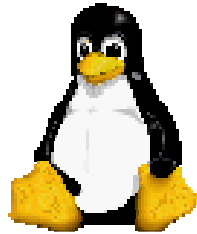


LINUX



KURSSTÜDENKURS 3



- 1 ZEITPLAN:..... 3**

- 2 AUFBAU DER SEMINARTOPOLOGIE..... 4**
 - 2.1 TESTEN DER NETZWERKVERBINDUNGEN: 4
 - 2.1.1 PING..... 4
 - 2.1.2 TRACEROUTE..... 4
 - 2.2 AUSBAU DER KONFIGURATION AUF LAYER 3 – ROUTING:..... 4
 - 2.3 STATISCHE NAMENSAUFLÖSUNG: 5

- 3 WINDOWS-UPDATE DIENST - WSUS 5**

- 4 SPIEGELN VON FTP/HTTP-SERVERN:..... 8**

- 5 PROXY-SERVER: 8**
 - 5.1 FILTERN VON WEBSEITEN: 9
 - 5.1.1 FILTERLÖSUNG DIREKT ÜBER SQUID:..... 9
 - 5.1.2 FILTERN ÜBER EIN EXTERNES PROGRAMM (SQUIDGUARD):..... 9
 - 5.2 TRANSPARENTER PROXY..... 12
 - 5.3 LOGFILES ANALYSIEREN: 12

- 6 FIREWALL – GRUNDLAGEN: 14**
 - 6.1 FIREWALL MIT IPTABLES:..... 15
 - 6.2 BLOCKADE EINZELNER VERBINDUNGEN: 16
 - 6.2.1 ICMP..... 16
 - 6.2.2 VERWENDUNG VON VARIABLEN IN SKRIPTEN 17
 - 6.2.3 PROTOKOLLIEREN VON PAKETEN: 18
 - 6.2.4 ÜBERPRÜFUNG DES VERBINDUNGSZUSTANDES: 18
 - 6.2.5 ZWEI BEISPIELE AUF DER PRAXIS: 18
 - 6.3 INTEGRATION IN DIE STARTSKRIPTEN 20
 - 6.4 MÖGLICHKEITEN DER SUSEFIREWALL2: 22
 - 6.5 ANHANG : MANPAGE ZU IPTABLES 24
 - 6.6 ERGÄNZUNG DER SUSE-FIREWALL: 31

- 7 PAKETSNIFFER 32**

- 8 SICHERE VERBINDUNGEN MIT VPN:..... 33**



1 ZEITPLAN:

Montag 18.02.2008	
09:30 – 10:15	Konfiguration der Seminartopologie Lokalen DHCP-Server einrichten
10:30 – 12:00	Windows – Update Service Spiegeln von FTP/HTTP-Servern
14:00 – 15:30	Konfiguration des Proxyservers SQUID Filtern mit Hilfe von Squidguard
15:45 – 17:15	erweiterte Optionen für den Filter div. Dateitypen filtern,... Analyse der Logfiles
Dienstag 19.02.2008	
08:45 – 10:15	Übersicht: Netzwerk-Layermodell Paketfilter – Grundlagen
10:30 – 12:00	Möglichkeiten des SuSE-Firewallskriptes transparenter Proxy, Portforwarding auf interne Server
13:45 – 16:00	Firewallskripte und Webmin Sniffer-Programme



Anmerkung: für dieses Seminar sollten am Linuxrechner folgende Pakete installiert werden: apache2, squid, squidguard, wireshark (Ethereal), tcpdump, dhcp-server

2 Aufbau der Seminartopologie

Auf den Arbeitsstationen befinden sich 3 VMwareImages:

- OpenSuSE10.3
- WS1
- WS2

Der Linuxrechner wird mit 2 Netzwerkkarten konfiguriert. Die 1. Netzwerkkarte wird via NAT mit dem Wirtssystem verbunden. Die 2. Netzwerkkarte wird gemeinsam mit den 2 Windowsrechnern auf ein virtuelles Netz gelegt und mit IP-Adressen versorgt. (z.B. mittels DHCP). Um den Windowsrechnern Internetzugang zu gewähren wird mittels Firewallkonfiguration eine Masquerade am Linuxrechner aufgesetzt.

2.1 Testen der Netzwerkverbindungen:

Bevor nun versucht wird, weitere Netzwerkdienste aufzubauen, sollte man versuchen, ob die Netzwerkverbindungen auf dieser Ebene bereits funktionieren. Dazu stehen (auf praktisch allen Systemen) zwei Routinen zur Verfügung: (Bevor man jedoch diese Tests durchführt empfiehlt sich eine Kontrolle der Netzwerkeinstellungen, die mit dem Befehl **ifconfig** an der Konsole durchgeführt werden kann.)

2.1.1 PING

ping 131.130.1.11: Schickt Pakete an die Adresse 131.130.1.11 und notiert die Antwortzeiten. Damit sollte man überprüfen, ob man die eingeschalteten Rechner in den Subnetzen, bzw. ob man Rechner im Internet erreichen kann. Sollten hier Fehler sichtbar werden, deren Ursache noch nicht ganz klar ist, kann man weitere Informationen mittels Traceroute erhalten.

2.1.2 TRACEROUTE

traceroute 131.130.1.11: Versucht ebenfalls den Rechner mit der Adresse 131.130.1.11 zu erreichen, wobei auch Informationen über den Weg zu diesem Rechner geliefert werden (Über welche Router). Damit kann man den Weg nachvollziehen und erkennen, bei welchem Router das Problem entsteht.

2.2 Ausbau der Konfiguration auf Layer 3 – Routing:

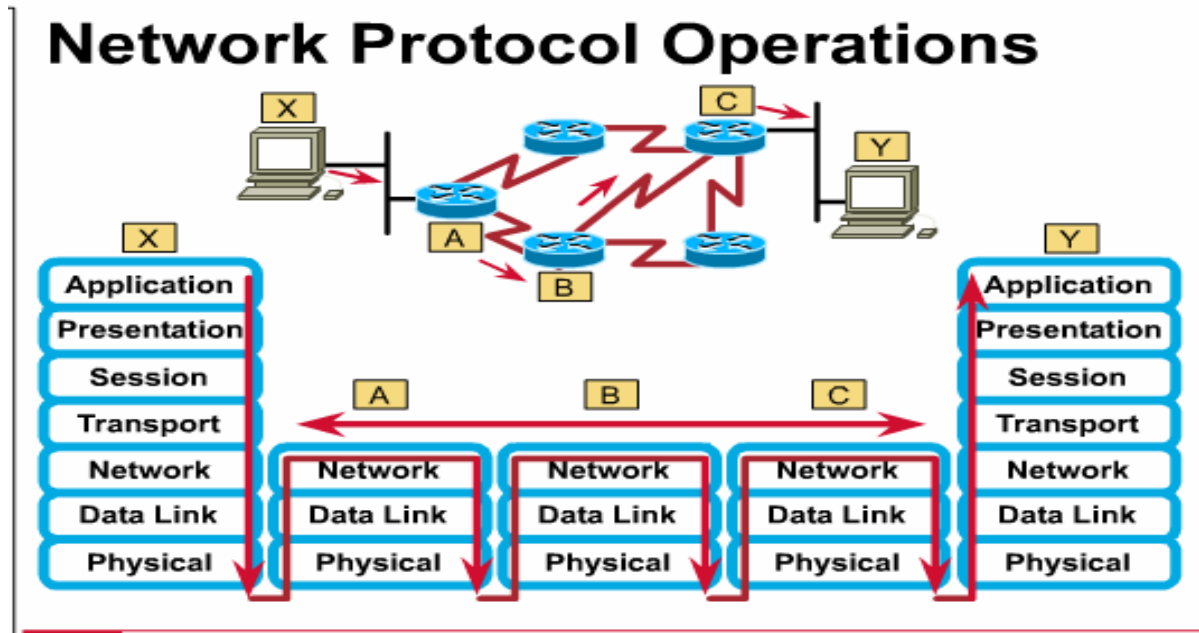
Damit die Rechner nicht nur Arbeitsstationen im eigenen Netz erreichen können, sind auf der sogenannten Netzwerkschicht (OSI – Modell Layer 3) noch einige Parameter anzupassen. Diese Einträge betreffen das Weiterleiten der Datenpakete. Dazu muss die Routingtabelle des Rechners angepasst werden.

Anzeigen der Routinginformationen: **ip route**

Zur Laufzeit kann diese Tabelle mit dem Befehl ip route verändert werden (siehe „man ip route“). Diese Veränderungen sind jedoch nach einem Neustart des Systems nicht mehr vorhanden. Damit diese Einträge dauerhaft gespeichert werden ist ein Eintrag in eine entsprechende Konfigurationsdatei erforderlich. Diese Einträge erstellt man über das YAST – Modul „Konfiguration der Netzwerkkarte“ → Routing.



Für den Router der Gruppe 1 ist z.B. einzutragen dass das Netz 10.0.12.0/255.255.255.0 über die Adresse 10.0.1.202 (externe Adresse des Gruppenrouters 2) zu erreichen ist. Ein Standardeintrag in dieser Tabelle ist das GATEWAY. Diese IP-Adresse ist das Ziel aller Datenpakete, die nicht in eine der anderen Routingeinträge passen.



2.3 Statische Namensauflösung:

Damit die Rechner nun nicht nur mit ihren IP-Adressen sondern auch mit ihren Namen erreicht werden können kann die Datei „etc/hosts“ mit zusätzliche Einträgen versehen werden. Diese Datei ist nicht nur auf Linuxrechnern zu finden. Unter Windows (versionsabhängig) liegt diese Datei z.B. unter c:\winnt\system32\drivers\etc und erfüllt dieselbe Funktion.

Auszug aus einer hosts-Datei:

```
# Zusätzliche Kommentare (so wie in dieser Datei) können in
# einzelnen Zeilen oder hinter dem Computernamen eingefügt werden,
# aber müssen mit dem Zeichen '#' eingegeben werden.
#
# Zum Beispiel:
#
# 102.54.94.97    rhino.acme.com        # Quellserver
# 38.25.63.10   x.acme.com           # x-Clienthost

127.0.0.1      localhost
193.170.207.133 mail.brg-wrn.ac.at
```

3 Windows-Update Dienst - WSUS

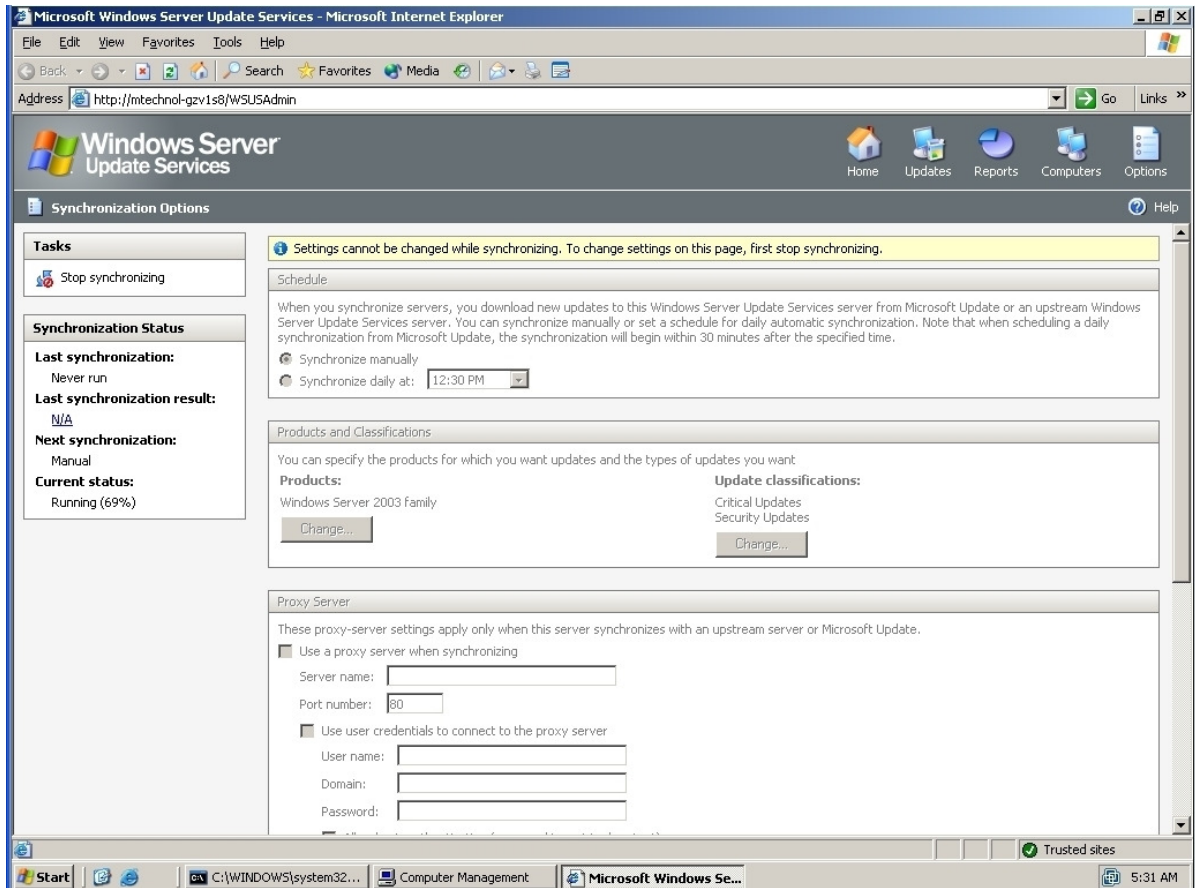
Mit Hilfe des Windows Server Update Dienstes können die Microsoft Updates lokal gespiegelt werden, um dann kontrolliert den lokalen Windows Arbeitsstationen zur Verfügung gestellt zu werden. Die dazu notwendige Software (und auch eventuell notwendige Dokumentationen) können unter:

[http://technet.microsoft.com/de-de/wsus/bb466193\(en-us\).aspx](http://technet.microsoft.com/de-de/wsus/bb466193(en-us).aspx) heruntergeladen werden.



Systemvoraussetzung: Windows 200x Server, BITS, .Net Framework 1.1, Datenbank,

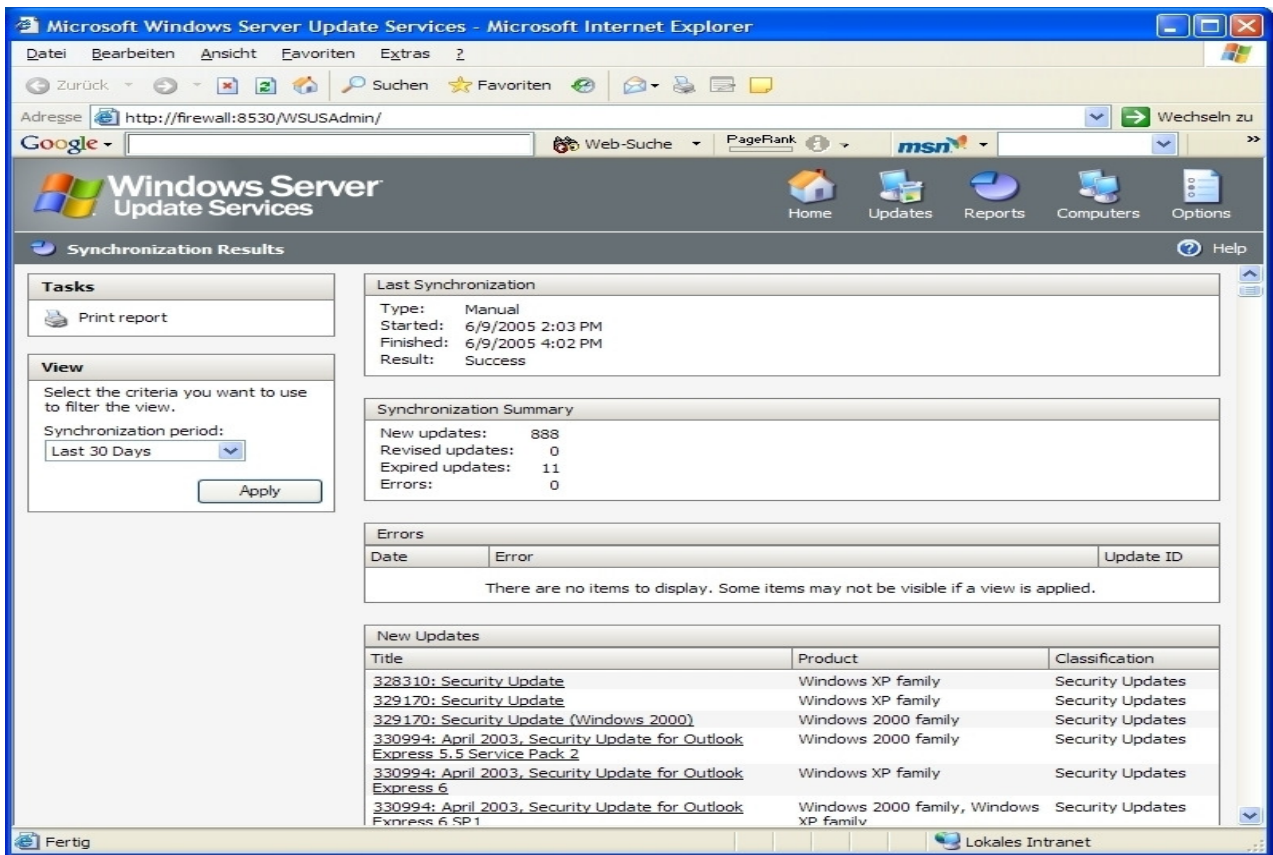
WSUS kümmert sich um aktuelle Windows-Versionen, Office XP, Office 2003, Exchange und den SQL Server



Der Serverdienst wird über ein Webinterface administriert, das man unter <http://UPDATESERVER:8530/WSUSAdmin> erreicht. Der Port kann z.B. einfach auf den Standardwebport (80) verlagert werden.

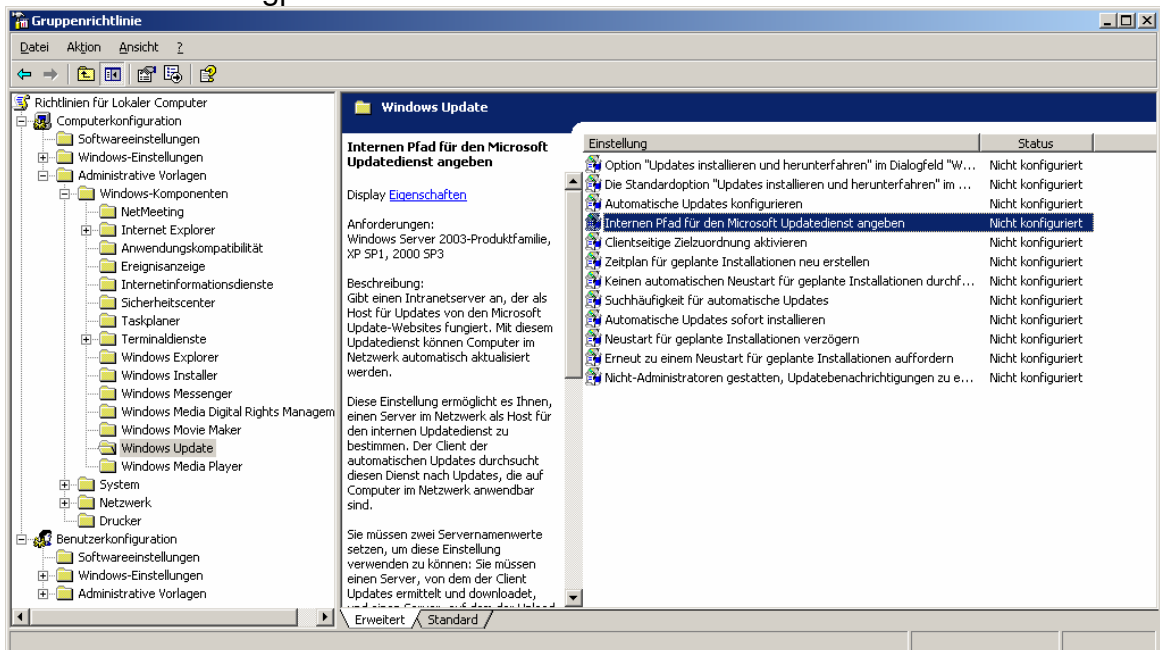
Es wird ein Aktualisierungsintervall eingestellt, in dem der Server sich mit dem Microsoft Updateserver synchronisiert. Die heruntergeladenen Updates können dann zentral vom Administrator den Arbeitsstationen zugewiesen werden.

Über den Updatedienst kann der Administrator auch zentral überprüfen, auf welchem Stand sich die jeweiligen Arbeitsstationen befinden.



Die Arbeitsstationen müssen noch eingestellt werden, sich mit dem lokalen Server (anstelle mit dem MS-Updateserver) zu verbinden. Diese Einstellungen werden über Gruppenpolicies gesetzt. Client-Einstellung:

- Active Directory: Group Policy → wuau.adm (kann auch mit ZEN unter Novell erfolgen)
- Lokal: gpedit.msc



Weitere Information siehe z.B.: <http://www.wsuswiki.com/>



4 Spiegeln von FTP/HTTP-Servern:

Um z.B.: die Updatefiles eines Antivirenprogramms (üblicherweise werden diese per FTP oder http abgeholt) auf einem lokalen Server zur Verfügung zu stellen kann man sich ein Skript erstellen, das wenn es im Verzeichnis /etc/cron.daily abgelegt wird einmal täglich abgearbeitet wird.

SKRIPT:

```
cd ~
wget -r http://speeddownload.nai.com/products/commonupdater
rm -r -f /srv/ftp/pub/CommonUpdater/*
mv ~/speeddownload.nai.com/products/commonupdater/* /srv/ftp/pub/CommonUpdater
```

nun muss nur noch vsftpd so konfiguriert werden, dass /srv/ftp/pub als anonymer FTP-Server zur Verfügung steht. Danach können die Virens Scanner der lokalen Workstations so konfiguriert werden, dass sie ihre Updates vom lokalen Spiegel abholen.

5 Proxy-Server:

Homepage: <http://www.squid-cache.org>

Ein Proxyserver (PROXY = Stellvertreter) richtet Webanfragen stellvertretend für seine Clients ins Internet. Der Vorteil dieser Lösung liegt darin, dass die Seiten aus dem Internet geholt werden und auch lokal (am Proxyserver) gespeichert werden. Ein nochmaliger Aufruf der Seite bewirkt beim Proxyserver nur eine Anfrage ins Internet, ob sich die Seite inzwischen geändert hat. Wenn dies nicht der Fall ist, wird die Seite aus dem lokalen Cache geliefert.

Der Standardproxyserver am Linuxrechner ist SQUID in der Version 2. Für den Start dieses Dienstes existiert in der Datei rc.config wieder eine Variable mit dem Namen START_SQUID (yes/no). Beim erstmaligen Start werden automatisch unter /var/squid die Cacheverzeichnisse angelegt.

Squid operiert standardmäßig am Port 3128 (dies kann jedoch in der Datei /etc/squid.conf auf den Port 8080 geändert werden; Standard bei vielen anderen Proxyserverdiensten). Es ist auch möglich Squid so zu konfigurieren, dass er auf beiden Ports lauscht.

Squid kann für Web-Abfragen (http) und für Downloadprozesse (ftp) als lokaler Zwischenspeicher verwendet werden.

Man muss nur noch bei den Clients (Netscape ' Einstellungen, IE ' Internetoptionen) das Verwenden eines Proxyserver einstellen. Als Proxyserver trägt man hier eine der IP-Adressen des Linuxrechners ein (am Besten die, die von der WS als erstes sichtbar wird). Der Proxyport (falls nicht geändert) ist mit 3128 gegeben. Wählen sie danach eine Internetseite, die noch nicht im lokalen Cache des Rechners ist, von mehreren Workstations, die alle den Proxy verwenden, an. Der Aufbau der ersten Seite ist noch immer so langsam wie vorher. Wenn man jedoch dieselbe Seite von einer anderen Workstation aus aufruft sollte sie deutlich schneller angezeigt werden.

Wenn sie nach der Erstinstallation Squid erstmalig verwenden wollen, werden sie möglicherweise eine Fehlermeldung in der Art "Access denied" erhalten. Der Grund dafür ist, dass die Verwendung in der Datei squid.conf gesperrt ist. Editieren sie diese Datei und suchen sie nach einem Eintrag der Art: "http_access deny all". Ändern sie in diesem Eintrag "deny" auf "allow" und sie sollten über einen verwendbaren Proxyserver verfügen.

(Eine ausführlich dokumentierte Version der Datei Squid.conf finden sie unter <http://www.gymmelk.ac.at/~berx/squid.html>)

Besser: (definiert welche Adressen zur Schule gehören)



```
acl brg src 193.170.207.0/255.255.255.0
acl brg src 10.0.0.0/16
.....
http_access allow brg
```

Die letzte Zeile erlaubt einen Zugriff von Rechnern aus dem Bereich brg. Eine Zeile der Form http_access deny all ist nicht notwendig, da falls die letzte Regel eine allow ist, implizit ein deny all danach angehängt wird.

5.1 Filtern von Webseiten:

5.1.1 Filterlösung direkt über Squid:

(Anleitung unter <http://www.htl-kapfenberg.ac.at/squid/index.html>)

Das von Rainer Grabher unter obiger Adresse angebotene Firewallkonzept kann ab der Version 6.2 nicht mehr verwendet werden (ipfwadm ' ipchains), die Konfiguration von Squid kann jedoch (vor allem seine Sammlung verbotener Adressen) übernommen werden. Die entscheidenden Zeilen, die in die Datei übernommen werden müssen lauten:

```
acl denied_hosts dstdomain "/etc/noaccess"
acl denied_phrases url_regex "/etc/nophrases"
```

```
http_access deny denied_hosts
http_access deny denied_phrases
```

Die ersten beiden Zeilen definieren die Variablen denied_hosts und denied_phrases, wobei der Inhalt aus 2 Dateien geholt wird. Wenn man eine Internetsite sperren will, trägt man die Adresse in /etc/noaccess ein, wenn man einen Begriff sperren will (egal in welcher Adresse er auftaucht) muss man ihn in /etc/nophrases ergänzen.

Die letzten beiden Zeilen verbieten es, dass der Inhalt von denied_hosts, bzw. denied_phrases angezeigt wird. Hierbei ist es entscheidend, dass diese Zeilen vor der allgemeinen Freigabe "http_access allow all" stehen.

Wenn (!!) die Clients nun den Proxyserver verwenden, werden Webzugriffe gefiltert. Es ist jedoch noch immer möglich ohne Proxy zu surfen (Direkte Verbindung ins Internet am Client einstellen). Diese Hintertür kann man nur durch Aktivieren einiger Firewallregeln schließen. (Im Teil 3 wird dafür eine elegante Möglichkeit vorgestellt.)

5.1.2 Filtern über ein externes Programm (SQUIDGUARD):

In den letzten SuSE - Versionen wird noch ein Paket mit der Bezeichnung "SquidGuard" mitgeliefert, das noch mehr Möglichkeiten als die soeben vorgestellte Filterlösung bietet. Dazu muss man das Paket installieren, in der Datei /etc/squid.conf die Zeile "redirect_program /usr/bin/squidguard" einfügen. Damit werden alle angeforderten Webseiten zunächst an das Programm SquidGuard übergeben, in dem dann über die weitere Vorgangsweise entschieden wird. Die Konfiguration dieses Dienstes erfolgt über die entsprechende Datei im Verzeichnis /etc.

In der Datei squidgrd.conf können nun zunächst Benutzergruppen und Ziele im Web (dest) definiert werden. Danach werden Zugriffsrechte (ACL's) festgelegt. Innerhalb dieser Regeln kann nun für jede Gruppe festgelegt werden, welche Zugriffe erlaubt und was verboten ist. Im Falle



eines Verbotes kann mit einer Redirect - Anweisung auf eine HTML-Seite verwiesen werden, die z.B. eines Fehlermeldung ausgibt.

Eine genauere Information findet man auf der Seite: <http://www.squidguard.org> und im Verzeichnis /usr/share/doc/packages/squidgrd.

Die Datei /etc/squidGuard.conf:

```
logdir /var/squidGuard/logs
dbhome /var/squidGuard/db/blacklists

src verwaltung {
    ip 10.1.0.0/24 # range 10.0.0.0 - 10.0.0.255
}

src schule {
    ip 10.0.0.0/16 # range 10.0.0.0 - 10.0.255.255
}

dest ads {
    domainlist ads/domains
    urllist ads/urls
}

dest freigabe {
    domainlist freigabe/domains
    urllist freigabe/urls
}

dest aggressive {
    domainlist aggressive/domains
    urllist aggressive/urls
}

dest av {
    domainlist audio-video/domains
    urllist audio-video/urls
}

dest drugs {
    domainlist drugs/domains
    urllist drugs/urls
}

dest gambling {
    domainlist gambling/domains
    urllist gambling/urls
}

dest hacking {
    domainlist hacking/domains
```



```

    urlist  hacking/urls
}

dest mail {
    domainlist mail/domains
}

dest porn {
    domainlist porn/domains
    urlist  porn/urls
    expressionlist  porn/expressions
}

dest proxy {
    domainlist proxy/domains
    urlist  proxy/urls
}

dest warez {
    domainlist warez/domains
    urlist  warez/urls
}

dest violence {
    domainlist violence/domains
    urlist  violence/urls
    expressionlist  violence/expressions
}

acl {
    verwaltung {
        pass all
    }

    schule {
        pass freigabe !ads !aggressive !av !drugs !gambling !hacking !mail !porn !proxy !violence
!warez all
        redirect http://10.0.0.1/warnung.html
    }

    default {
        pass none
        redirect http://10.0.0.1/warnung2.html
    }
}

```

Die folgende Zeile blockiert z.B. das Herunterladen von Video und Audio Dateien:

```
\. (ra?m|mpe?g?|mov|movie|qt|avi|dif|dvd?|mpv2|mp3) ($|\?)
```

The expressionlist file format is lines with regular expressions as described in regex(5). Of most interest is:



- Matches any single character (use "\." to match a ".").
- [abc] Matches one of the characters (" [abc]" matches a single "a" or "b" or "c").
- [c-g] Matches one of the characters in the range (" [c-g]" matches a single "c" or "d" or "e" or "f" or "g").
 "[a-z0-9]" matches any single letter or digit.
 "[-/.:?]" matches any single "-" or "/" or "." or ":" or "?").
- ? None or one of the preceding ("words?" will match "word" and "words").
 "[abc]?" matches a single "a" or "b" or "c" or nothing (i.e. "").
- * None or more of the preceding ("words*" will match "word", "words" and "wordssssss").
 ".*" will match anything including nothing).
- + One or more of the preceding ("xxx+" will match a sequence of 3 or more "x").
- (expr1|expr2) One of the expressions, which in turn may contain a similar construction ("(foo|bar)" will match "foo" or "bar".
 "(foo|bar)? will match "foo" or "bar" or nothing (i.e. "").
- \$ The end of the line ("(foo|bar)\$" will match "foo" or "bar" only at the end of a line).
- \x Disable the special meaning of x where x is one of the special regex characters
 ".?*() ^\$[]{}\" ("\" will match a single "\", "\\\" a single "\" etc.)

Thus a start to block possible sexual material by expression match could look like:

```
(^[ -\?+=/_]) (bondage|boobs?|busty?|hardcore|porno?|sex|xxx+) ([-\?+=/_]|$)
```

5.2 Transparenter PROXY

Durch eine Anpassung der Firewall kann man aus dem internen Netz nach außen (auf den Port 80) gerichtete Anfragen automatisch an den lokalen Port 3128 (SQUID) umleiten.

Damit der Proxyserver Squid nun als Transparenter Proxy (die Benutzer merken gar nicht, dass sie einen Proxyserver verwenden) agiert, muss in seiner Konfigurationsdatei noch eine kleine Veränderung vorgenommen werden:

Suchen sie in der Datei SQUID.CONF nach dem Abschnitt: HTTPD-ACCELERATOR OPTIONS und belegen sie den Wert folgender Variablen:

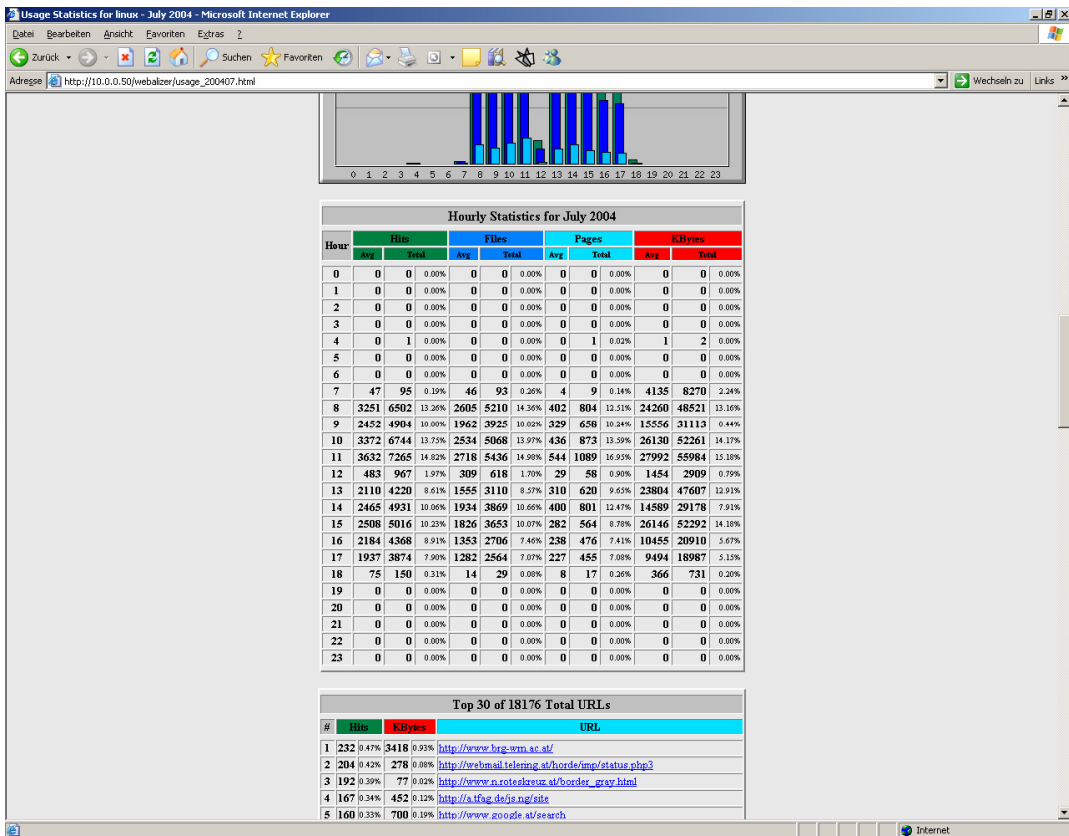
```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Starten sie danach Squid mit "rcsquid reload" neu. Danach sollte jede Workstation aus dem internen Netzwerk automatisch (unabhängig von ihren lokalen Einstellungen) einen Webzugriff über den Proxy vornehmen.

5.3 LogFiles analysieren:

Für Squid gibt es einige Dienstprogramme um die Logfiles (in /var/log/squid) zu analysieren. Bei bedarf kann man natürlich auch mit einem Editor direkt die Zugriffe verfolgen. Zu beachten ist, dass die Zeitangaben UNIX-Timestamps sind und erst in aktuelle Datumswerte umgewandelt werden müssen.

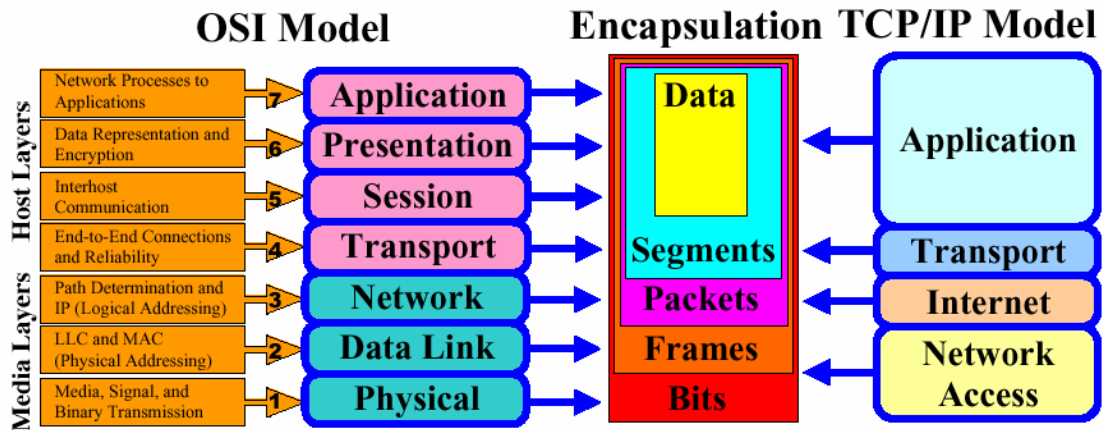
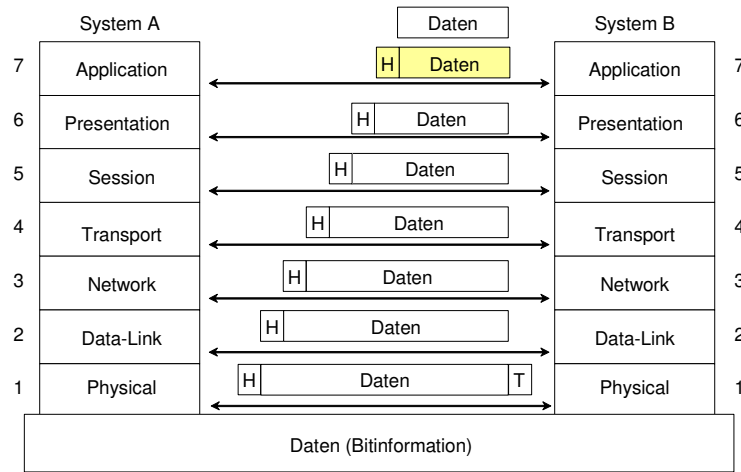
Man kann z.B. WEBALIZER verwenden um aus den LogFiles eine Statistik zu erstellen, die im HTML Format generiert wird. Webalizer wird über /etc/webalizer.conf konfiguriert. Für die Verwendung im Zusammenhang mit SQUID ist es notwendig in die Konfiguration den Parameter „LogType squid“ einzufügen.



Alternativen zu Webalizer sind z.B. Calamaris und SARG. Bei beiden empfiehlt sich die Verwendung von Webmin, da dort entsprechende Module für die Analyse via WEB vorhanden sind. Bei Sarg muss die Webminkonfiguration angepasst werden (Unter SuSE: /usr/share/sarg)



6 FIREWALL – GRUNDLAGEN:



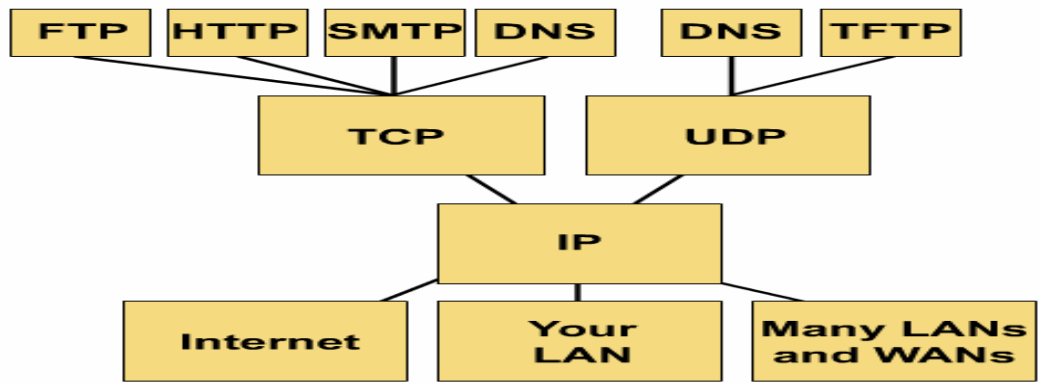
Auszug aus der Datei services:

```
# Diese Datei enthält die Portnummern für bekannte Dienste gemäß IANA.
#
# Format:
# <Dienstname> <Portnummer>/<Protokoll> [Alias...] [#<Kommentar>]
#
echo                7/tcp
echo                7/udp
discard             9/tcp    sink null
discard             9/udp    sink null
sysstat             11/tcp    users                #Active users
sysstat             11/tcp    users                #Active users
daytime             13/tcp
daytime             13/udp
```



gotd	17/tcp	quote	#Quote of the day
gotd	17/udp	quote	#Quote of the day
chargen	19/tcp	ttytst source	#Character generator
chargen	19/udp	ttytst source	#Character generator
ftp-data	20/tcp		#FTP, data
ftp	21/tcp		#FTP. control
telnet	23/tcp		
smtp	25/tcp	mail	#Simple Mail Transfer Protocol
time	37/tcp	timserver	
time	37/udp	timserver	
rlp	39/udp	resource	#Resource Location Protocol
nameserver	42/tcp	name	#Host Name Server
nameserver	42/udp	name	#Host Name Server
nicname	43/tcp	whois	
domain	53/tcp		#Domain Name Server
domain	53/udp		#Domain Name Server
bootps	67/udp	dhcps	#Bootstrap Protocol Server
bootpc	68/udp	dhcps	#Bootstrap Protocol Client
tftp	69/udp		#Trivial File Transfer
gopher	70/tcp		
finger	79/tcp		
http	80/tcp	www www-http	#World Wide Web
kerberos	88/tcp	krb5 kerberos-sec	#Kerberos
kerberos	88/udp	krb5 kerberos-sec	#Kerberos
hostname	101/tcp	hostnames	#NIC Host Name Server
iso-tsap	102/tcp		#ISO-TSAP Class 0
rtelnet	107/tcp		#Remote Telnet Service
pop2	109/tcp	postoffice	#Post Office Protocol - Version 2
pop3	110/tcp		#Post Office Protocol - Version 3

Protocol Graph: TCP/IP



6.1 Firewall mit IPTABLES:

Die Firewallkonfiguration mit iptables verwendet den Befehl iptables (Kommandozeile) um neue Regeln in die Filter einzutragen. Die Regeln sind in 3 Tabellen aufgeteilt:

Filter: Die Standardtabelle, die aus den 3 Ketten (Chains) INPUT, OUTPUT und FORWARD besteht. Die erlaubten Ziele (TARGETS) einer Regel sind ACCEPT, DROP, QUEUE oder RETURN.

Nat: Eine Tabelle die aus 3 Ketten besteht: PREROUTING, POSTROUTING und OUTPUT. Diese Tabelle ist speziell für NAT (Network Address Translation) gedacht.

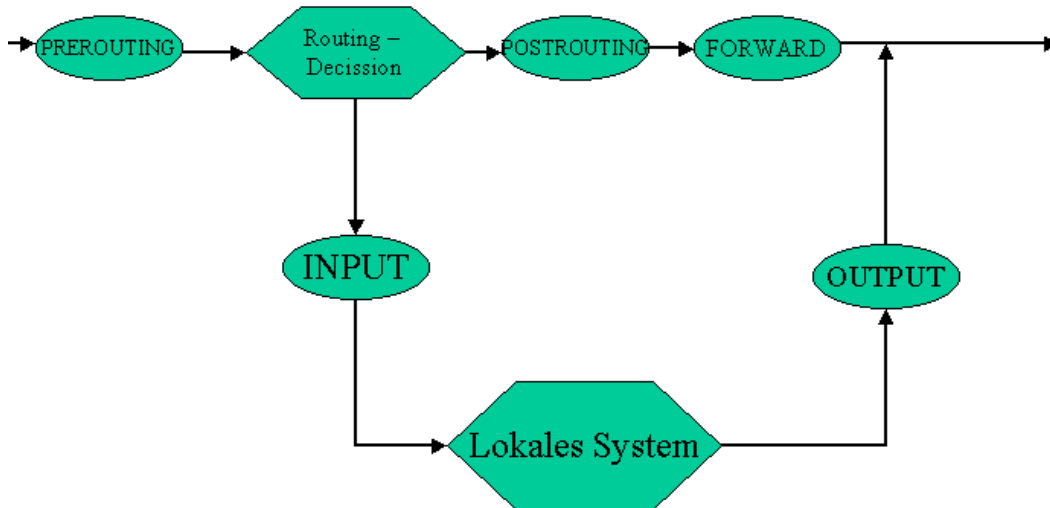
Mangle: Beinhaltet die PREROUTING und POSTROUTING Ketten. Mit dieser Tabelle ist es z.B. möglich einzelne Services gegenüber anderen zu bevorzugen. Nur in dieser



Tabelle kann das TOS (Type of Service) – Feld von Datenpaketen verändert werden. Als Beispiel kann hier dieses Feld auf Minimize-Delay oder Maximize Throughput ,...gestellt werden.

Die Befehle selbst besitzen eine gewisse Ähnlichkeit mit IPCHAINS Befehlen.

DATENFLUSS von Paketen bei Kernelversionen mit IPTABLE-Unterstützung



Über das Tool iptables können nun die Regeln in diesen Kerneltabellen verändert werden. Die wichtigsten Parameter sind:

- A append
- D delete
- I insert
- R replace

6.2 Blockade einzelner Verbindungen:

6.2.1 ICMP

Kontrollroutinen wie ping und traceroute verwenden das icmp Protokoll. Diese Datenpakete können ebenfalls von der Firewall kontrolliert werden. Zu Beachten ist, dass zur Funktion wesentlicher Dienste, Pakete vom Typ source-quench, parameter-problem, destination-unreachable (eingehend), fragmentation-needed (ausgehend) akzeptiert werden.

z.B.:

```
iptables -A INPUT -p icmp -icmp-type xxx -j ACCEPT
```

gültige icmp-Typen sind: (siehe „iptables -p icmp -h“)



```

echo-reply (pong)
destination-unreachable
  network-unreachable
  host-unreachable
  protocol-unreachable
  port-unreachable
  fragmentation-needed
  source-route-failed
  network-unknown
  host-unknown
  network-prohibited
  host-prohibited
  TOS-network-unreachable
  TOS-host-unreachable
  communication-prohibited
  host-precedence-violation
  precedence-cutoff
source-quench
redirect
  network-redirect
  host-redirect
  TOS-network-redirect
  TOS-host-redirect
echo-request (ping)
router-advertisement
router-solicitation
time-exceeded (ttl-exceeded)
  ttl-zero-during-transit
  ttl-zero-during-reassembly
parameter-problem
  ip-header-bad
  required-option-missing
timestamp-request
timestamp-reply
address-mask-request
address-mask-reply

```

6.2.2 Verwendung von Variablen in Skripten

Um nun z.B. mehrere icmp Typen zu erlauben sind einige gleichlautende Zeilen im Firewallskript notwendig. Durch die Verwendung von Variablen und Schleifen kann diese Datei deutlich besser lesbar gehalten werden:

```

CMD=/usr/sbin/iptables
allowed_icmp = "echo-request time-exceeded fragmentation-needed"
for x in $allowed_icmp
do
  $CMD -A INPUT -p icmp -icmp-type $x -j ACCEPT
done

```

Die Definition der notwendigen Variablen für solche Skripte kann auch in einer externen Datei erfolgen (siehe SuSEfirewall2....). Dazu werden die Zeilen

```

CMD=/usr/sbin/iptables
allowed_icmp = "echo-request time-exceeded fragmentation-needed"

```



in z.B. die Datei „firewall.conf“ eingetragen. Im dazugehörigen Skript wird diese Datei zunächst eingelesen und danach die Befehle abgearbeitet:

```
. /etc/firewall.conf
for x in $allowed_icmp
do
    $CMD -A INPUT -p icmp -icmp-type $x -j ACCEPT
done
```

6.2.3 Protokollieren von Paketen:

Pakete, die von der Firewall verworfen werden, können für eine spätere Analyse protokolliert werden. Dafür dient die Option „-j LOG“ die mit dem Parameter „--log-prefix 'Markierung'“ die Pakete auch mit einem persönlichen Kommentar versehen kann. Diese Informationen werden vom SYSLOG in die Datei „/var/log/messages“ geschrieben. Bei intensivem Gebrauch der LOG – Option würde diese Datei jedoch rasch anwachsen.

6.2.4 Überprüfung des Verbindungszustandes:

Wenn in einer Firewall der Zugriff auf einen bestimmten Dienst ermöglicht werden soll, ist darauf zu achten, dass auch die Antwortpakete erlaubt werden. Damit müssen z.B. bei einer bestehenden TELNET-Verbindung die Pakete jeweils die Filterregeln passieren, wodurch es bei einer komplexeren Firewall zu einer Verlangsamung des Datendurchsatzes kommen kann. Über das Modul conntrack (Connection Tracking) kann dieser Zustand verändert werden.

Das Modul ip_conntrack kann mittels „insmod ip_conntrack“ geladen werden. (Eine Liste der ladbaren Netfilter-Module befindet sich im Verzeichnis „/lib/modules/2.4.20-4GB/kernel/ipv4/netfilter“)

```
Pakete die zu bestehenden Verbindungen gehören werden akzeptiert:
iptables -A INPUT -m state - -state ESTABLISHED, RELATED -j ACCEPT
iptables -A OUTPUT -m state - -state ESTABLISHED, RELATED -j ACCEPT
```

Im restlichen Teil der Firewall sind nur noch Regeln für das erste Paket einer Verbindung notwendig:

```
.....
iptables -A INPUT -p tcp - - dport 23 -s 193.170.207.141 -m state - - state NEW -j ACCEPT
.....
```

6.2.5 Zwei Beispiele auf der Praxis:

MASQUERADE:

```
#!/bin/sh
# Masquerade-Skripten für AHS-NÖ-Standardausstattung
# (c) Gerald STACHL 2001
#
# Sollte nur alternativ zur Firewall installiert werden
# Durch dieses Skript werden keine Pakete gefiltert
# Die Variable World_Device wird in rc.config definiert
#
$WORLD_DEV = "eth0"
# Die folgende Zeile ist nur bis 7.3 notwendig
# test "$START_MASQ" = yes || exit 0
```



```
echo -n "Masquerade starten"
```

```
#Alle Regeln löschen
```

```
iptables -F
iptables -t nat -F
```

```
#Policy setzen
```

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

```
#Masquerade definieren
```

```
iptables -t nat -A POSTROUTING -s 10.0.0.0/8 -d ! 10.0.0.0/8 -j SNAT --to $WORLD_DEV
```

```
echo -e $src_done
```

Eine etwas komplexere Konfiguration:

```
#!/bin/sh
# Überarbeitung des Firewallskripts
# G. Stachl 02
#
# allgemeine Variable

CMD="/usr/sbin/iptables"
MDPR="/sbin/modprobe"
modules="_tables table_filter t_LOG t_state table_mangle table_nat _contrack _contrack_ftp _nat_ftp"
allowed_ports_tcp="ftp ftp-data ssh smtp domain bootps bootpc pop3 ntp imap 1024:"
allowed_ports_udp="fsp ftp-data ssh smtp domain bootps bootpc pop3 ntp imap 1024:"
allowed_ports_to_server_tcp="www"
allowed_ports_to_server_udp="www"
allowed_ports_from_intern_tcp="telnet 111 2049 3128 8080"
allowed_ports_from_intern_udp="111 2049 3128 8080"

EXTIP="127.0.0.1"
INTIP="192.168.1.1"
INTERNAL_NET="192.168.1.0/24"
PRIVILEGDED_IP="192.168.1.1/32 192.168.1.32/27"

# Alle Regeln löschen
$CMD -F
$CMD -t nat -F
$CMD -t mangle -F

# Policy setzen
$CMD -P INPUT ACCEPT
$CMD -P OUTPUT ACCEPT
$CMD -P FORWARD ACCEPT
$CMD -t nat -P PREROUTING DROP
$CMD -t nat -P POSTROUTING ACCEPT
$CMD -t nat -P OUTPUT ACCEPT
$CMD -t mangle -P PREROUTING ACCEPT
$CMD -t mangle -P OUTPUT ACCEPT

# Module für ftp,.. -Masquerade aktivieren
for mod in $modules
do
    $MDPR ip$mod
done

# ICMP akzeptieren
$CMD -t nat -A PREROUTING -p icmp -j ACCEPT

# Erlaubte Ports in PREROUTING akzeptieren
for x in $allowed_ports_tcp
do
    $CMD -t nat -A PREROUTING -p tcp --dport $x -j ACCEPT
done

for u in $allowed_ports_udp
do
    $CMD -t nat -A PREROUTING -p udp --dport $u -j ACCEPT
```



```
done

# Erlaubte Ports (auf Server) intern/extern in PREROUTING akzeptieren
for x in $allowed_ports_to_server_tcp
do
    $CMD -t nat -A PREROUTING -p tcp -d $EXTIP --dport $x -j ACCEPT
    $CMD -t nat -A PREROUTING -p tcp -d $INTIP --dport $x -j ACCEPT
done

for u in $allowed_ports_to_server_udp
do
    $CMD -t nat -A PREROUTING -p udp -d $EXTIP --dport $u -j ACCEPT
    $CMD -t nat -A PREROUTING -p udp -d $INTIP --dport $u -j ACCEPT
done
# Erlaubte Ports auf Server nur aus dem internen Netz
for x in $allowed_ports_from_intern_tcp
do
    $CMD -t nat -A PREROUTING -p tcp -s $INTERNAL_NET -d $INTERNAL_NET --dport $x -j ACCEPT
done

for u in $allowed_ports_from_intern_udp
do
    $CMD -t nat -A PREROUTING -p udp -s $INTERNAL_NET -d $INTERNAL_NET --dport $u -j ACCEPT
done

## MASQUERADE EINRICHTEN
# priv. IP dürfen direkt ins Internet (ohne Filter)
for x in $PRIVILEGED_IP
do
    $CMD -t nat -A PREROUTING -p tcp -s $x -d ! $INTERNAL_NET --dport www -j ACCEPT
    $CMD -t nat -A PREROUTING -p udp -s $x -d ! $INTERNAL_NET --dport www -j ACCEPT
    $CMD -t nat -A POSTROUTING -p tcp -s $x -d ! $INTERNAL_NET --dport www -j SNAT --to $EXTIP
    $CMD -t nat -A POSTROUTING -p udp -s $x -d ! $INTERNAL_NET --dport www -j SNAT --to $EXTIP
done
# Transparenter Proxy
$CMD -t nat -A PREROUTING -p tcp -s $INTERNAL_NET -d ! $INTERNAL_NET --dport www -j REDIRECT --to 3128
# Falls FTP nur über den Proxy möglich sein soll, dann die nächsten Zeilen aktivieren
$CMD -t nat -A POSTROUTING -p tcp -s $INTERNAL_NET -d ! $INTERNAL_NET --dport ftp -j DROP
$CMD -t nat -A POSTROUTING -p tcp -s $INTERNAL_NET -d ! $INTERNAL_NET --dport ftp-data -j DROP
# MASQUERADE für den Rest
$CMD -t nat -A POSTROUTING -s $INTERNAL_NET -d ! $INTERNAL_NET -j SNAT --to $EXTIP
```

6.3 INTEGRATION in die STARTSKRIPTEN

Wenn sie nun eine zur Zufriedenheit funktionierende Firewallkonfiguration haben, ist es sinnvoll sie so zu installieren, dass sie beim Systemstart automatisch aktiviert wird.

Dazu gehen sie z.B. folgendermaßen vor:

Bis SuSE 7.3:

Editieren sie die Datei rc.config, fügen sie in dieser Datei ein Variable mit Namen START_OWN_FIREWALL ein und geben sie dieser Variable den Wert „yes“.

Danach erzeugen sie im Verzeichnis /sbin/init.d zwei weitere ausführbare Dateien: fw_clear und own_firewall mit folgendem Inhalt:

Inhalt von OWN_FIREWALL:

```
# STARTEN der Firewall
./etc/rc.config
test "$START_OWN_FIREWALL" = yes || exit 0

#Welche Modules sollen bei der Masquerade geladen werden?

MSQ_MODULES="autofw cuseeme ftp irc mfw portfw quake raudio user vdolive"

INSMOD="/sbin/inssmod"
LSMOD="/sbin/lsmmod"
RMMOD="/sbin/rmmmod"
```



```

case "$1" in
  start)
    echo "Starting Firewall."
    modules() {
      ${LSMOD} | grep "^$1 " > /dev/null
    }
    /sbin/init.d/fw_set
    for i in ${MSQ_MODULES}; do
      if ! modules $i; then
        ${INSMOD} ip_masq_$i
      fi
    done
    ;;
  stop)
    echo -n "Firewall deaktivieren (alle Regeln werden geloescht)"
    /sbin/init.d/fw_clear
    echo
    ;;
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
esac
exit 0

```

Inhalt von FW_CLEAR:

```
IPCHAINS=/sbin/ipchains
```

```
lan="192.168.0.0/16"
Any="0.0.0.0/0"
localhost="127.0.0.1/32"
```

```
$IPCHAINS -F
$IPCHAINS -X
```

```
$IPCHAINS -P input ACCEPT
$IPCHAINS -P forward ACCEPT
$IPCHAINS -P output ACCEPT
```

```
# input rules
```

```
# forward rules
$IPCHAINS -A forward -s $lan -d ! $lan -i eth0 -j MASQ
```

```
# output rules
```

Nachdem sie diese 2 Skripte erstellt haben, müssen sie noch Links zur Datei own_firewall in die Startverzeichnisse jener Runlevels legen, in denen die Firewall gestartet werden soll (Level2, Level3) . In jedem dieser Verzeichnisse (/sbin/init.d/rcxx) wird eine Link zum Starten und einer zum Herunterfahren des Dienstes gelegt. (siehe vorhandene Links)
 Beim nächsten Bootvorgang müsste die Firewall automatisch gestartet werden.

Ab SuSE 8.0:

Kopieren sie das Startskript für die Firewall ins Verzeichnis /etc/init.d und setzen sie über den Runleveleditor von YAST die notwendigen Links (für Runlevel 3 und 5, bzw. eventuell auch für 2)



6.4 Möglichkeiten der SuSEfirewall2:

Nachdem die eigenständige Konfiguration einer Firewall relativ komplex und aufwendig ist kann man natürlich auf bereits vorhandene Skripte zurückgreifen. Diese müssen nur noch mit den richtigen Parametern versehen werden. Ein Beispiel dafür ist die von SuSE mitgelieferte SuSE Firewall2 die viele Möglichkeiten bis hin zur Masquerade und Forwarding auf interne Server ermöglicht.

```
# -----
#
# Configuration HELP:
#
# If you have got any problems configuring this file, take a look at
# /usr/share/doc/packages/SuSEfirewall2/EXAMPLES for an example.
#
# All types have to set enable SuSEfirewall2 in the runlevel editor
#
# If you are a end-user who is NOT connected to two networks (read: you have
# got a single user system and are using a dialup to the internet) you just
# have to configure (all other settings are OK): 2) and maybe 9).
#
# If this server is a firewall, which should act like a proxy (no direct
# routing between both networks), or you are an end-user connected to the
# internet and to an internal network, you have to setup your proxys and
# reconfigure (all other settings are OK): 2), 3), 9) and maybe 7), 11), 14)
#
# If this server is a firewall, and should do routing/masquerading between
# the untrusted and the trusted network, you have to reconfigure (all other
# settings are OK): 2), 3), 5), 6), 9), and maybe 7), 10), 11), 12), 13),
# 14), 20)
#
# If you want to run a DMZ in either of the above three standard setups, you
# just have to configure *additionally* 4), 9), 12), 13), 17), 19).
#
# If you know what you are doing, you may also change 8), 11), 15), 16)
# and the expert options 19), 20), 21), 22) and 23) at the far end, but you
# should NOT.
#
# If you use diald or ISDN autodialing, you might want to set 17).
#
# To get programs like traceroutes to your firewall to work is a bit tricky,
# you have to set the following options to "yes" : 11 (UDP only), 18 and 19.
#
# Please note that if you use service names, that they exist in /etc/services.
# There is no service "dns", it's called "domain"; email is called "smtp" etc.
#
# *Any* routing between interfaces except masquerading requires to set FW_ROUTE
# to "yes" and use FW_FORWARD or FW_ALLOW_CLASS_ROUTING !
#
# If you just want to do masquerading without filtering, ignore this script
# and run this line (exchange "ipp0" "ppp0" if you use a modem, not isdn):
# iptables -A POSTROUTING -t nat -j MASQUERADE -o ipp0
# echo 1 > /proc/sys/net/ipv4/ip_forward
# and additionally the following lines to get at least a minimum of security:
# iptables -A INPUT -j DROP -m state --state NEW,INVALID -i ipp0
# iptables -A FORWARD -j DROP -m state --state NEW,INVALID -i ipp0
# -----
FW_QUICKMODE="no"
```



```

FW_DEV_EXT="eth0"
FW_DEV_INT=""
FW_DEV_DMZ=""
FW_ROUTE="no"
FW_MASQUERADE="no"
FW_MASQ_DEV="$FW_DEV_EXT"
FW_MASQ_NETS=""
FW_PROTECT_FROM_INTERNAL="yes"
FW_AUTOPROTECT_SERVICES="yes"
FW_SERVICES_EXT_TCP="10000 23"
FW_SERVICES_EXT_UDP=""
FW_SERVICES_EXT_IP=""
FW_SERVICES_DMZ_TCP=""
FW_SERVICES_DMZ_UDP=""
FW_SERVICES_DMZ_IP=""
FW_SERVICES_INT_TCP=""
FW_SERVICES_INT_UDP=""
FW_SERVICES_INT_IP=""
FW_SERVICES_QUICK_TCP=""
FW_SERVICES_QUICK_UDP=""
FW_SERVICES_QUICK_IP=""
FW_TRUSTED_NETS=""
FW_ALLOW_INCOMING_HIGHPORTS_TCP="no"
FW_ALLOW_INCOMING_HIGHPORTS_UDP="DNS"
FW_SERVICE_AUTODETECT="no"
FW_SERVICE_DNS="no"
FW_SERVICE_DHCLIENT="no"
FW_SERVICE_DHCPD="no"
FW_SERVICE_SQUID="no"
FW_SERVICE_SAMBA="no"
FW_FORWARD=""
FW_FORWARD_MASQ=""

## Type:      string
# Beware to use this!

#
# 15.)
# Which accesses to services should be redirected to a localport on the
# firewall machine?
#
# This can be used to force all internal users to surf via your squid proxy,
# or transparently redirect incoming webtraffic to a secure webserver.
#
# Choice: leave empty or use the following explained syntax of redirecting
# rules, seperated by a space.
# A redirecting rule consists of 1) source IP/net, 2) destination IP/net,
# 3) protocol (tcp or udp) 3) original destination port and 4) local port to
# redirect the traffic to, seperated by a colon. e.g.:
# "10.0.0.0/8,0/0,tcp,80,3128 0/0,172.20.1.1,tcp,80,8080"
# Please note that as 2) destination, you may add '!' in front of the IP/net
# to specify everything EXCEPT this IP/net.
#
FW_REDIRECT=""
FW_LOG_DROP_CRIT="yes"
FW_LOG_DROP_ALL="no"
FW_LOG_ACCEPT_CRIT="yes"
FW_LOG_ACCEPT_ALL="no"
FW_LOG="--log-level warning --log-tcp-options --log-ip-option --log-prefix SuSE-FW"
FW_KERNEL_SECURITY="yes"
FW_STOP_KEEP_ROUTING_STATE="no"

```



```
FW_ALLOW_PING_FW="no"
FW_ALLOW_PING_DMZ="no"
FW_ALLOW_PING_EXT="no"
FW_ALLOW_FW_TRACEROUTE="no"
FW_ALLOW_FW_SOURCEQUENCH="yes"
FW_ALLOW_FW_BROADCAST="no"
FW_IGNORE_FW_BROADCAST="yes"
FW_ALLOW_CLASS_ROUTING="no"
FW_CUSTOMRULES=""
FW_REJECT="no"
FW_HTB_TUNE_DEV=""
```

6.5 ANHANG : MANPAGE zu IPTABLES

SYNOPSIS

```
Iptables  -[ADC]      chain      rule-specification [options]
iptables -[RI]      chain      rulenum    rule-specification [options]
iptables -D         chain      rulenum    [options]
iptables -[LFZ]    [chain]    [options]
iptables -[NX]     chain

iptables -P         chain      target     [options]
iptables -E         old-chain-name  new-chain-name
```

DESCRIPTION

Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. There are several different tables which may be defined, and each table contains a number of built-in chains, and may contain user-defined chains.

Each chain is a list of rules which can match a set of packets: each rule specifies what to do with a packet which matches. This is called a 'target', which may be a jump to a user-defined chain in the same table.

TARGETS

A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule in the chain is the examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain, or one of the special values ACCEPT, DROP, QUEUE, or RETURN.

ACCEPT means to let the packet through. DROP means to drop the packet on the floor. QUEUE means to pass the packet to userspace. RETURN means stop traversing this chain, and resume at the next rule in the previous (calling) chain. If the end of a built-in chain is reached, or a rule in a built-in chain with target RETURN is matched, the target specified by the chain policy determines the fate of the packet.

TABLES

There are current three tables (which tables are present at any time depends on the kernel configuration options and which modules are present).

-t, --table

This option specifies the packet matching table which the command should operate on. If the kernel is configured with automatic module loading, an attempt will be made to load the appropriate module for that table if it is not already there.

The tables are as follows: filter This is the default table, and contains the built-in chains INPUT (for packets coming into the box itself), FORWARD (for packets being routed through the box), and OUTPUT (for locally-generated packets). nat This table is consulted when a packet which is



creates a new connection is encountered. It consists of three built-ins: PREROUTING (for altering packets as soon as they come in), OUTPUT (for altering locally-generated packets before routing), and POSTROUTING (for altering packets as they are about to go out). mangle This table is used for specialized packet alteration. It has two built-in chains: PREROUTING (for altering incoming packets before routing) and OUTPUT (for altering locally-generated packets before routing).

OPTIONS

The options that are recognized by iptables can be divided into several different groups.

COMMANDS

These options specify the specific action to perform; only one of them can be specified on the command line, unless otherwise specified below. For all the long versions of the command and option names, you only need to use enough letters to ensure that iptables can differentiate it from all other options.

-A, --append

Append one or more rules to the end of the selected chain. When the source and/or destination names resolve to more than one address, a rule will be added for each possible address combination.

-D, --delete

Delete one or more rules from the selected chain. There are two versions of this command: the rule can be specified as a number in the chain (starting at 1 for the first rule) or a rule to match.

-R, --replace

Replace a rule in the selected chain. If the source and/or destination names resolve to multiple addresses, the command will fail. Rules are numbered starting at 1.

-I, --insert

Insert one or more rules in the selected chain as the given rule number. So, if the rule number is 1, the rule or rules are inserted at the head of the chain. This is also the default if no rule number is specified.

-L, --list

List all rules in the selected chain. If no chain is selected, all chains are listed. It is legal to specify the -Z (zero) option as well, in which case the chain(s) will be atomically listed and zeroed. The exact output is effected by the other arguments given.

-F, --flush

Flush the selected chain. This is equivalent to deleting all the rules one by one.

-Z, --zero

Zero the packet and byte counters in all chains. It is legal to specify the -L, --list (list) option as well, to see the counters immediately before they are cleared; see above.

-N, --new-chain

Create a new user-defined chain of the given name. There must be no target of that name already.

-X, --delete-chain

Delete the specified user-defined chain. There must be no references to the chain (if there are you must delete or replace the referring rules before the chain can be deleted). If no argument is given, it will attempt to delete every non-builtin chain.

-P, --policy

Set the policy for the chain to the given target. See the section

-E, --rename-chain

Rename the user specified chain to the user supplied name; this is cosmetic, and has no effect on the structure of the table. TARGETS for the legal targets. Only non-userdefined chains can have policies, and neither built-in nor user-defined chains can be policy targets.

-h



Help. Give a (currently very brief) description of the command syntax.

PARAMETERS

The following parameters make up a rule specification (as used in the add, delete, replace, append and check commands).

-p, --protocol [!] protocol

The protocol of the rule or of the packet to check. The specified protocol can be one of tcp, udp, icmp, or all, or it can be a numeric value, representing one of these protocols or a different one. Also a protocol name from /etc/protocols is allowed. A "!" argument before the protocol inverts the test. The number zero is equivalent to all. Protocol all will match with all protocols and is taken as default when this option is omitted. All may not be used in combination with the check command.

-s, --source [!] address[/mask]

Source specification. Address can be either a hostname, a network name, or a plain IP address. The mask can be either a network mask or a plain number, specifying the number of 1's at the left side of the network mask. Thus, a mask of 24 is equivalent to 255.255.255.0. A "!" argument before the address specification inverts the sense of the address. The flag --src is a convenient alias for this option.

-d, --destination [!] address[/mask]

Destination specification. See the description of the -s (source) flag for a detailed description of the syntax. The flag --dst is an alias for this option.

-j, --jump target

This specifies the target of the rule; ie. what to do if the packet matches it. The target can be a user-defined chain (not the one this rule is in), one of the special builtin targets which decide the fate of the packet immediately, or an extension (see EXTENSIONS below). If this option is omitted in a rule, then matching the rule will have no effect on the packet's fate, but the counters on the rule will be incremented.

-i, --in-interface [!] [name]

Optional name of an interface via which a packet is received (for packets entering the INPUT, FORWARD and PREROUTING chains). When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, the string "+" is assumed, which will match with any interface name.

-o, --out-interface [!] [name]

Optional name of an interface via which a packet is going to be sent (for packets entering the FORWARD, OUTPUT and POSTROUTING chains). When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, the string "+" is assumed, which will match with any interface name.

[!] -f, --fragment

This means that the rule only refers to second and further fragments of fragmented packets. Since there is no way to tell the source or destination ports of such a packet (or ICMP type), such a packet will not match any rules which specify them. When the "!" argument precedes the "-f" flag, the sense is inverted.

OTHER OPTIONS

The following additional options can be specified:

-v, --verbose

Verbose output. This option makes the list command show the interface address, the rule options (if any), and the TOS masks. The packet and byte counters are also listed, with the suffix 'K', 'M' or



'G' for 1000, 1,000,000 and 1,000,000,000 multipliers respectively (but see the -x flag to change this). For appending, insertion, deletion and replacement, this causes detailed information on the rule or rules to be printed.

-n, --numeric

Numeric output. IP addresses and port numbers will be printed in numeric format. By default, the program will try to display them as host names, network names, or services (whenever applicable).

-x, --exact

Expand numbers. Display the exact value of the packet and byte counters, instead of only the rounded number in K's (multiples of 1000) M's (multiples of 1000K) or G's (multiples of 1000M). This option is only relevant for the -L command.

--line-numbers

When listing rules, add line numbers to the beginning of each rule, corresponding to that rule's position in the chain.

MATCH EXTENSIONS

iptables can use extended packet matching modules. The following are included in the base package, and most of these can be preceded by a ! to invert the sense of the match.

tcp

These extensions are loaded if '--protocol tcp' is specified, and no other match is specified. It provides the following options:

--source-port [!] [port[:port]]

Source port or port range specification. This can either be a service name or a port number. An inclusive range can also be specified, using the format port:port. If the first port is omitted, "0" is assumed; if the last is omitted, "65535" is assumed. If the second port greater than the first they will be swapped. The flag --sport is an alias for this option.

--destination-port [!] [port[:port]]

Destination port or port range specification. The flag --dport is an alias for this option.

--tcp-flags [!] mask comp

Match when the TCP flags are as specified. The first argument is the flags which we should examine, written as a comma-separated list, and the second argument is a comma-separated list of flags which must be set. Flags are: SYN ACK FIN RST URG PSH ALL NONE. Hence the command

```
iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST SYN
will only match packets with the SYN flag set, and the ACK, FIN and RST flags unset.
```

[!] --syn

Only match TCP packets with the SYN bit set and the ACK and FIN bits cleared. Such packets are used to request TCP connection initiation; for example, blocking such packets coming in an interface will prevent incoming TCP connections, but outgoing TCP connections will be unaffected. It is equivalent to --tcp-flags SYN,RST,ACK SYN. If the "!" flag precedes the "--syn", the sense of the option is inverted.

--tcp-option [!] number

Match if TCP option set.

udp

These extensions are loaded if '--protocol udp' is specified, and no other match is specified. It provides the following options:

--source-port [!] [port[:port]]



Source port or port range specification. See the description of the --source-port option of the TCP extension for details.

--destination-port [!] [port[:port]]

Destination port or port range specification. See the description of the --destination-port option of the TCP extension for details.

icmp

This extension is loaded if '--protocol icmp' is specified, and no other match is specified. It provides the following option:

--icmp-type [!] typename

This allows specification of the ICMP type, which can be a numeric ICMP type, or one of the ICMP type names shown by the command

```
iptables -p icmp -h
```

mac

--mac-source [!] address

Match source MAC address. It must be of the form XX:XX:XX:XX:XX:XX. Note that this only makes sense for packets entering the PREROUTING, FORWARD or INPUT chains for packets coming from an ethernet device.

limit

This module matches at a limited rate using a token bucket filter: it can be used in combination with the LOG target to give limited logging. A rule using this extension will match until this limit is reached (unless the `!' flag is used).

--limit rate

Maximum average matching rate: specified as a number, with an optional '/second', '/minute', '/hour', or '/day' suffix; the default is 3/hour.

--limit-burst number

The maximum initial number of packets to match: this number gets recharged by one every time the limit specified above is not reached, up to this number; the default is 5.

multiport

This module matches a set of source or destination ports. Up to 15 ports can be specified. It can only be used in conjunction with -p tcp or -p udp.

--source-port [port[,port]]

Match if the source port is one of the given ports.

--destination-port [port[,port]]

Match if the destination port is one of the given ports.

--port [port[,port]]

Match if the both the source and destination ports are equal to each other and to one of the given ports.

mark

This module matches the netfilter mark field associated with a packet (which can be set using the MARK target below).

--mark value[/mask]

Matches packets with the given unsigned mark value (if a mask is specified, this is logically ANDed with the mark before the comparison).

owner



This module attempts to match various characteristics of the packet creator, for locally-generated packets. It is only valid in the OUTPUT chain, and even this some packets (such as ICMP ping responses) may have no owner, and hence never match.

--uid-owner userid

Matches if the packet was created by a process with the given effective user id.

--gid-owner groupid

Matches if the packet was created by a process with the given effective group id.

--pid-owner processid

Matches if the packet was created by a process with the given process id.

--sid-owner sessionid

Matches if the packet was created by a process in the given session group.

state

This module, when combined with connection tracking, allows access to the connection tracking state for this packet.

--state state

Where state is a comma separated list of the connection states to match. Possible states are INVALID meaning that the packet is associated with no known connection, ESTABLISHED meaning that the packet is associated with a connection which has seen packets in both directions, NEW meaning that the packet has started a new connection, or otherwise associated with a connection which has not seen packets in both directions, and RELATED meaning that the packet is starting a new connection, but is associated with an existing connection, such as an FTP data transfer, or an ICMP error.

unclean

This module takes no options, but attempts to match packets which seem malformed or unusual. This is regarded as experimental.

tos

This module matches the 8 bits of Type of Service field in the IP header (ie. including the precedence bits).

--tos tos

The argument is either a standard name, (use

iptables -m tos -h to see the list), or a numeric value to match.

TARGET EXTENSIONS

iptables can use extended target modules: the following are included in the standard distribution.

LOG

Turn on kernel logging of matching packets. When this option is set for a rule, the Linux kernel will print some information on all matching packets (like most IP header fields) via printk().

--log-level level

Level of logging (numeric or see [syslog.conf\(5\)](#)).

--log-prefix prefix

Prefix log messages with the specified prefix; up to 14 letters long, and useful for distinguishing messages in the logs.

--log-tcp-sequence

Log TCP sequence numbers. This is a security risk if the log is readable by users.

--log-tcp-options

Log options from the TCP packet header.

--log-ip-options

Log options from the IP packet header.



MARK

This is used to set the netfilter mark value associated with the packet. It is only valid in the mangle table.

--set-mark mark

REJECT

This is used to send back an error packet in response to the matched packet: otherwise it is equivalent to DROP. This target is only valid in the INPUT, FORWARD and OUTPUT chains, and user-defined chains which are only called from those chains. Several options control the nature of the error packet returned:

--reject-with type

The type given can be icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreachable, icmp-net-prohibited or icmp-host-prohibited, which return the appropriate ICMP error message (port-unreachable is the default). The option echo-reply is also allowed; it can only be used for rules which specify an ICMP ping packet, and generates a ping reply. Finally, the option tcp-reset can be used on rules in (or called from) the INPUT chain which only match the TCP protocol: this causes a TCP RST packet to be sent back.

TOS

This is used to set the 8-bit Type of Service field in the IP header. It is only valid in the mangle table.

--set-tos tos

You can use a numeric TOS values, or use

iptables -j TOS -h

to see the list of valid TOS names.

MIRROR

This is an experimental demonstration target which inverts the source and destination fields in the IP header and retransmits the packet. It is only valid in the INPUT, FORWARD and OUTPUT chains, and user-defined chains which are only called from those chains.

SNAT

This target is only valid in the nat table, in the POSTROUTING chain. It specifies that the source address of the packet should be modified (and all future packets in this connection will also be mangled), and rules should cease being examined. It takes one option:

--to-source <ipaddr>[-<ipaddr>][:port-port]

which can specify a single new source IP address, an inclusive range of IP addresses, and optionally, a port range (which is only valid if the rule also specifies -p tcp or -p udp). If no port range is specified, then source ports below 512 will be mapped to other ports below 512: those between 1024 will be mapped to ports below 1024, and other ports will be mapped to 1024 or above. Where possible, no port alteration will occur.

DNAT

This target is only valid in the nat table, in the PREROUTING and OUTPUT chains, and user-defined chains which are only called from those chains. It specifies that the destination address of the packet should be modified (and all future packets in this connection will also be mangled), and rules should cease being examined. It takes one option:

--to-destination <ipaddr>[-<ipaddr>][:port-port]

which can specify a single new destination IP address, an inclusive range of IP addresses, and optionally, a port range (which is only valid if the rule also specifies -p tcp or -p udp). If no port range is specified, then the destination port will never be modified.

MASQUERADE



This target is only valid in the nat table, in the POSTROUTING chain. It should only be used with dynamically assigned IP (dialup) connections: if you have a static IP address, you should use the SNAT target. Masquerading is equivalent to specifying a mapping to the IP address of the interface the packet is going out, but also has the effect that connections are forgotten when the interface goes down. This is the correct behavior when the next dialup is unlikely to have the same interface address (and hence any established connections are lost anyway). It takes one option:

--to-ports <port>[-<port>]

This specifies a range of source ports to use, overriding the default SNAT source port-selection heuristics (see above). This is only valid with if the rule also specifies -p tcp or -p udp).

REDIRECT

This target is only valid in the nat table, in the PREROUTING and OUTPUT chains, and user-defined chains which are only called from those chains. It alters the destination IP address to send the packet to the machine itself (locally-generated packets are mapped to the 127.0.0.1 address). It takes one option:

--to-ports <port>[-<port>]

This specifies a destination port or range or ports to use: without this, the destination port is never altered. This is only valid with if the rule also specifies -p tcp or -p udp)

6.6 Ergänzung der SuSE-Firewall:

Im Verzeichnis /etc/sysconfig/scripts liegt die Datei SuSEfirewall2-custom. Iptables-Befehle, die in diese Datei eingetragen werden, werden automatisch beim Starten der SuSEfirewall aktiviert.

Das Verändern von Firewallinstellungen obliegt dem root-Benutzer. Im Zusammenhang mit der Webminoberfläche gibt es eine elegante Möglichkeit Lehrern die Möglichkeit einzuräumen z.B. EDV-Räume zu sperren, ohne ihnen direkt Root-Rechte zu geben.

Notwendige Schritte:

- Als Root-Benutzer anlegen von „CustomCommands“
- Die Befehle unter der Benutzerkennung „root“ laufen lassen.
- Anlegen eines Webmin Benutzers „Lehrer“
- Einschränken der Webmin Rechte von „Lehrer“



7 PAKETSNIFFER

Zur Analyse von Datenpaketen werden Paketsniffer verwendet. Auf der Kommandozeile kann unter Linux z.B. das Programm tcpdump verwendet werden. Der folgende Screenshot zeigt einen typischen Output dieses Programms.

```

vpn:~ # tcpdump -i eth0 dst port 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
19:26:59.197755 IP 193.170.68.246.46386 > vpn.brg-wrn.ac.at.ssh: P 3706221378:3706221422(44) ack 1419147873 win 64899
19:26:59.228107 IP 193.170.68.246.46386 > vpn.brg-wrn.ac.at.ssh: P 44:88(44) ack 117 win 64783
19:26:59.233683 IP 193.170.68.246.46386 > vpn.brg-wrn.ac.at.ssh: P 88:132(44) ack 233 win 64667
19:26:59.257209 IP 193.170.68.246.46386 > vpn.brg-wrn.ac.at.ssh: P 132:176(44) ack 397 win 64503
19:26:59.278377 IP 193.170.68.246.46386 > vpn.brg-wrn.ac.at.ssh: P 176:220(44) ack 537 win 64363
19:26:59.286305 IP 193.170.68.246.46386 > vpn.brg-wrn.ac.at.ssh: P 220:264(44) a

```

Im Zusammenhang mit Snifferprogrammen ist es notwendig, den interessierenden Datenverkehr aus der Menge der Daten herauszufiltern. Dazu definiert man Filterkriterien (im Screenshot wird auf das Interface eth0 und den Zielport 22 gefiltert). Etwas komfortabler gelingt die Arbeit mit grafischen Frontends wie es etwa „ethereal“ bietet. Dieses Programm gibt es als Linux und auch als Windows-Version. Der nachfolgende Screenshot zeigt den Mitschnitt von Datenpaketen die auf den Zielport 80 (http) gerichtet sind.

The screenshot shows the Ethereal network sniffer interface. The top toolbar includes options like File, Edit, View, Go, Capture, Analyze, Statistics, and Help. Below the toolbar is a filter bar with the expression 'tcp.port==80'. The main pane displays a list of captured packets with columns for No., Time, Source, Destination, Protocol, and Info. Packet 30 is selected, showing details for an HTTP GET request to http://10.221.14.111/index.php?id=1&type=5. Below the packet list, a detailed view of the selected packet is shown, including Ethernet II, Internet Protocol, and Transmission Control Protocol headers, followed by the raw packet data in hexadecimal and ASCII.

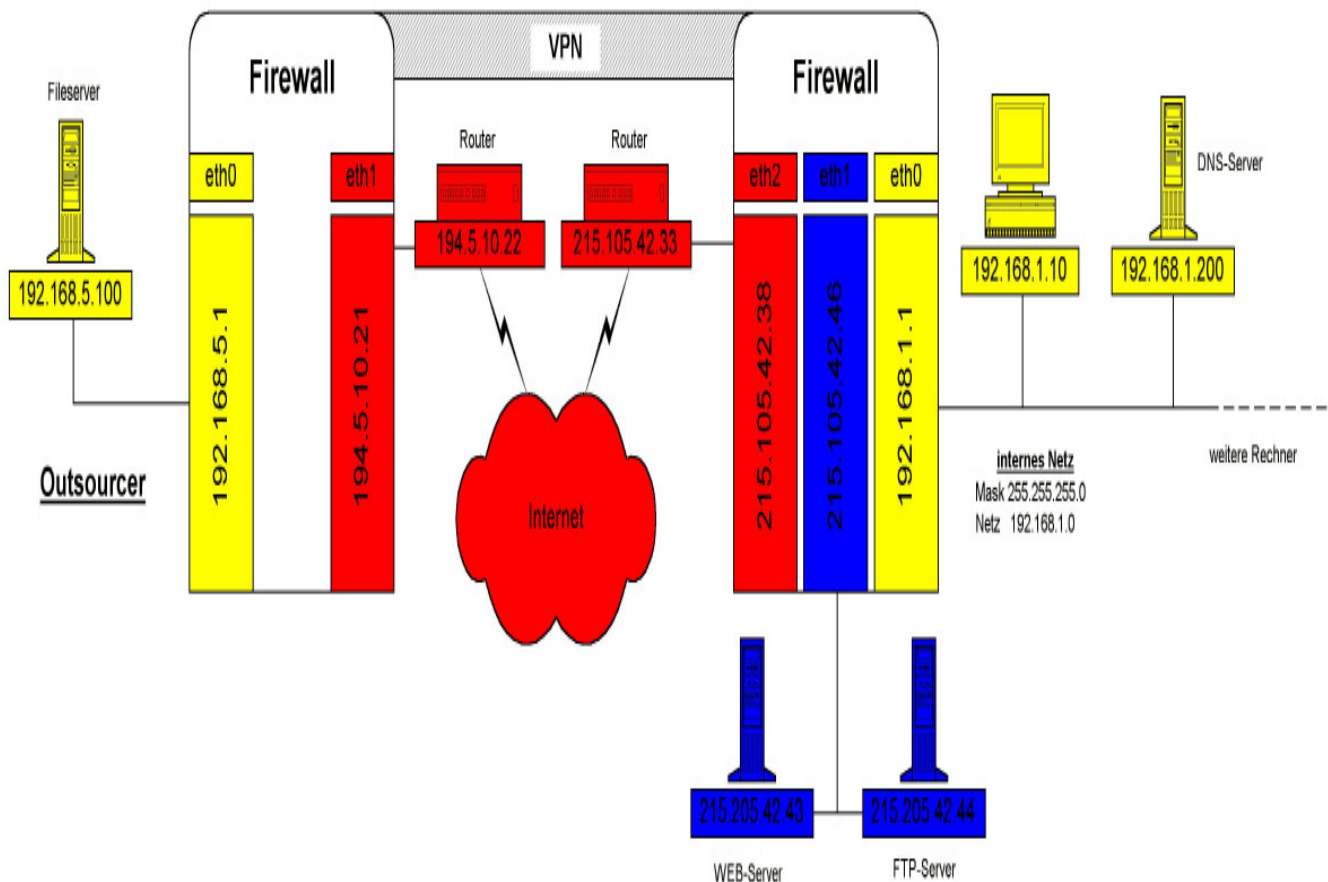
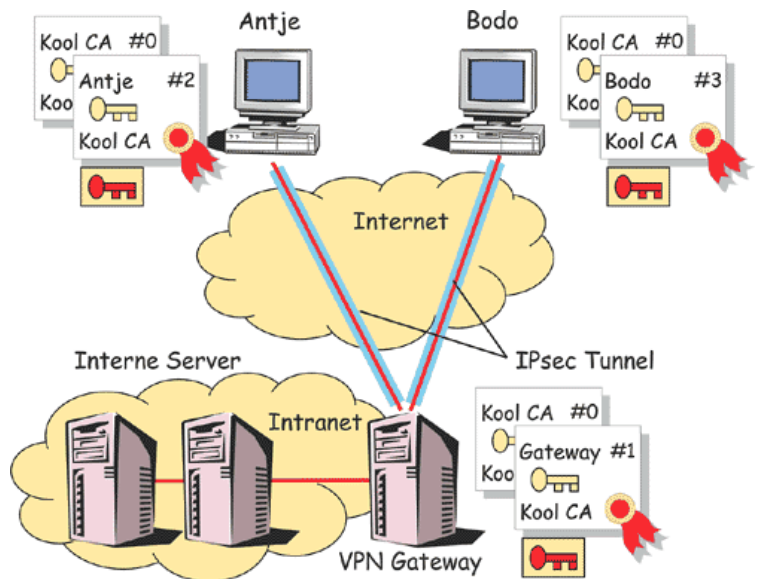


8 SICHERE VERBINDUNGEN mit VPN:

Die (sichere) Verbindung zweier privater Netzwerke über das Internet, kann über VPN (Virtual Private Network) erfolgen. Unter Linux kann man dafür z.B. das Paket FreeSWAN (Free Secure WAN) verwenden, das über die Datei ipsec.conf konfiguriert wird. Bei der Installation von IPSEC wird automatisch ein Schlüsselpaar generiert. Der öffentliche Teil dieses Schlüssels muss nun der jeweiligen Gegenstelle bekannt gegeben werden.

Für die Bereitstellung der entsprechenden Schlüssel kann der Befehl „ipsec showhostkey“ verwendet werden. Ohne weiteren Parameter liefert dieser Befehl die Information, die über einen DNS Server publiziert werden kann um den Schlüssel allgemein zur Verfügung zu stellen. Mit dem Parameter „—left“ bzw. „—right“ werden die Schlüssel für das rechte bzw. linke Ende erzeugt, die gleich in eine Datei umgeleitet werden können.

Wenn auf den VPN Rechnern auch eine Firewall läuft ist zu beachten, dass der upd-Port 500 und das Protokoll 50 (ESP) freigeschaltet werden müssen.





Weiters ist zu beachten, dass sich die beiden VPN-Rechner selbst nicht sehen. (Dies kann jedoch durch ein nachträgliches Anpassen der Routingtabelle erfolgen)

Beispiel für eine RoadWarrior – Lösung (Ende mit dyn. Adresse)

```
# /etc/ipsec.conf - FreeS/WAN IPsec configuration file

# More elaborate and more varied sample configurations can be found
# in FreeS/WAN's doc/examples file, and in the HTML documentation.

# basic configuration
config setup
    # THIS SETTING MUST BE CORRECT or almost nothing will work;
    # %defaultroute is okay for most simple cases.
    interfaces=%defaultroute
    # Debug-logging controls: "none" for (almost) none, "all" for lots.
    klipsdebug=none
    plutodebug=none
    # Use auto= parameters in conn descriptions to control startup actions.
    plutoload=%search
    plutostart=%search
    # Close down old connection when new one using same ID shows up.
    uniqueids=yes

# defaults for subsequent connection descriptions
# (these defaults will soon go away)
conn %default
    keyingtries=0
    authby=rsasig

conn sta-home
    left=193.170.207.190
    leftsubnet=10.0.0.0/24
    leftid=@vpn.brg-wrn.ac.at
    leftrsasigkey= .....
    right=%defaultroute
    rightsubnet=10.1.1.0/24
    rightid=@stachl.road.brg-wrn.ac.at
    rightrsasigkey= .....
    auto=start
```

Die entsprechende Gegenstelle (fixe IP) wurde folgendermaßen konfiguriert:

```
config setup
interfaces="ipsec0=eth0"
klipsdebug=none
plutodebug=none
# Use auto= parameters in conn descriptions to control startup actions.
plutoload=%search
plutostart=%search
# Close down old connection when new one using same ID shows up.
uniqueids=yes
# defaults for subsequent connection descriptions
# (these defaults will soon go away)
```



```
conn %default
keyingtries=0
disablearrivalcheck=no
authby=rsasig
conn sta-home
    left=193.170.207.190
    leftsubnet=10.0.0.0/24
    leftid=@vpn.brg-wrn.ac.at
    leftnexthop=193.170.207.189
    leftrsasigkey=0sAQOUT.....
    right=%any
    rightsubnet=10.1.1.0/24
    rightid=@stachl.road.brg-wrn.ac.at
    rightrsasigkey=0sAQOpIJBpi.....
    auto=add
conn pfw-home
    left=193.170.207.190
    leftsubnet=10.0.0.0/24
    leftnexthop=193.170.207.189
    leftid=@vpn.brg-wrn.ac.at
    leftrsasigkey=0sAQOUTh.....
    right=%any
    rightsubnet=192.168.3.0/24
    rightid=@pfw.home.brg-wrn.ac.at
    rightrsasigkey=0sAQOccil0.....
    auto=add
```