Firewalling und Proxy Server

Jürgen Steiner (`js@barbar.augusta.de`)

v0.1-2, 18. März 1997

Dieses Dokument wurde entwickelt, um die Grundlagen von Firewall-Systemen zu vermitteln und bietet einige Details zur Konfiguration von Filter- und Proxy-Firewalls auf einem PC-System, basierend auf Linux.

Inhaltsverzeichnis

1 Einführung

Das original FIREWALL-HOWTO wurde von David Rudder geschrieben und von Mark Grennan erweitert. Diese erweiterte Version wurde von mir ins Deutsche übersetzt.

Firewall-Systeme haben einen großen Erfolg als das Ultimative in Internet-Sicherheit gewonnen. Wie die meisten Dinge die Erfolg haben, ist mit dem Erfolg auch ein falsches Verständnis gekommen. Dieses Dokument wird die Grundlagen vermitteln, was Firewall und Proxy-Server sind und wie sie konfiguriert werden. Es werden auch Anwendungen außerhalb des Sicherheitsbereiches vorgestellt.

1.1 Feedback

Ich bitte mir jede Ungenauigkeit in diesem HOWTO zu berichten. Jeglicher Feedback ist willkommen.

meine eMail-Adresse ist: `js@barbar.augusta.de`

1.2 Disclaimer

Ich bin nicht verantwortlich für Folgen die durch die Anwendung dieses HOWTOs enstehen.

Dieses Dokument ist als eine Einführung in die Funktion von Firewall und Proxy-Servern gedacht. Ich bin kein Experte für Sicherheitsfragen und möchte hier auch keine ultimative Anleitung für den Betrieb von Firewall und Proxy-Servern bieten.

1.3 Copyright

1.4 Die Gründe für die Entstehung dieses HOWTO's

Es gab eine Menge Diskussionen auf `comp.os.linux.*` in letzter Zeit über Firewalls. Und Informationen darüber einen Firewall zu administrieren, waren sehr dürftig. Dieses HowTo soll den Wissenshungrigen zielgerecht an das Thema heranführen.

2 Allgemeine Einführung

2.1 Vorbereitende Lektüre als Empfehlung

- Das *NET-3 HOWTO*
- Das *Ethernet HOWTO*
- Das *Mehrere Ethernetkarten mini HOWTO*
- *Networking with Linux*
- Das *PPP HOWTO*
- *TCP/IP Network Administrator's Guide* by O'Reilly and Associates
- Die Dokumentation zum TIS Firewall Toolkit

Die Trusted Information System's (TIS) Website hat eine große Sammlung an Informationen, Dokumentationen und Programmen um ein sicheres Linux-System aufzubauen: `http://www.tis.com/`

3 Beschreibung des Firewalls

Ein Firewall in Computern ist eine logische Vorrichtung, die ein privates Netz vor dem öffentlichen Teil (Internet) schützt.

Funktionsprinzip:

Ein Computer mit installierter Routing-Software und 2 Schnittstellen (z.B. serielle Schnittstellen, Ethernet, Token Ring, usw.). Das Internet ist mit einer Schnittstelle verbunden, das zu schützende private Netz mit der anderen Schnittstelle. Jetzt, haben Sie zwei verschiedene Netze, die sich einen Computer teilen.

Der Firewall-Computer, von jetzt an »Firewall« genannt, kann beide Seiten erreichen, das geschützte private Netz und das Internet. Niemand aus dem geschützten Netz kann das Internet erreichen, und aus dem Internet kann niemand in das geschützte Netz.

Damit jemand das Internet vom geschützten Netz aus erreichen kann, muß er eine Telnet-Verbindung zum Firewall aufbauen und das Internet von dort aus benutzen. Entsprechend, um eine Verbindung vom Internet aus in das geschützte Netz zu bekommen, muß man auch durch den Firewall gehen.

Dieses stellt eine ausgezeichnete Sicherheit gegen Angriffe aus dem Internet dar. Falls jemand einen Angriff gegen das geschützte Netz machen will, muß er zuerst durch den Firewall gehen. Ein 2-stufiger Zugang zum gesicherten Netz ist resistent gegen Angriffe. Falls jemand das geschützte Netz über eine gemeinere Methode angreifen will, wie z.B. Mail-Bombe oder den berüchtigten »Internet Wurm«, werden sie nicht in der Lage sein das geschützte Netz zu erreichen. Dies ist ein ausgezeichneter Schutz.

4 Nachteile eines Firewall

Das größte Problem eines Firewalls ist, daß er den Zugang zum Internet von der Innenseite kommend stark hemmt. Grundsätzlich reduziert er den Gebrauch des Internets dahingehend, als ob man nur einen »Dialup Shell Zugang« haben würde. Sich in den Firewall einloggen zu müssen um vollen Internet-Zugang zu haben ist eine starke Beeinschränkung.

Programme wie Netscape, die eine direkte Internet-Verbindung benötigen, werden hinter einem Firewall nicht arbeiten. Die Antwort zu diesem Problem hat ein Proxy-Server. Proxy-Server erlauben ein direktes Erreichen des Internets hinter einem Firewall.

Um die Möglichkeit, von einem Computer im geschützten Netz mit Netscape im Web zu lesen, anbieten zu können, setzt man einen Proxy-Server auf den Firewall auf. Der Proxy-Server würde so konfiguriert werden, daß ein Computer vom eigentlichen Port 80 des Firewalls zum Port 1080 des Proxy verbunden wird, um alle Verbindungen zu den richtigen Adressen umzuleiten.

Der große Vorteil von Proxy-Servern ist die absolute Sicherheit, wenn sie korrekt konfiguriert sind. Sie werden niemanden erlauben sie zu umgehen. Der Proxy-Server ist vor allem eine Sicherheitsvorrichtung. Seine Benutzung für einen Internet-Zugang mit begrenzten IP-Adressen wird viele Nachteile haben.

Ein Proxy-Server bietet Zugang von innerhalb des geschütztes Netzes zur Außenseite, aber die Innenseite wird völlig unerreichbar für die Außenseite bleiben. Dieses bedeutet keine Server-, Talk- oder Archie-Verbindungen, oder direkte Emails zu den Computern im geschützten Netz. Diese Nachteile können gering erscheinen, aber bedenken sie: Es liegt ein Dokument auf Ihrem Computer innerhalb eines per Firewall geschütztem Netzes.

FTP verursacht ein anderes Problem mit einem Proxy-Server. Beim Absenden des `ls`-Befehls, öffnet der FTP-Server einen Port auf der Kundenmaschine und übermittelt die Information. Ein Proxy-Server wird das nicht erlauben, somit arbeitet FTP nicht zuverlässig. Proxy-Server arbeiten langsam. Wegen dem größeren Protokollaufwandes werden fast alle anderen Mittel, um einen Internet-Zuganges zu bekommen, schneller sein.

Grundsätzlich, falls Sie ausreichend IP Adressen haben und ihnen macht die Sicherheit keine Sorgen, wird kein Firewall und/oder Proxy-Server benutzt. Falls Sie zu wenig IP Adressen haben und ihnen macht die Sicherheit keine Sorgen, wird man eher einen IP-Emulator benutzen wie Term, Slirp oder TIA. Diese Pakete arbeiten schneller, erlauben bessere Verbindungen, und stellen ein hochwertigen Zugang vom Internet zum geschützten Netz bereit.

Proxy-Server sind gut für jene Netzwerke mit vielen Hosts, welche einen transparenten Internetzugang wollen. Sie benötigen nur eine kleine Konfiguration und wenig Verwaltungsarbeit im laufenden Betrieb.

5 Funktion und Prinzip eines Firewalls

5.1 Firewall Varianten

Es gibt zwei Arten von Firewalls.

1. IP oder Filter Firewalls - die alles sperren bis auf ausgewählten Netzwerkverkehr
2. Proxy-Server - die einem die Netzwerkverbindung übernehmen

5.1.1 IP Filter Firewalls

Ein IP Filter Firewall arbeitet auf Paket-Ebene, er ist so konstruiert, daß er den Datenstrom kontrolliert auf Grund von Ursprung, Ziel, Port und Paket-Typ-Information die in jedem Daten-Paket enthalten sind.

Diese Art des Firewalls ist sehr sicher, aber es mangelt an einer brauchbaren Protokollierung. Er verbietet den Zugang zum privaten Netzwerk aber es wird nicht protokolliert wer auf das private Netzwerk zugreifen will oder wer vom privaten Netz Zugriff ins Internet haben will.

Filter Firewalls sind absolute Filter. Gibt man einem von außen den Zugriff auf das private Netz hat automatisch jeder Zugriff darauf.

In Linux ist packet-filtering-software seit Kernel 1.2.13 enthalten

5.1.2 Proxy Server

Proxy Server erlauben indirekten Zugang zum Internet durch den Firewall. Als Beispiel zur Funktion startet man einen Telnet zu einem System um von dort aus einen Telnet zu einem anderen zu starten. Nur mit einem Proxy-Server kann man diesen Vorgang automatisieren. Wenn man mit einer Client-Software einen connect zum Proxy-Server macht, startet der automatisch seine Client(Proxy)-Software und reicht die Daten durch.

Weil Proxy-Server ihre Kommunikation duplizieren, können sie alles mitprotokollieren was sie tun.

Der große Vorteil von Proxy-Server ist, sofern sie korrekt konfiguriert sind, daß sie absolut sicher sind. Sie erlauben keinem ein Durchkommen. Sie haben keine direkten IP-Routen.

6 Firewall Konfiguration

6.1 Benötigte Hardware

In diesem Beispiel ist der Computer ein 486-DX66 mit 16 MB RAM und einer 500 MB Linux Partition. Dieses System hat zwei Netzwerkkarten. Eine ist an das private LAN angeschlossen. Die Andere ist an ein LAN angeschlossen, das wir die »Demilitarisierte Zone« (DMZ) nennen wollen. An die DMZ ist ein Router angeschlossen mit Verbindung zum Internet.

Das ist eine übliche Standardkonfiguration für ein Büro. Man kann auch nur eine Netzwerkkarte und dafür ein Modem mit PPP nehmen. Hauptsache der Firewall hat zwei IP-Netzwerk-Adressen.

Es gibt ein paar Leute mit kleinen LANs zuhause mit zwei oder drei Computern. Manche könnten in Erwägung ziehen alle Modems an eine Linux-Box (vielleicht ein verstaubter 386er) zu stecken um sie ans Internet anzubinden, vielleicht sogar mit Load-Balancing. Der Datendurchsatz von nur einer Person würde sich mit dieser Konfiguration glatt verdoppeln.

7 Firewall Software

7.1 Verfügbare Pakete

Wenn man nur einen Filter Firewall will, reicht Linux mit dem Standard-Netzwerk-Paket. Das »IP Firewall Administration Tool« kann möglicherweise nicht bei allen Linux-Distributionen dabei sein.

IPFWADM ist zu holen bei:

```
http://www.xos.nl/linux/ipfwadm/
```

Um einen Proxy-Server zu installieren benötigt man eines der beiden Packages:

1. SOCKS
2. TIS Firewall Toolkit (FWTK)

7.2 Der TIS Firewall Toolkit vs. SOCKS

Trusted Information System (`http://www.tis.com`) veröffentlichte eine Kollektion von Programmen, die einem das »Firewalling« erleichtern. Die Programme haben grundsätzlich die selbe Funktion wie das »SOCKS« Paket, haben aber eine unterschiedliche Strategie im Programmaufbau. SOCKS bedient mit nur einem Programm alle Internet-Transportaufgaben, wobei TIS für jedes Utility, welches den Firewall benützen möchte, ein eigenes Programm anbietet.

Um die beiden zu vergleichen, nehmen wir als Beispiel eine WWW- und eine Telnet-Verbindung. Mit SOCKS muß man nur eine Konfigurationsdatei anpassen und hat nur einen Dämon. Durch diese Datei und dem Dämon hat man die Möglichkeit für WWW und Telnet geschaffen, ebenso für alle anderen Dienste die nicht gesperrt sind.

Mit dem TIS-Toolkit startet man für WWW und Telnet jeweils einen eigenen Dämon mit seiner eigenen Konfigurationsdatei. Jeder weitere Internetdienst ist solange nicht möglich bis er explizit freigegeben wird. Wird ein Dämon für einen bestimmten Dienst nicht angeboten (z.B. TALK) gibt es einen »plug-in« daemon, er ist aber dann weder flexibel, noch einfach zu konfigurieren wie die anderen Utilities.

Das klingt vielleicht einfach, macht aber einen großen Unterschied. SOCKS erlaubt einem nachlässiger zu sein. Mit einem nachlässig installiertem SOCKS-Server kann ein Anwender aus dem LAN mehr Zugriff zum Internet erreichen als eigentlich beabsichtigt. Mit dem TIS-Toolkit haben die Anwender im LAN nur den Zugriff, den der System-Administrator erlaubt.

SOCKS ist einfacher zu konfigurieren, einfacher zu kompilieren und erlaubt größere Flexibilität. Das TIS-Toolkit ist weitaus sicherer wenn man die Anwender im geschützten LAN regulieren will. Beide bieten absoluten Schutz vor der Außenwelt.

Eine Möglichkeit der Installation und Konfiguration beider Systeme wird in diesem HOWTO beschrieben.

8 Vorbereitung des Linux Systems

8.1 Kompilieren des Kernels

Beste Voraussetzung ist eine saubere Linuxinstallation.(Die Beispiele hier sind auf einer RedHat 3.0.3 Distibution entstanden). Je weniger Software man installiert hat, desto weniger Schlupflöcher, Hintertürchen und/oder Fehler bilden die Grundlage für Sicherheitsprobleme in Ihrem System. Idealerweise installiert man nur die unbedingt nötigte Software.

Als Kernel verwendet man einen bekannt Stabilen, wie z.B 2.0.14 (auf den sich die folgende Konfiguration abstützt).

1. Under General setup
   (a) Turn Networking Support ON
2. Under Networking Options
   (a) Turn Network firewalls ON
   (b) Turn TCP/IP Networking ON
   (c) Turn IP forwarding/gatewaying OFF (UNLESS you wish to use IP filtering)
   (d) Turn IP Firewalling ON
   (e) Turn IP firewall packet loggin ON (this is not required but it is a good idea)
   (f) Turn IP: masquerading OFF (I am not covering this subject here.)
   (g) Turn IP: accounting ON
   (h) Turn IP: tunneling OFF
   (i) Turn IP: aliasing OFF
   (j) Turn IP: PC/TCP compatibility mode OFF
   (k) Turn IP: Reverse ARP OFF
   (l) Turn Drop source routed frames ON
3. Under Network device support
   (a) Turn Network device support ON
   (b) Turn Dummy net driver support ON
   (c) Turn Ethernet (10 or 100Mbit) ON
   (d) Select your network card

Nun kann man den Kernel neu kompilieren und installieren und Linux neu starten. Die Netzwerkkarte sollte in den Boot-Meldungen erscheinen. Wenn nicht, nochmal einen Blick in die entsprechende HOWTOs werfen.

8.2 Konfiguration von zwei Netzwerkkarten

Sind zwei Netzwerkkarten im Computer, kann man einen Zusatzeintrag in `/etc/lilo.conf` machen, der die IO-Adressen und die IRQ's der beiden Karten bekannt gibt.

z.B.

```
append="ether=12,0x300,eth0 ether=15,0x340,eth1"
```

8.3 Konfiguration der Netzwerkadressen

Dies ist der interessante Teil, weil es ein paar Entscheidungen zu machen gilt. Nachdem erwünscht ist, daß aus dem Internet kein Zugriff auf irgendeinen Teil des LAN stattfindet, braucht man auch keine offiziellen Adressen für das LAN . Es gibt ein paar Adressen nur für den Gebrauch in privaten Netzwerken. Weil immer mehr Adressen gebraucht werden und diese private Adressen nicht geroutet werden, sind sie eine gute Wahl.

Einer dieser Adressbereiche ist 192.168.2.x , der auch hier in diesem Beispiel benützt wird.

Der Beispiel-Proxy-Server hier ist ein Bestandteil in beiden Netzwerken und kann somit Daten von und ins private Netzwerk übergeben.

```
         199.1.2.10    ___________    192.168.2.1
   _  __  _          \ |           | /           ________________
  | \/  \/ |          \| Firewall  |/            |                |
 / Internet \--------|  System   |------------| Workstation/s |
 \_/\_/\_/\_/         |___________|            |________________|
```

Will man einen »Filter Firewall« benützen, kann man diese Adressen beibehalten. Man muß allerdings IP-Masquerading benützen um dies zu ermöglichen. Mit diesem Zusatz wird der Firewall die Pakete weiter transportieren und dabei in »Offizielle« IP-Adressen umsetzen, um sie ins Internet zu schicken.

Die Netzwerkkarte zum Internet muß die offizielle IP-Adresse bekommen und die Karte zum LAN bekommt die 192.168.2.1 zugewiesen. Das wird die Proxy/Gateway-Adresse. Alle anderen Systeme im geschützten LAN können Adressen aus dem Bereich 192.168.2.xxx bekommen (192.168.2.2 bis 192.168.2.254).

Bei RedHat kann man, um das Netzwerk beim starten von Linux zu konfigurieren, eine Datei in `/etc/sysconfig/network-scripts` hinzufügen. Diese Datei wird beim starten gelesen und konfiguriert das Netzwerk und die Routing-Tabelle.

Ein Beispiel für `ifcfg-eth1`:

```
#!/bin/sh
#>>>Device type: ethernet
#>>>Variable declarations:
DEVICE=eth1
IPADDR=192.168.2.1
NETMASK=255.255.255.0
NETWORK=192.168.2.0
BROADCAST=192.168.2.255
GATEWAY=199.1.2.10
ONBOOT=yes
#>>>End variable declarations
```

Man kann auch das Script dazu verwenden um per Modem automatisch eine Verbindung zum Provider aufzubauen. Als Beispiel kann das `ipup-ppp` Skript dienen.

Bei Benutzung eines Modems für die Internetverbindung kann die IP-Adresse zum Internet vom Provider dynamisch beim Connect übermittelt werden.

8.4 Testen des Netzwerkes

Begonnen wird mit der Kontrolle von `ifconfig` und `route`. Bei zwei Netzwerkkarten kann `ifconfig` so aussehen:

```
#ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.0  Bcast:127.255.255.255  Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
          RX packets:1620 errors:0 dropped:0 overruns:0
          TX packets:1620 errors:0 dropped:0 overruns:0

eth0      Link encap:10Mbps Ethernet  HWaddr 00:00:09:85:AC:55
          inet addr:199.1.2.10 Bcast:199.1.2.255  Mask:255.255.255.0
```

```
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
          Interrupt:12 Base address:0x310

eth1      Link encap:10Mbps Ethernet  HWaddr 00:00:09:80:1E:D7
          inet addr:192.168.2.1  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:0 errors:0 dropped:0 overruns:0
          Interrupt:15 Base address:0x350
```

Und die Tabelle von route:

```
  #route -n
  Kernel routing table
  Destination     Gateway         Genmask         Flags MSS    Win-
dow Use Iface
  199.1.2.0       *               255.255.255.0   U     1500   0        15 eth0
  192.168.2.0     *               255.255.255.0   U     1500   0         0 eth1
  127.0.0.0       *               255.0.0.0       U     3584   0         2 lo
  default         199.1.2.10      *               UG    1500   0        72 eth0
```

Zur Erinnerung: 199.1.2.0 ist die Internetseite des Firewalls und 192.168.2.0 ist die private Seite.

Nun kann versucht werden vom Firewall einen ping zum Internet zu starten. Ein guter Testpunkt wäre ns.nic.de. Es ist eigentlich ein guter Test, hat sich aber als nicht sehr zuverlässig herausgestellt. Falls der ping nicht funktioniert, einfach mal eine Adresse des Providers versuchen. Falls auch das nicht funktioniert, dann ist die IP-Verbindung nicht richtig konfiguriert und muß mit Hilfe des *NET-3 HOWTOs* berichtigt werden.

Als nächstes versucht man einen ping vom Firewall zu einem System innerhalb des geschützten Netzwerkes. Falls dies auch nicht klappt, dann wiederum mit Hilfe des *NET-3 HOWTOs* das LAN neu konfigurieren.

Dann versucht man von einem System innerhalb des LAN's eine Adresse außerhalb des Firewalls anzupingen. (Dies sollte keine der 192.168.2.xxx IP-Adressen sein) Wenn das funktioniert, hat man IP-Forwarding nicht gesperrt. Man sollte sichergehen, daß dies gewünscht ist. Ist es freigegeben, sollte man sich den IP-Filtering Absatz in diesem Dokument näher ansehen.

Am Besten versucht man jetzt eine Internetadresse von einem Rechner im LAN anzupingen. Idealerweise nimmt man die Adresse die schon beim Test am Firewall funktioniert hat (ns.nic.de). Nur zur Erinnerung, bei gesperrten IP-Forwarding funktioniert der Test nicht, nur wenn IP-Forwarding freigegeben ist.

Wenn IP-Forwarding freigegeben ist und »Offizielle IP-Adressen« (keine 192.168.2.*) verwendet werden, kann man das Internet nicht direkt anpingen, sondern nur die Internetseite des Firewalls. [*]

Hat man als Adressen für das geschützte LAN 192.168.2.* gewählt, können auch keine Datenpakete geroutet werden. Ausser IP-Masquerading ist installiert, dann ist der Test wiederum möglich.

Jetzt ist das Basis-System installiert.

8.5 IP-Filter Konfiguration (IPFWADM)

Um zu beginnen soll IP-Forwarding im Kernel konfiguriert sein und das System sollte in der Lage sein, alle Pakete weiter zu schicken. Die Routing-Tabellen sollten passen und man soll in der Lage sein, vom LAN aus alle Zielsysteme im Internet zu erreichen und zurück.

Da hier ein Firewall gebaut werden soll, ist es notwenig jeglichen freien Zugang zu sperren.

Es ist nützlich ein paar Shell-Scripte zu schreiben die beim booten die Firewall-forwarding Regeln und das Accounting festlegen.

Das IP-Forwarding im Kernel leitet unkonfiguriert alles weiter. Aus diesem Grund soll das Boot-Script beim Systemstart jeglichen Zugriff verbieten und jegliche ipfw-Regeln vom letzten Start löschen. Ein Beispiel-Script soll dies erklären:

```
#
# setup IP packet Accounting and Forwarding
#
#   Forwarding
#
# By default DENY all services
ipfwadm -F -p deny
# Flush all commands
ipfwadm -F -f
ipfwadm -I -f
ipfwadm -O -f
```

Das ist jetzt der ultimative Firewall. Keine Daten kommen jetzt durch. Um jetzt ein paar Internet-Dienste durchzulassen hier ein paar Beispiele:

```
# Forward email to your server
ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/0 1024:65535 -D 192.1.2.10 25

# Forward email connections to outside email servers
ipfwadm -F -a accept -b -P tcp -S 196.1.2.10 25 -D 0.0.0.0/0 1024:65535

# Forward Web connections to your Web Server
/sbin/ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/0 1024:65535 -
D 196.1.2.11 80

# Forward Web connections to outside Web Server
/sbin/ipfwadm -F -a accept -b -P tcp -S 196.1.2.* 80 -D 0.0.0.0/0 1024:65535

# Forward DNS traffic
/sbin/ipfwadm -F -a accept -b -P udp -S 0.0.0.0/0 53 -D 196.1.2.0/24
```

Um eine Übersicht des Datenverkehrs zu erhalten, der durch den Firewall geht, hier ein Beispiel:

```
# Flush the current accounting rules
ipfwadm -A -f
# Accounting
/sbin/ipfwadm -A -f
/sbin/ipfwadm -A out -i -S 196.1.2.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -A out -i -S 0.0.0.0/0 -D 196.1.2.0/24
/sbin/ipfwadm -A in -i -S 196.1.2.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -A in -i -S 0.0.0.0/0 -D 196.1.2.0/24
```

Wenn das gewünschte ein Firewall-Filter war, dann ist hier das Ende.

9 Installation des TIS Proxy Servers

9.1 Beschaffung der Software

Der TIS Firewall Tool Kit (FWTK) ist erhältlich bei `ftp.tis.com`

Der FWTK wird von TIS in einem versteckten Verzeichnis auf ihrem Server bereitgehalten. TIS verlangt, daß eine **E-Mail an `fwtk-request@tis.com`** gesandt wird mit dem einzigen Wort **SEND** im Mailbody. Ein Subject wird nicht benötigt. TIS schickt eine E-Mail zurück mit dem Pfad zu dem Verzeichnis, wo man den FWTK herunter laden kann. Das Verzeichnis ist für ca. 12h zugänglich.

Zur Zeit ist die Version 2.0 beta vom FWTK aktuell, welche in diesem Beispiel hier auch verwendet wird. Wenn die endgültige Version veröffentlicht wird, gibt es auch eine neuere Version von diesem HOWTO.

Um den FWTK zu installieren muß man unter `/usr/src` ein Verzeichnis mit `fwtk-2.0` anlegen. Danach das Archiv `fwtk-2.0.tar.gz` dort hin kopieren und mit tar auspacken.

Der FWTK erlaubt keine SSL Web Dokumente aber es ist ein Zusatzmodul von Jean-Christophe Touvet entwickelt worden. Erhältlich unter

```
ftp.edelweb.fr:/pub/contrib/fwtk/ssl-gw.tar.Z
```

Touvet gibt aber keine Unterstüzung zu dem Modul.

Es gibt aber eine modifizierte Version von Eric Wedel das Zugriff zu Netscapes »Secure News Server« erlaubt. Erhältlich unter

```
mdi.meridian-data.com:/pub/tis.fwtl/ssl-gw/ssl-gw2.tar.Z
```

In diesem Beispiel wird Eric Wedel's Version verwendet.

Um es zu installieren, einfach im `/usr/src/fwtk-2.0` Verzeichnis einen Ordner `ssl-gw` anlegen und die Dateien dort hinkopieren. Es braucht dann noch einige Änderungen damit es mit dem FWTK kompiliert.

Die erste Änderung ist in der `ssl-gw.c` Datei, dort fehlt noch ein Include:

```
#if defined(__linux)
#include        <sys/ioctl.h>
#endif
```

Zweitens fehlt noch ein Makefile. Man kann eines von den anderen Gateway-Verzeichnissen kopieren und den original Gateway-Namen mit dem des ssl-gw überschreiben.

9.2 Kompilieren des TIS FWTK

Die Version 2.0 des FWTK kompiliert viel einfacher als die älteren Versionen. Es müssen aber noch einige Änderungen gemacht werden bis die Beta-Version sauber kompiliert.

Zu Beginn muß im `/usr/src/fwtk/fwtk` Verzeichnis das `Makefile.config.linux` über das originale `Makefile.config` kopiert werden.

NICHT FIXMAKE BENÜTZEN. In der Anleitung steht es zwar, es zerstört aber die Makefile's in jedem Verzeichnis.

Das sed-Skript fügt ein '.' und " zu der include-Zeile in jedem Makefile hinzu. Dieses sed-Skript funktioniert:

```
sed 's/^include[        ]*\([^  ].*\)/include \1/' $name .proto > $name
```

Im Makefile.config müssen noch zwei Änderungen gemacht werden.

Der Autor nahm sein Home-Verzeichnis als Source-Verzeichnis. Hier wird aber in `/usr/src` gearbeitet. Aus diesem Grund muß die FWTKSRCDIR Variable umgesetzt werden.

```
FWTKSRCDIR=/usr/src/fwtk/fwtk
```

Weiterhin benützen einige Linux-Systeme die gdbm Datenbank. Im `Makefile.config` steht dbm und muß in diesem Fall auch umgesetzt werden:

```
DBMLIB=-lgdbm
```

Die letzte Änderung ist im `x-gw`. Der Fehler in der BETA-Version ist in der `socket.c`. Um ihn zu beheben müssen folgende Zeilen gelöscht werden:

```
#ifdef SCM_RIGHTS  /* 4.3BSD Reno and later */
                    + sizeof(un_name->sun_len) + 1
#endif
```

Wenn das ssl-gw benützt wird, muß man das Verzeichnis mit ins Makefile aufnehmen:

```
DIRS=   smap smapd netacl plug-gw ftp-gw tn-gw rlogin-gw http-gw x-gw ssl-gw
```

Jetzt kann man **make** starten.

9.3 Installation des TIS FWTK

Jetzt beginnt der Spaß erst richtig! Man muß jedem System mitteilen, daß es die neuen Dienste nutzen soll und es müssen Tabellen erstellt werden, um dies zu kontrollieren.

Nun ein Beispiel für Einstellungen die erprobt sind. Danach noch Beschreibungen der Probleme die dabei auftreten können und wie man sie beseitigt.

Es gibt drei Dateien die die Kontrolltabellen enthalten:

- `/etc/services`
  - Die Information für das System welcher Dienst auf welchem Port liegt
- `/etc/inetd.conf`
  - Teilt dem inetd mit welches Programm zu starten ist wenn eine
  - Verbindung an einem Port ansteht
- `/usr/local/etc/netperm-table`
  - Teilt den FWTK Diensten mit, wem welcher Dienst erlaubt wird oder nicht.

Um die Funktion des FWTK zu gewährleisten müssen diese Dateien von Grund auf neu erstellt werden. Ein falsche Einstellung in den drei Dateien kann das Netzwerk unerreichbar machen.

9.3.1 Die netperm-table Datei

Diese Datei regelt wer Zugriff auf die Dienste des TIS FWTK hat. Man soll auch den Datenverkehr zu beiden Seiten des Firewalls im Auge behalten. Der Authentifizierungsabschnitt der netperm-table zeigt wo die Datenbank zu finden ist und wer Zugriff darauf hat.

```
#
# Proxy configuration table
#
# Authentication server and client rules
authsrv:      database /usr/local/etc/fw-authdb
authsrv:      permit-hosts localhost
authsrv:      badsleep 1200
authsrv:      nobogus true
# Client Applications using the Authentication server
*:            authserver 127.0.0.1 114
```

Mit der permit-hosts Zeile kann es Probleme beim sperren des Zugriffs auf diesen Dienst geben. Eine mögliche Lösung des Problems wäre folgende Zeile: `authsrv:  permit-hosts *`

Um die Datenbank zu initialisieren muß man als root das Script `./authsrv` in `/var/local/etc` starten um den »administrative user record« zu ertellen.

In der FWTK-Dokumentation ist beschrieben wie man neue User und Groups hinzufügt.

Ein Beispiel der Vorgehensweise:

```
#
# authsrv
authsrv# list
authsrv# adduser admin "Auth DB admin"
ok - user added initially disabled
authsrv# ena admin
enabled
authsrv# proto admin pass
changed
authsrv# pass admin "plugh"
Password changed.
authsrv# superwiz admin
set wizard
authsrv# list
Report for users in database
user   group  longname           ok?    proto   last
------ ------ ------------------ -----  ------  -----
admin         Auth DB admin      ena    passw   never
authsrv# display admin
Report for user admin (Auth DB admin)
Authentication protocol: password
Flags: WIZARD
authsrv# ^D
EOT
#
```

Die telnet-gateway (tn-gw) controls sind in einer Reihenfolge und der Erste ist zu konfigurieren.

Im folgenden Beispiel ist es den Systemen im LAN erlaubt ohne Authentifikation den Firewall zu passieren. (permit-hosts 19961.2.* -passok)

Einem anderem System (196.1.2.202) wird der Zugang direkt zum Firewall gestattet ohne gleich durchgereicht zu werden. Die netacl-in.telnetd Zeile erlaubt dies.

Der Telnet timeout sollte kurz gehalten werden.

```
# telnet gateway rules:
tn-gw:              denial-msg      /usr/local/etc/tn-deny.txt
tn-gw:              welcome-msg     /usr/local/etc/tn-welcome.txt
tn-gw:              help-msg        /usr/local/etc/tn-help.txt
tn-gw:              timeout 90
tn-gw:              permit-hosts 196.1.2.* -passok -xok
tn-gw:              permit-hosts * -auth
# Only the Administrator can telnet directly to the Firewall via Port 24
netacl-in.telnetd: permit-hosts 196.1.2.202 -exec /usr/sbin/in.telnetd
```

Die r-commands arbeiten in der selben Weise wie der telnet.

```
# rlogin gateway rules:
rlogin-gw:    denial-msg      /usr/local/etc/rlogin-deny.txt
rlogin-gw:    welcome-msg     /usr/local/etc/rlogin-welcome.txt
rlogin-gw:    help-msg        /usr/local/etc/rlogin-help.txt
rlogin-gw:    timeout 90
rlogin-gw:    permit-hosts 196.1.2.* -passok -xok
rlogin-gw:    permit-hosts * -auth -xok
# Only the Administrator can telnet directly to the Firewall via Port
netacl-rlogind: permit-hosts 196.1.2.202 -exec /usr/libexec/rlogind -a
```

Es sollte niemand Zugriff direkt auf den Firewall haben, wie es bei FTP der Fall ist. Somit darf kein FTP-Server auf dem Firewall installiert sein.

Die permit-hosts Zeile erlaubt jedem im LAN den freien Zugriff ins Internet und alle anderen müssen sich authentisieren. In den controls ist logging von jeder Datei, welche gesendet oder empfangen wird, integriert (-log { retr stor }).

Der ftp timeout reguliert wie lange es dauert bis eine schlechte Verbindung beendet wird oder wie lange eine Verbindung ohne Aktivität gehalten wird.

```
# ftp gateway rules:
ftp-gw:             denial-msg      /usr/local/etc/ftp-deny.txt
ftp-gw:             welcome-msg     /usr/local/etc/ftp-welcome.txt
ftp-gw:             help-msg        /usr/local/etc/ftp-help.txt
ftp-gw:             timeout 300
ftp-gw:             permit-hosts 196.1.2.* -log { retr stor }
ftp-gw:             permit-hosts * -authall -log { retr stor }
```

FTP-Verbindungen mittels Web, Gopher oder Browser werden durch den http-gw umgesetzt. Die ersten beiden Zeilen im folgenden Beispiel erzeugen einen Ordner in dem die ftp-Dateien und Web-Dokumente, die den Firewall passieren, gespeichert werden. Der Datei- und Ordner-Zugriff sollte nur root gestattet sein.

Die Web-Verbindung sollte kurz gehalten werden. Dies steuert, wie lange der Anwender bei einer schlechten Verbindung warten muß.

```
# www and gopher gateway rules:
http-gw:      userid          root
http-gw:      directory       /jail
http-gw:      timeout 90
http-gw:      default-httpd   www.afs.net
http-gw:      hosts           196.1.2.* -log { read write ftp }
http-gw:      deny-hosts      *
```

Der ssl-gw ist ein Gateway der alles durchläßt, aus diesem Grund wird zur Vorsicht geraten. Im folgenden Beispiel wird jedem Anwender innerhalb des LANs erlaubt eine Vebindung zu allen Servern außerhalb erlaubt außer zu den Adressen 127.0.0.* und 192.1.1.* und auch nur auf den Ports 443 und 563. Diese Ports sind bekannte SSL-Ports.

```
# ssl gateway rules:
ssl-gw:         timeout 300
ssl-gw:         hosts           196.1.2.* -dest { !127.0.0.* !192.1.1.* *:443:563 }
ssl-gw:         deny-hosts      *
```

Im folgendem Beispiel ist die Benützung des plug-gw beschrieben um Verbindungen zu einem News-Server zu erlauben. In diesem Beispiel ist jedem Anwender im LAN nur die Verbindung zu einem System erlaubt und auch nur zum News-Port.

Die zweite Zeile erlaubt dem Newsserver die Daten zurück zum LAN passieren zu lassen.

Da die meisten News-Clients die Verbindung während dem News-Lesens bestehen lassen ist ein längerer timeout für den Newsserver zu wählen.

```
# NetNews Pluged gateway
plug-gw:        timeout 3600
plug-gw: port nntp 196.1.2.* -plug-to 199.5.175.22 -port nntp
plug-gw: port nntp 199.5.175.22 -plug-to 196.1.2.* -port nntp
```

Der finger-gateway ist sehr einfach. Jeder Anwender innerhalb des LANs muß sich in den Firewall einloggen um den dortigen finger zu verwenden. Jeder andere bekommt einfach eine Meldung.

```
# Enable finger service
netacl-fingerd: permit-hosts 196.1.2.* -exec /usr/libexec/fingerd
netacl-fingerd: permit-hosts * -exec /bin/cat /usr/local/etc/finger.txt
```

9.3.2 Die inetd.conf Datei

Das folgende Beispiel ist eine komplette `/etc/inetd.conf` Datei. Alle unbenötigten Dienste sind auskommentiert. Es ist aus dem Grund komplett um zu zeigen was nicht benötigt wird und wie die neuen Dienste eingefügt werden.

```
#echo stream  tcp  nowait  root       internal
#echo dgram   udp  wait    root       internal
```

```
  #discard      stream  tcp  nowait  root       internal
  #discard      dgram   udp  wait    root       internal
  #daytime      stream  tcp  nowait  root       internal
  #daytime      dgram   udp  wait    root       internal
  #chargen      stream  tcp  nowait  root       internal
  #chargen      dgram   udp  wait    root       internal
  # FTP firewall gateway
  ftp-gw      stream  tcp  nowait.400  root  /usr/local/etc/ftp-gw ftp-gw
  # Telnet firewall gateway
  telnet        stream  tcp  nowait      root  /usr/local/etc/tn-
gw /usr/local/etc/tn-gw
  # local telnet services
  telnet-a    stream  tcp  nowait      root  /usr/local/etc/netacl in.telnetd
  # Gopher firewall gateway
  gopher        stream  tcp  nowait.400  root /usr/local/etc/http-
gw /usr/local/etc/http-gw
  # WWW firewall gateway
  http  stream  tcp  nowait.400  root  /usr/local/etc/http-
gw /usr/local/etc/http-gw
  # SSL firewall gateway
  ssl-gw  stream  tcp     nowait  root /usr/local/etc/ssl-gw ssl-gw
  # NetNews firewall proxy (using plug-gw)
  nntp    stream  tcp     nowait  root    /usr/local/etc/plug-gw plug-gw nntp
  #nntp stream  tcp     nowait  root    /usr/sbin/tcpd  in.nntpd
  # SMTP (email) firewall gateway
  #smtp stream  tcp     nowait  root    /usr/local/etc/smap smap
  #
  # Shell, login, exec and talk are BSD protocols.
  #
  #shell        stream  tcp     nowait  root    /usr/sbin/tcpd in.rshd
  #login        stream  tcp     nowait  root    /usr/sbin/tcpd in.rlogind
  #exec stream  tcp     nowait  root    /usr/sbin/tcpd  in.rexecd
  #talk dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
  #ntalk        dgram   udp     wait    root    /usr/sbin/tcpd in.ntalkd
  #dtalk        stream  tcp     waut    nobody  /usr/sbin/tcpd in.dtalkd
  #
  # Pop and imap mail services et al
  #
  #pop-2   stream  tcp  nowait  root  /usr/sbin/tcpd    ipop2d
  #pop-3   stream  tcp  nowait  root  /usr/sbin/tcpd    ipop3d
  #imap    stream  tcp  nowait  root  /usr/sbin/tcpd    imapd
  #
  # The Internet UUCP service.
  #
  #uucp    stream  tcp  nowait  uucp  /usr/sbin/tcpd /usr/lib/uucp/uucico -l
  #
  # Tftp service is provided primarily for booting.  Most sites
  # run this only on machines acting as "boot servers." Do not uncomment
  # this unless you *need* it.
  #
```

```
  #tftp dgram   udp     wait    root    /usr/sbin/tcpd  in.tftpd
  #bootps       dgram   udp     wait    root    /usr/sbin/tcpd bootpd
  #
  # Finger, systat and netstat give out user information which may be
  # valuable to potential "system crackers."  Many sites choose to disable
  # some or all of these services to improve security.
  #
  # cfinger is for GNU finger, which is currently not in use in RHS Linux
  #
  finger        stream  tcp  nowait  root   /usr/sbin/tcpd in.fingerd
  #cfinger      stream  tcp  nowait  root   /usr/sbin/tcpd in.cfingerd
  #systat       stream  tcp  nowait  guest  /usr/sbin/tcpd  /bin/ps -auwwx
  #netstat      stream  tcp  nowait  guest  /usr/sbin/tcpd /bin/netstat -
f inet
  #
  # Time service is used for clock syncronization.
  #
  #time stream  tcp  nowait  root  /usr/sbin/tcpd  in.timed
  #time dgram   udp  wait    root  /usr/sbin/tcpd  in.timed
  #
  # Authentication
  #
  auth          stream  tcp  wait    root  /usr/sbin/tcpd  in.identd -w -t120
  authsrv       stream  tcp  nowait  root  /usr/local/etc/authsrv authsrv
  #
  # End of inetd.conf
```

9.3.3 Die /etc/services Datei

Hier ist aller Anfang. Ein Client baut die Verbindung zu einem Firewall immer zu einem bekannten Port auf (unterhalb 1024), zum Beispiel der telnet auf Port 23. Der inet-Dämon bemerkt die Verbindung und sucht den Dienst in der /etc/services Datei. Danach startet er das Programm, welches in der /etc/inetd.conf Datei dem Dienst zugewiesen ist.

Einige der Dienste, die hier erstellt werden, sind üblicherweise nicht in der `/etc/services` Datei. Somit kann man einigen davon einen Port eigener Wahl zuweisen. Zum Beispiel kann man den telnet-Port vom Administrator (telnet-a) dem Port 24 zuweisen. Man kann ihn aber auch dem Port 2323 zuweisen, je nach Belieben. Damit sich der Administrator direkt zum Firewall verbinden kann muß er einen telnet zum Port 24, und nicht zum Port 23, starten und wenn die netperm-table richtig aufgesetzt ist kann man dies nur von einem Systems innerhalb des geschützten Netzwerkes tun.

```
  telnet-a        24/tcp
  ftp-gw          21/tcp                # this named changed
  auth            113/tcp   ident       # User Verification
  ssl-gw          443/tcp
```

10 Der SOCKS Proxy Server

10.1 Installation des Proxy Servers

Der SOCKS Proxy Server ist hier erhältlich:

```
metalab.unc.edu:/pub/Linux/system/Network/misc/socks-linux-src.tgz
```

Im selben Verzeichnis ist auch eine Beispiel-Konfigurationsdatei mit folgendem Namen `socks-conf`. Mit gzip und tar das Archiv auspacken und den Anweisungen im README folgen. Manche hatten Probleme beim kompilieren, einfach aufpassen ob die Makefiles passen.

Unbedingt zu beachten ist, daß der Proxy Server in der `/etc/inet.d` Datei eingetragen wird:

```
socks  stream  tcp  nowait  nobody  /usr/local/etc/sockd  sockd
```

Damit wird der Server bei Anforderung gestartet.

10.2 Konfiguration des Proxy Servers

Das SOCKS Programm benötigt zwei verschiedene Konfigurationsdateien. Eine um den Zugriff zu erlauben (access file) und eine um die Anforderungen zum betreffenden Proxy Server zu leiten (routing file). Die Access Datei soll im Server installiert sein und die Routing Datei auf jedem Unix-Rechner. DOS-Rechner und MacIntosh-Rechner machen ihr eigenes Routing.

10.2.1 Die Access Datei

Mit SOCKS 4.2 Beta wird die Access Datei `sockd.conf` genannt. Diese Datei soll zwei Zeilen enthalten: eine Erlaubnis- (permit) und eine Ablehnungs- (deny) Zeile.

Jede Zeile hat drei Einträge.

- Den Bezeichner (identifier, permit/deny)
- Die IP Adresse (address)
- Den Adressen Modifizierer (modifier)

Der Bezeichner ist entweder permit oder deny. Man soll beides haben, eine permit- und eine deny-Zeile.

Die IP-Adresse enthält eine 4 Byte Adresse in typischer IP-Notation, z.B. 192.168.2.0.

Der Adress-Modifizierer ist ebenso eine typische 4 Byte IP-Adresse. Sie arbeitet wie eine Netzmaske. Gehen wir davon aus, daß diese Zahl 32 Bit (Einsen und Nullen) entspricht. Ist das Bit eine Eins muß das entsprechende Bit der zu prüfenden Adresse zum entsprechenden Bit des IP-Adreß-Feldes passen.

Wenn zum Beispiel die Zeile so lautet:

```
permit 192.168.2.23 255.255.255.255
```

werden nur die IP-Adressen zugelassen, bei denen jedes einzelne Bit passen muß. Bei 192.168.2.23 eben nur 192.168.2.23.

Bei folgender Zeile

```
permit 192.168.2.0 255.255.255.0
```

wird jede Nummer innerhalb der Gruppe von 192.168.2.0 bis 192.168.2.255 zugelassen, eben eines kompletten Klasse-C Bereiches.

Folgende Zeile sollte man niemals zulassen:

```
permit 192.168.2.0 0.0.0.0
```

Damit bekommt ohne Rücksicht jeder die Zulassung.

Aus diesem Grund wird nur jede Adresse durchgelassen die eine Erlaubnis hat, der Rest wird einfach abgewiesen. Damit jeder im Bereich 192.168.2.xxx zugelassen wird sind folgende Zeilen passend:

```
permit 192.168.2.0 255.255.255.0
deny 0.0.0.0 0.0.0.0
```

Wenn man das erste »0.0.0.0« in der deny-Zeile betrachtet, ist zu beachten, daß bei einem Modifizierer von 0.0.0.0 die IP-Adresse nicht mehr wichtig ist. In diesem Fall ist 0.0.0.0 der Standard, da es einfacher zu tippen ist.

Es ist auch mehr als eine Zeile mit den beiden Möglichkeiten erlaubt.

Speziellen Anwendern kann der Zugriff erlaubt oder auch abgelehnt werden. Dies geschieht mit der ident Authentifizierung. Nicht alle Systeme unterstützen ident, unter anderem Trumpet Winsock. Aus diesem Grund wird hier nicht näher darauf eingegangen. Die Dokumentation zu SOCKS bietet hier selbst gute Unterstützung.

10.2.2 Die Routing Datei

Die Routing Datei in SOCKS wird leider `socks.conf` genannt. Warum wird sie »leider« so genannt? Sie unterscheidet sich kaum im Namen von der Access Datei und ist somit schnell verwechselt.

Die Aufgabe der Routing Datei ist den SOCKS-Clients mitzuteilen wann SOCKS zu benützen ist und wann nicht. Zum Beispiel wird bei einer Verbindung von 192.168.2.3 zu 192.168.2.1 der SOCKS Firewall nicht benützt, es wird direkt über Ethernet verbunden. Der Loopback, 127.0.0.1, wird automatisch definiert. Außerdem benötigt man SOCKS bei einer Verbindung zu sich selber auch nicht. Es gibt drei Einträge:

- deny
- direct
- sockd

Mit deny wird SOCKS mitgeteilt wann eine Anforderung abgewiesen wird. Dieser Eintrag hat die selben drei Felder wie in `sockd.conf`: identifier, address und modifier. Da dies auch von `sockd.conf`, der Access-Datei, verwaltet wird, kann das modifier Feld 0.0.0.0 bleiben. Um sich selbst davon auszuschließen irgendeinen Rechner zu erreichen, kann man es hier tun.

Der direct Eintrag nennt die Adressen die SOCKS nicht benötigen. Das sind alle Adressen die ohne den Proxy Server erreicht werden können.

Im folgenden Beispiel

```
direct 192.168.2.0 255.255.255.0
```

kann jeder direkt den anderen rechner im geschützten Netzwerk erreichen.

Der folgende sockd - Eintrag zeigt dem Computer welcher Host den SOCKS Server Dämon installiert hat:

```
sockd @=<serverlist> <IP address> <modifier>
```

Zu beachten ist der @= Eintrag. Dies erlaubt die IP-Adressen einer Liste von Proxy-Servern einzutragen. In diesem Beispiel wird aber nur ein Proxy-Server verwendet. Man kann aber mehrere haben um die Last zu verteilen oder eine Sicherheits-Redundanz zu haben.

Das IP-Adreß und modifier Feld hat die selbe Form wie im vorigen Beispiel. Es müssen die Adressen bekannt gemacht werden die durch den Proxy-Server dürfen.

10.2.3 DNS hinterhalb des Firewalls

Einen Domain Name Service hinterhalb des Firewalls zu installieren ist eine einfache Aufgabe. Man muß lediglich den DNS auf dem Firewall-Computer installieren und alle Rechner hinterhalb des Firewalls diesen DNS benutzen lassen.

10.3 Arbeiten mit einem Proxy Server

10.3.1 Unix

Damit die Applikationen mit dem Proxy-Server arbeiten, müssen sie »SOCKSifiziert« werden. Man benötigt zwei verschiedene telnet, einen für direkte Kommunikation und einen für die Kommunikation mit dem Proxy-Server. Socks enthält eine Anleitung wie man ein Programm SOCKSifiziert und ein paar fertigen SOCKSifizierten Programmen. Wenn man eine SOCKSifizierte Version eines Programmes verwendet um innerhalb des LANs eine Verbindung aufzubauen, lenkt SOCKS automatisch zur Version für direkte Kommunikation um. Aus diesem Grund werden alle Programm im Netzwerk umbenannt und durch SOCKSifizierte Versionen ersetzt. `finger` wird zu `finger.orig`, `telnet` wird zu `telnet.orig` usw. Diese Änderungen müssen in die `include/socks.h` eingetragen werden.

Manche Programmen führen das Routing und die SOCKSifizierung selber durch, wie z.B. NetScape. Man kann mit NetScape einen Proxy-Server verwenden indem man die IP-Adresse des Servers in dem entsprechenden Feld der Netzwerkeinstellungen einträgt (in diesem Beispiel 192.168.2.1). Jede Applikation benötigt eine gewisse Umstellung, unabhängig davon wie es mit einen Proxy Server funktioniert.

10.3.2 MS Windows mit Trumpet Winsock

Trumpet Winsock hat schon eingebaute Proxy-Server Möglichkeiten. Im »setup« Menü gibt man die IP-Adresse des Servers ein, ebenso alle Adressen der Rechner, die man direkt erreichen kann. Trumpet leitet dann alle Pakete weiter.

10.3.3 Konfiguration des Proxy Servers zur Unterstützung von UDP-Paketen

Das SOCKS-Paket arbeitet nur mit TCP-Paketen, nicht mit UDP-Paketen. Das schränkt ein klein wenig den Nutzen ein. Viele nützliche Programme, wie talk und archie, benützen UDP. Es gibt ein Paket das von Tom Fitzgerald (`fitz@wang.com`) entwickelt wurde, um wie ein Proxy Server für UDP-Pakete zu funktionieren, genannt UDPrelay. Leider ist es zur Zeit nicht kompatibel zu Linux.

10.4 Nachteile mit Proxy Servern

Ein Proxy Server ist vor allem eine *Sicherheits Einheit*. Die Benutzung, um den Internetzugang mit limitierten IP-Adressen zu erweitern, hat viele Nachteile. Ein Proxy Server bietet guten Zugriff von innerhalb des geschützten

Netzwerkes nach außen und läßt das LAN für Anwender von Außen komplett unerreichbar sein. Das bedeutet keine Server, Archie, Talk Verbindungen oder direktes Mailing zu den inneren Computern. Diese Nachteile können schwerwiegend sein, aber man soll folgendes beachten:

- Sie bearbeiten gerade einen Bericht auf einem Computer innerhalb des geschützten Netzwerkes. Zuhause kommt der Wunsch auf, den Bericht nochmals zu überarbeiten. Sie haben aber keinen Zugriff auf Ihren Bürocomputer, da er hinter einer Firewall ist. Sie versuchen eine Verbindung zum *Firewall* aufzubauen, aber es wurde versäumt Ihnen einen Zugang zum Firewall einzurichten.
- Ihre Tochter geht zur Uni. Sie wollen ihr eine E-Mail schicken. Sie haben wirklich private Themen mit ihr zu besprechen und wollen die Antwort direkt auf Ihren Computer gesendet haben. Sie vertrauen dem System-Administrator aber es ist eben private Mail.
- Die fehlende Möglichkeit UDP zu benützen, zeigt einen großen Nachteil von Proxy-Servern auf. Der Einsatz von UDP wird aber wohl bald möglich sein.

FTP bereitet ein weiteres Problem mit einem Proxy-Server. Bei Empfang oder Verwendung von `ls` öffnet der FTP-Server einen Socket auf dem Client-Rechner und reicht die Informationen durch. Ein Proxy-Server wird das nicht erlauben, darum wird FTP nicht sonderlich gut funktionieren.

Aufgrund des größeren Overheads arbeiten Proxy-Server relativ langsam. Die Allgemeinheit ist der Meinung daß sich das zukünftig ändert.

Grundsätzlich, wenn man IP-Adressen hat und die Sicherheit ist nicht das wichtigste, dann sollte man Proxy-Server und Firewalls nicht verwenden. Wenn keine ausreichende Anzahl an IP-Adressen vorhanden ist und die Sicherheit immer noch keine große Rolle spielt, dann ist ein IP-Emulator, wie Term, Slirp oder TIA, eine interessante Wahl.

Term ist erhältlich bei: `metalab.unc.edu` und Slirp bei: `blitzen.canberra.edu.au:/pub/slirp`. TIA gibts bei `marketplace.com`.

Diese Pakete arbeiten schneller, erlauben bessere Verbindungen und bieten einen besseren Zugriff vom Internet ins private Netzwerk. Proxy-Server sind gut für Netzwerke mit vielen Computern die einen einfachen Zugriff zum Internet haben wollen mit nur einer einmaligen Konfiguration und mit wenig Arbeit im laufenden Betrieb.

11 Erweiterte Konfigurationen

Das nächste Beispiel zeigt eine erweiterte Konfiguration die vielen genügen wird und einige Fragen klärt.

11.1 Ein grosses Netzwerk mit Sicherheit als Schwerpunkt

Als Beispiel sei angenommen ein Netzwerk mit 50 Computern und ein Subnetz von 32 IP-Adressen (5 Bits). Benötigt werden unterschiedliche Zugangsstufen und Teile des Netzwerkes sollen vom Rest geschützt sein.

Die Stufen wären:

1. **Die externe Zugangsstufe**. Diese Stufe bekommt jeder zu sehen.
2. **Mitarbeiter**. Dies ist die Stufe für alle die aus der externen Stufe aufgestiegen sind.
3. **Experten**. Hier werden die wichtigsten Daten verwaltet.

11.1.1 Die Netzwerk Konfiguration

Die IP-Adressen sind folgendermaßen eingeordnet:

- Die Adresse 192.168.2.255 (Broadcast Adresse) ist unbenützbar.
- 23 der 32 IP-Adressen werden 23 Computern mit Zugriff zum Internet zugewiesen.
- Eine Adresse wird dem Linux-Computer zugewiesen.
- Eine Adresse wird einem weiteren Linux-Computer im Netzwerk zugewiesen.
- 2 Adressen bekommt der Router
- 4 Adressen bleiben übrig und werden für zukünftige Aufgaben reserviert.
- Die geschützten Netzwerke bekommen beide die Adressen 192.168.2.xxx

Es werden außer dem externen öffentlichen noch 2 separate Netzwerke gebildet, jedes in verschiedenen Räumen. Sie werden mit Ethernet verbunden.

Diese Netzwerke werden jeweils an einen Linux-Computer mit eigener IP-Adresse angebunden.

Ein Daten-Server wird mit beiden Netzwerke vebunden, damit einige aus der restlichen Mitarbeiter-Gruppe Zugriff auf die wichtigen Daten haben. Der Daten-Server hat die IP-Adressen 192.168.2.17 für das Mitarbeiter-Netzwerk und 192.168.2.23 für das Experten-Netzwerk. Er hat 2 IP-Adressen weil er 2 Ethernetkarten hat, eine jeweils für das Mitarbeiter- und das Experten-Netzwerk. IP-Forwarding ist ausgeschaltet.

IP-Forwarding ist auch an den Linux-Computern ausgeschaltet. Der Router wird keine IP-Pakete für 192.168.2.xxx weiterleiten, solange ihm das nicht erlaubt wird, somit kann niemand aus dem Internet rein. Der Grund für das Ausschalten von IP-Forwarding besteht darin, daß keine Daten-Pakete aus dem Mitarbeiter-Netzwerk in das Experten-Netzwerk geraten und ebenso umgekehrt.

Der Daten-Server kann so konfiguriert werden um beide Netzwerke mit unterschiedlichen Daten zu bedienen. Mit symbolischen Verweisen kann man allgemeine Daten beiden Netzwerken zugänglich machen.

11.1.2 Die Proxy-Konfiguration

Alle 3 Stufen wünschen eine Überwachung des Netzwerkes über eventuelle falsche Absichten. Das externe öffentliche Netzwerk wird direkt mit dem Internet verbunden somit gibt es hier keine Probleme mit dem Proxy-Server. Die Experten- und Mitarbeiter-Netzwerke sind hinter dem Firewall, somit ist es wichtig hier einen Proxy-Server zu installieren.

Beide Netzwerke werden ungefähr gleich konfiguriert. Beide bekommen die selbe IP-Adresse zugewiesen um es ein hier ein wenig interessanter zu machen.

1. Keiner kann den Daten-Server für den Internet-Zugnang benützen. Dies schützt den Daten-Server vor Viren und anderen üblen Dingen und ist somit sehr wichtig.
2. Den Mitarbeitern wird kein Zugriff aufs World Wide Web gestattet.

Die sockd.conf Datei im Mitarbeiter-Linux-Computer hat folgenden Eintrag:

```
deny 192.168.2.17 255.255.255.255
```

Der Experten-Computer folgenden:

```
deny 192.168.2.23 255.255.255.255
```

Der Mitarbeiter-Linux-Computer hat noch diesen Eintrag:

```
deny 0.0.0.0 0.0.0.0 eq 80
```

Dies verbietet den Zugriff aller Computer auf den Port gleich (eq) 80, dem http Port. Es erlaubt jeden anderen Dienst, nur eben nicht den Web-Zugriff.

Die Datei auf beiden Computern hat noch jenen Eintrag:

```
permit 192.168.2.0 255.255.255.0
```

damit allen Computern aus dem 192.168.2.xxx Netzwerk erlaubt wird den Proxy-Server zu benützen außer den schon Abgewiesenen (den Daten-Server und der Web-Zugriff aus dem Mitarbeiter-Netzwerk).

Die Mitarbeiter sockd.conf sieht demnach so aus:

```
deny 192.168.2.17 255.255.255.255
deny 0.0.0.0 0.0.0.0 eq 80
permit 192.168.2.0 255.255.255.0
```

und die Experten sockd.conf so:

```
deny 192.168.2.23 255.255.255.255
permit 192.168.2.0 255.255.255.0
```

Dies sollte alles korrekt konfigurieren. Jedes Netzwerk sollte entsprechend dem Maß an Wichtigkeit isoliert sein.