Sicherheit unter Linux
Workshop

Hergen Harnisch
harnisch@rrzn.uni-hannover.de

Mark Heisterkamp
heisterkamp@rrzn.uni-hannover.de

19. Juni 2006

IPTables

IPTables ist eine Filterfunktion des Kernels unter Linux.

Mittels Regeln wird der Verlauf eines TCP/IP-Datenpaketes beeinflusst. Dabei kann eine Datenpaket entweder

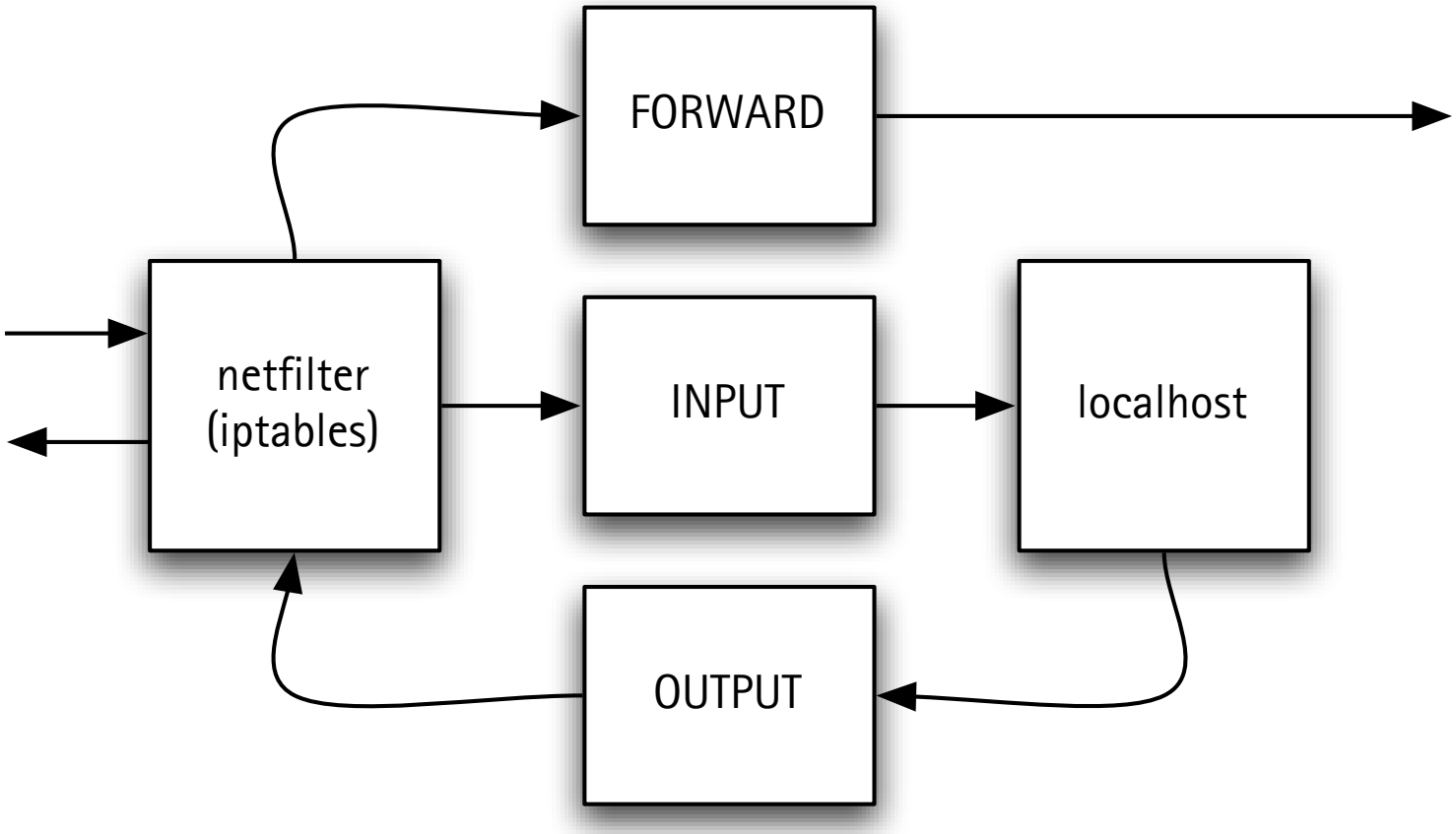
- verworfen (DROP),
- zurückgewiesen (REJECT) oder
- angenommen (ACCEPT)

werden.

Ein Paket durchläuft normalerweise eine von drei möglichen sogenannten *Ketten*:

- FORWARD
- INPUT
- OUTPUT

IPTables



IPTables-Regeln

Eine IPTables-Regel hat den allgemeinen Aufbau:

```
iptables [-t Tabelle] -A <Kette> <Regel>
```

wobei drei Tabellen zur Verfügung stehen:

- filter (Standard)
- mangle
- nat

Über entsprechende Abkürzungen werden dann die Regeln aufgebaut.
Wird eine Regel eingegeben, so wird sie *sofort* wirksam.

Regelbeispiel

Das folgende Beispiel für einen Satz Filterregeln sollte als Quasi-Standard am RRZN gelten, sobald ein Server aufgesetzt wird noch bevor er endgültig ans Netz geht:

```
iptables -L
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -m state --state ESTABLISHED\
         -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -s 130.75.5.0/255.255.255.0\
         -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -s 127.0.01 -i lo -j ACCEPT
```

Speichern und Laden von Filterregeln

Mittels

```
iptables-save > <DATEI>
```

und

```
iptables-restore < <DATEI>
```

können aktuelle Regeln in `DATEI` gespeichert werden bzw. daraus gelesen werden.

Aufgabe

- Sperren Sie Ihren Rechner für alle Zugriffe von außen.
- Verbieten Sie das Surfen im Internet auf unverschlüsselten Seiten.
- Erlauben Sie ssh-Zugriff ausschließlich vom Rechner dozpc (`192.168.0.13`).
- Lassen Sie den Zugriff auf ihren Rechner lokal zu.

SSH

SSH steht für „Secure-Shell“ und ermöglicht die verschlüsselte Verbindung zwischen zwei Rechnern im Netz.

Syntax:

```
ssh [OPTIONS] USER@HOST
```

Optionen:

`-l <USER>`
`-p <PORT>`
`-L <LOCALPORT>:<HOST>:<ZIELPORT>` („Tunneling“)
`-X`

Konfigurationsdateien

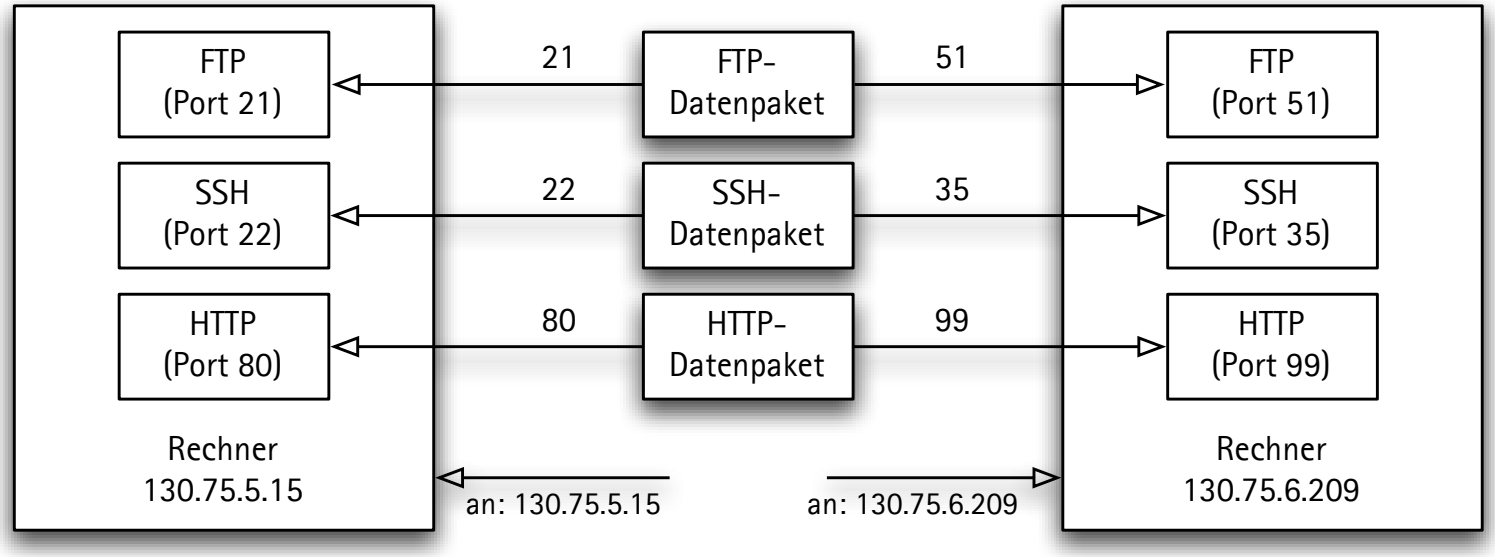
Konfigurationsdateien für SSH:

`/etc/ssh_config` und `/etc/sshd_config` oder
`/etc/ssh/ssh_config` und `/etc/ssh/sshd_config`,

Wichtige Optionen in `sshd_config`

- `Port 22`
- `Protocol 2`
- `PermitRootLogin no`
- `StrictModes yes`

Anwendungen und ihre Ports



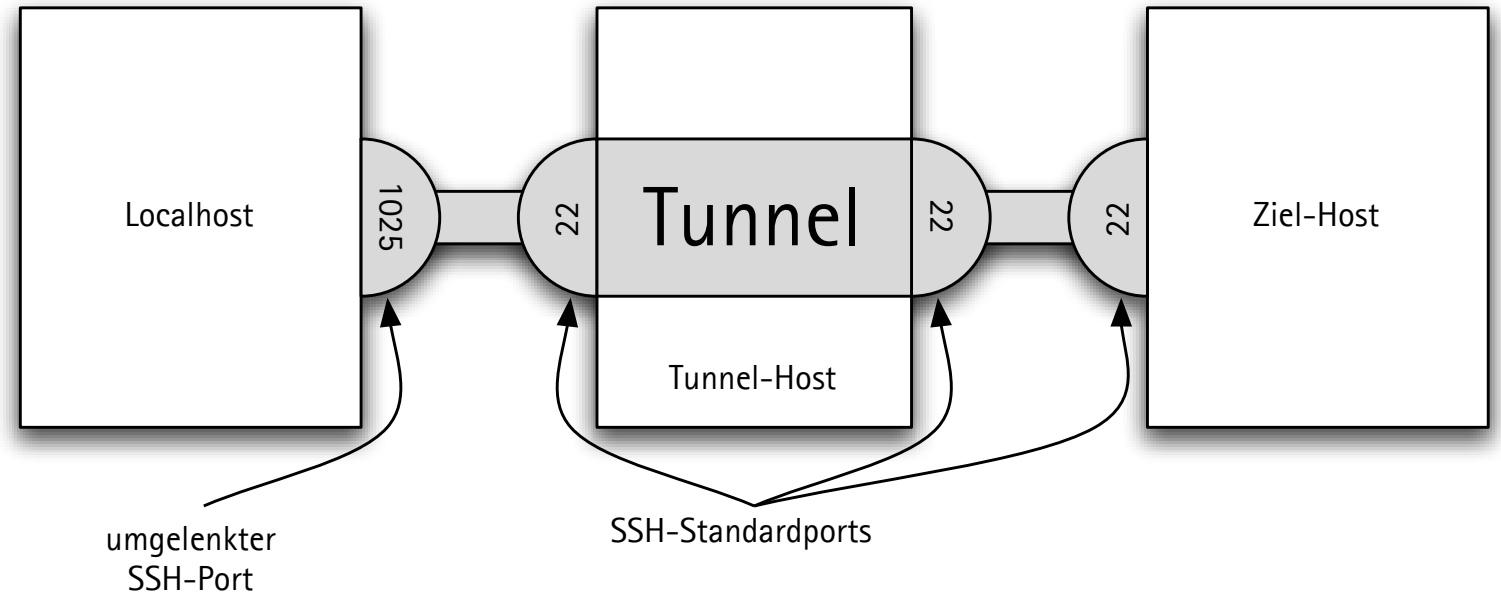
Ein Datenpaket enthält

- Absenderadresse
- Empfängeradresse
- Port der Anwendung am Zielrechner

Im Falle von SSH geschieht eine vom Standard abweichende Port-Einstellung in der Datei `sshd_config`.

Tunnel

Ist ein Rechner beispielsweise durch eine Firewall abgeschirmt, es gibt aber einen dezidierten Eingangsrechner, von dem aus auf den eigentlichen abgeschirmten Rechner zugegriffen werden kann, so kann über diesen Zugangsrechner ein verschlüsselter „Tunnel" aufgebaut werden:



Tunnelaufbau

Zunächst wird in einem eigenen Terminalfenster (oder einer separaten Konsole) die Verbindung zu einem Host aufgebaut, der einerseits von mir erreichbar ist, der aber andererseits auch den eigentlichen Zielrechner erreichen kann.

Ich erzeuge einen lokalen Port, der tatsächlich auf den ssh-Port des eigentlichen Zielrechners zeigt:

```
ssh -L <LOCALPORT>:<ZIEL-IP>:<ZIELPORT> \
    <ID>@<TUNNEL-HOST>
```

Die obige Verbindung lässt man bestehen und baut nun in einem neuen Fenster (einer neuen Konsole) folgende Verbindung auf:

```
ssh -p <LOCALPORT> ZIEL-ID@localhost
```

Nun ist eine verschlüsselter Tunnel zum eigentlichen Zielrechner aufgebaut.

Public Key Authentifizierung

Es gibt die Möglichkeit, eine Verbindung so zu definieren, dass keine Authetifizierung mehr per Nutzername und Passwort notwendig ist.

1. Lokal: `ssh-keygen -t dsa`
2. Kopieren des Schlüssels auf den Server:
   `scp $HOME/.ssh/id_dsa.pub user@server:`
3. Login auf den Server:
   `ssh user@server`
4. Auf dem Server:
   `mkdir .ssh` und `chmod 700 .ssh`,
   falls .ssh noch nicht existiert
   `cat id_dsa.pub >>.ssh/authorized_keys`
   `rm id_dsa.pub`

Public Key Authentifizierung

Binden eines Public-Keys an eine bestimmte IP-Adresse:

```
from="foo.bar.de" ssh-dss AAAB3NzaC1 [...] kc3
MAAACBALT5QXyWa7WLokEqQi8+t0= mheiste@pc225h
```

Ohne die Option

```
PasswordAuthentication no
```

in `sshd_config` nicht sinnvoll.

Aufgabe

- Verbieten Sie den Root-Login per SSH.
- Erzeugen Sie sich ein einen öffentlichen Schlüssel.
- Bauen Sie einen Tunnel zu einem Nachbarrechner auf.
- Binden Sie Ihren öffentlichen Schlüssel auf dem Zielrechner an Ihren lokalen Rechner.
- Nutzen Sie die Public-Key-Authentifizierung, um bei zukünftigen Tunnel-Verbindungen weder auf dem Tunnel-Host, noch auf dem Ziel-Host ein Passwort eingeben zu müssen.

Datensicherung mit tar

Packen:

```
tar cvzf <archiv.tgz> <Quellverzeichnis>
```

Inhalt auflisten:

```
tar tvzf <archiv.tgz>
```

Entpacken:

```
tar xvzf <archiv.tgz> [-C <Zielverzeichnis>]
```

Umlenkung mit tar

Nach STDOUT packen:

```
tar cvzf - <Quellverzeichnis>
```

Nach STDOUT entpacken:

```
tar xvzf <archiv.tgz> -O
```

Von STDIN entpacken:

```
tar xvzf - [-C <Zielverzeichnis>]
```

tar, ssh, Umlenkung und Pipes

tar und ssh sind für Kommandoverknüpfungen geeignet:

```
ssh <ID>@<HOST> \
"tar cf - <Quellverzeichnis>" > archiv.tar
```

oder

```
tar cf - <Quellverzeichnis> | ssh <ID>@<HOST> \
"cat > archiv.tar"
```

oder

```
cat archiv.tar | ssh <ID>@<HOST> \
"tar xf - -C <Zielverzeichnis>"
```

Aufgabe

- Packen Sie per Pipe durch `ssh` das `/tmp`-Verzeichnis des Tunnel-Host (`192.168.0.13`) in Ihr Home-Verzeichnis.
- Entpacken Sie das soeben erzeugte Archiv durch eine ssh-Pipe in ein `/tmp`-Verzeichnis (das Sie ggf. noch erzeugen müssen) eines anderen Accounts auf Ihrem lokalen Rechner.