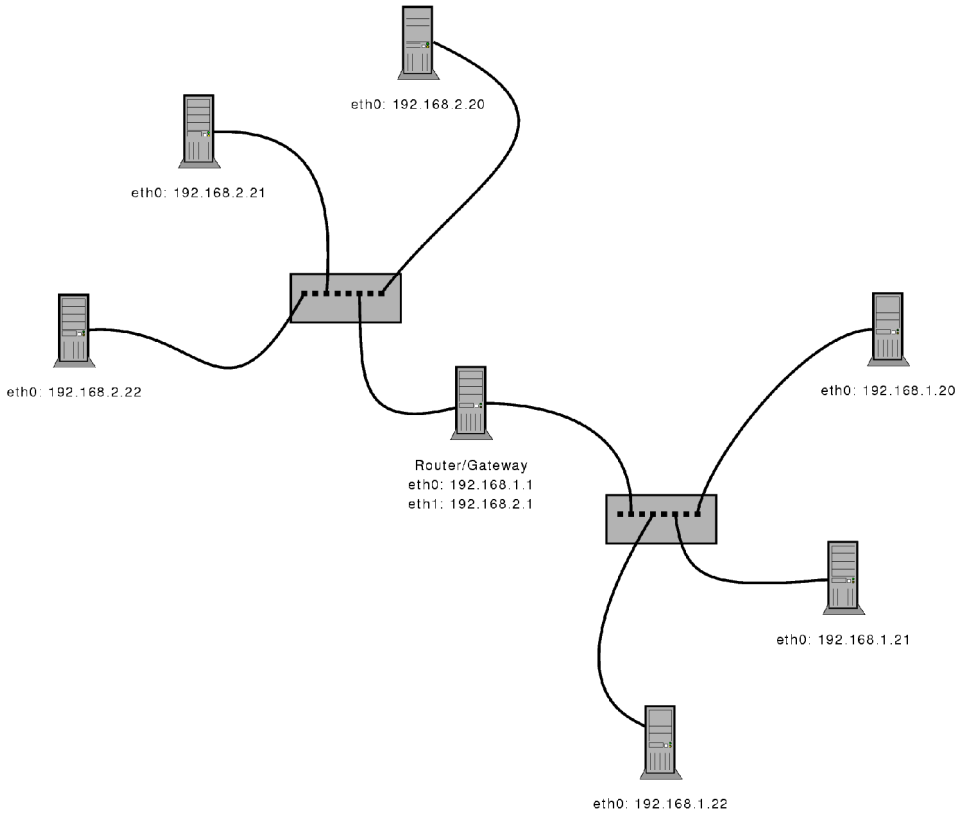
Grundsätzlich ist jeder Rechner nur in der Lage, andere Rechner aus dem gleichen Subnetz zu kontaktieren. Das Subnetz wird festgelegt durch die sogenannte Netzmaske. IP-Adresse und Netzmaske sind jeweils vier Bytewerte (IP-Version 4), setzen sich also aus insgesamt 32 Bit zusammen.

Beispiel:

IP-Adresse, dezimal:	192.168.0.10
IP-Adresse, hexadezimal:	C0.A8.00.0A
IP-Adresse, binär:	1100000.10101000.00000000.00001010

Alle Bits der Netzmaske müssen bei allen Rechnern im gleichen Subnetz gleich gesetzt sein. Die Standardnetzmaske ist die 255.255.255.0, also die Bitmaske „11111111.11111111.11111111.00000000“. Damit müssen die ersten drei Byte einer IP-Adresse in diesem Fall im Subnetz identisch sein, während das letzte Byte abweichen kann. Das ermöglicht 256 Verschiedene IP-Adressen (0-255), wobei '0' und '255' nicht verwendet werden.

Um Rechner aus einem anderen Netzwerk zu erreichen, muss ein Rechner Zugang zu beiden Netzen haben. Dieser Rechner wird als sog. „Gateway“ (Übergang) für die Übersetzung der Netzwerkadressen (NAT) konfiguriert. Er hat normalerweise zwei Netzwerkkarten, von denen jeweils eine eine IP-Adresse des jeweiligen Subnetzes hat.



8.1 Konfiguration der normalen Rechner (nicht Gateway)

Zunächst wird der Rechner normal im eigenen Subnetz mit einer IP-Adresse angegliedert. Das kann in der Datei „/etc/network/interfaces“ geschehen (siehe Kapitel 1) oder durch Eingabe des folgenden Kommandos am Beispiel des Rechners 192.168.1.21:

Beispiel 20: Netzwerkkarte konfigurieren mit ifconfig

```
$> ifconfig eth0 192.168.1.21 up
```

Schließlich wird noch eine Route, als eine Art „Weg“ in das andere Subnetz definiert. Das kann anhand einer sogenannten Default-Route erfolgen, also der Weg in jedes Netz für das keine eigene Route existiert, oder in Form einer einzelnen Route in das jeweilige Netz. In Beispiel 21 wird eine Route in das Subnetz 192.168.2.0 über den Rechner 192.168.1.1 definiert. Beispiel 22 deklariert diesen Rechner als Standardgateway. Eine der beiden Varianten reicht für den Weg ins andere Netz aus.

Beispiel 21: Definition einer Route ins Subnetz 192.168.2.0

```
$> route add –net 192.168.2.0 netmask 255.255.255.0 gw 192.168.1.1
```

Beispiel 22: Alternativ mit Standardgateway

```
$> route add default gw 192.168.1.1
```

Mit dem Befehl „route -n“ kann man die auf dem Rechner festgelegten Routen kontrollieren.

Beispiel 23: Routing-Tabelle des Linux-Kernels

```
Kernel IP Routentabelle
Ziel         Router      Genmask        Flags  Metric  Ref  Use    Iface
192.168.1.0  0.0.0.0     255.255.255.0  U      0       0    0      eth0
0.0.0.0      192.168.1.1 0.0.0.0        UG     0       0    0      eth0
```

8.2 Konfiguration des Routers (Gateway)

Wenn jeder Rechner innerhalb der beteiligten Netze eine eindeutige IP besitzt, dann genügt auf dem Gateway die folgende Eingabe, um das Routing bzw. das sogenannte „Forwarding“ zu aktivieren:

Beispiel 24: Aktivieren des Forwardings

```
$> echo 1 > /proc/sys/net/ipv4/ip_forward
```

Wenn die im eigenen Subnetz vorhandenen IP-Adressen darüber hinaus keine Verwendung finden sollen, dann kann durch das sogenannte „Masquerading“ die IP des Routers vorgegaukelt werden. Dafür sind zwei weitere Eingaben erforderlich:

Beispiel 25: Aktivieren des Masqueradings

```
$> modprobe ipt_MASQUERADE
$> iptables -A POSTROUTING -t nat -j MASQUERADE
```

Um einen Router automatisch als Gateway von einem privaten in ein öffentliches Netz zu konfigurieren, steht unter Debian das Paket 'ipmasq' zur Verfügung, das anhand eigener Regeln entsprechende Einstellungen der Firewall vornimmt.

Bei der Arbeit an einer eigenen Firewall ist in diesem Zusammenhang darauf zu achten, dass die eigenen Regeln sinnvoll in die ggf. von 'ipmasq' vorgegebene Regelstruktur eingegliedert wird.

Darüberhinaus muss der Router natürlich beide Netzwerkkarten korrekt konfiguriert haben (siehe Kapitel 1).

9. Firewallkonfiguration mit „iptables“

Bei dem Programm „iptables“ handelt es sich um ein Programm zum Anlegen von Filterregeln für das Akzeptieren oder Verwerfen von Datenpaketen, die über ein Netzwerk den Rechner erreichen.

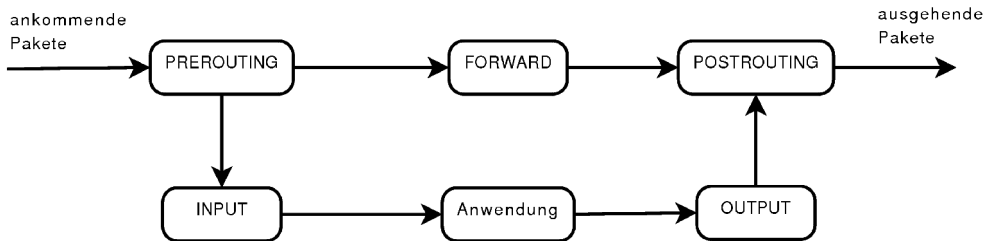
Die grundsätzliche Vorgehensweise ist die, dass Pakete, die am Rechner ankommen zunächst nach Herkunft und Ziel auf drei Regelketten (chains) aufgeteilt werden.

Pakete, die von aussen kommen, und für den Rechner bestimmt sind, werden von den Regeln der INPUT-Kette verarbeitet.

Pakete, die von einer Anwendung auf dem Rechner erzeugt und und verschickt werden, folgen den Regeln in der OUTPUT-Kette.

Pakete, die für andere Rechner bestimmt sind (z.B. beim Masquerading bzw. Forwarding) werden in der FORWARD-Kette behandelt. Wenn das Forwarding verwendet wird, stehen für die Vorverarbeitung noch die PREROUTING-Kette und die POSTROUTING-Kette zur Verfügung.

Die folgende Grafik verdeutlicht die möglichen Wege eines Datenpaketes.



Um Regeln zu einer Regelkette hinzuzufügen, die Regeln aufzulisten, zu löschen oder zu verwalten, wird der Befehl 'iptables' verwendet. 'iptables' benötigt für die meisten Operationen als Parameter die betroffene Kette sowie spezielle Informationen über die betreffende Regel.

9.1 Grundlegende iptables-Aufrufe und Beispiele

1. Auflisten der aktuellen Regeln der INPUT, OUTPUT und FORWARD-Kette

```
$> iptables -L -n
```

2. Löschen der INPUT, OUTPUT und FORWARD-Kette

```
$> iptables -F
```

3. Setzen der Grundeinstellung für eine Kette (hier auf ACCEPT)

```
$> iptables -P INPUT ACCEPT
```

4. Sperren von eingehenden Verbindungen auf Port 80 (http)

```
$> iptables -A INPUT -p tcp --dport 80 -j DROP
```

5. Löschen der Sperre auf Port 80

```
$> iptables -D INPUT -p tcp --dport 80 -j DROP
```

6. Eingehenden Datenverkehr aus dem Subnetz 192.168.2.0 sperren

```
$> iptables -A INPUT -s 192.168.2.0/24 -j DROP
```

7. Ausgehenden Datenverkehr auf Port 27015 blockieren

```
$> iptables -A OUTPUT -p tcp --dport 27015 -j DROP
```

8. Neue Regelkette anlegen

```
$> iptables -N NEUEKETTE
```

9. Alle Pakete aus dem Subnetz '10.0.0.0' auf die neue Kette umleiten

```
$> iptables -A INPUT -s 10.0.0.0/8 -j NEUEKETTE
```

10. Alle Pakete in NEUEKETTE loggen und dann erlauben

```
$> iptables -A NEUEKETTE -j LOG
$> iptables -A NEUEKETTE -j ACCEPT
```

11. Eine eigene Kette löschen

```
$> iptables -X NEUEKETTE
```

12. Pakete an den Rechner 192.168.0.42 nicht weiterleiten

```
$> iptables -A FORWARD -d 192.168.0.42 -j DROP
```

9.2 Beispiel einer Basis-Firewall

Beispiel 26: Einfaches Firewall-Skript

```
#!/bin/sh

IPT=/sbin/iptables

$IPT -F
$IPT -P OUTPUT DROP
$IPT -P INPUT DROP
$IPT -P FORWARD DROP
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A INPUT -p icmp -j ACCEPT
$IPT -A INPUT -p tcp --dport 22 -j ACCEPT
$IPT -A OUTPUT -j ACCEPT
$IPT -L -n
```

Die Konfiguration in Beispiel 26 läßt eingehende Verbindungen auf Port 22 (ssh) sowie icmp-Pakete (ping) zu. Darüberhinaus ist jegliche Kommunikation mit dem eigenen Rechner über das Loopback-Device gestattet. Pakete, die sich auf eine bestehende Verbindung beziehen, werden akzeptiert. Ausgehende Pakete passieren immer.

Diese Konfiguration ist als Basis für angepasste Firewalls geeignet, wenn die Regeln für die OUTPUT-Kette etwas differenzierter ausfallen.

Ein etwas komplexeres, aber immer noch überschaubares Beispiel, das für einen Gateway Verwendung findet, der ein privates Netz schützt, lautet wie folgt:

Beispiel 27: DSL-Router-Firewall

```
#!/bin/sh

EXTIF=ppp+
INTIF=eth1
FW=/sbin/iptables

$FW -F
$FW -F PREROUTING -t nat
$FW -F POSTROUTING -t nat
$FW -P INPUT DROP
$FW -P OUTPUT DROP
$FW -P FORWARD ACCEPT
$FW -P PREROUTING ACCEPT -t nat
$FW -P POSTROUTING ACCEPT -t nat

# Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
# Masquerading
$FW -A POSTROUTING -t nat -j MASQUERADE
# Antispoofing
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 1 > $f; done

# INPUT
$FW -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$FW -A INPUT -p icmp -j ACCEPT # ping usw. zulassen
$FW -A INPUT -i $EXTIF -p tcp --dport 22 -j LOG # ssh von aussen: loggen
$FW -A INPUT -p tcp --dport 22 -j ACCEPT # ssh zulassen
$FW -A INPUT -i $INTIF -j ACCEPT # internes Interface: akzeptieren
$FW -A INPUT -s 192.168.0.0/16 -j ACCEPT # internes Netz: akzeptieren

# OUTPUT
$FW -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$FW -A OUTPUT -p icmp -j ACCEPT # ping zulassen
$FW -A OUTPUT -p tcp --dport 22 -j ACCEPT # ssh
$FW -A OUTPUT -p tcp --dport 53 -j ACCEPT # dns
$FW -A OUTPUT -p udp --dport 53 -j ACCEPT # dns
$FW -A OUTPUT -p tcp -d checkip.dyndns.org --dport 80 -j ACCEPT
$FW -A OUTPUT -p udp -d checkip.dyndns.org --dport 80 -j ACCEPT
$FW -A OUTPUT -p udp --dport 123 -j ACCEPT # ntpdate
$FW -A OUTPUT -d security.debian.org -j ACCEPT
$FW -A OUTPUT -d non-us.debian.org -j ACCEPT
$FW -A OUTPUT -d ftp.de.debian.org -j ACCEPT
$FW -A OUTPUT -j LOG --log-prefix "IPTABLES OUTPUT: "
```

9.3 Portforwarding

Mit iptables besteht die Möglichkeit, den ankommenden Datenverkehr einzelner Ports auf andere Rechner und auch auf andere Ports umzuleiten. Dies ermöglicht u.a. im Betrieb mehrere Dienste auf unterschiedliche Rechner mit einer nach aussen hin sichtbaren IP-Adresse zu verteilen.

Im folgenden Beispiel wird eine auf dem Server auf Port 80 ankommende TCP-Verbindung auf den Rechner 192.168.0.2 ebenfalls auf Port 80 weitergeleitet.

```
$> iptables -A PREROUTING -t nat -p tcp --dport 80 \
      -j DNAT --to-destination 192.168.0.2:80
```

9.4 Sichern und Wiederherstellen einer Firewall

Der Befehl 'iptables-save' gibt die aktuelle Konfiguration der Firewall auf der Standardausgabe (normalerweise der Bildschirm) aus. Diese Ausgabe kann mit dem '>' - Zeichen oder dem Befehl '| tee' in eine Datei umgeleitet werden.

```
$> iptables-save | tee meine-firewall.txt
```

oder

```
$> iptables-save > meine-firewall.txt
```

Die so entstandene Datei kann vom Befehl 'iptables-restore' wieder verarbeitet werden. iptables-restore erwartet die Eingabe über die Standardeingabe, die ebenfalls umgelenkt werden kann, so dass die Eingabe aus einer Datei erfolgt.

```
$> cat meine-firewall.txt | iptables-restore
```

oder

```
$> iptables-restore < meine-firewall.txt
```

Eine ausführliche Beschreibung der Funktionen und Möglichkeiten von IP-Tables ist in der Manpage (man iptables) sowie in der Dokumentation im Verzeichnis „/usr/share/doc/iptables/html“ verfügbar.

10. SSH

SSH (Secure Shell) stellt den Königswerk der verschlüsselten Kommunikation dar. SSH kann verwendet werden, um auf entfernten Rechnern zu arbeiten, Dateien zu kopieren und sonst unverschlüsselte Daten durch einen SSH-Tunnel zu schützen. Im folgenden werden die gebräuchlichsten Anwendungen dargestellt.

Installation von Server und Client

Um einen SSH-Server und einen SSH-Client zu installieren, stehen die Pakete „openssh-server“ und „openssh-client“ zur Verfügung. Um beide Pakete in einem Zug zu installieren, kann das Meta-Paket „ssh“ verwendet werden.

```
$> sudo apt-get install ssh
```

Auf einem entfernten Rechner einloggen

Die Eingabe der folgenden Zeile loggt den lokalen Benutzer als Benutzer „johndoe“ auf dem Rechner „asgard.org“ ein. Auf dem Rechner „asgard.org“ muss dazu ein SSH-Server laufen. Wenn der Port nicht extra angegeben wird, läuft dieser auf Port 22.

```
$> ssh johndoe@asgard.org
```

Bei der erstmaligen Verbindung zu diesem Rechner muss noch der sogenannte „Host-Key“ akzeptiert werden. Dabei handelt es sich um den öffentlichen Teil eines Schlüsselpaares, das den entfernten Rechner eindeutig identifiziert.

Nach Eingabe des Passwortes kann man schließlich auf dem entfernten Rechner arbeiten. Wenn der lokale Benutzername gleich dem entfernten Benutzername ist, kann „johndoe@“ wegfallen.

Dateien mit entfernten Rechnern austauschen

Um die Datei „test.txt“ aus dem aktuellen Verzeichnis in das Homeverzeichnis des gleichnamigen Benutzers auf „asgard.org“.

```
$> ssh test.txt asgard.org:
```

Die Datei „/tmp/test.bat“ auf dem entfernten Rechner kann wie folgt in das aktuelle Verzeichnis kopiert werden:

```
$> ssh asgard.org:/tmp/test.bat .
```

Einen SSH-Tunnel einrichten

Um beliebige Netzwerkkommunikation über einen verschlüsselten Tunnel umzuleiten, kann ein bestimmter Port auf einem entfernten Rechner (z.B. www.heise.de:80) über einen SSH-Server geleitet auf einen lokalen Port umgemappt werden.

Der SSH-Server baut seinerseits die Verbindung auf und leitet die Daten verschlüsselt an den lokalen Rechner weiter. Im folgenden Beispiel heisst der SSH-Server „asgard.org“.

```
$> ssh -L 4711:www.heise.de:80 asgard.org
```

Nach Eingabe des Passwortes ist der Server www.heise.de:80 auf dem lokalen Port 4711 erreichbar. Das kann man z.B. im Internetbrowser durch die Eingabe von „http://localhost:4711“ testen.

In Wahrheit wird daraufhin über eine Verschlüsselte Verbindung zu asgard.org der Datenverkehr zwischen asgard.org und www.heise.org an den lokalen Computer geleitet.

Auch unverschlüsselte E-Mails von einem POP3-Server können so verschlüsselt abgeholt werden.

Schlüsselpaar generieren und öffentliche Schlüssel austauschen

Um die ständige Eingabe des Passwortes bei einer SSH-Verbindung (z.B. zu Rechner asgard.org) zu vermeiden, kann ein öffentlicher Schlüssel ausgetauscht werden. Die folgende Befehlsfolge kann eingesetzt werden.

```
$> cd $HOME
$> ssh-keygen -t rsa
$> # Hier nun alle Fragen mit „ENTER“ bestätigen
$> scp .ssh/id_rsa.pub asgard.org:
$> ssh asgard.org
$> # nun muss nun letztmalig das Passwort eingegeben werden
$> # Die weiteren Schritte finden auf dem Server statt
$> mkdir -p .ssh
$> chmod 700 .ssh
$> cat id_rsa.pub >> .ssh/authorized_keys
$> rm id_rsa.pub # Die Datei wird nicht mehr benötigt
$> logout # Rechner wieder verlassen
```

Die Kommandos gehen davon aus, dass der Benutzername auf beiden Rechnern der gleiche ist. Wenn man sich in Zukunft mit den Rechner „asgard.org“ verbindet, muss kein Passwort mehr eingegeben werden. Der

entfernte Rechner prüft, ober der Schlüssel zu Ihrem Rechner (bzw. zu Ihrem Account) passt und gewährt den Zugang.

11. SMB/NetBIOS-Server „samba“

Mit der Paketfamilie „Samba“ ist es möglich

- Dateien auf einem Netzlaufwerk zu speichern und zu lesen
- Entfernte Drucker als Netzwerkdrucker mit zu nutzen
- Eine zentrale Userverwaltung durch PDC zu realisieren

Die meisten dieser Aufgaben werden kombiniert wahrgenommen. So stellt zum Beispiel ein PDC die zentrale Benutzer-Authentifizierung, ein Netzlaufwerk als Homeverzeichnis und ggf. weitere Verzeichnisse sowie einen Netzwerkdrucker zur Verfügung.

11.1 Samba als Primary Domain Controller (PDC)

Zunächst muss das Paket „samba“ installiert werden. Evtl. Abhängigkeiten (wie zum das Paket „samba-common“) werden mitinstalliert.

Nun sollte man, nach Anfertigung einer Sicherungskopie, die Datei „/etc/samba/smb.conf“ durch die im Beispiel 28 aufgelistete Variante ersetzen.

Beispiel 28: /etc/samba/smb.conf für einen PDC

```
[global]
  workgroup = MEINEDOMAIN
  passdb backend = smbpasswd:/etc/samba/smbpasswd
  name resolve order = wins bcast hosts
  logon drive = H:
  logon path = \\%L\%U\profile
  domain logons = Yes
  preferred master = Yes
  wins support = Yes

[homes]
  comment = Home Verzeichnisse
  valid users = %S
  read only = No
  browseable = No
```

Nach einem Neustart des Servers steht Der PDC bereits zur Benutzerauthentifizierung zur Verfügung, jedoch existieren noch keine Benutzer und noch ist kein Rechner der Domäne „MEINEDOMAIN“ beigetreten.

```
$> /etc/init.d/samba restart
```

Um auf dem Rechner Benutzer anzulegen und anderen Rechnern das Beitreten zur Domäne zu erlauben, benötigt man root-Rechte. Alle Benutzer, die sich über diesen PDC authentifizieren wollen, müssen auf dem Server einen gültigen Linux-Account haben. Ausserdem müssen diese Benutzer ein Passwort für SMB-Verbindungen erhalten.

Benutzer auf Server anlegen:

```
$> useradd –m johndoe
```

Dem Benutzer ein SMB-Passwort vergeben:

```
$> smbpasswd –a johndoe
```

Damit ein Rechner einer Domäne beitreten kann, muss für ihn ein sogenannter Machine-Account angelegt werden. Es handelt sich dabei um einen normalen Unix-Account, der jedoch mit einem '$' endet.

Machine-Account anlegen:

```
$> useradd –m ASGARD$
$> smbpasswd –m –a ASGARD$
```

Nun muss der entfernte Rechner der Domäne beitreten. Er muss sich hierfür mit dem Root-Passwort beim PDC authentifizieren, welches noch zu vergeben ist.

SMB-Root-Passwort vergeben:

```
$> smbpasswd –a root
```

Die folgenden Schritte sind erforderlich, um den Windowsrechner (ASGARD) zur Domäne hinzuzufügen:

- Rechtsklick auf Arbeitsplatz -> Eigenschaften
- Computername -> Domäne
- Den Domänennamen „MEINEDOMAIN“ angeben
- 'root' und das SMB-Root-Passwort angeben

Nach einem Neustart des Windowsrechners ist eine Anmeldung über die Domäne möglich.

Die folgend Schritte ermöglichen eine Authentifizierung auf Linux-Rechner an einer PDC-Domäne durch „winbind“:

- Installation von Samba mit folgender /etc/samba/smb.conf :

```
[global]
workgroup = MEINEDOMAIN
idmap uid = 10000-20000
idmap gid = 10000-20000
security = domain
encrypt passwords = yes
winbind enum groups = yes
winbind enum users = yes
winbind use default domain = yes
template homedir = /home/%U
obey pam restrictions = yes
```

- In der Datei „/etc/nsswitch.conf sind zwei Zeilen abzuändern:

```
group:          compat winbind
group:          compat winbind
```

- Installation des Paketes „winbind“
- Im Verzeichnis „/etc/pam.d“ ist in den Dateien „common-auth“, „common-session“, „common-password“ und „common-account“ folgende Zeile anzufügen ('auth' ist ggf. mit 'session', 'password' bzw. 'account' zu ersetzen):

```
auth  sufficient  pam_winbind.so
```

Hinweis:
Vor jeder Veränderung in den Dateien des Verzeichnisses „/etc/pam.d/“ sollte dieses Verzeichnis gesichert werden. Um Fehler zu vermeiden sollte das Handbuch zu pam konsultiert werden. Nach der Installation des Paketes „libpam-doc“ findet man ausführliche Dokumentation im Verzeichnis „/usr/share/doc/libpam-doc/html“ .

- Im Verzeichnis „/lib/“ sind die für NSS erforderlichen Bibliotheken so zu verlinken, dass die entsprechenden Module gefunden werden können.

```
$> cd /lib
$> ln -s libnss_winbind.so libnss_winbind.so.2
$> ln -s libnss_wins.so libnss_wins.so.2
```

- Nach einem Neustart von winbind und samba muss der Linux-Rechner der Domäne beitreten

```
$> net join -U root
```

Weitere Informationen zu diesem Thema finden Sie in der Datei file:///usr/share/doc/samba-doc/htmldocs/using_samba/ch09.html wenn das Paket „samba-doc“ installiert wurde. (Kapitel „Authentication with winbind“).

Hinzufügen weiterer Freigaben

Um das lokale Verzeichnis „/mnt/freigabe“ des SMB-Servers für jeden freizugeben, ist der folgende Eintrag auf dem Samba-Server in die smb.conf hinzuzufügen:

Beispiel: Anzufügen an „/etc/samba/smb.conf“

```
[freigabe]
      path = /mnt/freigabe
      read only = no
      public = yes
```

Weitere Informationen finden Sie in der Manpage von smb.conf (man smb.conf).

12. VPN - Virtual Private Network

Für die Verbindung zweier Netzwerke über ein unsicheres Drittnetz eignet sich VPN, und in unserem Fall die Version „OpenVPN“ besonders gut. Die Installation erstreckt sich für diese Konfiguration auf folgende Schritte.

1. Installation der Software auf Client und Server
2. Erstellung und Austausch von Zertifikaten
3. Anpassung der Konfigurationsdateien

Um die Software zu installieren wird das Paket openvpn wie gewohnt installiert, z.B. bei Debian mit der folgenden Kommandozeile.

```
$> sudo apt-get install openvpn
```

Als nächstes sind die notwendigen Zertifikate zu erstellen. Das Paket openvpn bietet einige Skripte, die den Umgang mit OpenSSL zu diesem Zweck vereinfachen. Im folgenden Ordner befindet sich auch eine Anleitung (**README.gz**), die die Erstellung detailliert beschreibt.

```
$> cd /usr/share/doc/openvpn/examples/easy-rsa
```

Die dort befindliche Datei README.gz kann z.B. mit **zcat** oder **zless** gelesen werden.

```
$> zless README.gz
```

12.1 Vorbereitungen

Im einzelnen sind folgende Schritte durchzuführen (es folgt eine geraffte Übersetzung der o.a. Anleitung aus dem Englischen):

1. Editieren der Datei 'vars'.

   Die Variable KEY_CONFIG muss auf die Datei openssl.cnf zeigen, die sich normalerweise ebenfalls im Verzeichnis easy-rsa befindet.

   Die Variable KEY_DIR muss auf das Verzeichnis zeigen, in den die erzeugten Schlüssel und Zertifikate abgespeichert werden sollen.

2. Eingabe des Befehls „. vars“ (Punkt, Leerzeichen, „vars“). Dies übernimmt die darin definierten Umgebungsvariablen in die aktuelle Terminalsitzung.

3. Aufruf des Skriptes „./clean-all“

   Die Im folgenden erstellten Dateien haben die Endungen 'key' (privater Schlüssel), 'csr' (Certificate Sign Request) und 'crt' (Certificate). Es ist darauf zu achten, dass die Dateien mit der Endung 'key' nicht über unsichere Kanäle versendet werden!

4. Erstellen virtueller Netzwerkgeräte

12.2 Erstellung der SSL-Zertifikate

Um zwei Netze zu verbinden, ist die sogenannte Selbstzertifizierung, also das Signieren des eigenen Schlüssels, eine legitime Vorgehensweise für die Erlangung von Zertifikaten. Hierfür wird der Schlüssel der Certificate Authority (CA, die Zertifizierungsstelle) benötigt. Es wird durch den Aufruf

```
$> ./build-ca
```

erstellt. Man ist nun selbst in der Lage, als CA zu fungieren. Die benötigten Schlüssel (und alle folgenden erstellten Dateien) befinden sich in dem Ordner, der in der Datei 'vars' der Variable KEY_DIR zugewiesen wurde.

Nun wird jeweils für den Client und den Server eine Zertifizierungsanfrage (Certificate Sign Request) erstellt und durch signieren (mit dem Schlüssel der CA) zum gültigen Zertifikat gemacht.

Anfrage erstellen:

```
$> ./build-req <zertifikatsname>
```

Nun wird eine Datei '**<zertifikatsname>.csr**' erstellt, die wie folgt zu einem Zertifikat gemacht werden kann:

```
$> ./sign-req <zertifikatsname>
```

Diese Schritte sind auf dem Server und dem Client vorzunehmen. Die Signierung muss durch die selbe CA, d.h. mit dem oben erstellten Schlüssel der CA erfolgen.

Auf dem Server ist nun noch der Diffie-Hellmann-Parameter zu erzeugen, und zwar durch Aufruf des folgenden Skriptes:

```
$> ./build-dh
```

12.3 Konfigurationsdateien erstellen

Das Paket OpenVPN liefert unter Debian für den o.a. Anwendungsfall Beispielkonfigurationen mit. Auf dem Server bzw. dem Client sind sie lediglich in ein geeignetes Verzeichnis zu kopieren (**/etc/openvpn/**) und dann gemäß der darin enthaltenen Kommentare anzupassen (IP's anpassen, virtuelles Netzwerkgerät (s.u.) angeben und Pfade zu den Zertifikaten eintragen).

```
$> cp /usr/share/doc/openvpn/examples/sample-config-files/client.cfg \
     /etc/openvpn
```

bzw. auf dem Server

```
$> cp /usr/share/doc/openvpn/examples/sample-config-files/client.cfg \
    /etc/openvpn
```

12.4 Erstellung der virtuellen Netzwerkgeräte

Ein virtuelles Netzwerkgerät kann mit dem Befehl 'tunctl' erstellt werden. Hierzu ist das Paket 'uml-utilities' (User-Mode-Linux-Utilities) zu installieren. Das Gerät (z.B. tap0) kann dann von OpenVPN verwendet werden.

```
$> sudo apt-get install uml-utilities
$> tunctl -t tap0
```

Der Aufruf von OpenVPN erfolgt schließlich unter der Angabe der Konfigurationsdatei.

Weitere Informationen sind in der Manpage von openvpn, im o. a. Dokumentationsverzeichnis und auf der Website http://de.wikipedia.org/wiki/OpenVPN verfügbar.