Beschreibung Netzwerklabor (S)

Version 0.1.4 vom 06.07.2003

Fachbereich Informatik der FH Karlsruhe

Beschreibung Netzwerklabor (Studierende)

Fachbereich Informatik der FH Karlsruhe

SS 2003

Prof. Dr. rer. nat. Lothar Gmeiner
Dipl. Inf. (FH) Georg Magschok
Dipl. Inf. (FH) Michael Fischer
Henning Pfeiffer

Inhalt

1 Laboraufbau 2

Ausstattung 2
Erreichbarkeit der VMware-Instanzen 2
Domain Name Service 4
Cisco Telnet Zugriffe 4

2 Cisco Tutorial 5

Basiswissen 5
Startup und Running 5
User und Privileged 5
Kurzreferenz 8
Interface-Konfiguration 8
Routing-Konfiguration 11
ACL-Konfiguration 14

3 Linux Tutorial 17

bash und vim 17
Netcat und Telnet 22
nslookup und dig 23
TCPdump/Headers 26
Netfilter/IPtables 29
Übersicht 29
Beispiele 30

A Literaturverzeichnis 32

Kapitelinhalt:

- Ausstattung
- VMware-Instanzen
- Domain Name Service
- Telnet auf Ciscos

1

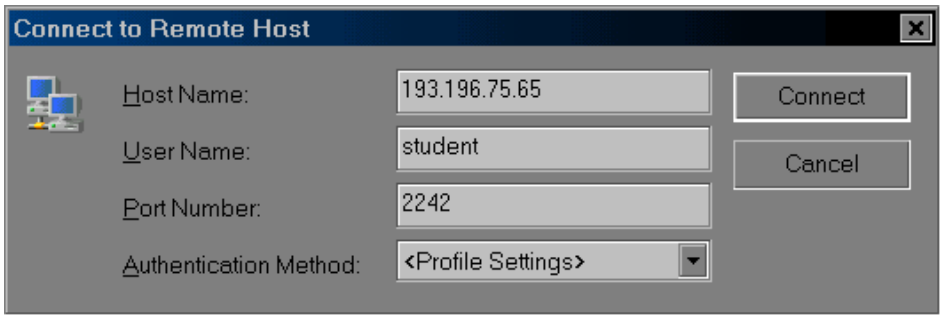
Laboraufbau

Ausstattung

Das Zentrum des Labors besteht aus einem geschachtelten Ring von vier Cisco 1720 Routern. Diese sind jeweils paarweise mit seriellen Leitungen und über Standard Ethernet (10baseT) verbunden. An jedem Router gibt es einen lokalen Fast Ethernet (100baseTX) Abzweig, der an je einem Cisco Catalyst 2600 Fast Ethernet Switch angeschlossen ist. Vier PCs runden das Gesamtbild der Hardware ab. Zwei von ihnen enthalten jeweils vier VMware-Instanzen. Diese sind, jede für sich, mit einer eigenen Netzwerkkarte an unterschiedliche der oben genannten Catalyst-Switche angeschlossen. Ein anderer der vier PCs dient als Zugangsserver zum FH-Netz. Über diesen sind per Secure Shell direkte Terminalverbindungen zu allen VMware-Instanzen auf den anderen beiden PCs möglich. Auf allen PCs und VMware-Instanzen ist Mandrake Linux 9.0, ein Linux Betriebssystem, das auf Redhat Linux basiert, installiert.

Erreichbarkeit der VMware-Instanzen

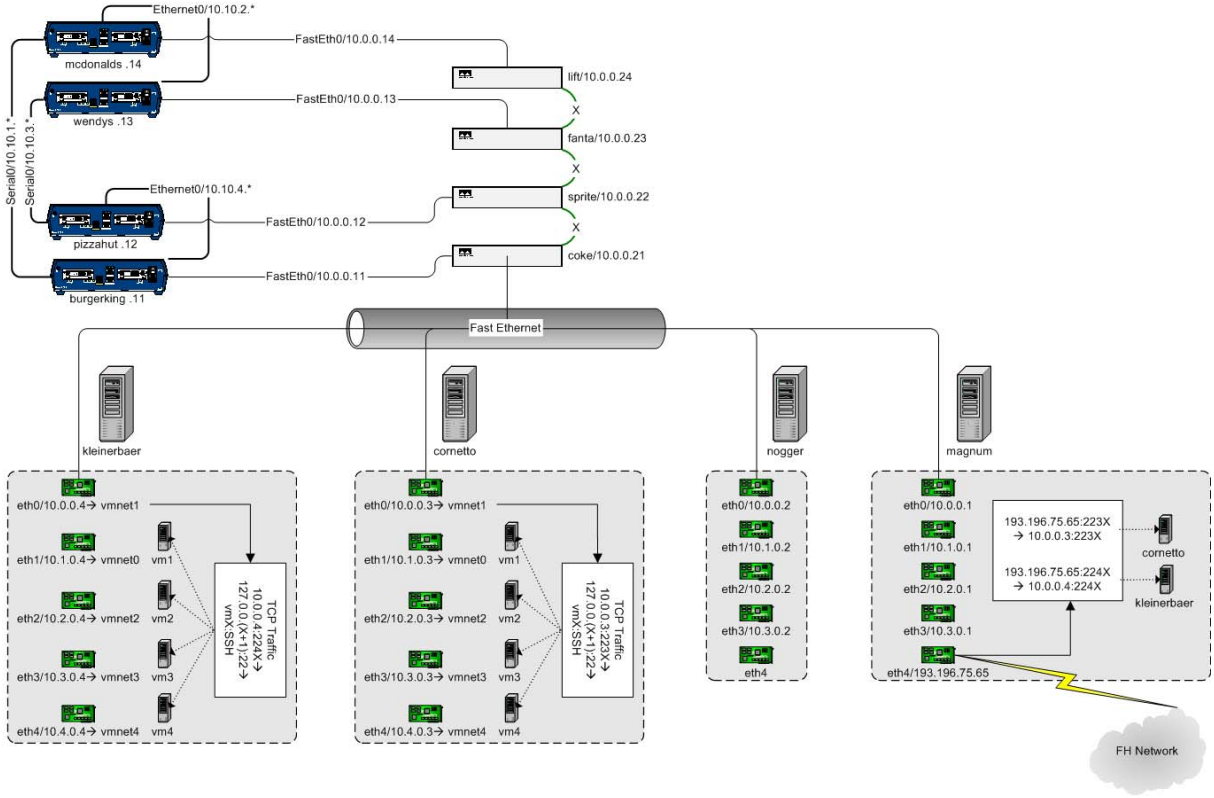
Das Labor verfügt über die öffentliche IP-Adresse 193.196.75.65, welche dem externen Interface des ersten PC (PC#1) zugeordnet ist. Dieser PC agiert als Zugangsrouter für Verkehr zum und vom FH-Netzwerk. Da das Netzlabor über das externe Interface von PC#1 maskiert wird, ist nur dieser Rechner direkt von außerhalb des Labors erreichbar. Damit jedoch die VMware-Instanzen auf den dedizierten PCs für VMware, PC#3 und PC#4, administrativ erreichbar sind, wurde ein kaskadiertes Port-Forwarding auf PC#1 und jeweils auf PC#3 und PC#4 konfiguriert. Außer dem SSH-Server sind standardmäßig jedoch keine anderen Dienste von außerhalb des Labors auf den VMware-Instanzen erreichbar.
Um nun eine bestimmte VMware-Instanz von einem Rechnerpool der Informatik zu erreichen, ist eine SSH Verbindung zu *FbI-Netlab-01.FH-Karlsruhe.DE* mit einem vom Standard Well-Known Port für SSH (tcp/22) abweichenden Server-Port nötig. Entsprechend der Gruppeneinteilung ergibt sich eine Zuteilung einer bestimmten Instanz (vm1 bis vm4) auf PC#3 oder PC#4. Der Server-Port setzt sich nun wie folgt (als String gesehen) zusammen: 22(3|4)[1-4]. Bei einer Zuordnung von PC#4 und VMware-Instanz vm2 ergibt sich also 2242 als Serverport. Bei Verwendung eines graphischen SSH Clients, wie z.B. der von SSH Communications [1-1] an der FH eingesetzte, läßt sich der Port beim Verbindungsaufbau mit angeben:



Unter Linux/UNIX wird der SSH Client mit folgenden Parametern aufgerufen:

```
[student@host student]# ssh -l student -p 2242 193.196.75.65
```

Weitere Details bezüglich des Netzwerkaufbaus und des Port-Forwardings für SSH sind der nachfolgenden Graphik zu entnehmen.



Der Secure Shell Client kann später dazu verwendet werden, Dienste, die auf den VMware-Instanzen installiert wurden über eine SSH-Verbindung zu tunneln. So kann z.B. ein aufgesetzter Apache Webserver auf der zweiten VMware-Instanz von PC#3 über ein Port-Forwarding von localhost:8032 (gleiche Syntax/Semantik wie bei SSH Port 22, nun jedoch HTTP Port 80) zur VMware-Instanz auf Port 80, von außen erreichbar gemacht werden.

Domain Name Service

Innerhalb der Subdomain *netlab.fh-karlsruhe.de* wurde ein eigener DNS-Server aufgesetzt, welcher auf PC#1 seinen Dienst verrichtet. Dieser leistet sowohl Vorwärts- als auch Rückwärtsauflösung. Er kann von außerhalb des Netzwerklabors mit der öffentlichen IP-Adresse auf TCP/UDP Port 53 kontaktiert werden. Hier eine Überprüfung der Vorwärts- und Rückwärtsauflösung von PC#3:

```
bash-2.05$ nslookup
Default Server:  rz01.FH-Karlsruhe.DE
Address:  193.196.64.1

> server FbI-Netlab-01.FH-Karlsruhe.DE
Default Server:  FbI-Netlab-01.FH-Karlsruhe.DE
Address:  193.196.75.65

> set type=A
> nogger.netlab.fh-karlsruhe.de
Name:    nogger.netlab.fh-karlsruhe.de
Address:  10.0.0.3

> set type=PTR
> 10.0.0.3
3.0.0.10.in-addr.arpa   name = nogger.netlab.fh-karlsruhe.de
10.in-addr.arpa nameserver = nogger.netlab.fh-karlsruhe.de
nogger.netlab.fh-karlsruhe.de   internet address = 10.0.0.3
> exit
```

Utilities wie *nslookup* oder *dig* werden im späteren Verlauf des Labors zur Fehlerdiagnose zum Einsatz kommen, wenn eigene Nameserver und Mailserver aufgesetzt oder verwaltet werden sollen.

Cisco Telnet Zugriffe

Abhängig von der Aufgabenstellung sind bestimmte Cisco Router oder Switches von den VMware-Instanzen erreichbar. Diese werden im Labor über das Cisco Command Line Interface interaktiv konfiguriert. Nachfolgend ist der Authentifizierungsdialog nach erfolgreichem Verbinden gezeigt:

```
[student@vm3 student]# telnet burgerking
Trying 10.0.0.11...
Connected  to  burgerking.netlab.fh-karlsruhe.de
(10.0.0.11).
Escape character is '^]'.

User Access Verification

Password: ********
BurgerKing>
```

Kapitelinhalt:

- Startup und Running
- User und Privileged
- Interfaces und Routing
- Cisco Access Lists

2

Cisco Tutorial

Im Labor befinden sich jeweils vier Cisco 1720 Router (IOS 12.2) und Cisco 2600 Catalyst Switche (IOS 12.1). Diese verfügen über proprietäre Betriebssysteme, das sogenannte Cisco IOS, welches in einer *flash-fähigen* Firmware vorliegen. Die Konfiguration der Cisco Hardware erfolgt über ein Kommandozeilen-Interface (CLI). Von der Verwendung eines Web-Interfaces im Labor wird abgesehen, da der Lernerfolg von der direkten Interaktion mit dem Cisco IOS abhängt.

Basiswissen

Startup und Running

Es wird zwischen den Konfigurationen *startup* und *running* unterschieden. Die *running-config* ist eine flüchtige zur Laufzeit aktive Konfiguration. Sie verliert nach einem Reset der Hardware ihre Gültigkeit. Bei einem Neustart wird die *startup-config* geladen, die sich im *NVRAM* (non-volatile RAM) befindet. Deswegen sollten Konfigurationsänderungen an der running-config, die nach einem Neustart des Geräts noch gültig sein sollen, in die startup-config übernommen werden. Umgekehrt kann von einer Standard startup-config eine modifizierte running-config wieder in den Ursprungszustand versetzt werden.
Wenn also eine gewisse Unsicherheit bezüglich der Korrektheit einer durchgeführten Konfigurationsarbeit herrscht, sollte ein Rückschreiben der running auf die startup-config unterlassen werden. Bei einem Konnektivitätsverlust kann dann im Notfall durch einen Reset der Cisco die funktionierende startup-config wieder geladen werden

User und Privileged

Um bestimmte Konfigurationen einer Cisco anzuschauen reicht ein einfacher Telnet Login mit Passwort. Danach befindet man sich im Benutzer Modus. Um z.B. jedoch Änderungen an der Interface-Konfiguration vorzunehmen oder die startup-config oder running-config einzusehen, bedarf es des Wechsels vom Benutzer in einen privilegierten Modus. Dies ist eine ähnliche Sicherheitsvorkehrung wie beim Benutzerwechsel zum *superuser* bei UNIX.

Nach dem Anmelden auf der Cisco kann die eingebaute Hilfe verwendet werden, die rudimentäre Informationen zu den möglichen Kommandos innerhalb des jeweiligen Modus (privilegiert oder Benutzer) gibt. Hier ein Ausschnitt:

```
[student@vm3 student]# telnet burgerking
Trying 10.0.0.11...
Connected to burgerking.netlab.fh-karlsruhe.de
(10.0.0.11).
Escape character is '^]'.

User Access Verification

Password: ********

BurgerKing>?
Exec commands:
  disable          Turn off privileged commands
  enable           Turn on privileged commands
  exit             Exit from the EXEC
  help             Description of the interactive help system
  ping             Send echo messages
  ppp              Start IETF Point-to-Point Protocol (PPP)
  show             Show running system information
  ssh              Open a secure shell client connection
  systat           Display information about terminal lines
  telnet           Open a telnet connection
  traceroute       Trace route to destination

BurgerKing>show ?
  clock          Display the system clock
  flash:         display information about flash: file system
  history        Display the session command history
  hosts          IP domain-name, lookup style, nameservers, and host table
  sessions       Information about Telnet connections
  snmp           snmp statistics
  ssh            Status of SSH server connections
  users          Display information about terminal lines
  version        System hardware and software status
```

Ein angefügtes *?* nach einem Befehl, z.B. *show ssh ?* gibt mögliche Parameter mit einer Kurzerläuterung wieder. Befehle und Parameter können desweiteren abgekürzt und mit der Tabulatortaste erweitert werden. Dies erspart sehr viel Tipparbeit und hilft fehlendem Erinnerungsvermögen auf die Sprünge.

Um nun in den privilegierten Modus zu gelangen, muß das Kommando *enable* verwendet werden. Es findet ein weiterer Authentifizierungsdialog statt, soweit für den privilegierten Modus ein Passwort vorgesehen wurde:

```
BurgerKing>enable
Password: ********
BurgerKing#
```

Wie unter eine UNIX-Shell wird der Wechsel des Modus durch einen anderen Prompt signalisiert. Es stehen nun alle vorherigen Kommandos und viele zusätzliche bereit. Von diesem Punkt an, sollte mit entsprechender Vorsicht vorgegangen werden. Wichtige Befehle zum Anzeigen von Konfigurationen sind:

```
BurgerKing#show running-config
BurgerKing#show startup-config
BurgerKing#show interfaces (FastEthernet|Ethernet|Serial) 0-9
```

Um nun die Cisco und ihre Interfaces über die Telnet-Sitzung interaktiv zu konfigurieren ist folgender Befehl notwendig:

```
BurgerKing#configure terminal
```

Danach wechselt der Prompt (config). Nun kann ein Interface zur Konfiguration angegeben werden:

```
BurgerKing(config)#interface Ethernet 0
```

Als Beispiel eine einfache Interface Konfiguration:

```
BurgerKing(config-if)#ip address 10.22.33.66 255.255.255.252
BurgerKing(config-if)#ip unreachables
BurgerKing(config-if)#mtu 1460
BurgerKing(config-if)#no shutdown
```

Damit ist das Interface bereits aktiviert (no shutdown). Leider kann von der momentanen Position aus die Konfiguration des Interfaces nicht eingesehen werden. Um wieder zur Ausgangsposition zu gelangen ohne den privilegierten Modus zu verlassen, betätigt man die Tastenkombination <CTRL-Z> oder gibt das Kommando *end* ein:

```
BurgerKing(config-if)#^Z
BurgerKing(config-if)#end
```

Nun kann die aktuelle Konfiguration des Interfaces eingesehen werden:

```
BurgerKing#show interfaces Ethernet 0
```

Alle Konfigurationsänderungen befinden sich nun in der running-config. Sie können auf zwei Arten in die startup-config übernommen werden:

```
BurgerKing#write
Building configuration...
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!![OK]
BurgerKing#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!![OK]
```

Um nach erfolgter Konfiguration wieder in den Benutzer Modus zu gelangen und danach die Telnet-Sitzung zur Cisco zu beenden sind folgende Kommandos notwendig:

```
BurgerKing#disable
BurgerKing>exit
Connection closed by foreign host.
```

Wobei vor Beendigung mit *exit* der Moduswechsel nicht zwingend notwendig ist. Unter [2-1] finden sich weitreichende Beschreibungen aller Konfigurationsmodi mit Beispielen. Desweiteren sind alle IOS-Kommandos aufgeführt.

Kurzreferenz

Für die Arbeit im Labor reicht zunächst ein kleinerer Ausschnitt aus der Menge von IOS-Kommandos. Hauptsächlich werden Interfaces und Routing zu konfigurieren sein.

Interface-Konfiguration

Die Router und Switches im Netzwerklabor verfügen über Ethernet-, FastEthernet- und Serial-Interfaces, wobei die Catalyst-Switche nur FastEthernet-Interfaces beherbergen.

Ethernet Interface

Die Konfiguration eines Ethernet- oder FastEthernet-Interfaces ist sehr geradlinig. Im einfachsten Falle sind ledglich die IP-Adresse und Netzmaske festzulegen:

```
BurgerKing (config)# interface Ethernet 0
BurgerKing (config-if)# ip address 10.1.0.254 255.255.255.0
BurgerKing (config-if)# description LAN zu VMnet0
BurgerKing (config-if)# no shutdown
```

Bei Bedarf können fortgeschrittene Parameter, wie MTU, Duplexverhalten, Keepalive und Traffic-Shaping angegeben werden.

Serial Interface

Bei einem synchronen seriellen Interface wie zweifach in den Routern BurgerKing und McDonalds vorhanden, ist zunächst festzustellen, welcher der beiden Kommunikationspartner den Synchronisierungstakt vorgibt. Bei diesem Router muss das serielle Interface als *DCE* (Data Communication Equipment) definiert werden. Dies wird über das Kommando *clock rate* erreicht:

```
BurgerKing (config)# interface Serial 0
BurgerKing (config-if)# ip address 10.10.1.11 255.255.255.0
BurgerKing (config-if)# clock rate 2000000
BurgerKing (config-if)# encapsulation hdlc
BurgerKing (config-if)# description Verbindung zu McDonalds
BurgerKing (config-if)# no shutdown
```

Das Interface des anderen Routers wird durch das Weglassen des Kommandos *clock rate* bzw. der Negation mittels *no clock rate* automatisch zum *DTE* (Data Terminal Equipment):

```
McDonalds (config)# interface Serial 0
McDonalds (config-if)# ip address 10.10.1.14 255.255.255.0
McDonalds (config-if)# no clock rate
McDonalds (config-if)# encapsulation hdlc
McDonalds (config-if)# description Verbindung zu BurgerKing
McDonalds (config-if)# no shutdown
```

Die Clock Rate spezifiziert die Anzahl der Bits, die pro Sekunde über die serielle Leitung übertragen werden können. Im einfachsten Falle kann bei zwei Cisco Routern die serielle Leitungsenkapsulierung mit einem Cisco proprietärem HDLC-Protokoll aufgebaut werden.

Interfaces können administrativ abgeschaltet werden, dabei jedoch ihre Konfiguration behalten. Das jeweils letzte Kommando in den Beispielkonfigurationen – no shutdown – aktiviert ein Interface. Um zu überprüfen, dass sowohl das Interface als auch die Leitung aktiv sind, kann der Status des Interfaces oder seines Controllers angezeigt werden:

```
BurgerKing#show interfaces Serial 0
Serial0 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 10.10.1.11/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, loopback not set
  Keepalive set (10 sec)
  ...
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     206482 packets input, 4955952 bytes, 0 no buffer
     Received 206482 broadcasts, 0 runts, 0 giants, 0 throttles
     1 input errors, 0 CRC, 1 frame, 0 overrun, 0 ignored, 0 abort
     240933 packets output, 16766239 bytes, 0 underruns
     0 output errors, 0 collisions, 5 interface resets
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions
     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Die folgenden Informationen helfen bei der Fehlerbehebung. Einen ersten wichtigen Aufschluß über den Status erhält man bereits in der ersten (fett markierten) Zeile. Hier wird sowohl der Status des Interface Controllers als auch der des Leitungsprotokolls angezeigt. Es können folgende Zustände unterschieden werden:

1. **Serial0 is up, line protocol is up**
   Dieses Interface befindet sich im operablen Zustand.

2. **Serial0 is down, line protocol is down**
   Der Router kann kein Leitungsträgersignal (CD – Carrier Detect) erkennen. In diesem Falle liegt entweder ein Verkabelungsproblem oder eine defekte Hardware vor.

3. **Serial0 is up, line protocol is down**
   Hier kann zwar das Leitungsträgersignal erkannt werden, der Aufbau des Leitungsprotokolls (Cisco HDLC) schlägt jedoch fehl. Da DCE und DTE direkt über die Interface Hardware abgehandelt werden liegt hier höchstwahrscheinlich ein Hardwarefehler beim lokalen oder entfernten Router vor.

4. **Serial0 is up, line protocol is down (looped)**
   Der Router hat eine Schleifenkonfiguration erkannt, die normalerweise von Providern und Telcos zur Fehlereingrenzung verwendet wird. In Labor zeigt dieser Status schlichtweg eine Fehlkonfiguration des Interfaces an.

5. **Serial0 is up, line protocol is down (disabled)**
   Das Interface wurde vom lokalen Router selbständig deaktiviert. Die Ursachen können in hohen Fehlerraten (Bitübertragungsfehler) der Verbindungsleitung oder einem Hardwarefehler des lokalen Routers liegen.

6. **Serial0 is administratively down, line protocol is down**
   Das Interface wurde manuell absichtlich deaktiviert oder vom Router automatisch in diesen Zustand versetzt, da z.B. die gleiche IP- oder MAC-Adresse im Netzwerk oder auf dem Router mehrfach vergeben wurde.

Einige dieser Zustände können im Labor nicht beobachtet werden, da die seriellen Leitungen direkte Verbindungen zwischen den Routern sind und nicht über weite Überlandstrecken mit Telco-Repeatern und –Switche geführt werden. Es folgt nun eine Beschreibung der wichtigsten Fehlerzähler:

1. **Input Errors**
   Beschreibt die Summe von CRC- und Frame-Errors. Macht diese Summe mehr als 1% des gesamten eingehenden Verkehrsaufkommens aus, liegen entweder beträchtliche Störeinflüsse vor, die auf die Verbindung einwirken oder die Hardware (Interface, Router, Kabel, Stecker) unterliegt einem Defekt. CRC- und Frame-Errors sind meist auf rauschende bzw. auf zu lange Leitungen zurückzuführen. Bei solchen Bedingungen läßt der Verlust der Signalstärke die Rahmengrenzen verschwimmen.

2. **Interface Resets**
   Auf den Interfaces ist standardmäßig die Versendung sogenannter Keepalive-Pakete konfiguriert. Ein Interface-Reset findet dann statt, wenn solch ein erwartetes Keepalive-Paket verpaßt wird. Auch hier kann wiederum eine gestörte Leitung verantwortlich sein, jedoch ist die Wahrscheinlichkeit höher, daß die Leitung schlichtweg mit Verkehr überladen ist, so daß die Antwort auf ein Keepalive-Paket nicht mehr rechtzeitig beim Kommunikationspartner eintrifft.

3. **Carrier Transitions**
   Dieser Fehler tritt bei einer Unterbrechung des Trägersignals auf. Ein Interface-Reset kann solch einen Fehler verursachen. Ein einfacher Wackelkontakt am Kabel oder Stecker kann ebenso der Grund sein wie starke Störeinflüsse. Bei Überlandstandleitungen kann z.B. ein Gewitter entlang der Verbindungsstrecke das Trägersignal stören.

Routing-Konfiguration

Es können statische Routen und nahezu alle Formen dynamischer Routingprotokolle konfiguriert werden. Dabei findet standardmäßig das Classless-Inter-Domain-Routing (CIDR) Anwendung. CIDR führt jedoch bei einfachen Routingprotokollen wie RIP oder IGRP zu Problemen führt. Diese werden hier deshalb nicht behandelt.

Statisches Routing

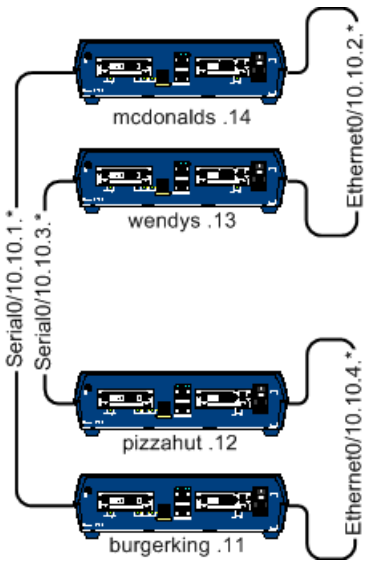
Grundsätzlich kann eine Route zu einem Zielsystem direkt über ein Interface des Routers oder über eine IP-Adresse des nächsten Hops definiert werden. Ein mit einer IP-Adresse und Netzmaske versehenes Interface bildet in der lokalen Routingtabelle automatischen einen Eintrag für das angeschlossene Netz und muss nicht weiter konfiguriert werden:

```
BurgerKing#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 3 subnets
C       10.10.1.0 is directly connected, Serial0
C       10.10.4.0 is directly connected, Ethernet0
C       10.0.0.0 is directly connected, FastEthernet0
```

Die erste wichtige Information, ist das Fehlen einer Default-Route. Dies wird durch die Angabe *Gateway of last resort* angezeigt. Insgesamt ist das Class A Network 10.0.0.0 in drei Subnetze zerlegt, welche alle mit der gleichen Netzmaske (255.255.255.0 oder CIDR /24) versehen sind. Cisco Router unterstützen jedoch auch die Zerlegung in variable Subnetzmasken, was das folgende Beispiel aufzeigen wird.



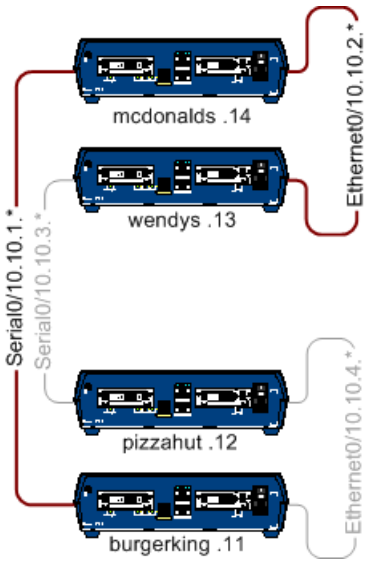
Es soll eine statische Route von BurgerKing zu Wendys gesetzt werden. Die beiden Router sind nicht direkt miteinander verbunden, jedoch über zwei verschiedene Wege erreichbar – einmal über PizzaHut und einmal über McDonalds. Als erste Variante soll Wendys über McDonalds erreicht werden. McDonalds ist über Standard Ethernet mit Wendys verbunden und Wendys hat die Adresse 10.10.2.13 in diesem Netzwerk. Wir konfigurieren dementsprechend eine statische Route auf BurgerKing für die Host-Adresse 10.10.2.13/32 über die Host-Adresse 10.10.1.14/32:

```
BurgerKing(config)#ip route 10.10.2.13 255.255.255.255 10.10.1.14
BurgerKing#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP

Gateway of last resort is not set
```

```
     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C       10.10.1.0/24 is directly connected, Serial0
C       10.10.4.0/24 is directly connected, Ethernet0
C       10.0.0.0/24 is directly connected, FastEthernet0
S       10.10.2.13/32 [1/0] via 10.10.1.14
```

Da eine Host-Route zur IP-Adresse des Ethernet0 Interfaces von Wendys gesetzt wurde, ist das Class A Netzwerk 10.0.0.0 nun variabel unterteilt. Bei einer Netz-Route zu 10.10.2.0/24 wären 4 Subnetze mit der gleichen Maske das Resultat gewesen. Der Test der Route über ICMP Echo-Request/Reply (Ping) schlägt in diesem Stadium aber fehl:




```
BurgerKing#ping 10.10.2.13

Sending 5, 100-byte ICMP Echos to 10.10.2.13, timeout is
2 seconds: .....
Success rate is 0 percent (0/5)
```

Der Grund hierfür liegt, daß die erwarteten ICMP Echo-Reply Pakete vom Router Wendys verworfen werden, da dieser keine Route zur Absenderadresse von BurgerKing kennt. Eine Möglichkeit diesen Fehler zu beheben, bestünde darin, den gleichen Weg zurück zu nehmen. Auf Wendys muss dementsprechend ebenfalls eine statische Route zu BurgerKing gesetzt werden, die wiederum über McDonalds geht. In diesem Falle ist die Alternativroute über Pizzahut ausgeblendet und hellgrau dargestellt.

```
Wendys(config)#ip route 10.10.1.11 255.255.255.255 10.10.2.14
Wendys#ping 10.10.1.11

Sending 5, 100-byte ICMP Echos to 10.10.1.11, timeout is 2 seconds:.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/4/4 ms
```

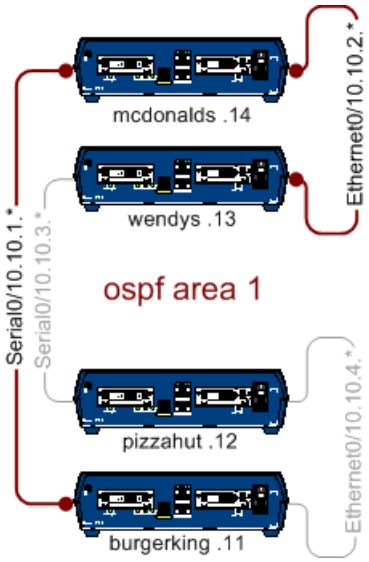
Beim ersten Ping geht noch ein Paket verloren, was höchstwahrscheinlich mit dem Fehlen von ARP-Informationen (IP-Adressen zu Link-Layer-Adressen) verbunden ist. Bei einem darauffolgenden Ping in die umgekehrte Richtung, also von BurgerKing ausgehend, sind die ARP-Caches der Router bereits gefüllt und es kommt zu keinerlei Paketverlusten mehr:

```
BurgerKing#ping 10.10.2.13

Sending 5, 100-byte ICMP Echos to 10.10.2.13, timeout is 2 seconds:!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

Dynamisches Routing

Die Router BurgerKing und Wendys sollen die notwendigen Routen nun ohne explizite statische Konfiguration selbst in Erfahrung bringen. Hierzu verwenden sie ein dynamisches Routingprotokoll. In diesem Beispiel wird OSPF verwendet, welches nach dem Link-State-Verfahren arbeitet und sich



eine Topologie des gesamten Netzwerkes zusammenbaut. Jeder Router, der am OSPF-Routing teilnimmt hat im Idealfall Kenntnis des gesamten Netzwerkes.
Damit die Router BurgerKing und Wendys den oben bereits statisch erzwungenen Weg über McDonalds nun via OSPF herausfinden, müssen diese Router wie folgt konfiguriert werden:
Die Router bilden eine gemeinsame OSPF Area. Innerhalb dieser Area nehmen bestimmte Router-Interfaces an der OSPF-Kommunikation teil. Diese sind in der Abbildung als dicke (rote) Punkte gekennzeichnet. Jeder Router muß sich bei OSPF desweiteren über eine ID ausweisen. Wir verwenden hier die letzte Ziffer der jeweiligen Router-IP, die neben den Namen des Routers steht. Die notwendigen Konfigurationskommandos sind nun:

```
BurgerKing(config)#router ospf 11
BurgerKing(config-router)#network 10.10.1.0 0.0.0.255 area 1

Wendys(config)#router ospf 13
Wendys(config-router)#network 10.10.2.0 0.0.0.255 area 1

McDonalds(config)#router ospf 14
McDonalds(config-router)#network 10.10.1.0 0.0.0.255 area 1
McDonalds(config-router)#network 10.10.2.0 0.0.0.255 area 1
```

Über die inversen Netzmasken werden für die jeweiligen Netzwerke, für die OSPF verwendet werden soll, die zutreffenden Interfaces erkannt. Zwei nützliche Befehle, um den Status von OSPF Interfaces und erkannter Nachbar-Router zu ersehen sind:

```
BurgerKing#show ip ospf interface
Serial0 is up, line protocol is up
  Internet Address 10.10.1.11/24, Area 1
  Process ID 11, Router ID 10.10.4.11, Network Type POINT_TO_POINT, Cost: 64

BurgerKing#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.10.2.14        1   FULL/  -        00:00:33    10.10.1.14      Serial0
```

Der einzige Nachbar von BurgerKing ist hier McDonalds, was auch genau unseren Vorstellungen entspricht, da über PizzaHut nicht geroutet werden soll. OSPF fügt nun automatisch eine Route für das Netzwerk, welches hinter McDonalds liegt, und diesen mit Wendys verbindet, hinzu:

```
BurgerKing#show ip route
     10.0.0.0/24 is subnetted, 4 subnets
C       10.10.1.0 is directly connected, Serial0
O       10.10.2.0 [110/74] via 10.10.1.14, 00:05:49, Serial0
C       10.10.4.0 is directly connected, Ethernet0
C       10.0.0.0 is directly connected, FastEthernet0
```

Wendys hat es seinerseits geschafft, OSPF auf dem Standard Ethernet Interface zu aktivieren und eine Route über McDonalds zum gemeinsamen Netz von BurgerKing und McDonalds aufzubauen:

```
Wendys#show ip ospf interface
Ethernet0 is up, line protocol is up
  Internet Address 10.10.2.13/24, Area 1
  Process ID 13, Router ID 10.10.3.13, Network Type BROADCAST, Cost: 10

Wendys#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.10.2.14        1   FULL/BDR        00:00:39    10.10.2.14      Ethernet0

Wendys#show ip route
C    192.168.13.0/24 is directly connected, FastEthernet0
     10.0.0.0/24 is subnetted, 4 subnets
O       10.10.1.0 [110/74] via 10.10.2.14, 00:07:07, Ethernet0
C       10.10.2.0 is directly connected, Ethernet0
C       10.10.3.0 is directly connected, Serial0
C       10.0.0.0 is directly connected, FastEthernet0
```

Der Router McDonalds hat OSPF gleich an zwei Interfaces gebunden, erhält aber keine Routing-Updates mehr, da er beide Netze über seine Interfaces bereits kennt:

```
McDonalds#show ip ospf interface
Ethernet0 is up, line protocol is up
  Internet Address 10.10.2.14/24, Area 1
  Process ID 14, Router ID 10.10.2.14, Network Type BROADCAST, Cost: 10
Serial0 is up, line protocol is up
  Internet Address 10.10.1.14/24, Area 1
  Process ID 14, Router ID 10.10.2.14, Network Type POINT_TO_POINT, Cost: 64

McDonalds#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.10.3.13        1   FULL/DR         00:00:35    10.10.2.13      Ethernet0
10.10.4.11        1   FULL/  -        00:00:32    10.10.1.11      Serial0
```

ACL-Konfiguration

Hier soll kurz der Einsatz und die Verwendung von sogenannten Access Lists (Zugriffslisten) beschrieben werden. Solche Listen können bei einem Cisco Router auf vielfältige Art und Weise verwendet werden. In ihrer ursprünglichen Bedeutung werden sie hauptsächlich für das Packet-Filtering auf Basis einer Default Policy Deny verwendet. Zum Traffic-Filtering wird der gleiche Mechanismus auf Basis einer Default Policy Allow verwendet, um bei bestimmten Verbindungen die Bandbreite zu konservieren.
Generell fungieren die ACLs als Beschreibung für zu identifizierende Pakete. Solche Beschreibungen, auch Filter genannt, werden in Reihe geschaltet und bilden dann eine Durchlaufkette (siehe auch Netfilter). Ein Paket wird solange durch die Kette gereicht, bis ein Filter auf das Paket zutrifft. Zu

jedem Filter existiert eine zugeordnete eindeutige Aktion (Deny oder Permit), die bei einem Treffer ausgeführt wird. Stimmt kein einziger der Filter in der Kette mit dem Paket überein, greift die zuvor definierte Default Policy implizit ein und eine Entscheidung/Aktion wird erzwungen.

Als Beispiel für den Einsatz von ACLs soll jeglicher Verkehr von BurgerKing's seriellem Interface auf dem Router Wendys verworfen werden. Für die Definition von ACLs sind wieder inverse Netzmasken notwendig (Cisco nennt diese Wildcards). Die inversen Netzmasken werden über eine bitweise OR-Operation auf die IP-Adressen angewandt. Ein gesetztes Bit in solch einer inversen Netzmaske spiegelt also eine Don't Care Condition wieder, nicht gesetzte Bits erzwingen eine Übereinstimmung.

```
Wendys(config)#access-list 1 deny 10.10.1.11 0.0.0.0
Wendys(config)#interface Ethernet 0
Wendys(config-if)#ip access-group 1 in
```

Die ACL muß nun noch zugeordnet werden. In diesem Falle dem Standard Ethernet Interface von Wendys in eingehender Richtung. Dies ist der Weg, den das statische oder durch OSPF etablierte Routing, den Paketen vorschreibt. Versuchen wir nun die IP-Adresse von Wendys Ethernet Interface zu pingen, erwartet uns eine böse Überraschung:

```
BurgerKing#ping 10.10.2.13

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.2.13, timeout is 2 seconds:U.U.U
Success rate is 0 percent (0/5)
```

In diesem Falle scheint Wendys die Pakete nicht einfach nur zu verwerfen sondern antwortet mit einem ICMP Reject (Destination Unreachable) Paket auf jeden ICMP Echo Request von BurgerKing. Dies kann für das entsprechende Interface auch deaktiviert werden:

```
Wendys(config)#interface Ethernet 0
Wendys(config-if)#no ip unreachables
```

Damit laufen die Pakete wirklich ins Leere und der Absender erhält einen Effekt wie bei einem schwarzen Loch. Als nächstes sollen spezifischere Pakete erkannt werden. Hierzu muss eine andere Kategorie von ACLs verwendet werden. Standard IP ACLs haben eine Nummer von 1-99. Erweiterte IP ACLs laufen von 100-199. Es soll nun auf allen seriellen Verbindungen aller Router die Weiterleitung von Bandbreitenkonsumierenden Anwendungen wie *kazaa* unterbunden werden:

```
Router(config)#access-list 101 deny tcp any any eq 1214
Router(config)#interface Serial 0
Router(config-if)#ip access-group 101 out
Router(config-if)#ip access-group 101 in
```

Diese ACL verwendet speziell die TCP-Erweiterung, mit der auch der Status einzelner TCP-Flags (SYN, ACK, FIN) überprüft werden kann. In diesem Fall wird schlichtweg jeglicher Verkehr, der von einer beliebigen Ursprungsadresse (any) zu einer beliebigen Zieladresse (any) fließt und einen TCP-

Server-Port 1214 (eq 1214) anspricht, vom Router verworfen. Die ACL wird dann auf dem seriellen Interface in beide Richtungen aktiviert. Es wird nicht auf einen bestimmten TCP-Client-Port oder einen Bereich, z.B. unprivilegierter Bereich (1024-65535) überprüft.

Mehrere Filter in einer Kette ergeben sich einfach durch Definition einer neuen ACL mit der gleichen Kategorienummer. Die neue ACL wird dann ans Ende der Kette angehängt. Eine ganze Kette kann wie folgt gelöscht werden:

```
Router(config)#no access-list 101
```

Es ist nicht möglich einzelne Einträge aus der Kette herauszulöschen oder ACLs an bestimmten Stellen in einer Kette einzutragen. Die Bedienung durch das Command Line Interface ist hier sehr träge und umständlich. Es bietet sich daher ein Zusammenfassen der Filter ACLs in einer Textdatei an. Diese kann anschließend via TFTP auf den Router übertragen werden:

```
no access-list 101
access-list 101 permit tcp 10.10.1.0 0.0.0.255 gt 1023 host 10.0.0.1 eq 22
access-list 101 permit tcp any 10.0.3.0 0.0.0.255 established
access-list 101 permit udp 10.10.2.0 0.0.0.255 host 10.0.0.1 eq 161
access-list 101 deny icmp 10.10.4.0 0.0.0.255 any 3 9
access-list 101 deny ospf 10.10.2.0 0.0.0.255 any
access-list 101 permit udp any eq 500 any eq 500
access-list 101 permit 51 any any
access-list 101 permit 50 any any
```

In der ersten ACL wird ein neues Schlüsselwort *host* verwendet, um die inverse Netzmaske 0.0.0.0 überflüssig zu machen. Desweiteren findet sich hier ein Operator *gt* ähnlich bereits oben vorgestellten Operator *eq*. Während *eq* (equal) einen einzelnen Port spezifiziert, wird über *gt* (greater) hier der unprivilegierte Bereich von 1024-65535 angegeben. Alternativ kann mit dem Operator *range* und zwei folgenden Portnummern ein beliebiger Bereich angegeben werden.

Die zweite ACL verwendet am Ende das Schlüsselwort *established* und sorgt hiermit dafür, daß keine Neuverbindungen von einem beliebigen Netz (any) zum Netzwerk 10.0.3.0/24 aufgenommen werden können. Es gestattet nur Pakete nach 10.0.3.0/24, die in Antwort auf eine von diesem Netzwerk aus aufgebaute Verbindung erfolgen. Solche Pakete tragen somit kein alleiniges SYN-Flag, sondern müssen ein SYN/ACK, ACK oder FIN aufweisen.

Die dritte ACL erlaubt über einen UDP-Filter die SNMP-Kommunikation vom ersten Ethernet Netzwerk zum ersten PC. Bei der vierten ACL handelt es sich um einen Filter für ICMP-Nachrichten. Bei solch einem Filter folgen auf die Quell- und Zieladressen ein ICMP Message Type (hier 3 : Destination Unreachable) und ein optionaler ICMP Message Code (hier 9 : Network Administratively Forbidden).

Mit der fünften ACL wird der Austausch von dynamischen Routingprotokollinformationen via OSPF (IP-Protokoll 89) ausgehend vom Ethernet Netzwerk unterbunden. Die letzten drei ACLs gestatten sämtliche IPsec-Kommunikation (IKE mit UDP Client- und Server-Port 500, AH mit IP-Protokoll 51 und ESP mit IP-Protokoll 50).

Kapitelinhalt:

- bash und vim
- Netcat und Telnet
- nslookup und dig
- TCPdump/Headers
- Netfilter/IPtables

3

Linux Tutorial

Nachdem nun die Cisco Seite des Netzwerklabors beleuchtet wurde, wird hier abschließend noch etwas auf Linux Netzwerkkonfiguration und Fehlerbehebung eingegangen, sowie kurz der Umgang mit der Shell und dem Standard-Editor erläutert.

bash und vim

Ein rudimentär installiertes Linux-System verfügt heutzutage mit hoher Wahrscheinlichkeit über zwei Komponenten. Die eine ist die *bash*, eine Shell, die auf die altehrwürdige UNIX-Bourne-Shell aufbaut. Die andere Komponente ist *vim*, ein antiquiert anmutender, aber leistungsfähiger Texteditor.

bash

Die bash [3-1] (bourne again shell) ist ein moderner UNIX-Kommandointerpreter mit weitreichender Skriptfunktionalität. Auf die Programmierung mit der bash wird hier nicht eingegangen, es werden jedoch einige Hinweise zur Verwendung von Ein- und Ausgabeumlenkung sowie der Interprozesskommunikation gegeben.

Eingabehilfen und Historie

Die bash hält eine Reihe von Funktionen bereit, um bereits getippte Kommandos wiederzuverwenden oder unfertig eingegebene Dateinamen zu expandieren. Die Bedienung der bash ist somit ähnlich bequem wie beim Cisco Command Line Interface, bei dem ebenfalls durch eine Historie von Kommandos geblättert werden und Kommandozeilen expandiert werden können. Konkret kann dies mit den Cursortasten `<up>` und `<down>` und der Tabulatortaste `<tab>` erreicht werden:

```
bash-2.05$ ser<tab> httpd reload
```

wird dann zu

```
bash-2.05$ service httpd reload
```

und ein Kommando wie

```
bash-2.05$ /etc/init.d/h<tab><tab>
halt   httpd
```

listet alle Dateien in dem Verzeichnis /etc/init.d/ auf, die mit dem Zeichen „h“ anfangen. Wird der Bildschirm irgendwann zu voll und unübersichtlich, kann dieser mit <CTRL>+<l> gelöscht werden. Soll nach früheren Kommandos gesucht werden, kann mittels <CTRL>+<r> eine Suche über die Kommandohistorie aktiviert werden. Für das Benutzereigene Home-Verzeichnis gibt es eine Abkürzung, die von der bash ebenfalls expandiert werden kann:

```
bash-2.05$ cat ~/.vimrc
```

Wird dann implizit von der bash nach */home/username/.vimrc* umgesetzt, jedoch nicht mehr auf der Kommandozeile angezeigt.

Ein- und Ausgabeumleitung

Sowohl die Standardeingabe (normalerweise über die Tastatur) als auch die Standardausgabe (normalerweise über den Bildschirm) können jederzeit auf ein anderes Ziel umgelenkt werden. Gleiches gilt für die Standardfehlerausgabe. Oft soll z.B. die erwartete Eingabe eines Programmes durch den Inhalt einer Datei realisiert werden. Oder die Ausgabe eines Programmes soll nicht auf dem Bildschirm angezeigt sondern in eine Datei geschrieben werden.

```
bash-2.05$ netstat –anp > /var/log/netstat.log
bash-2.05$ ls -1 *.sh > scriptfiles.txt
```

Das erste Beispiel lenkt die Ausgabe mit dem Operator > von *netstat* in eine Datei im Verzeichnis */var/log* um. Das zweite listet alle Dateien im aktuellen Verzeichnis, die über die Endung .sh verfügen und schreibt die Aufzählung ebenfalls in eine Datei.

Der umgekehrte Weg, die Eingabeumlenkung wird über den Operator < erreicht. Damit können Programme, die normalerweise Eingaben von der Tastatur erwarten, diese stattdessen aus einer Datei beziehen. Für die Umlenkung der Ausgabe bestehen noch zwei weitere Möglichkeiten. Eine ist das Anhängen an bereits bestehende Dateien über den Operator >>, wie das folgende Beispiel zeigt:

```
bash-2.05$ ls –hals /etc/init.d >> ~/filelist.txt
```

Alternativ kann auch die Standardfehlerausgabe einzeln über den Operator 2> oder kombiniert mit der Standardausgabe über den Operator >& in eine Datei umgelenkt werden:

```
bash-2.05$ gcc source.c 2> /var/log/error.log
bash-2.05$ gcc source.c >& /var/log/compile.log
bash-2.05$ gcc source.c >>& /var/log/running.log
```

Der Superuser

Auch auf Linux/UNIX-Systemen wird wie bei einem Cisco-IOS zwischen verschiedenen User-Privilegien unterschieden. Es gibt normale User und den sogenannten Superuser, auch root genannt. Um zu erkennen, in welchem Modus ein Benutzer sich gerade befindet, zeigt die bash über ein besonderes Zeichen im Prompt an:

```
[user@host user]$ ...
[root@host root]# ...
```

Auch hier wird der Privilegienwechsel wie beim Cisco CLI mit einem #-Zeichen angezeigt. Der Wechsel erfolgt mit dem Kommando:

```
[user@host user]$ su -
```

Damit kann vom Useraccount zu root gewechselt werden, sofern der User über die notwendigen Rechte verfügt und das Passwort für root kennt. Das Minuszeichen nach dem Befehl gibt an, daß eine sogenannte Login-Shell von root verwendet werden soll. Von root aus kann zu jedem anderen beliebigen Useraccount ohne Kenntnis des Passworts gewechselt werden:

```
[root@host root]# su - user
```

Eine so erhaltene oder auch jede andere Shell kann entweder über die Befehle *exit*, *logout* oder die Tastenkombination <CTRL>+<d> verlassen werden.

Interprozesskommunikation mit Pipes

Mit Pipes lassen sich Ausgaben eines Programmes über eine Zwischenpipeline speichern. Solche Informationen können von einem gleichzeitig aktiven zweitem Programm aus der Pipe gelesen werden:

```
[root@host root]# iptables -L -n | less
```

Der Pipe-Operator | steht hierbei zwischen dem ausgebenden und dem einlesenden Programm. In diesem Beispiel werden die über *iptables* gesetzten Paketfilterregeln, die sich über mehrere Bildschirmseiten erstrecken können, an das Anzeigeprogramm less weitergegeben.

```
[root@host root]# find / -name ´*sh´ | grep scripts > filelist.txt
```

In diesem Beispiel werden alle Dateien gefunden, deren Namen die Zeichen sh am Ende tragen (also auch bash). In der Ausgabeliste wird dann via *grep* nach solchen Zeilen gesucht, die das Wort scripts enthalten. Anschließend wird die gefiltertete Ausgabe in eine Datei umgelenkt.

vim

Der vim-Editor [3-2] (vi-Improved) ist die freie Linux Variante des berüchtigten vi-Editors. Sein Einsatz im Labor ist zwingend erforderlich, um frisch installierte Basissysteme überhaupt konfigurieren zu können. Desweiteren bietet der original vi-Editor quasi einen UNIX-weiten Standard, bedarf geringer Ressourcen und funktioniert auch in Abwesenheit einer graphischen Benutzeroberfläche. Der vim-Editor hat bereits einen beträchtlichen Funktionsumfang, u.a. Syntax-Highlighting und Window-Splitting eingebaut.

Es existiert ein symbolischer Link von vi auf vim, so daß ein Aufruf von

```
bash-2.05 $ vi [dateiname]
```

in Wirklichkeit den vim-Editor im vi-Modus startet. Wird vim direkt und nicht über den symbolischen Link gestartet

```
bash-2.05 $ vim [dateiname]
```

stehen erweiterte Funktionen wie das bereits oben erwähnte Syntax-Highlighting zur Verfügung. Solche Funktionen können in einer Konfigurationsdatei *~/.vimrc* festgelegt werden und sind dann bei jedem Start von vim verfügbar. Als Beispiel hier zunächst eine sinnvolle Konfigurationsdatei für vim:

```
set vb       # visual-bell
set ruler    # cursor position
set number   # line numbers
syntax on    # syntax highlighting
```

Um diese Datei anzulegen, muss bereits der vim-Editor benutzt werden. Dabei ist die folgende Befehlssequenz absolutes Grundwissen zur Bedienung des Editors: <ESC><:><!><q>

Damit kann der Editor jederzeit verlassen werden, bzw. bei mehreren geöffneten Fenstern wird das jeweils aktive geschlossen. Der wesentliche Unterschied von vim zu anderen Editoren ist das Vorhandensein von verschiedenen Betriebsmodi. Es gibt selbstverständlich einen Editiermodus, der jederzeit mit <ESC> verlassen werden kann. Jetzt befindet sich der Editor in einem neutralen Modus, von dem aus wieder in den Editiermodus oder in einen Kommandozeilenmodus mittels <:> gewechselt werden kann. Auch dieser Modus kann mittels <ESC> wieder verlassen werden. Wie wird nun der Editiermodus aktiviert?

<i>	Einfügen vor aktueller Cursorposition
<I>	Einfügen am Anfang der Zeile
<a>	Anhängen nach aktueller Cursorposition
<A>	Anhängen am Ende der Zeile
<R>	Überschreibmodus ab aktueller Position aktivieren
<o>	Neue Zeile unter aktueller Zeile einfügen
<O>	Neue Zeile oberhalb aktueller Zeile einfügen

Alle diese Befehle versetzen den Editor entweder in den Insert- oder Replace Editier-Modus. Dieser muß explizit mit <ESC> wieder verlassen werden, um in den neutralen Modus zu gelangen. Aus dem neutralen Modus können weitere Befehle verwendet werden. Nach dem Abschluß des jeweiligen Befehls kehrt der Editor automatisch in den neutralen Modus zurück:

<r>	Zeichen an aktueller Cursorposition ersetzen
<d><d>	Aktuelle Zeile löschen
<y><y>	Aktuelle Zeile kopieren
<p>	Gerade gelöschte oder kopierte Zeile an Cursorposition einfügen
<u>	Letzte Änderung rückgängig machen, mehrmals anwendbar
<x>	Wie Tastenfunktion <Entfernen>
<X>	Wie Tastenfunktion <Backspace>

Abhängig vom Terminaltyp und der Interpretation der Richtungstasten, muss gegebenenfalls der Editiermodus zum Bewegen des Cursors verlassen werden. Normalerweise funktioniert die Steuerung des Cursors über die Richtungstasten sowohl im Editiermodus als auch im neutralen Modus. Sollte dies nicht der Fall sein, müssen im neutralen Modus folgende Tasten verwendet werden:

<h>,<j>,<k>,<l>	Cursor links, runter, hoch, rechts
<w>, <b>	nächstes, vorheriges Wort
<e>	Ende des aktuellen Wortes

Um sich zu einer bestimmten Zeilennummer zu bewegen, stehen folgende Befehle im neutralen Modus zur Verfügung:

<G>	Zur letzten Zeile gehen
<1><G>	Zur ersten Zeile gehen
<*n*><G>	Zur Zeile *n* gehen

Möchte man zwei Editorfenster haben, kann das Fenster horizontal getrennt werden:

<CTRL>+<w><s>	Fenster horizontal splitten
<:><q>	Aktuelles Fenster schließen

Desweiteren kann nach bestimmten Mustern über reguläre Ausdrücke gesucht werden:

</>*pattern*	Sucht vorwärts nach Muster *pattern* durch das Dokument
<?>*pattern*	Sucht rückwärts nach Muster *pattern* durch das Dokument
</>	Sucht vorwärts nach zuletzt angegebenen Muster
<?>	Sucht rückwärts nach zuletzt angegebenen Muster

Der vim-Editor beherbergt außerdem einen sogenannten Visual-Mode, in dem Cut-and-paste Funktionen bereitstehen:

<v>	Visual-Mode aktivieren, mit Cursor Tasten markieren
<V>	Visual-Mode aktivieren und ganze Zeile markieren
<x>	Markierten Bereich ausschneiden
<y>	Markierten Bereich kopieren
<p>	Bereich wieder einfügen

Abschließen sollen noch kurz die wichtigsten Kommandozeilen-Befehle erläutert werden:

<:><w><q>	Speichern und Editor/Fenster schließen
<:><q>	Editor/Fenster ohne Speichern verlassen
<:><q><!>	Editor/Fenster ohne Speichern verlassen, Sicherheitsabfrage ignorieren
<:><e>*name*	Neue Datei mit Namen *name* editieren
<:><w>	Aktuelle Datei speichern
<:><w>*name*	Aktuelle Datei unter anderem Namen abspeichern

Netcat und Telnet

Mittels Telnet und Netcat lassen sich in erster Linie ganz hervorragend auf Klartext-Interaktion basierende Internet-Services wie SMTP und POP3 analysieren:

```
bash-2.05$ telnet smtp.server.net 25
Trying...
Connected to 12.13.14.15.
Escape character is '^]'.
220 texas.cowboy.org ESMTP Sun, 6 Jul 2003 15:20:41 +0200
EHLO relaytest
250-texas.cowboy.org Hello rz06.FH-Karlsruhe.DE [193.196.64.6]
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-AUTH LOGIN PLAIN
250-HELP
mail from: someone@somewhere.org
250 2.1.0 someone@somewhere.org... Sender ok
rcpt to: relaytest@orbs.org
550 5.7.1 relaytest@orbs.org... Relaying denied
quit
221 2.0.0 texas.cowboy.org closing connection
Connection closed.
```

Selbiges hätte mittels Verwendung von Netcat erreicht werden können:

```
bash-2.05$ nc smtp.server.net 25
```

Netcat hat aber einige Vorzüge gegenüber Telnet. Es kann zum einen auch für UDP-Dienste verwendet werden und zum anderen selbst einen Serverdienst für UDP oder TCP bereitstellen. Desweiteren interpretiert Netcat nicht den geschriebenen oder gelesenen Datenstrom und hat

dementsprechend die Möglichkeit, Binärdaten zu übertragen. Eine hervorragende Anwendung ist das Kopieren von Dateien über das Netzwerk mittels Netcat. Hierzu wird eine Serverseite mit Netcat wie folgt aufgesetzt:

```
[root@server root]# nc -l -p 4444 > /tmp/netcat.dat
```

Das Netcat tatsächlich auf diesem TCP-Socket horcht, kann wie folgt überprüft werden:

```
[root@server root]# netstat -anp |grep :4444
tcp    0      0      0.0.0.0:4444        0.0.0.0:*     LISTEN       16195/nc
```

Von einem anderen Host aus, kann nun eine Datei an das Socket geschickt werden:

```
[root@client root]# nc –w 1 server 4444 < /bin/ls
```

Auf dem Server kann nun die übertragene Datei aufgerufen werden, nachdem sie zunächst ausführbar gemacht wurde:

```
[root@server root]# chmod 755 /tmp/netcat.dat
[root@server root]# /tmp/netcat.dat –help
Usage: /tmp/netcat.dat [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuSUX nor --sort.
...
```

Es ist also tatsächlich möglich Binärdateien auf diese Art und Weise über das Netzwerk zu übertragen.

nslookup und dig

Zwei sehr hilfreiche Tools, um DNS-Informationen zu sammeln und zu analysieren sind nslookup und dig. Ersteres ist auch auf Windows NT/2000 Betriebssystemen vorhanden, wird jedoch auf neueren Linux-Betriebssystemen als veraltet gekennzeichnet.

nslookup

Im einfachsten Falle reicht ein Aufruf in der Form:

```
$ nslookup login.fh-karlsruhe.de
Server:         217.172.162.246
Address:        217.172.162.246#53

Non-authoritative answer:
login.fh-karlsruhe.de   canonical name = rz06.RZ-FDDI.fh-karlsruhe.de.
Name:   rz06.RZ-FDDI.fh-karlsruhe.de
Address: 193.196.125.6
```

Dies funktioniert jedoch nur, wenn ein oder mehrere Nameserver in der Datei */etc/resolv.conf* konfiguriert wurden. Alternativ kann nslookup im interaktiven Modus aufgerufen werden:

```
$ nslookup
>
```

Es erscheint ein eigener Eingabeprompt. Wichtig sind die set-Kommandos, mit denen sich ein abzufragender DNS-Ressource-Type festlegen läßt, z.B. Adresse, Pointer, MX-Record, authorativer Nameserver.

```
> set type=MX
> fh-karlsruhe.de
Server:         217.172.162.246
Address:        217.172.162.246#53

Non-authoritative answer:
fh-karlsruhe.de mail exchanger = 9 rz06.fh-karlsruhe.de.

Authoritative answers can be found from:
fh-karlsruhe.de nameserver = dns1.BelWue.de.
fh-karlsruhe.de nameserver = dns3.BelWue.de.
fh-karlsruhe.de nameserver = rz01.fh-karlsruhe.de.
fh-karlsruhe.de nameserver = iraun1.ira.uni-karlsruhe.de.
rz06.fh-karlsruhe.de    internet address = 193.196.64.6
```

Soll ein anderer Nameserver abgefragt werden, kann dies über das server-Kommando erreicht werden:

```
> server rz01.fh-karlsruhe.de
Default server: rz01.fh-karlsruhe.de
Address: 193.196.64.1#53
> set type=PTR
> 193.196.75.65
Server:         rz01.fh-karlsruhe.de
Address:        193.196.64.1#53

65.75.196.193.in-addr.arpa      name = FbI-Netlab-01.FH-Karlsruhe.DE.
```

Mit dem ls-Kommando kann zu einer angegebenen Domain eine Liste aller zugehörigen Einträge eines bestimmten Typs geholt werden, sofern dies die Sicherheitsrichtlinien des DNS-Servers gestatten:

```
> server 193.196.75.65
Server:  FbI-Netlab-01.FH-Karlsruhe.DE
Address:  193.196.75.65

> ls -d netlab.fh-karlsruhe.de
[FbI-Netlab-01.FH-Karlsruhe.DE]
$ORIGIN netlab.fh-karlsruhe.de.
```

```
@                       1H IN SOA       @ hostmaster (
                                        2003011001      ; serial
                                        1H              ; refresh
                                        30M             ; retry
                                        1W              ; expiry
                                        1H )            ; minimum

                        1H IN NS        magnum
burgerking              1H IN A         10.0.0.11
coke                    1H IN A         10.0.0.21
cornetto                1H IN A         10.0.0.3
fanta                   1H IN A         10.0.0.23
kleinerbaer             1H IN A         10.0.0.4
lift                    1H IN A         10.0.0.24
magnum                  1H IN A         10.0.0.1
mcdonalds               1H IN A         10.0.0.14
nogger                  1H IN A         10.0.0.2
pizzahut                1H IN A         10.0.0.12
sprite                  1H IN A         10.0.0.22
wendys                  1H IN A         10.0.0.13
@                       1H IN SOA       @ hostmaster (
                                        2003011001      ; serial
                                        1H              ; refresh
                                        30M             ; retry
                                        1W              ; expiry
                                        1H )            ; minimum
```

Zusammenfassend, die wichtigsten set type Kommandos:

A	Name zu Adresse
CNAME	Canonical Name, ein Alias-Name auf einen bereits bestehenden DNS-Eintrag
MX	Mail-Exchanger mit Priorität für zuständige Mailserver einer Domain
NS	Zuständige authorative Nameserver für eine Domain
PTR	Adresse zu Name, falls Reverse-Eintrag in-addr.arpa vorhanden
SOA	Start of Authority, Informationen über Hostmaster und primären Nameserver

dig

Im einfachsten Falle reicht ein Aufruf in der Form:

```
$ dig login.fh-karlsruhe.de

; <<>> DiG 9.2.1 <<>> login.fh-karlsruhe.de
;; global options:  printcmd

;; QUESTION SECTION:
;login.fh-karlsruhe.de.         IN      A

;; ANSWER SECTION:
login.fh-karlsruhe.de.  86400   IN      CNAME   rz06.RZ-FDDI.fh-karlsruhe.de.
rz06.RZ-FDDI.fh-karlsruhe.de. 86400 IN  A       193.196.125.6
```

```
;; AUTHORITY SECTION:
fh-karlsruhe.de.        86400   IN      NS      rz01.fh-karlsruhe.de.
fh-karlsruhe.de.        86400   IN      NS      iraun1.ira.uni-karlsruhe.de.
fh-karlsruhe.de.        86400   IN      NS      dns1.BelWue.de.
fh-karlsruhe.de.        86400   IN      NS      dns3.BelWue.de.

;; ADDITIONAL SECTION:
dns1.BelWue.de.         66869   IN      A       129.143.2.1
dns3.BelWue.de.         66869   IN      A       131.246.119.18
rz01.fh-karlsruhe.de.   86400   IN      A       193.196.64.1
```

Die Ausgabe wurde hier etwas verkürzt, da dig sehr viele Informationen über einen DNS-Lookup liefert. Dig läßt sich besser als Kommandozeilentool verwenden als nslookup. Die Syntax ist kurz und bündig mit:

```
$ dig @server domain type
$ dig @localhost fh-karlsruhe.de mx
$ dig @rz01.fh-karlsruhe.de FbI-Netlab-01.FH-Karlsruhe.DE a
```

TCPdump

Je nachdem, ob TCPdump [3-3] im netzwerkweiten oder hostspezifischen Rahmen betrieben werden soll, kann der sogenannte promiscuous-mode verwendet oder deaktiviert (-p) werden. Dies ist die erste Option, die neben dem Interface (-i ethX), auf dem gehorcht werden soll, gegebenenfalls beim Aufruf von TCPdump nötig ist. Weiterhin sollten DNS-Lookups (-n) und Service-Lookups (-nn) deaktiviert werden, damit die Analyse zeitlich nicht negativ beeinflußt wird. Die sich ergebende Kommandozeile für eine Analyse sämtlichen Verkehrs auf dem ersten Ethernet Interface lautet dann einfach:

```
[root@host root]# tcpdump -p -i eth0 -nn
```

Die Ausgabe erfolgt nun auf die Standardausgabe. Die Umlenkung in eine Datei im Binärformat des Berkeley Packet Filter erfolgt mit (-w filename). Ein Wiederauslesen ist mit (-r filename) möglich. Außerdem soll nun nicht sämtlicher Verkehr ausgegeben werden, sondern über eine Filterdatei (-F filename) selektiert werden:

```
# tcpdump -p -i eth0 -nn -w /tmp/dump.log -F /etc/dump.fil
```

Als Beispiel hier eine Filterdatei, die auf directed broadcast pings reagiert. Dafür muß TCPdump das letzte Byte einer IPv4-Zieladresse prüfen. TCPdump beginnt immer mit einer „0“ als Index für das erste Byte. Das letzte Byte der Zieladresse im IP-Header ist demnach 19. Ein directed broadcast hat entweder die Form x.y.z.0 oder x.y.z.255, so daß die Filterdatei folgenden Inhalt haben kann:

```
ip[19]=0 or ip[19]=255
```

Dies ergibt einen sehr einfachen Smurf-Amplifier-Mißbrauchsdetektor, der jedoch auch eine Windows NetBIOS Kommunikation als falsche Erkennung liefern würde:

```
19:55:46.901078 192.168.0.11 > 192.168.0.255: icmp: echo request (DF)
20:07:27.700526 192.168.0.102.138 > 192.168.0.255.138: NBT UDP PACKET(138)
20:07:27.700581 192.168.0.102.137 > 192.168.0.255.137: NBT UDP PACKET(137):
QUERY; REQUEST; BROADCAST
```

Dementsprechend wäre die erste Erweiterung, nicht auf NetBIOS Broadcasts (UDP-Pakete von und zu Port 137 und 138) zu matchen. Hierbei können eingebaute Makros (src port, dst port) verwendet werden, die die entsprechenden Portfelder im UDP-Header adressieren:

```
(ip[19]=0 or ip[19]=255)
 and not
((udp src port 137 and udp dst port 137)
  or
 (udp src port 138 and udp dst port 138))
```

Nun soll der Filter noch etwas erweitert werden, um darüberhinaus die OS-Detection eines nmap-scanners zu erkennen. Nmap verwendet einen NULL-Scan auf einen offenen Port und einen XMAS-Scan auf einen vermutlich geschlossenen Port. Der neue Filter muß nun die gesetzten TCP-Flags eines Paketes überprüfen, welche im 14. Byte des TCP-Headers liegen:

```
tcp[13]=41 or tcp[13]=0
```

Die OS-Detection von nmap hinterläßt folgende Spuren, wobei der XMAS-Scan auf Ziel-Port 1 (tcpmux) mit den Flags FIN(1), PUSH(8), URG(32) erfolgte:

```
21:22:28.741801 192.168.0.101.50408 > 192.168.0.11.22: .
win 3072 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
21:22:28.743181 192.168.0.101.50413 > 192.168.0.11.1: FP 1568435430:1568435430(0)
win 3072 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
21:22:31.217786 192.168.0.101.50408 > 192.168.0.11.22: .
win 3072 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol>
```

Damit können generisch mißgestaltete TCP-Pakete erkannt werden, die keine Flags oder die illegale Kombination von FIN, PUSH, URG gesetzt haben. Die zwei Filter für Broadcasts und Scans analysieren Protokollanomalien. Sie können in einer Datei über eine logische oder-Verknüpfung kombiniert werden:

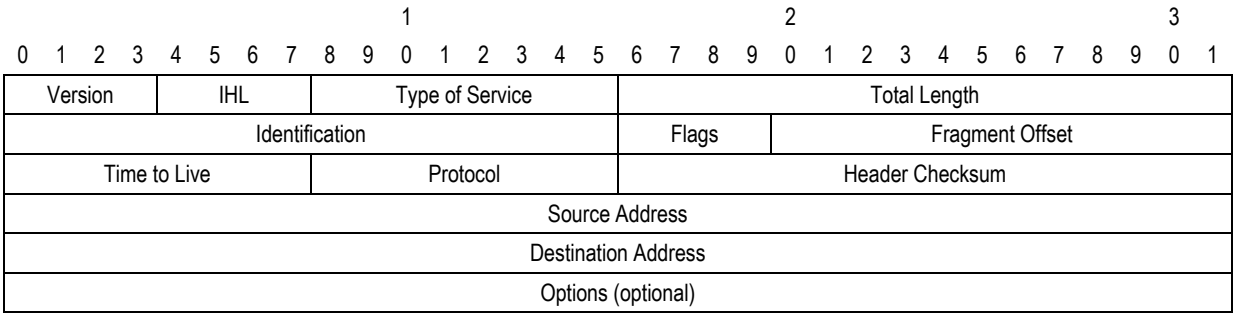
```
(
  (ip[19]=0 or ip[19]=255)
   and not
  ((udp src port 137 and udp dst port 137)
    or
   (udp src port 138 and udp dst port 138))
)
or
(
  tcp[13]=41 or tcp[13]=0
)
```

TCPdump Kurzreferenz

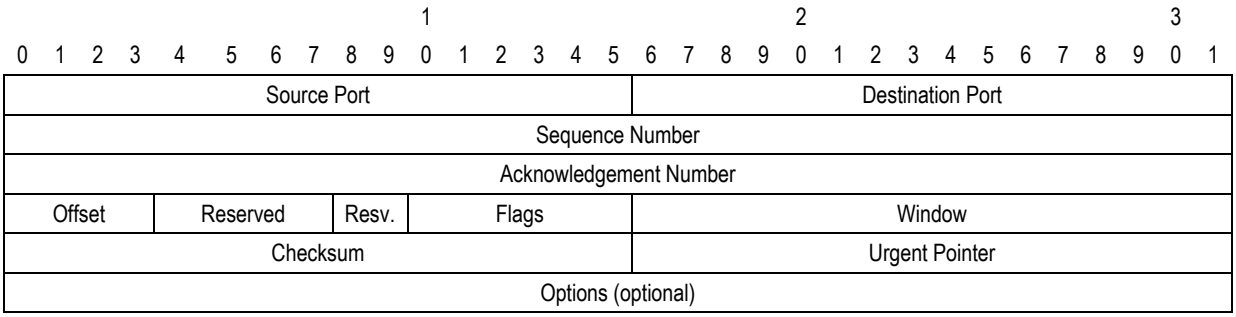
```
tcpdump [-npSvx] [-i itf] [-s snaplen] [-F file] [-r file] [-w file] [expression]

-i     Interface auf dem gehorcht werden soll
-F     Filterdatei statt Ausdruck (expression) von der Kommandozeile verwenden
-n     keine DNS Auflösung, (-nn) um weder DNS noch Services aufzulösen
-p     keinen promiscuous-mode verwenden, nur gerichtete Pakete und Broadcasts
-r     von einer BPF-Binärdatei post-mortem lesen
-s     um (snaplen) bytes der Nutzlast mitzuprotokollieren
-S     absolute statt relative TCP-Sequenznummern verwenden
-v     detaillierte Informationen geben, bis zu dreimal wiederholbar (-vvv)
-w     in eine Datei im Berkeley Packet Filter Binärformat schreiben
-x     den durch (snaplen) angebenen Nutzlastinhalt in Hexadezimal ausgeben
```

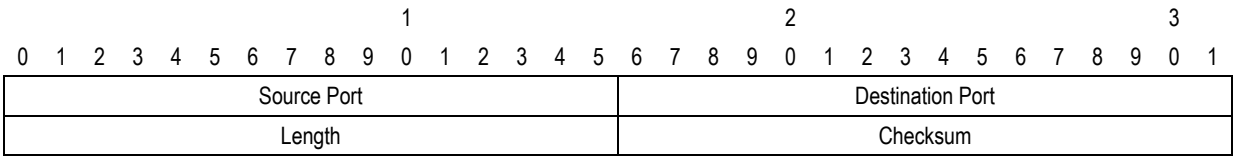
IP Header



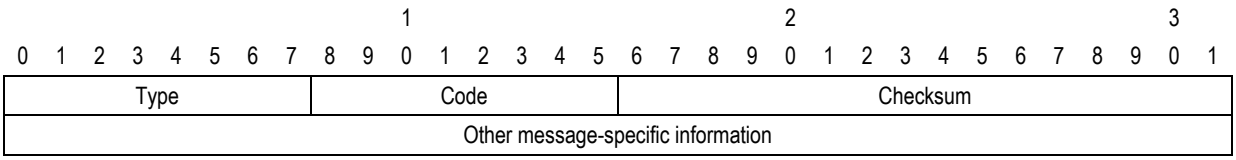
TCP Header



UDP Header



ICMP Header



Netfilter/IPtables

Netfilter ist ein Rahmenwerk zur Verarbeitung und Analyse von Paketen auf Kernel-Ebene des Linux-Betriebssystems. Es ermöglicht Stateful-Packet-Firewalling, Network Address Translation und Packet-Mangling. IPtables ist ein Kommandozeilen-Frontend zur Konfiguration von Netfilter.

Übersicht

Tabellen

Mit IPtables lassen sich alle Netfilter-Tabellen verwalten, also für Packet-Filtering, NAT und Packet-Mangling. Standardmäßig wird immer die Tabelle für Packet-Filtering ansprochen. Soll eine der anderen Tabelle verwaltet werden, muss diese explizit beim Einsatz von IPtables mit angegeben werden. Die folgenden Erläuterungen beziehen sich zunächst nur auf Packet-Filtering [3-4], später wird noch kurz NAT [3-5] behandelt.

Kollabierte Filterung

Der Hauptunterschied zur Filterung mit einem Cisco IOS Router liegt bei Linux Netfilter darin, daß nicht per Interface nach ein- und ausgehendem Verkehr gefiltert werden kann. Netfilter verwendet stattdessen ein einzelnes zentrales (virtuelles) Interface, in dem alle anderen physikalischen Interfaces scheinbar kollabiert sind. D.h. das Regelketten wie die ACLs einer Cisco nicht auf ein spezielles Interface sondern immer auf eine bestimmten Richtung des zentralen Interfaces gebunden werden.

Filterketten

Netfilter hat einige vordefinierte, feste, Regelketten (Chains). Diese sind u.a. INPUT, OUTPUT und FORWARD in der Tabelle für Packet-Filtering. Um einen eingehenden Filter zu definieren, kann dieser einfach der INPUT-Chain angehängt werden. Ganze Filterketten können separat definiert und dann in eine vordefinierte Regelkette eingeklinkt werden. Die vordefinierten Chains sind zunächst auf eine Default-Policy Allow gesetzt.

Aktionen

Die Aktionen permit und deny heißen bei Netfilter stattdessen ACCEPT und DROP. Ein DROP hat keine ICMP-Nachricht zur Folge wie es bei einem Cisco-Interface automatisch der Fall ist. Stattdessen muß diese Eigenschaft für jede Filterregel separat definiert werden und wird dann als REJECT Aktion gehandhabt.

Beispiele

Packet-Filter

Als erstes ist es wichtig, sich den Status der derzeitigen Filtertabelle anzuschauen:

```
# iptables –L –n –t filter
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Standardmäßig sind keine Filter definiert. Es soll nun sämtlicher eingehender Verkehr, der nicht als Antwort auf Verbindungen, die vom Host selbst initiiert wurden, geblockt werden:

```
# iptables --new INBOUND
# iptables --append INBOUND --match state --state ESTABLISHED,RELATED --jump ACCEPT
# iptables --append INBOUND --jump DROP
```

Es wurde gleich eine neue Kette mit --new (-N) angelegt, die jetzt noch referenziert werden muß. Doch zunächst soll die zweite Zeile näher erläutert werden. Die Option --append (-A) hängt die Regel ans Ende einer Kette. Die Option --match lädt ein zusätzliches Vergleichsmodul, hier *state*, um Stateful-Packet-Inspection via Connection-Tracking durchführen zu können. Die Option --state gibt nun an, welche Stati innerhalb des Connection-Tracking berücksichtigt werden sollen. Hierbei fällt das Schlüsselwort *ESTABLISHED* auf, welches auch schon bei den Cisco ACLs auftauchte, hier jedoch unabhängig von TCP verwendet werden kann. Das nächste Schlüsselwort, *RELATED*, ermöglicht die Verknüpfung von neuen Verbindungen mit bereits etablierten Verbindungen, z.B. dem Datenkanal bei FTP mit einer bereits aktiven Kontrollkanalverbindung.

Nun muß die Kette noch angesprungen werden. Hierbei ist zu beachten, daß eingehender Verkehr zum einen über die INPUT-Chain, aber auch über die FORWARD-Chain gelangen kann, falls die Pakete für ein anderes Ziel als den lokalen Host bestimmt sind:

```
# iptables --append INPUT --jump INBOUND
# iptables --append FORWARD --jump INBOUND
```

Desweiteren sollen nun einige ausgehende Regeln gesetzt werden. Der Host soll z.B. keine Peer-to-Peer Dienste oder bestimmte ICMP-Nachrichten weiterleiten:

```
# iptables -N OUTBOUND
# iptables -A OUTBOUND –p tcp --dport 1214 -j DROP
# iptables –A OUTBOUND –p icmp –icmp-type ! fragmentation-needed -j DROP
```

Wiederum muß die neue Kette in die vordefinierten Ketten eingehängt werden, diesmal in OUTPUT und FORWARD.

Network Address Translation

Um sich den Status der derzeitigen NAT-Tabelle anzuschauen verwendet man:

```
# iptables –L –n –t nat
Chain PREROUTING (policy ACCEPT)
target prot   opt    source       destination
DNAT   tcp    --     0.0.0.0/0    193.196.75.65       tcp dpts:2231:2234 to:10.0.0.3
DNAT   tcp    --     0.0.0.0/0    193.196.75.65       tcp dpts:2241:2244 to:10.0.0.4

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

In der Kette *PREROUTING* befinden sich bereits zwei Einträge, die für das Port-Forwarding von SSH auf die PCs mit den VMware-Instanzen verantwortlich sind. Die Kette PREROUTING und das damit verbundene Destination-NAT wird durchgeführt, bevor die Paketfilter-Regeln für die entsprechende Verkehrsrichtung durchlaufen werden. D.h., daß etwaige Filterregeln mit den bereits umgesetzten Adressen und Ports arbeiten müssen.
Entgegengesetzt würden Einträge im *POSTROUTING* erst nach etwaigen Paketfilter-Regeln angewandt. In dieser Kette befinden sich Source-NAT, also auch Masquerading-Einträge. Eine Definition von Paketfilter-Regeln müßte hierbei auf noch nicht umgesetzte Adressen und Ports abzielen.

Das bedeutet, daß bei der Definition von Paketfilter-Regeln stets mit internen/privaten Adressen gearbeitet wird, und daß NAT diese internen Adressen erst nach dem Paketfilter zu externen Adressen umsetzt oder bereits externe Adressen zu internen umgesetzt hat. Somit sind Paketfilter und NAT weitestgehend in ihrer Definition voneinander entkoppelt.

Es soll nun noch ein einfacher Masquerading-Eintrag vorgenommen werden, der ausgehenden Verkehr auf dem externen Interface des Hosts maskiert:

```
# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Save und Restore

IPtables verfügt über zwei praktische Shell-Skripte, mit denen der aktuelle Zustand aller Tabellen gespeichert und zu einem späteren Zeitpunkt wieder eingespielt werden kann. Diese sind *iptables-save* und *iptables-restore*. Desweiteren sei die Möglichkeit zur Steuerung von IPtables über das *service* Shell-Skript erwähnt:

```
# service iptables (status|start|stop|restart|save|panic)
```

A

Literaturverzeichnis

[1-1] SSH Communications – Secure Shell for Workstations
http://www.ssh.com/support/downloads/secureshellwks/non-commercial.html

[2-1] Cisco IOS Release 12.0 Configuration Guides and Command References
http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/

[3-1] BASH – GNU Project – Free Software Foundation
http://www.gnu.org/software/bash/bash.html

[3-2] welcome home: vim online
http://www.vim.org/

[3-3] TCPdump Public Repository
http://www.tcpdump.org/

[3-4] Netfilter/Iptables – Documentation
http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html

[3-5] Netfilter/Iptables – Documentation
http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html