1 Was ist eine Firewall?

Eine Firewall schützt ein internes Netz (Intranet) vor Angriffen aus einem externen Netz (Internet).
Sie hat die Eigenschaften einer elektronischen Brandschutzmauer und eines elektronischen Pförtners.

Aufgaben eines Firewall-Systems

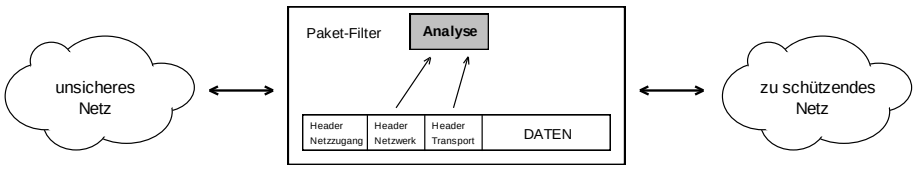
- abschotten bestimmter Bereiche
- bereitstellen eines sicheren, besonders bewachten Übergangs zwischen beiden Netzen
- erlauben eines definierten Zugangs durch Identifikation und Authentifikation
- kontrollieren der ein- und ausgegenden Daten
- protokollieren aller ein- und ausgehenden Daten und Ereignisse

Sicherheitskonzept
Damit eine Firewall ein internes Netz schützen kann, muss man vorher schon genau wissen, wer oder was wovor geschützt wird und was erlaubt bzw. verboten werden soll. Es muss deshalb eine entsprechende Sicherheitspolitik entwickelt und in ein Sicherheitskonzept umgesetzt werden.

Elemente eines Firewall-Systems

1.) Paket-Filter
Analysiert und kontrolliert die ein- und ausgehenden Datenpakete auf der Netzwerk- und Transport-Ebene (OSI-Schichten 3 und 4). Die entsprechenden Prüfinformationen werden einem Regelwerk (*Firewall-Regeln*) entnommen und mit den Analyse-Ergebnissen verglichen.

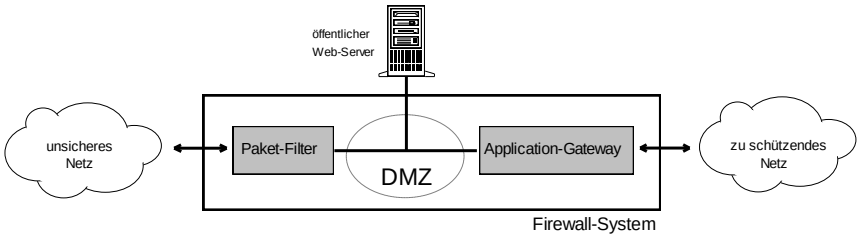
Problem: Die eigentlichen Nutzdaten, die oberhalb der Transportebene sind (Anwendungs-Schicht), werden nicht analysiert!



Paket-Filter

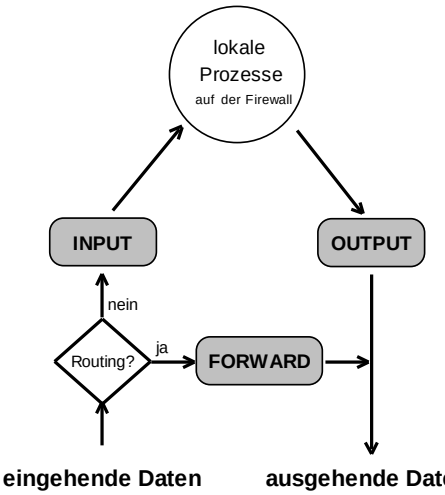
2.) Application-Gateway (Proxies)
Das Application-Gateway filtert auf den Netzwerk-Schichten 3 - 7. Es analysiert nicht nur die Inhalte der Protokoll-Header (z.B. Adressen, Protokolle und Ports) nach bestimmten Regeln, sondern überprüft auch die Nutzdaten der Pakete. Einzelne Anwendungsprotokolle (z.B. E-Mail, FTP, HTTP usw.) werden von sog. *Proxies* überwacht, welche die übertragenen Inhalte überprüfen.
Proxies ermöglichen es beispielsweise, die Inhalte von E-Mails zu analysieren, diese z.B auf SPAM zu testen und deren Anhänge nach Malware zu durchsuchen. Für Application-Gateways sind leistungsfähige Rechner notwendig!

Konzepte von Firewall-Systemen
Einfache Netze mit geringem Schutzbedarf benutzen oft nur einen Paket-Filter, der sich im DSL-Router befindet.
Bei mittlerem Schutzbedarf wird eine Kombination aus Paket-Filter und Application-Gateway verwendet. Das Netz-Segment zwischen Paket-Filter und Application-Gateway wird *demilitarisierte Zone* (DMZ), *Perimeter-Netz* oder *screened subnet* genannt. In der DMZ wird meist der öffentliche Web-Server angeschlossen.



Firewall-System für mittleren Schutzbedarf

2 Firewall-Regeln bei Linux



Das Grundprinzip der Linux-Firewall besteht darin, jedes Daten-Paket zu überprüfen, bevor es passieren darf.

Die Überprüfung erfolgt anhand von Regelsätzen (Ketten, *chains*), die man für Pakete formuliert, welche die Firewall erreichen (INPUT) und für die Pakete, die sie ausgibt (OUTPUT).

Zusätzlich existiert eine weitere Regel-Kette mit FORWARD-Regeln. Diese Regeln betreffen Pakete, die nicht an die Firewall selbst gerichtet sind, sondern entweder von Rechnern aus dem LAN an das Internet weitergereicht werden oder, wenn sie aus dem Internet kommen, für Rechner im LAN bestimmt sind.
Die Firewall arbeitet in diesem Fall als Router.

Kriterien für die Überprüfung der Paketinhalte durch Regeln sind beispielsweise die IP-Adressen von Absender und Empfänger, das verwendete Transportprotokoll sowie die angesprochene Portnummer.

Alle Regeln einer Kette werden nacheinander abgearbeitet, bis eine zutreffende Regel gefunden wird. Die zutreffende Regel enthält eine auszuführende Aktion z. B: ACCEPT, DROP, REJECT, LOG.

ACCEPT	lässt das Paket passieren
DROP	lehnt das Paket ohne Rückmeldung ab
REJECT	lehnt das Paket ab und schickt dem Absender per ICMP die Rückmeldung *destination unreachable*
LOG	Paket im Logfile des Systems protokollieren

Wird keine zutreffende Regel gefunden, wird die Standard-Aktion (*default policy*) eines Regelsatzes ausgeführt.

Strategie zum Erstellen von Firewall-Regeln
Zunächst muss jeglicher Verkehr verboten werden. Dann werden nur die IP-Adressen, Protokolle und Dienste zugelassen, die auch wirklich benötigt werden. Dadurch verhindert man unbekannte Angriffe (für die sich im Momemt noch keine passende Verbotsregel definieren lässt) und erhält somit ein fehlertoleranteres System.

> Das schrittweise Abarbeiten der Regeln in einer Kette bedingt,
> dass spezielle Regeln stets **vor** allgemeineren stehen müssen,
> denn wenn die allgemeinere Regel zutrifft, wird die spezielle nicht mehr erreicht!

Informationen und Links zu Firewalls mit Linux und iptables

- Die man-Pages zu iptables und die Dokumentation in `/usr/share/doc/packages/iptables`
- Firewall mit Linux 2.4 in c't 26/01, S.230 und c't 04/02, S.202
- Linux 2.4 Packet Filtering HOWTO (www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html)
- Firewall-Workshop bei Pro-Linux (http://www.pro-linux.de/work/)

Komplette freie bzw. zum Teil im privaten Bereich frei verwendbare Firewall-Lösungen

- IPFire - An Open Source Firewall Distribution (www.ipfire.org)
- Shorewall - iptables made easy (www.shorewall.net)
- Smoothwall Express (www.smoothwall.org)
- fli4l, kompletter Router mit Firewall und Zusatzdiensten, der früher mal auf eine Floppy passte (www.fli4l.de)

3 Firewall Regeln mit `iptables`

Für das Setzen der Firewall-Regeln wird seit den 2.4.x-Kerneln das Programm `iptables` verwendet
Hilfe gibt es mit **`iptables --help`** bzw. **`man iptables`**.
Im Verzeichnis `/usr/share/doc/packages/iptables` finden Sie u.a. das Netfilter-Howto.

Mit `iptables` Regeln erstellen
Das Programm `iptables` ermöglicht das Konfigurieren der Netfilter im Linux-Kernel. Die Regeln von `iptables` werden standardmäßig in die 3 Ketten INPUT (für eingehende Pakete), OUTPUT (für ausgehende Pakete) und FORWARD (für weitergeleitete Pakete) abgelegt. Man kann auch eigene Regel-Ketten erstellen.

Um Regeln in Ketten einzugliedern oder zu löschen gibt es eine Reihe von Programm-Optionen
In den meisten Fällen wird man von den Optionen `-A` für Anfügen bzw. `-D` für das Löschen einer Regel Gebrauch machen.

Nach diesen Optionen werden die Kriterien angegeben, die ein Paket erfüllen muss. Hier gibt es sehr viele Möglichkeiten, die wichtigsten werden nachfolgend genannt:

- Mit `-d` kann man die Zieladresse des Paketes angeben, mit `-s` die Absender-Adresse.
- Durch `-p` kann das Protokoll des Paketes spezifiziert werden. Zur Auswahl stehen `tcp`, `udp`, `icmp` oder `all`.
- Mit `--dport` bzw. `--sport` wird der Ziel- bzw. Quellport des Paketes angegeben, hierzu ist die Option `-p` zwingend erforderlich.
- Mit `-i` bzw. `-o` lässt sich das Input- bzw. Output-Interface spezifizieren.
- Abschließend wird mit `-j` bestimmt, was mit einem Paket geschieht, wenn die angegebenen Kriterien zutreffen. Als Parameter sind `ACCEPT`, `DROP` und `REJECT` erlaubt.
  `ACCEPT` lässt ein Paket ganz normal passieren, mit `DROP` wird das Paket verworfen.
- Mit `REJECT` wird ein Paket nicht einfach verworfen, sondern der Sender bekommt eine Meldung zurück. Standardmäßig ist das die ICMP-Fehlermeldung *destination unreacheable*.

Eine komplette Regel könnte zum Beispiel so aussehen:

```
iptables -A INPUT -s 193.99.144.85 -p tcp --sport 80 -j ACCEPT
```

Damit werden alle Pakete angenommen, die vom Webserver (Port 80) mit der Adresse 193.99.144.85 (www.heise.de) kommen.

Default Policy
Jede Chain von `iptables` besitzt eine sog. Policy. In der Policy steht, was mit Paketen geschieht, auf die keine Regel zutrifft. Standardmäßig ist die Policy jeder Kette auf `ACCEPT` gestellt. **Sie sollte immer auf `DROP` geändert werden**, da bei einer vergessenen Regel ein Sicherheitsloch in der Firewall entsteht, weil das Datenpaket sonst durch die Policy-Einstellung automatisch akzeptiert wird.

Modularer Aufbau von `iptables`
Das Programm `iptables` ist so aufgebaut, dass es durch nachladbare Kernel-Module um Funktionen erweitert werden kann. Es gibt Module wie z.B. `reject` (eine ICMP-Meldung zurücksenden), `limit` (volumenbedingtes Filtern), `owner` (filtern nach bestimmten Benutzern), `mac` (filtern anhand der MAC-Adresse der Netzwerkkarte). Das Modul `state` (Stateful Paket Inspection) ist dabei sicherlich das interessanteste.

Stateful Packet Inspection (SPI)
Stateful Packet Inspection bedeutet, dass man ein Paket anhand seiner Beziehung zu einer bestehenden Verbindung filtern kann (dynamischer Paketfilter).
Ein Paket hat bezüglich einer bestehenden Verbindung immer einen bestimmten Status (*state*):
`NEW` (das Paket baut eine neue Verbindung auf), `ESTABLISHED` (das Paket gehört zu einer bereits bestehenden Verbindung), `RELATED` (das Paket nutzt eine verwandte Verbindung) und `INVALID` (unbekannter Status).
Um die Stateful Paket Inspection nutzen zu können, muss in den Regeln mit `--state` eine oder mehrere Status-Arten angegeben werden (siehe Beispiel 5.1 auf der übernächsten Seite).

4 Die wichtigsten Optionen zu `iptables`

Syntax: **`iptables`** ***`Befehl Kette Optionen`*** **`-j`** ***`Ziel`***

Befehl	**-A** hängt eine neue Regel an eine Regel-Kette an (*append*) **-P** setzt die Standard-Aktion für eine Regel-Kette (*policy*) **-F** löscht alle Regeln einer Regel-Kette (*flush*) **-D** löscht eine Regel einer Regel-Kette (*delete*) **-N** benutzerdefinierte Regel-Kette erstellen (*new*) **-L** alle Regeln einer Regel-Kette anzeigen (*list*)
Kette	Name der Regel-Kette z.B. **INPUT**, **FORWARD**, **OUTPUT** bzw. der Name einer benutzerdefinierten Kette
Optionen *Dies ist nur eine kleine Auswahl!*	**-s** *IP* — Herkunftsadresse (*source*) des Pakets **-d** *IP* — Zieladresse (*destination*) des Pakets (Anstatt IP-Adressen sind auch IP-Netze, Hostnamen, FQHN , DNS-Domänen-Namen und FQDN möglich) **-p** *protocol* — mögliche Protokoll-Typen: `tcp`, `udp`, `icmp`, `all` **--icmp-type** *typ* — ICMP-Typ (die möglichen Typen mit **`iptables -p icmp --help`** abfragen!) **--sport** *port[:port]* — Herkunfts-Port(*source port*) (-Bereich) bei TCP/UDP **--dport** *port[:port]* — Ziel-Port(*destination port*) (-Bereich) bei TCP/UDP **--syn** — TCP-Pakete mit gesetztem SYN-Bit, diese fordern einen Verbindungsaufbau an **!** — Ein Ausrufezeichen kann vor jeder der Angaben `name`, `ip`, `port`, `protocol` und auch vor `--syn` stehen und bedeutet "alle außer..." **-i** *name* — Name des Netzwerk-Interface, durch das die Pakete in den PC hineingehen, z.B: **`-i eth0`** erste Ethernet-Karte, **`-i ppp0`** erstes Analog-Modem, **`-i ippp0`** erste ISDN-Karte **-o** *name* — Name des Netzwerk-Interface, durch das die Pakete den PC verlassen
Ziel	**ACCEPT, DROP, REJECT, LOG, MASQUERADE** bzw. der Name einer benutzerdefinierten Regelkette

Beispiele für Regeln

`iptables -F FORWARD`	Alle Regeln der FORWARD-Chain löschen
`iptables -F`	Alle Regeln löschen
`iptables -L -v --line-numbers`	ausführliche Ausgabe der bestehenden Ketten mit Regelnummern
`iptables -D OUTPUT 5`	Regel 5 in OUTPUT löschen

`iptables -P INPUT ACCEPT`
`iptables -P OUTPUT ACCEPT`
`iptables -P FORWARD DROP`
	Standard-Aktionen (*default policies*) setzen: Pakete der INPUT und OUTPUT-Chain akzeptieren, Pakete aber nicht weiterleiten (routen)

`iptables -A INPUT -i eth0 -p TCP --syn -j DROP`	erlaubt keinen ankommenden Verbindungs-Aufbau (SYN) von außen (eth0 ist externes Interface)
`iptables -A INPUT -i eth0 -s 0/0 -p udp --dport 0:1023 -j DROP` `iptables -A INPUT -i eth0 -s 0/0 -p tcp --dport 0:1023 -j DROP`	Von allen Adressen eingehende TCP- und UDP-Pakete an privilegierte Ports (Portnummer1024) abweisen
`iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE`	Dieser Befehl reicht aus, dass ein Linuxrechner, der mit dem Modem (ppp0) ins Internet eingewählt ist (dyn. IP-Adresse), alle Rechner, die an seiner Netzwerkkarte angeschlossen sind, maskiert und ihnen so den Internet-Zugang ermöglicht.
`iptables -A INPUT -i lo -j ACCEPT` `iptables -A OUTPUT -o lo -j ACCEPT`	Kommunikation über das Loopback-Netz (127.0.0.0) erlauben --> wichtig für die lokale Nutzung von Netzwerkdiensten

5 Beispiele zu `iptables`

Diese Beipiele können Sie sich mit **`wget --no-proxy http://intranet/files/5.1.sh`** herunterladen
(aber nur, wenn die Firewall offen ist...)

5.1 SPI - Einfache lokale Firewall mit Stateful Paket Inspection (`bsp1.sh`)

```
# Alles abweisen: Die Standard-Policy ist DROP
iptables -P INPUT   DROP
iptables -P OUTPUT  DROP
iptables -P FORWARD DROP

# Löschen aller Regeln
iptables -F

# Nach außen hin darf alles gemacht werden
iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Von außen werden nur Antwortpakete und Pakete von bestehenden oder verwandten Verbindungen hereingelassen
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

5.2 Stateful Paket Inspection über eine benutzerdefinierte Regelkette (`bsp2.sh`)

```
#!/bin/bash

# Löschen aller möglicherweise noch vorhandenen Regelketten
iptables -F
iptables -t nat -F
iptables -X

# Loopback für lokale Prozesse offenhalten
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A INPUT  -i lo -j ACCEPT

# Erzeugen der benutzerdefinierten Regelkette wall
iptables -N wall

# Lässt Pakete bereits bestehender Verbindungen
# in beiden Richtungen passieren (ESTABLISHED)
# und solche, die in Beziehung dazu stehen (RELATED)
iptables -A wall -m state --state ESTABLISHED,RELATED -j ACCEPT

# Erlaube keinen Verbindungsaufbau von außen nach innen. Das sind Pakete,
# die über das nach außen hin angeschlossene Interface eth0 (=erste Netzwerkkarte) hereinkommen
iptables -A wall -m state --state NEW ! -i eth0 -j ACCEPT

# Verwerfe alle Pakete, auf die keine der Regeln zutrifft
iptables -A wall -j DROP

# Sprung zu der benutzerdefinierten Regelkette wall über die "eingebauten" Regelketten INPUT und FORWARD
iptables -A INPUT   -j wall
iptables -A FORWARD -j wall
```

5.3 Wieder alles erlauben (`allow-all.sh`)

```
# Set the default policies of all standard chains to accept
iptables -P INPUT   ACCEPT
iptables -P OUTPUT  ACCEPT
iptables -P FORWARD ACCEPT

iptables -t nat -P PREROUTING  ACCEPT
iptables -t nat -P OUTPUT      ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

iptables -t mangle -P PREROUTING ACCEPT
iptables -t mangle -P OUTPUT     ACCEPT

# Remove any existing rules from the built-in chains
iptables -F
iptables -t nat -F
iptables -t mangle -F

# Remove any pre-existing user-defined chains
iptables -X
iptables -t nat -X
iptables -t mangle -X
```

6 Übungen

1.) Einfache Regeln mit `iptables` erstellen und testen

Hinweise

- Schreiben Sie die Regeln jeweils in eine eigene Datei. Machen Sie diese mit **chmod +x** ausführbar und führen Sie die Datei durch Voranstellen von **./** aus.
- In der ersten Zeile Ihrer Firewall-Skripte sollten Sie immer zuerst mit **`iptables -F`** alle bestehenden Regeln löschen.

a) Beispiel 1: - schreiben Sie in die Datei **`erlaube_ping.sh`** folgende Befehle,
- machen Sie die Datei mit **`chmod +x erlaube_ping.sh`** ausführbar und
- führen Sie sie mit **`./erlaube_ping.sh`** aus.

```
iptables -P INPUT   DROP
iptables -P OUTPUT  DROP
iptables -P FORWARD DROP
iptables -F

iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A INPUT  -p icmp -j ACCEPT
```

Jetzt können Sie alle PCs anpingen und Sie können von allen PCs angepingt werden. Testen Sie das!

b) Beispiel 2: Ändern Sie die obige Datei wie folgend dargestellt ab und speichern Sie sie unter **`outgoing_ping_only.sh`** , machen Sie sie ausführbar und führen Sie sie dann aus.

```
iptables -P INPUT   DROP
iptables -P OUTPUT  DROP
iptables -P FORWARD DROP
iptables -F

iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT  -p icmp --icmp-type echo-reply   -j ACCEPT
```

Jetzt können Sie alle PCs anpingen, aber Sie können von niemandem angepingt werden. Warum?

c) Beispiel 3: Ändern Sie die obige Datei wie folgend dargestellt ab und speichern Sie sie unter **`ping_und_http.sh`** , machen Sie sie ausführbar und führen Sie sie dann aus.

```
iptables -P INPUT   DROP
iptables -P OUTPUT  DROP
iptables -P FORWARD DROP
iptables -F

iptables -A OUTPUT  -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT   -p icmp --icmp-type echo-reply   -j ACCEPT

iptables -A OUTPUT  -d 192.168.1.242 -p tcp --dport 80 -j ACCEPT
iptables -A INPUT   -s 192.168.1.242 -p tcp --sport 80 -j ACCEPT
```

Hier ist folgendes eingestellt:
- Die Standard Policies der Ketten sind auf Abweisen aller Pakete ohne Rückmeldung eingestellt
- Ping nach außen und die Ping-Antworten von außen sind erlaubt
- HTTP-Verbindungen sind nur zum Rechner `192.168.1.242` erlaubt

Testen Sie die Firewall, indem Sie andere PCs und den Rechner `192.168.1.242` anpingen.
Testen Sie auch den HTTP-Zugriff auf `192.168.1.242`

Hinweis: HTTP-Server kann man auch mit **`telnet IPServer 80`** testen (hier: **`telnet 192.168.1.242 80`**) dann blind **GET /** eingeben und zweimal die RETURN-Taste drücken....

2.) Aufgabe: Erfragen Sie die IP-Adresse der VM auf dem Lehrerrechner: _____

Hinweis für Lehrer: Download von: http://intranet/files/VMs/IPtables-test-VM.exe

a) Ergänzen Sie **`ping_und_http.sh`** so, dass zusätzlich noch der Zugriff auf die FTP-/HTTP- und SSH-Server der VM möglich ist. Hinweis: SSH benötigt den TCP-Port 22, FTP die TCP-Ports 20 und 21.

b) Realisieren Sie die Firewall nun über eine *Stateful Paket Inspection* mit folgenden Anforderungen:

Hinweis: siehe Beispiel 5.1
- Die Standard Policies der Ketten auf Abweisen aller Pakete ohne Rückmeldung einstellen
- Nach außen dürfen alle Pakete passieren
- Von außen dürfen nur Pakete von *bereits bestehenden* bzw. dazu *verwandten* Verbindungen hereingelassen werden