Erweiterte Firewall-Techniken

Rene Mayrhofer

Grazer Linuxtage

14.5.2005

13:00 – 14:00

Vortragsinhalt

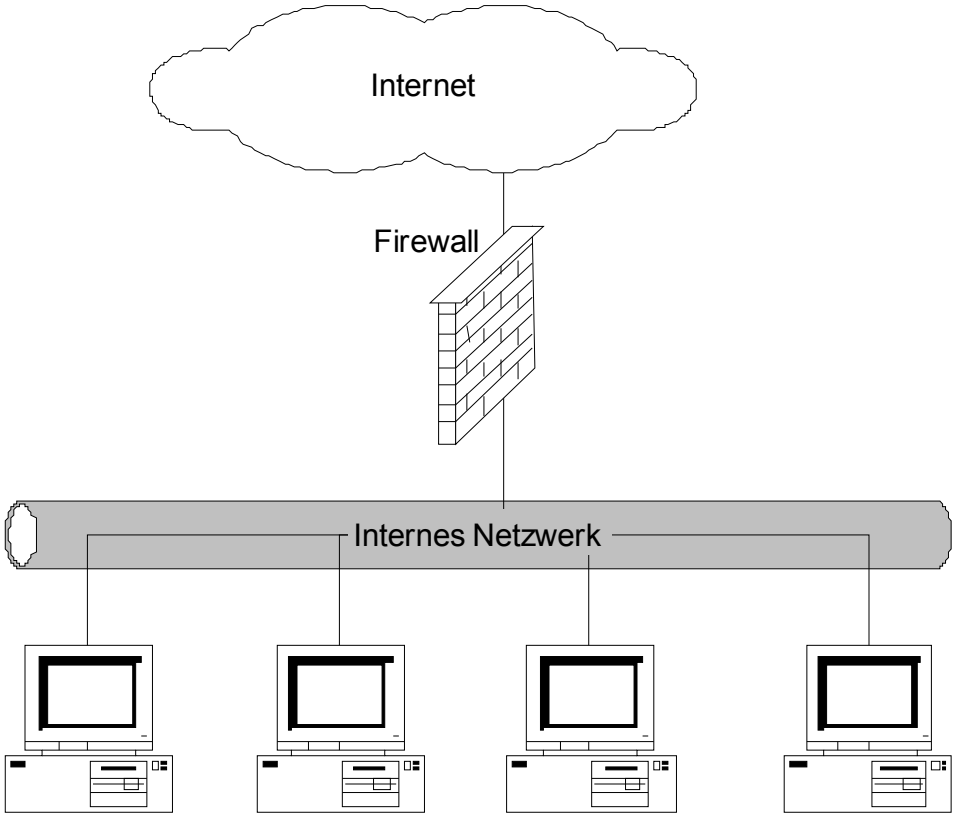
- **Transparente Firewalls**
  - Layer 2 vs. Layer 3 Firewalls
  - Bridging unter Linux
  - netfilter auf einer Bridge
- **VLANs**
- **Traffic Shaping**
  - Was ist überhaupt möglich?
  - Grundmethodik
  - Traffic Shaping unter Linux
  - Zusammenspiel mit Bridging
- **Hot-Standby Firewalls**
  - Prinzipielle Funktionsweise
  - Verschiedene Methoden des Failovers
  - Problem: Connection Tracking
- **Ausblick**

Transparente Firewalls

Firewall, Paketfilter, Proxy-Server, NAT, VPN

GIBRALTAR

Was ist eine Firewall?



ISO/OSI – 7-Schichtenmodell

Schicht	ISO/OSI-Modell		TCP/IP-Modell
7	Applikationsschicht	Applikations-Protokolle	telnet, ftp, nfs rlogin, DNS smtp, snmp X-Windows Socket library
6	Präsentationsschicht		
5	Kommunikations-Steuerungsschicht		
4	Transportschicht	Transport-Protokolle	TCP UDP
3	Netzwerkschicht	Internetwork-Protokolle	IP EGP, RIP ICMP ARP, RARP
2	Sicherungsschicht	Network-Access-Protokolle	Ethernet CSMA/CD Token Ring FDDI
1	Bitübertragungsschicht		

ISO/OSI – 7-Schichtenmodell

Schicht	ISO/OSI-Modell		TCP/IP-Modell
7	Applikationsschicht	Applikations-Protokolle	telnet, ftp, nfs rlogin, DNS smtp, snmp X-Windows Socket library
6	Präsentationsschicht		
5	Kommunikations-Steuerungsschicht		
4	Transportschicht	Transport-Protokolle	TCP UDP
3	Netzwerkschicht	Internetwork-Protokolle	IP EGP, RIP ICMP ARP, RARP
2	Sicherungsschicht	Network-Access-Protokolle	Ethernet CSMA/CD Token Ring FDDI
1	Bitübertragungsschicht		

Mögliche Ebenen für Filterung

- Erinnerung: ISO/OSI Schichtenmodell bietet verschiedene Punkte zum Filtern von Netzwerkverkehr

- Für Firewalls interessant sind:
  - Layer 2: sog. “Bridging” Firewalls, auch als “transparente” oder “unsichtbare” Firewalls beschrieben
  - Layer 3: “normale” Firewalls
  - Layer (5 –) 7: “Deep Inspection” / “Content Inspection” / “Intelligent” / “Smart” / (hier bitte das aktuelle Marketing Buzz-Word der Woche einsetzen) Firewalls

Layer 2 Firewalls

- Firewall funktioniert wie Bridge, d.h. die angeschlossenen Ethernet-Segmente sind transparent auf Ethernet-Ebene miteinander verbunden
- Allerdings: nicht jedes Paket wird weitergeleitet, sondern es wird nach üblichen Firewall-Regeln gefiltert (also z.B. Quell-/Ziel-MAC, -IP, -Port, etc.)

- Vorteile:
  - Kein Routing, also muss auch kein Gateway bei den angeschlossenen Computern eingetragen werden
  - Daher bei bestehenden Netzwerkstrukturen keinerlei Aufwand zur Rekonfiguration – eine Layer 2 Firewall kann als Ersatz für ein Netzwerkkabel „dazwischengesteckt“ werden
  - Firewall selbst benötigt keine IP-Adressen und ist daher über das Netzwerk auch nicht (zumindest nicht direkt!) angreifbar

Layer 3 Firewalls

- Übliche Firewalltechnik, d.h. die Firwall arbeitet wie ein Router
- Mindestens eine IP-Adresse pro Netzwerkschnittstelle
- Angeschlossene Computer verwenden die Firewall als Gateway

- Vorteil: bekannte Struktur

Layer 7 Firewalls

- Einbindung in Netzwerk entweder als Layer 2 oder Layer 3 Firewall
- Untersuchung der Pakete zusätzlich auf höheren ISO/OSI Schichten (Anwendungsschichten 5-7), also im Datenbereich aus Sicht von IP bzw. TCP/UDP ⇒ „Deep Inspection”
- Daher: mehr Information zur Entscheidung ob Paket weitergeleitet oder verworfen/zurückgewiesen werden soll

- Vorteile:
  - Applikationsprotokoll wird in Entscheidung mit einbezogen
    ⇒ mehr Freiraum und Sicherheit
  - Zusatzdienste auf Applikationsebene möglich, die direkt auf den übertragenen Daten arbeiten (applikationsabhängig!)
    z.B.: transparenter Virenschutz (HTTP, FTP, SMTP, POP3, IMAP4, ...), Blockieren von Cookies (HTTP), Blockieren von Javascript etc. (HTTP)
- Nachteile:
  - deutlich höherer Ressourcenbedarf (CPU, RAM, HDD)
  - erhöhte Latenz

Bridging unter Linux

- Prinzip:
  - mehrere (physikalische) Netzwerkinterfaces unter einem virtuellen Bridge-Interface vereint
  - Linux Kernel führt MAC-Tabellen
  - Zusatzfunktionen einer Bridge: STP

- Und wie geht's jetzt im Detail?

```
$ brctl addbr br_int
$ brctl addif br_int eth0
$ brctl addif br_int wlan0
$ brctl setfd br_int 5
$ brctl stp br_int on
$ ip link set eth0 up
$ ip link set wlan0 up
$ ip link set br_int up
$ ip addr add 10.0.0.1/24 dev br_int
```

... just use your shiny new bridge ...

Bridging mit Unterstützung

- Debian

```
/etc/network/interfaces:
      auto br_int
      iface br_int inet static
          address 10.0.0.1
          netmask 255.255.255.0
          bridge_ports eth0 wlan0
          bridge_stp on
          bridge_fd 5
$ ifup br0
```

- Gibraltar

```
      auto br_int
      iface br_int inet scripted
         method scripted
         address 10.0.0.1/24
         route0 default 10.0.0.254
         device br_int
         bridge_ports ext int
         bridge_stp on
         bridge_fd 5
```

Bridging + netfilter

- gebridgte Pakete sollen netfilter Hooks passieren:
  - Ab Kernel 2.6 im Standard, für 2.4 ebtables Patch (inkludiert in aktueller Version den bridge-nf Patch): *http://ebtables.sourceforge.net/*
  - `CONFIG_BRIDGE=m`
    `CONFIG_BRIDGE_NF_EBTABLES=m`

- INPUT, OUTPUT und FORWARD Chains funktionieren dann auf dem virtuellen Bridge-Interface wie erwartet
- Unterscheidung von welchem physikalischen Interface ein Paket empfangen wurde bzw. wohin es gesendet würde?
  - physdev Match für iptables ⇒ patch-o-matic-ng

VLANs

Warum VLANs?

- Viele verschiedene Netzwerksegmente an einer Firewall
  - z.B. extern, intern, DMZ Fileserver, DMZ Authentication Server, DMZ Log Server, WLAN, ...
  - Extremfall: jedes Netzwerkgerät in eigenem Ethernet-Segment zur kompletten Trennung (Angriffe von Innen, Wurmbefall etc.)
- Problem: Nach 12 – 16 Netzwerkinterfaces ist bei PC Hardware doch meist Schluss...
- Lösung bei Ethernet: VLANs
  - mehrere virtuelle Segmente auf einem physischen Switch / einer physischen Netzwerkkarte
  - Trennung durch (12 Bit) ID im Ethernet Header
  - Switch wertet ID aus und sendet in entsprechendes Segment
  - „Tagged“ Ports auf dem Switch dürfen IDs mitsenden bzw. bekommen Pakete aus mehreren Segmenten mit dem ID Header
  - „Untagged“ Ports bekommen Ethernet-Pakete ohne ID aus dem jeweils zugewiesenen Segment

VLANs unter Linux

- Achtung: MTU-Größen!
- Linux-Kernel erzeugt virtuelle Interfaces für jede VLAN ID

```
$ vconfig set_name_type VLAN_PLUS_VID_NO_PAD
$ vconfig add eth0 3
$ ip link set eth0 up
$ ip link set vlan3 up

$ vconfig set_name_type DEV_PLUS_VID_NO_PAD
$ vconfig add eth0 3
$ ip link set eth0 up
$ ip link set eth0.3 up
```

- Virtuelles Interface kann dann so verwendet werden wie physikalisches (Adresse zuweisen, netfilter Regeln, etc.)

VLANs mit Unterstützung

- Debian / Gibraltar

/etc/network/interfaces:

```
auto vlan3
iface vlan3 inet static
        pre-up vconfig set_name_type VLAN_PLUS_VID_NO_PAD
        pre-up vconfig add int 3
        pre-up vconfig set_flag vlan3 1 1
        address 10.0.0.129
        netmask 255.255.255.224
        broadcast 10.0.0.159
        # this hangs the 2.6 kernel (works perfectly on 2.4 though!)
        #post-down vconfig rem vlan3
```

oder (mit neuem vlan Paket):

```
auto eth0.3
iface eth0.3 inet static
        address 10.0.0.129
        netmask 255.255.255.224
        broadcast 10.0.0.159
```

Traffic Shaping

Was geht?
Grundprinzipien,
Traffic Shaping + Bridging

Warum Traffic Shaping?

- Netzwerkverbindungen sind nie schnell genug
- Aber: verschiedene Arten von Traffic müssen nicht gleich schnell behandelt werden ⇒ Priorisierung kann bestehende Verbindung besser ausnutzen
- Daher: Übertrage ausgewählte Pakete vor anderen

Grenzen des Traffic Shaping

- **Mantra**: Man kann nur Traffic kontrollieren, den man selbst erzeugt! (outgoing)
- Grundprinzip des Internet: Paketvermittlung, nicht Leitungsherstellung
- Daher: es kann nicht kontrolliert werden, was Andere an das eigene System senden (incoming)
- z.B. Media Streaming Server (Webradio, Video, ...) sendet UDP Stream schneller als die eigene Downstream-Bandbreite $\Rightarrow$ Leitung überlastet, keine Kontrollmöglichkeit
- (D)DoS Attacken auf Server können verhindert bzw. gemindert werden, (D)DoS Attacken auf die Internet-Anbindung aber nicht!

Grundprinzipien

- Grundprinzip ist **bewusste Verzögerung** von Paketen
- Daher: Traffic Shaping senkt typischerweise die maximale Auslastung einer Leitung (in Bezug auf Gesamtbandbreite)
- verzögere jene Pakete, die niedriger priorisiert werden und versende höher priorisierte Pakete zuerst

- Wiederhole Mantra $\Rightarrow$ Traffic Shaping wird auf outgoing Interface angewandt
- Verzögere Pakete bevor sie das System verlassen (aber nehme sie so schnell wie möglich an)

Traffic Shaping unter Linux

- Wird durch „Queuing Disciplines“ implementiert
- Jedes Interfaces hat immer eine qdisc – wenn nicht explizit angegeben dann naive „pfifo_fast“ qdisc

- Unterscheidung zwischen classless und classful qdiscs:
  - classless: Ziel ist möglichst „faire“ Ausnutzung der zur Verfügung stehenden Bandbreite
    z.B. TBF (Drosselung), SFQ (Fairness)
  - classful: Unterscheidung in verschiedene Klassen von Traffic mit verschiedenen Anforderungen bzw. Prioritäten
    z.B. PRIO (nach TOS), HTB (flexibel), ...
- Filterung entweder über eigene, einfache Filter oder durch Integration mit iptables

Beispiel: Garantierte Bandbreite für ICA (Citrix)

- Verwendung von HTB für Flexibilität und SFQ für Fairness

```
tc qdisc del dev ext root
tc qdisc add dev ext root handle 1: htb default 102
tc class add dev ext parent 1: classid 1:10 htb rate 240kbit ceil 240kbit
   prio1 burst 1k
tc class add dev ext parent 1:10 classid 1:101 htb rate 215kbit ceil
   240.0kbit prio 1 burst 1k
tc filter add dev ext parent 1: protocol ip prio 1 u32 \
        match ip protocol 6 0xff \
        match ip dport 1494 0xffff \
        flowid 1:101
tc qdisc add dev ext parent 1:101 handle 101: sfq perturb 10
tc class add dev ext parent 1:10 classid 1:102 htb rate 25kbit ceil
   150.0kbit prio 2 burst 1k
tc filter add dev ext parent 1: protocol ip prio 2 u32 \
        match ip src 0.0.0.0/0 \
        flowid 1:102
tc qdisc add dev ext parent 1:102 handle 102: sfq perturb 10
```

Hot-Failover Firewalls

Prinzip, Methoden, Probleme

Prinzip des Hot-Failovers

- „Hot“ Failover bedeutet dass ein zweites System neben dem Hauptsystem kontinuierlich mitläuft und „sofort“ übernehmen kann
- Ziel
  - Verfügbarkeit der Firewall erhöhen
- Realisiert
  - Mittels Heartbeat
    - Bereits in Gibraltar integriert
    - Ident konfigurierte Knoten (1 Knoten aktiv)
    - Knoten senden sich Lebenszeichen
    - Stirbt Primary Knoten übernimmt der Stellvertreter
  - andere Möglichkeiten: z.B. ucarp (CARP), keepalived (VRRP)

Möglichkeit zur Adressübernahme

- Virtuelle IP-Adresse:
  - beide (oder auch mehrere) Hosts haben zusätzlich zu eigener IP eine gemeinsame virtuelle IP
  - fällt Master aus, wird virtuelle IP dem Slave zugewiesen
  - Slave sendet „gratitous ARP“ um Hosts von Änderung der IP/MAC Zuordnung zu informieren

- Virtuelle MAC-Adresse:
  - beide Hosts verwenden virtuelle MAC Adresse zusätzlich zu virtueller IP Adresse
  - fällt Master aus, wird virtuelle IP **und** virtuelle MAC dem Slave zugewiesen
  - gratitous ARP nicht mehr nötig

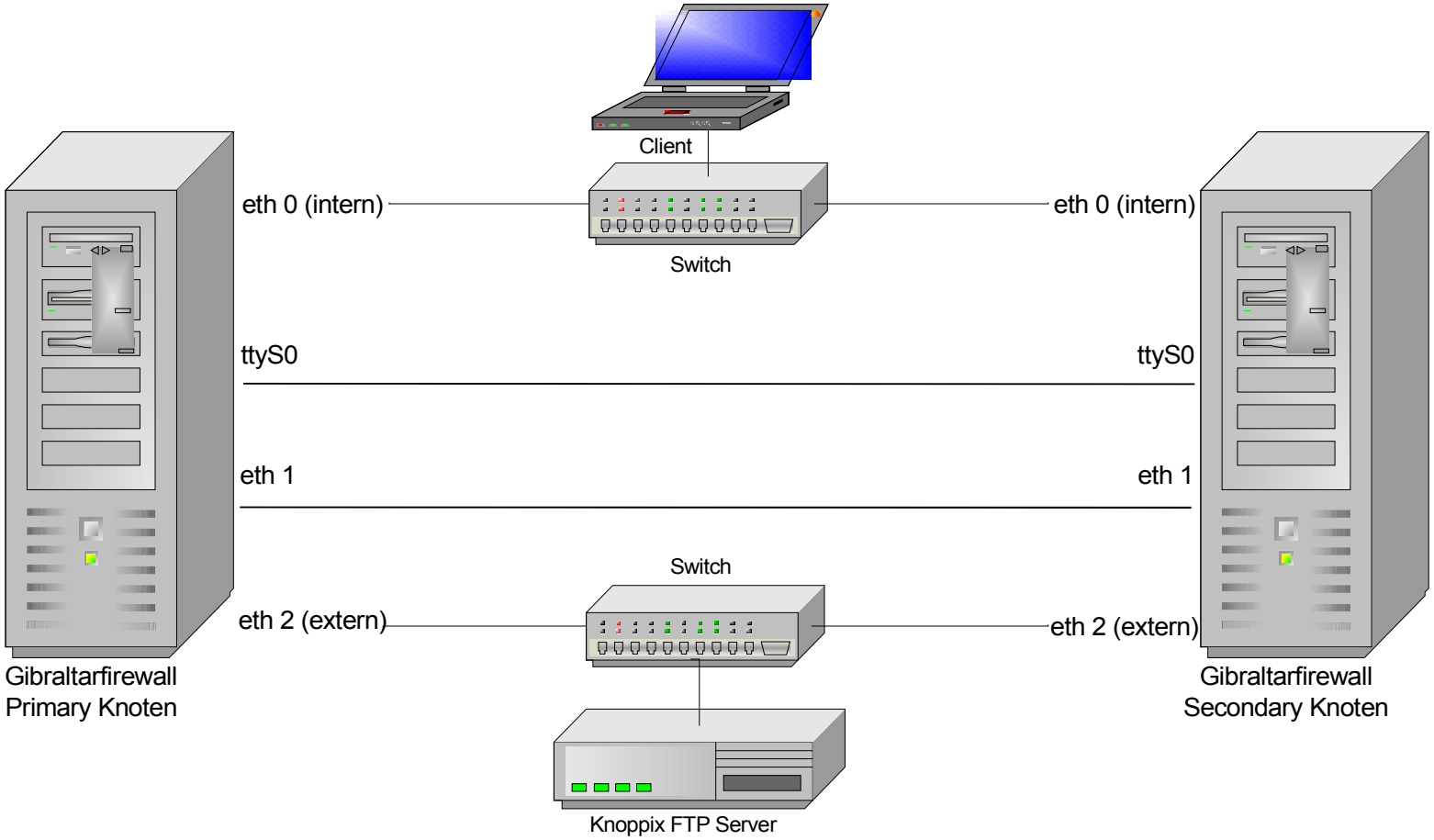
Synchrones Connection Tracking

- Konzept
  - Beide Knoten erhalten identischen Verkehr. Nur einer der beiden Knoten schickt tatsächlich Antwortpakete.
  - Inaktiver Knoten ist geblockt und wird erst nach einem Takeover freigegeben.
  - Einschränkung: Nur der hostübergreifende Verkehr wird mitverfolgt (Forward Chain).

Aufteilung des Verkehrs

- Möglichkeit A:
  - Verwendung eines HUB´s
- Möglichkeit B:
  - Verwendung eines Switches der in den Flooding Mode gebracht wurde (beispielsweise durch Verwendung von Multicast-MAC Adressen.)

Aufbau


Client
Switch
eth 0 (intern)
eth 0 (intern)
ttyS0
ttyS0
eth 1
eth 1
Switch
eth 2 (extern)
eth 2 (extern)
Gibraltarfirewall
Primary Knoten
Gibraltarfirewall
Secondary Knoten
Knoppix FTP Server

Ausblick

Failover der Internetanbindung

- Hot-Failover der Systeme ist ein Teil, aber sind sie auch immer erreichbar?
- Für Ethernet gelöst: Trunking

- Problem bei Internetanbindung: ülicherweise „hilft“ Provider nicht (dynamische Routingprotokolle nur für Großabnehmer)

Load Balancing der Internetanbindung

- Mehrere günstige ADSL Leitungen können deutlich mehr Gesamtbandbreite liefern als eine (teurere) Standleitung
- Idee: verwende verschiedene Provider (bietet auch Failover) und kombiniere verfügbare Bandbreiten

- Problem bei Internetanbindung: ülicherweise „hilft“ Provider nicht (dynamische Routingprotokolle nur für Großabnehmer)

Fragen?

Rene Mayrhofer

Thomas Mayrhofer
eSYS Informationssysteme GmbH
Steinhüblstraße 1
4800 Attnang-Puchheim

www.gibraltar.at

office@gibraltar.at