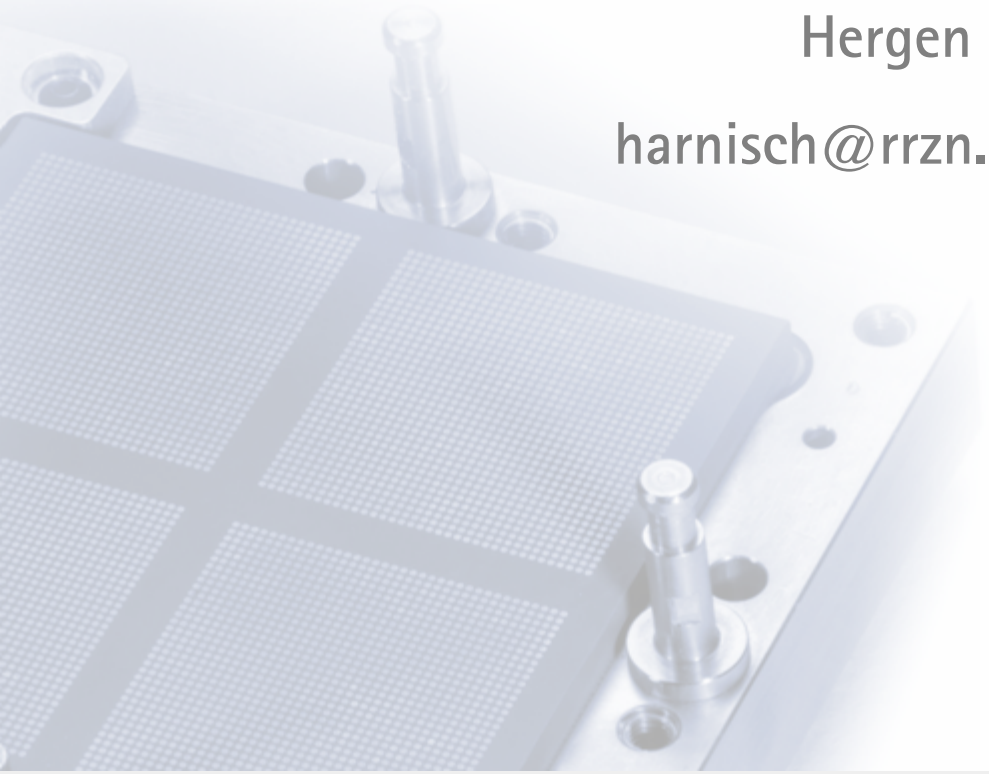
Personal-Firewall

Hergen Harnisch

harnisch@rrzn.uni-hannover.de



1 Grundlagen:

- IP-Kommunikation
- Firewall

2 Personal Firewall:

- Funktionalitäten

3 Implementationen:

- Windows
- Linux
- OpenBSD

4 Schlussbemerkung

Adress-Informationen

Ethernet

Identifikation durch Hardware-Adressen (MAC)

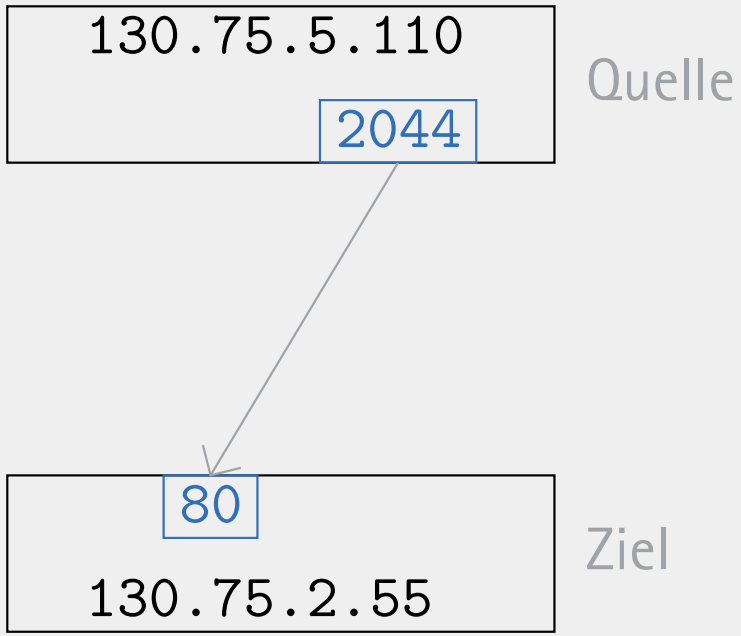
TCP/IP

IP-Pakete haben:

- Quell-IP-Adresse
- Ziel-IP-Adresse

TCP & UDP zudem:

- Quell-Port
- Ziel-Port



R|R|Z|N

IPv4

Jeder Netzwerkanschluss erhält eine 32-bittige IP-Adresse, z.B. 130.75.5.110.

- einige Adressen sind reserviert
  - 127.*.*.* ist rechnerintern (loopback)
  - private Adressen werden nicht geroutet und sind lokal frei wählbar aus definierten Bereichen (vgl. RFC 1918):

    10.*.*.* zukünftig unintern evt. koordiniert vergeben
    172.16.*.*–172.31.*.* teilweise in LUH schon übergreifend im Einsatz
    192.168.*.* meist an DSL-Routern im Einsatz, frei verwendbar
  - 169.254.*.* sind Link-Local (automatische Adressaushandlung)
  - Multicast 224.0.0.0/4 (kaum in Nutzung)
- 130.75.*.* ist der Adressbereich der LUH (Vergabe über RRZN)
- Zusammenfassung von Adressen in „Netzwerke“ mit Bitmasken:
  - eher älter: vorne beginnend führende Bits mit 1 kennzeichnen
    LUH: 130.75.0.0/255.255.0.0 RRZN: 130.75.6.0/255.255.240.0
  - moderner ist CIDR: Zahl der fixen führenden Bits angeben
    LUH: 130.75.0.0/16 RRZN: 130.75.0.0/20

CIDR-Notation

Früher gab es nur feste Klassen und eher „Byte-Masken“:

Klasse	Netzmaske	Zahl Adressen
A	8 (255.0.0.0)	
B	16 (255.255.0.0)	65
C	24 (255.255.255.0)	256

Jetzt beliebige Maskenlänge (*classless*):

$$130.75.5.110 = \underbrace{\underbrace{\underbrace{1000\,0010}_{16=\text{LUH}}\,|\,0100\,1011}_{24=\text{RRZN}-\text{Mitarbeiter}}\,|\,0000\,1001}_{130.75.5.108-130.75.5.111}\,|\,0110\,11\,10$$

CIDR-Notation auch unabhängig vom Routing verwendbar!

R|R|Z|N

IPv6

Derzeit in der LUH nicht im Einsatz, kurzfristig auch nicht geplant.
Aber Betriebssysteme haben es implementiert und verwenden es.

128 Bit statt 32 Bit, Netzmasken-Schreibweise mit CIDR-Notation.
(„::“ heißt mit 0 auffüllen, „:“ trennt 16-Bit-Blöcke)

- Loopback: ::1/128
- 6to4-Tunnel: 2002::/16 bzw. 2002:824b::/32 für die LUH
- Teredo-Tunnel: 2001:0::/32
- nativ die LUH: 2001:0638:0606::/48
- IPv4-mapped IPv6: ::FFFF:824b:56e (130.75.5.110)

⟶ derzeit zu beachten: IPv6 auf dem Loopbackdevice muss funktionieren!

Grundlagen: IP-Kommunikation

R|R|Z|N

ICMP (z.B. ping)

neben Quell- & Ziel-IP noch Pakettyp (z.B. 8 für Echo-Request)

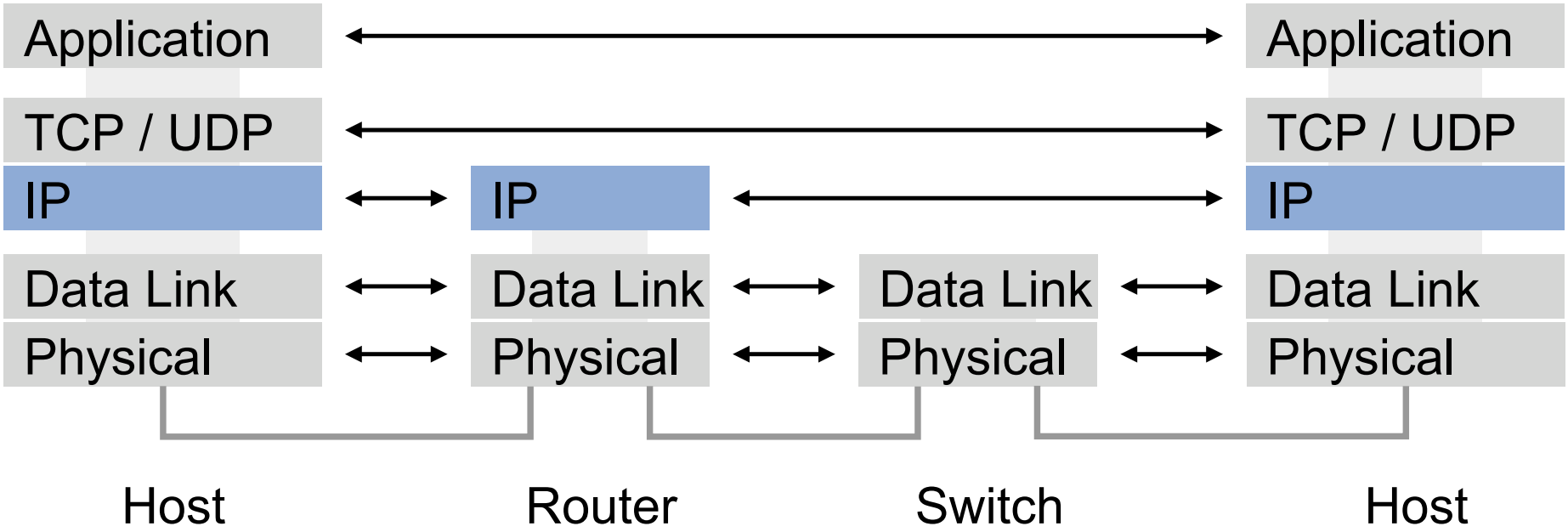
TCP (z.B. http) / UDP (z.B. dns)

neben Quell- & Ziel-IP zusätzlich einen Quell- und einen Ziel-Port

UDP zustandslos (aber trotzdem „Kommunkations-Verbindungen“)
Absender-IP sehr leicht fälschbar

TCP verbindungsorientiert (klar definiert: Aufbau, Paket-Reihenfolge, Ende)

Protokoll-Schichten



(Quelle: C. Grimm, RRZN)

R|R|Z|N

TCP/IP & OSI

relevante Layer des OSI-Modells (üblicher Vergleich, nicht exakt passend):

7 Applikationsprotokoll (z.B. http, ftp)
4 TCP- & UDP-Kommunikation
3 IP-Kommunikation, auch ICMP
2 Ethernet-Kommunikation (MAC-Adresse, ARP)

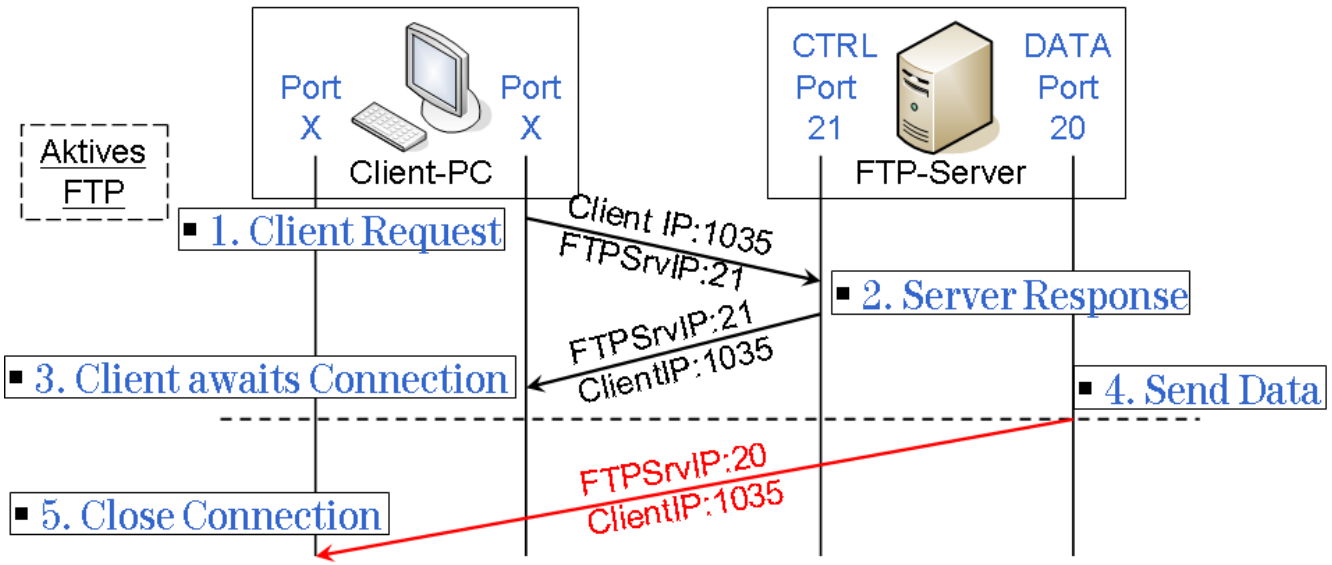
- Layer 2 eigentlich nur bei transparenten Hardware-Firewalls
- stateless Filtering für Layer 3 & 4
- statefull Firewalling betrifft Layer 3–5
- proxy-basierte Gateways oder Deep-Packet-Inspection sind Layer 7

PFW: meist statefull, im Netzwerksinne nur bis Layer 4/5

R|R|Z|N

ftp (Beispiel für komplexere Kommunikationsbeziehung)

1. Client baut Control-Verbindung (TCP mit Zielport 21) auf
2. Ziel-IP (Client bei active-mode, Server bei passive-mode) und Ziel-Port für Datenverbindung werden auf der Control-Verbindung ausgehandelt
3. Datenverbindung wird zusätzlich aufgebaut (durch Server im active-mode, durch Client bei passive-mode)



(Quelle: Wikipedia)

ftp und Firewalling

Problem

Zielport der Datenverbindung ist nicht festgelegt sondern wird dynamisch gewählt.

Lösung

Kommunikation auf Control-Verbindung wird auch inhaltlich durch Firewall belauscht und Datenverbindung dynamisch als ebenfalls zulässig eingestuft.

R|R|Z|N

trotzdem Probleme

- Verschlüsselung (SSL): Lauschen nicht möglich
- auch andere Protokolle benötigen zusätzliche Ports (z.B. sip, H.323), Firewall muss jedes dieser Protokolle kennen und gesondert behandeln
- einige dieser Protokolle einfach zu komplex (z.B. H.323)

einfache aber unschöne Lösung

freischalten von allen möglichen Ports (z.B. alle TCP $\geq$ 1024),
teilweise in Servern/Clients die Port-Range einschränkbar

einzige sichere Lösung

dedizierte Proxies für jedes komplexere Protokoll

übliches Setup

verbreitete Art der Filterung ist stufig:

- stateless durch ACLs in Routern und Switchen
- statefull in dedizierten Firewalls (teilweise auch Router)
- statefull mit Unterstützung für komplexere Protokolle wie ftp oder sip nur in dedizierten Firewalls
- Layer-7-Filterung durch Sperrung des Direktzugriffs und Nutzung eines speziellen Proxies

Layer 7

Häufig wird Layer-7-Firewalling mit Intrusion-Prevention (eigentlich auch > Layer-7) gekoppelt, teilweise werden Begriffe uneinheitlich verwendet. Manchmal ist Layer-7-Filterung / IPS nur ein Blacklist-Ansatz.

LUH

eigentlich recht ähnlich, aber relativ offenes Netz

- ACLs in Routern/Switchen für Flooding-Ports, Windows-Ports
- statefull mit Unterstützung für ftp etc. in Gateway- und ggf. Netzschutz-Firewalls
- Layer-7 nur die Ausnahme, teilweise z.B. bei Mail und Update per WSUS

R|R|Z|N

Grenzen einer Firewall

Gegen viele Angriffe kann eine Firewall nicht helfen

- Lokale Angriffe am PC / Server
- Angriffe aus dem LAN
- Angriffe auf den Dienst eines Servers (z.B. php-Exploits)
- Drive-By-Downloads
- Software zur Umgehung von Firewalls (z.B. bei Skype):
  - UDP-Hole-Pinning (möglich da zustandslos)
  - Nutzung von Standard-Ports (gerne TCP-Zielport 80)

Dennoch: Firewall ist ein wichtiger Basis-Schutz für ein Netzwerk (z.B. durch RRZN-Netzschutz) *und* einen Rechner selbst (Personal-Firewall)

Begriff

Filter-Software auf einem Server

- Teil des Betriebssystems oder eine Anwendungssoftware
- zwischen TCP/IP-Stack des OS und Anwendungssoftware
  oder Teil des TCP/IP-Stacks
  (normalerweise recht tief im System verankert)
- keine eigenständige Einheit sondern unterliegt dem OS

R|R|Z|N

Zusatzmöglichkeiten

- Entscheidungen können nicht nur auf Netzparametern beruhen:
  - je nach Nutzer
  - je nach Prozess oder Programm
- Filterregeln ggf. abhängig vom Standort, (ggf. dynamischer) IP
- Nutzerinteraktion
- Interaktion mit komplexen Programmen zur Entscheidungsfindung

R|R|Z|N

Filterung eingehenden Datenverkehrs

- Prüfung erfolgt in einfacherer Software und früher als in Dämon
- Fehlkonfiguration kann abgefangen werden
- teilweise einzige Möglichkeiten der IP-basierten Filterung
  - kein anderer Einfluss auf Start eines Dämon
  - Dämon hat keine Filterfunktionalität

Es sollten immer Dämon *und* Firewall zur Filterung verwendet und entsprechend konfiguriert werden.

R|R|Z|N

Filterung ausgehenden Datenverkehrs

Idee

Nur zugelassen Software darf Verbindungen nach Außen aufbauen,
Trojaner und Bot-Software darf nicht und fällt auf.

Problem

Malware

- umgeht Firewall,
- simuliert Zustimmung des Nutzers,
- tarnt sich als erlaubte Applikation,
- schleust zusätzliche Daten in laufende Verbindungen ein (covert channel)

R|R|Z|N

Bewertung Ausgehend-Filterung

Beides richtig, wichtig ist Aufwand-Nutzen-Betrachtung:

Statische Regeldefinition

- auf Port-Ebene machbar, auf Programmen (und deren Versionen) eher aufwändig
- meist nur bei Servern sinnvoll

Nutzernachfrage

- Auf Servern unmöglich, normaler Desktop-Nutzer ist überfragt.
- Aufwand für Nutzer groß

Unterdrückte Malware-Verbindungen aber unbedingt reporten!

R|R|Z|N

Empfehlung

- ausgehende Filterung nur bei (einfachen) Servern und dann auch nur auf Port-Ebene
- unbedingt Verstöße loggen, um über Malware informiert zu werden

Beispiel Spamming

Ist Malware, die Spam verschickt, auf einem Rechner, so hilft meist eine Filterung, die ausgehend Zielport TCP-25 (smtp) nur auf RRZN-Server zulässt.

- Ohne Filterung wird zwar Spam verschickt, das fällt aber auf
  - durch Spam-Meldungen von extern
  - durch Netzwerk-Überwachung
- Mit Filterung ohne Logging bleibt Malware unentdeckt.

R|R|Z|N

Umgehung ausgehender Filterung

Malware

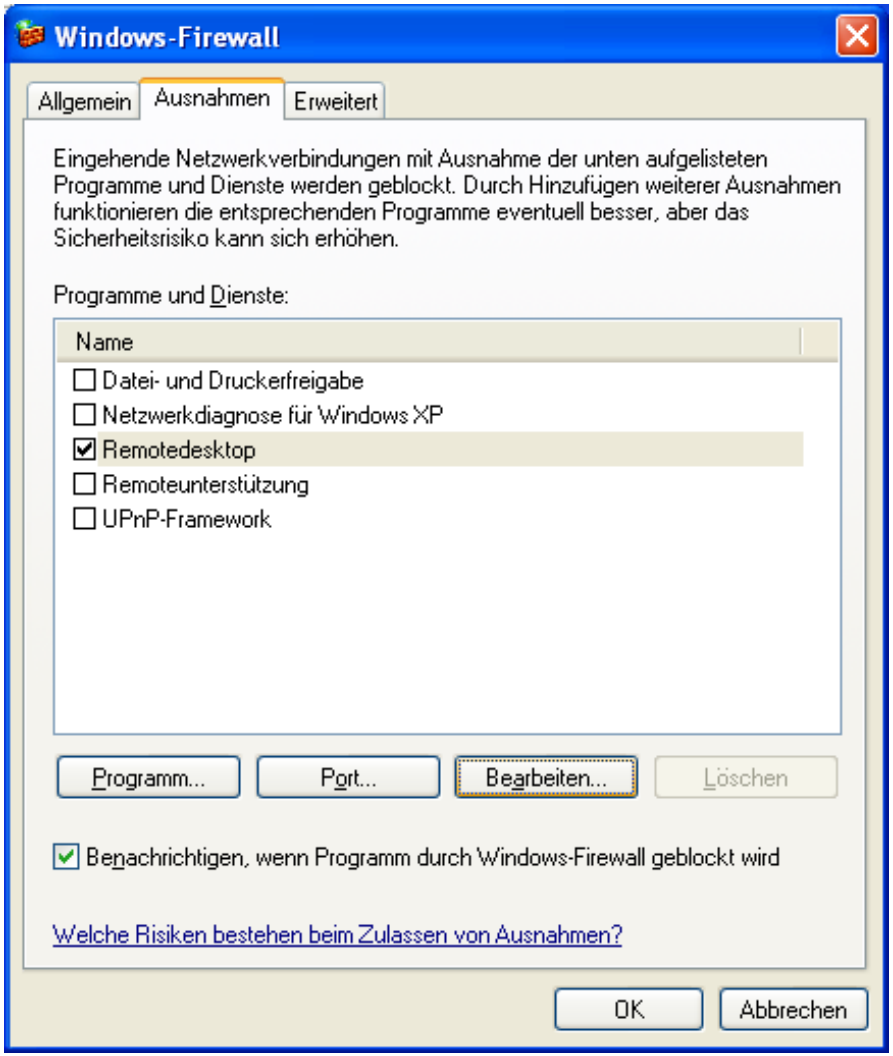
- umgeht Firewall:
  lokale Privilegien-Erhöhung leicht, Nutzer arbeiten häufig mit Administrator-Rechten
- simuliert Zustimmung des Nutzers:
  Malware kennt Firewall-Produkte und „klickt„ auf Zulassen
  (bei Vista dank UAC deutlich schwieriger)
- tarnt sich als erlaubte Applikation:
  Malware-Prozess/-Datei heißt einfach wie der Internet-Explorer
  (Firewall-Produkte kennen z.T. die normalen Applikationen bereits, aber (Sprach-)Versionen und Patchstände machen es schwer)

Windows-Firewall (XP, 2003)

- Verwaltung über GUI, Gruppenrichtlinien, CLI (`netsh firewall`)
- statefull
- kann nur eingehenden Datenverkehr filtern
- wie die Netzwerkeinstellungen an sich: nicht mehrere Standorte
- ACLs sind dämon- oder port-basiert
- bei Aktivierung von Windows-Diensten werden Ausnahmen teilweise automatisch konfiguriert, die dann voreingestellt weltweit
- Einschränkung auf zugelassene IP-Bereiche / Liste möglich, nicht expliziter Ausschluss
- Regelwerk hat keine Reihenfolge, ist Vereingung von zugelassenen Ausnahmen

Implementationen: Windows

Beispiel RDP



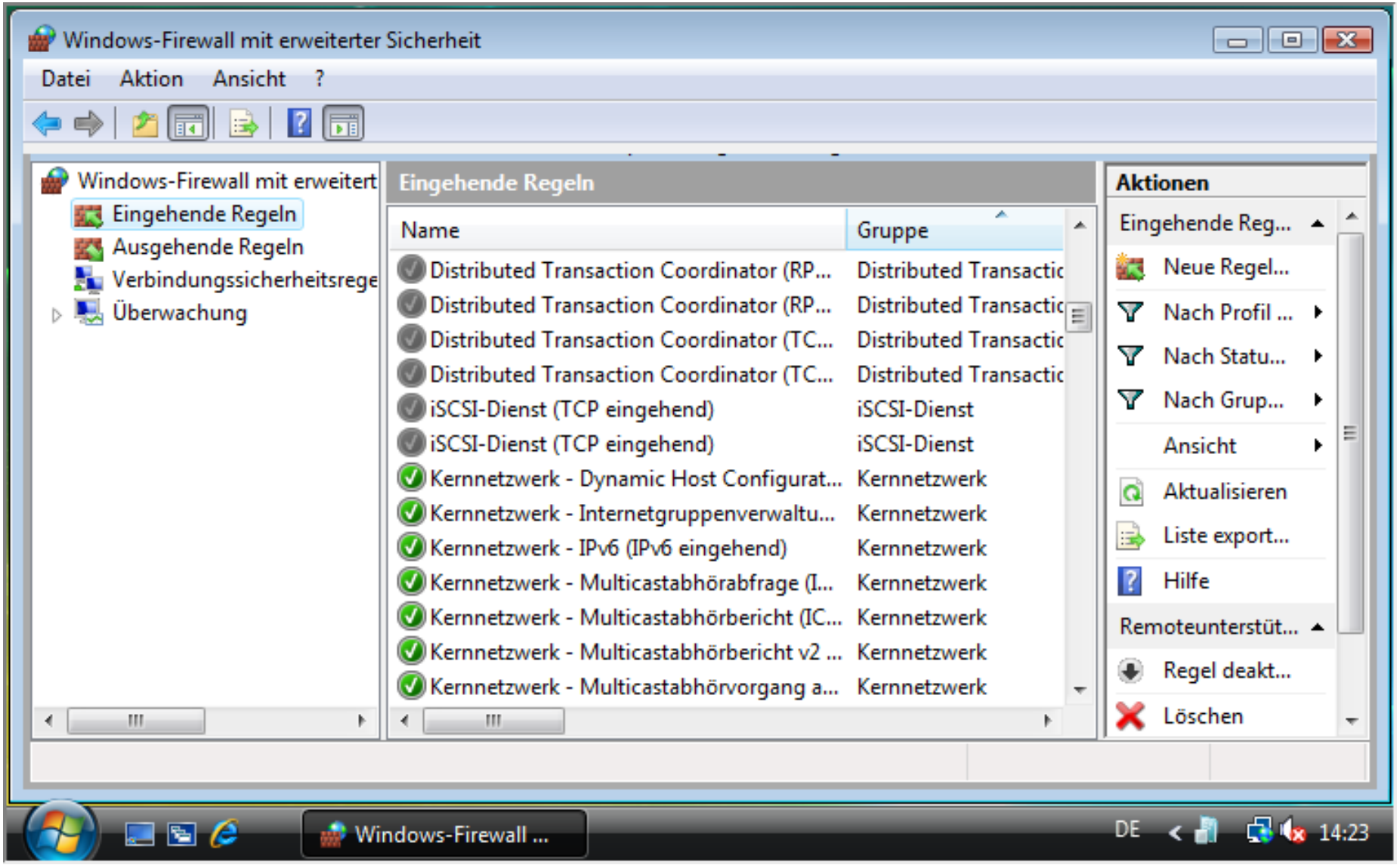
Firewall in Vista, 2008

- Verwaltung nun über MMC-Plugin oder CLI (`netsh advfirewall`)
- statefull
- kann auch ausgehenden Datenverkehr filtern
- mit VPN-Verwaltung zusammengelegt
- mehrere Profile möglich
- Regeln & deren Namen: schwer zu finden, sprachabhängig, nicht port-basiert
  (z.B. „Datei- und Druckerfreigabe (Echoanforderung – ICMPv…“)

… insgesamt sehr komplex geworden

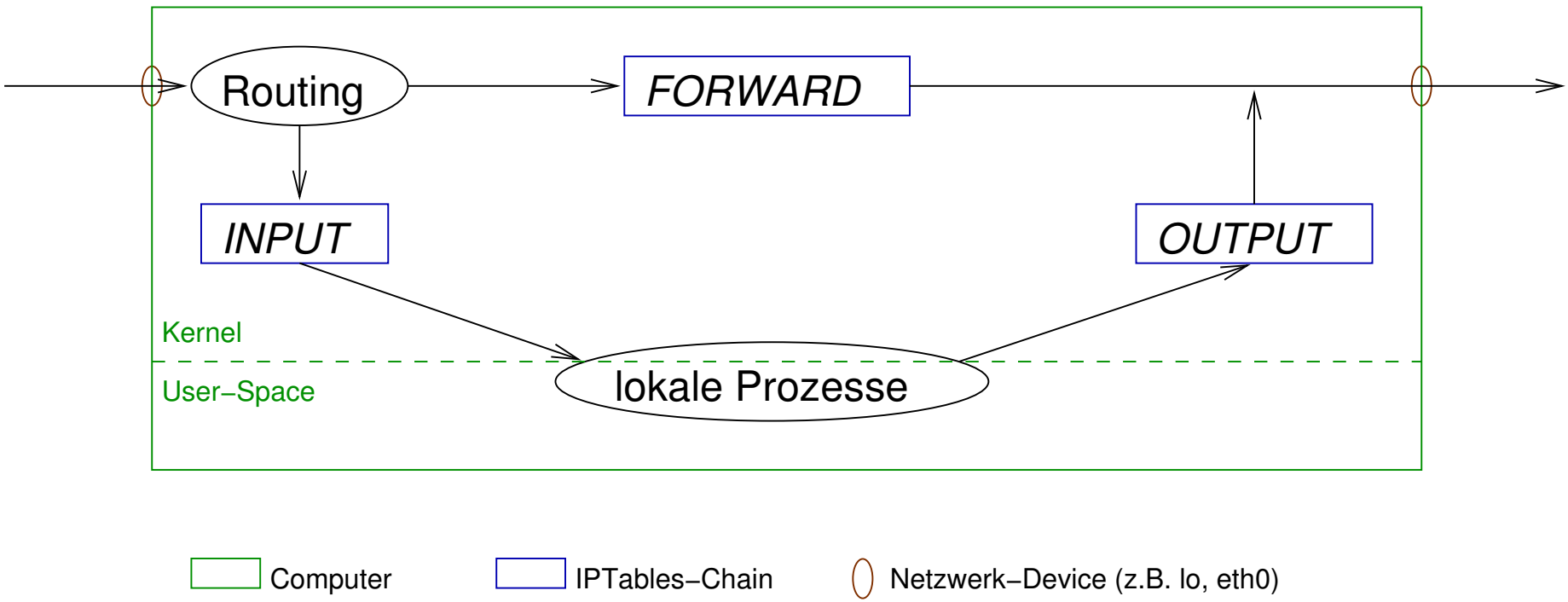
Implementationen: Windows

Bsp. Vista



Schema

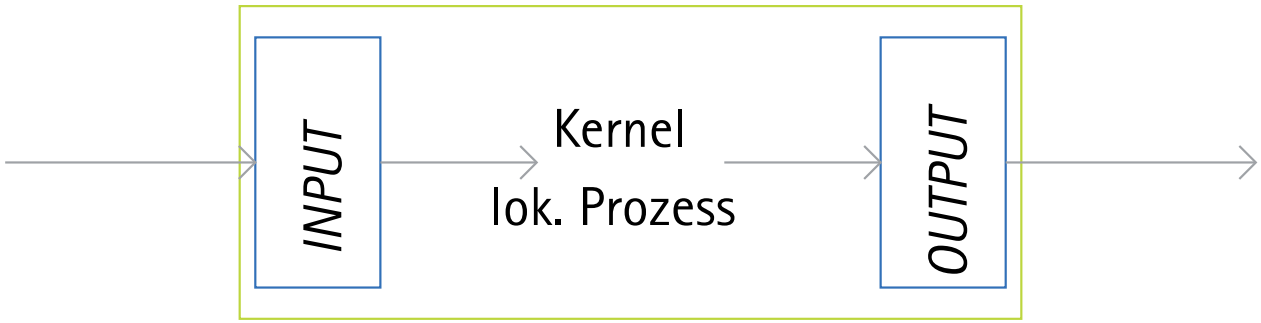
Aufbau IPTables



Implementationen: Linux

für Server / Clients

- kein Routing: `FORWARD` uninteressant
- jedes reinkommende Paket geht durch `INPUT`
- jedes rausgehende Paket geht durch `OUTPUT`
- nach Eintrag der „Stateful-Regel“ in `INPUT` & `OUTPUT`: nur noch „Verbindungs“-Aufbauten
  - nach Innen / zum Rechner in `INPUT`
  - nach Außen / vom Rechner in `OUTPUT`



R|R|Z|N

Grund-Setup

Standard-Policy

```
iptables -P INPUT DROP
# erstmal raus alles, spaeter ggf. restriktiver
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
# ggf. alle chains komplett leeren
iptables -F
```

Stateful

```
iptables -A INPUT -m state \
    --state RELATED,ESTABLISHED -j ACCEPT
```

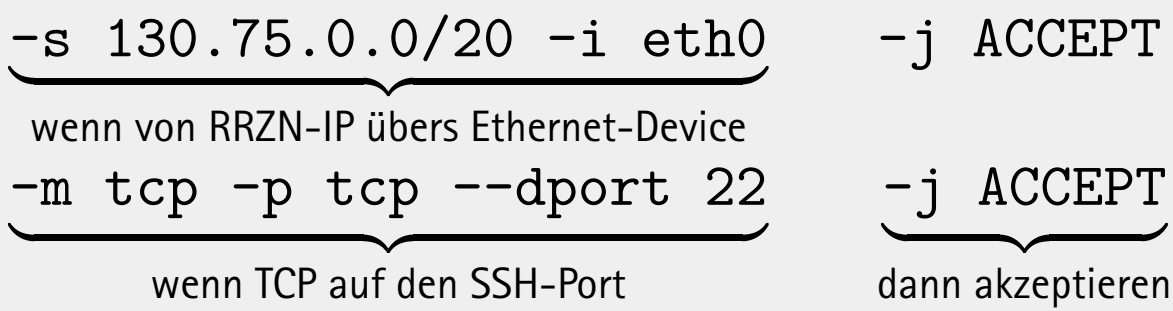
$\longrightarrow$ bereits gute Absicherung eines Clients, aber noch zulassen:

```
iptables -A INPUT -i lo -j ACCEPT
```

Prinzip

Einzelregel

Jede Regel besteht aus einer *Bedingung* und einem *Ziel*:



Das Ziel bestimmt das weitere Schicksal eines passenden Paketes.

R|R|Z|N

Prinzip

Abfolge

Jede *chain* besteht aus einer Abfolge von Regeln,
Reihenfolge ist entscheidend: Abarbeitung bis *first match*,
d.h. nach einem „Match“ wird Regelverarbeitung abgebrochen

Beispiel

```
1 -p tcp -d 130.75.2.0/24 --dport 25 -j ACCEPT
2 -p tcp --dport 25 -j DROP
3 -j ACCEPT
```

akzeptiert Mail an Mailserver des RRZN,
Zustellung an andere Mailserver nicht erlaubt;
restlicher Verkehr wieder erlaubt

Ziele

`ACCEPT` Das Paket wird durchgelassen.

`DROP` Das Paket wird ohne Benachrichtigung/Rückmeldung verworfen.

`REJECT` Das Paket wird verworfen, aber der Absender wird benachrichtigt:

- standardmäßig per ICMP `port-unreachable`
- alternativ auch mit Beendigung einer TCP-Verbindung, sinnvoll für Ident-Port (Vermeidung von Timeouts):
  `-p tcp -dport 113 -j REJECT --reject-with tcp-reset`

`REJECT` kann nicht Chain-Policy sein und benötigt Zusatzmodul.

R|R|Z|N

Matches

IP-Adresse

Quelle: `-s [!] address[/mask]` Untersuchung der Absender-IP-Adresse (! negiert)

z.B. nicht aus LUH-Netz: `-s ! 130.75.0.0/16`

Ziel: `-d [!] address[/mask]`

TCP/UDP-Ports

Quelle: `-p tcp|udp --sport [!] port`

Ziel: `-p tcp|udp --dport [!] port`

z.B. alle TCP-Verbindungen außer ssh: `-p tcp --dport ! 22`

Matches

ICMP

Typ: `-p icmp --icmp-type [!] type`

z.B. ping-Anfrage: `-p icmp --icmp-type echo-request`

state-Modul

Status: `-m state --state state`

wobei *state* eine Kombination aus

- ESTABLISHED Teil einer existierenden Verbindung
- RELATED zwar neue Verbindung, aber zu existierender gehörend (z.B. FTP-Data zu FTP-Control)
- NEW gehört zu nichts Bekanntem
- INVALID ungültig (oder auch Speicher-Problem des Moduls)

Beispielregeln

```
01 iptables -P INPUT DROP
02 iptables -P OUTPUT ACCEPT
03 iptables -P FORWARD DROP; iptables -F
04 iptables -A INPUT -m state -state RELATED,ESTABLISHED \
                                                    -j ACCEPT
# rein: lokal, ping, ssh aus LUH
05 iptables -A INPUT -i lo -j ACCEPT
06 iptables -A INPUT -p icmp -icmp-type 8 -j ACCEPT
07 iptables -A INPUT -s 130.75.0.0/255.255.0.0 -p tcp \
                                          --dport 22 -j ACCEPT
# raus: alles, aber Mail/DNS nur LUH
08 iptables -A OUTPUT -o lo -j ACCEPT
09 iptables -A OUTPUT -d ! 130.75.0.0/16 -p tcp \
                                        --dport 25,465 -j DROP
10 10 iptables -A OUTPUT -d 130.75.1.32/32 -j ACCEPT
11 iptables -A OUTPUT -d 130.75.1.40/32 -j ACCEPT
12 iptables -A OUTPUT --dport 53 -j DROP
```

Implementationen: OpenBSD

pf

Grundprinzip ähnlich zu IPTables, aber

- andere Match-Philosophie: last-match statt first-match
- Protokoll-Unterstützung nicht im Kernel(-Modul), sondern in separaten Proxies (userspace)
  $\longrightarrow$ komplexe Auswertung läuft mit wenigen Rechten, aber Performance-Verlust
- stateless oder statefull je Regel definierbar
  $\longrightarrow$ Sehr praktisch bei DNS- und Log-Servern (viele UDP-„Verbindungen“ mit Timeout als Ende)
- Traffic-Shaping direkt mit Regeln angebbar
- gute Unterstützung für CIDR-Tabellen

R|R|Z|N

Empfehlung

- Verwenden Sie PFW auch, wenn Sie am Netzschutz teilnehmen.
  (mehrstufige Sicherheit, Absicherung gegen LAN)
- Filtern Sie eingehenden Datenverkehr, ausgehenden nicht.
  (Aufwand-Nutzen, Bedienbarkeit)
- Erlauben Sie keinesfalls allen Datenverkehr aus dem LAN.
  (wichtiger Zusatznutzen gegenüber Hardware-Firewall)
- Lassen Sie immer ICMP-echo-request zu, zumindest aus dem LUH-Netz.
  (heute ungefährlich, ping gut für Störungsbeseitigung / Monitoring)
- Ignorieren Sie IDS-Warnungen für Quellen außerhalb der LUH.
  (Portscan durch Externe sind allgegenwärtig)

Vertrauen Sie nicht auf Firewalls allein.

(DSL-Natting-Router verbreitet, modere Malware umgeht das)