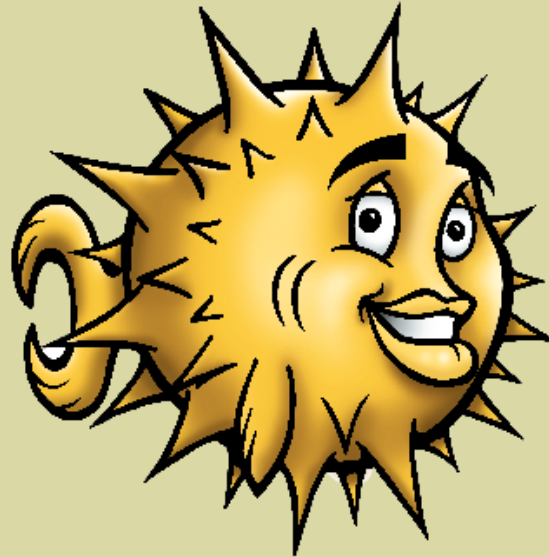


# OpenBSD in VoIP Umgebungen



amoocon 2009

Felix Kronlage

kronlage@bytemine.net  
fkr@openbsd.org



# Fahrplan

- Einführung OpenBSD
- OpenBSD im Netzwerk
- Paketfilter
- Bandbreitenbegrenzung
- Redundanz
- VoIP-Software auf OpenBSD



# Überblick OpenBSD

- Freies unixoides Betriebssystem
- Basierend auf 4.4 BSD lite
- '95 von Theo de Raadt gegründet
- Entwickler von OpenSSH
- Die mit dem Kugelfisch und den schicken T-Shirts
- Aktueller Release: **OpenBSD 4.5**
- Details: <http://www.openbsd.org/>



# Projektziele OpenBSD

- Ein sicheres, BSD-lizensiertes Betriebssystem
  - Zwei Remote-Holes in ~~10 Jahren~~ langer Zeit
- Bestmögliche Entwicklungsplattform
- Komponenten sollen unter freien Lizenzen stehen (BSD, ISC)
- Einhaltung von Standards (POSIX, X/Open)
- Regelmässiger Releasezyklus
  - Alle sechs Monate ein neue Version
  - Die letzten zwei Versionen werden mit Sicherheitspatches versorgt



# OpenBSD auf die Schnelle

- Entwickelt von einem Team
  - Knapp 90 aktive Entwickler
  - Hackathons und Peer-Review
- Evolution statt Revolution
- Kryptographie und Sicherheit
- Entwickler sind für Ihre Dokumentation verantwortlich
  - Fehlende Dokumentation ist ein Bug
- OpenBSD Ports und Pakete



# Secure by default

- Audits
- `strncpy / strlcat`
- $W^X$
- ProPolice StackSmash Schutz
- Privilege Separation / Privilege Revocation
- Sichere Defaults
- `/etc/daily`
- `systrace`



# BSD ist nicht Linux

- Ethernet Interfaces nach Treibern benannt

- */etc/hostname.if*

- */etc/rc.conf, /etc/rc.local*

- */etc/sysctl.conf*

- `net.inet.ip.forwarding=1`

- Systemdokumentation in Form von man-pages

- `bsd make`

- `gmake`



# Infrastruktur mit OpenBSD

- Routing
  - bgpd(8), ospfd(8), ripd(8)
- DNS
  - named(8)
- ntp – Zeitsynchronisation
  - ntpd(8)
- Apache (Fork der 1.3er Linie)
- OpenSMTPd aktuell in Entwicklung





# OpenBSD im VoIP Umfeld

- Absicherung der Infrastruktur
  - Exzellenter und verständlicher Paketfilter
- Transportsicherung
  - Verschiedene VPN Möglichkeiten
- Transportbeschleunigung
  - Traffic-Shaping mittels Alternate Queueing (ALTQ)
- Redundanz
  - Common Address Redundancy Protocol (CARP)



# OpenBSD im VoIP Umfeld

- Last – but certainly not least:

OpenBSD kann auch VoIP servieren!



# OpenBSD Firewall

- ipf(4) wurde 2001 durch pf(4) abgelöst
  - Lizenzprobleme mit Darren Reed
- pf(4)
- sehr mächtiger Paketfilter
  - Mittlerweile auch auf Free-, Net- und DflyBSD
- Last-match-counts
- Lesbare Syntax



# pf(4) Beispiel

```
01  ext_if="vr0"
02  internal_net="192.168.1.0/24"
03
04  set skip on { lo enc0 }
05  scrub in all no-df
06
07  nat pass on $ext_if from $internal_net to any -> ($ext_if)
08
09  block in all
10
11  pass quick inet proto icmp all icmp-type echoreq keep state
12
13  pass in on $ext_if proto tcp from any to port ssh
14
15  pass out on $ext_if
```



# Transportsicherung

- Verschiedene VPN Möglichkeiten
  - IPsec
  - OpenVPN via packages
  - Ad-hoc VPN via ssh-vpn
  - (pptp)



# IPsec

- IPsec wurde bei OpenBSD in-tree entwickelt
  - Kame Implementierung
- `isakmpd(8)` und `ipsecctl(8)`
  - Seit OpenBSD 3.8 dank `ipsec.conf(5)` einfach
  - Syntax ähnlich der von `pf(4)`
- IPsec in der Default Installation bereits vorbereitet
  - *`/etc/isakmpd/local.pub`, `/etc/isakmpd/private/local.key`*



# IPsec(4) Beispiel

*/etc/ipsec.conf* auf Host *foo*:

```
ike esp from 10.1.1.0/24 to 10.2.2.0/24 peer 192.168.10.2
ike esp from 192.168.10.1 to 10.2.2.0/24 peer 192.168.10.2
ike esp from 192.168.10.1 to 192.168.10.2
```

*/etc/ipsec.conf* auf Host *bar*:

```
ike passive esp from 10.2.2.0/24 to 10.1.1.0/24 peer 192.168.10.1
ike passive esp from 192.168.10.2 to 10.1.1.0/24 peer 192.168.10.1
ike passive esp from 192.168.10.2 to 192.168.10.1
```



# Beispielsetup 1

- Zwei Standorte
- Nach aussen reguläre Telefonie -
  - Innerhalb des Unternehmens via IP
- Problem:
  - Telefone machten selbst schon IPsec
  - Im Zusammenspiel mit Cisco IPsec gab es Probleme
- Erster Lösungsansatz:
  - Klartexttunnel (böse!)





# Simple Bridge-Setup

- Auf Host *foo*:

- /etc/hostname.gif0

- ```
tunnel 10.0.0.1 10.0.2.1 up
```

- /etc/bridge/bridge0

- ```
add gif0
```

- ```
add vr1
```

- ```
blocknonip vr1
```

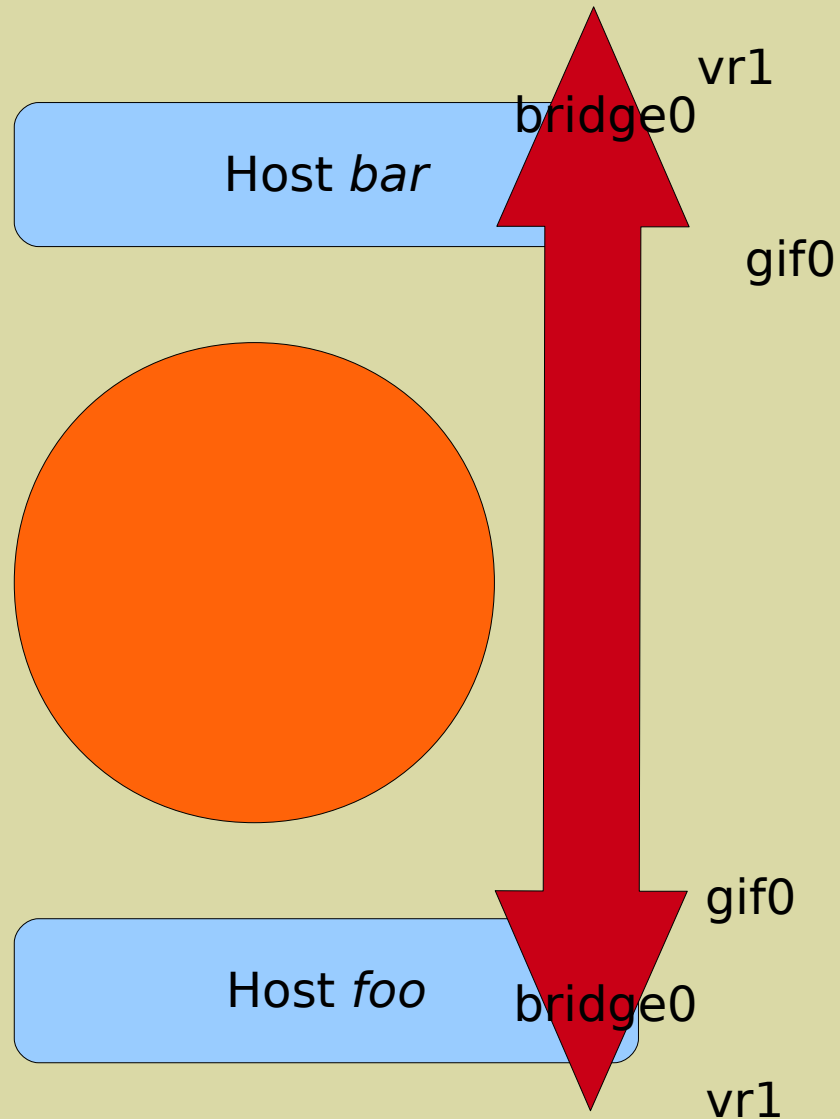
- ```
up
```

- Auf Host *bar*:

- Entsprechend umdrehen → fertig.



# Simple Bridge-Setup



# Beispiel um IPsec erweitert

- Generell: Public Keys austauschen
  - */etc/isakmpd/local.pub* muss auf die Remote Maschine als */etc/isakmpd/pubkeys/ipv4/IP*

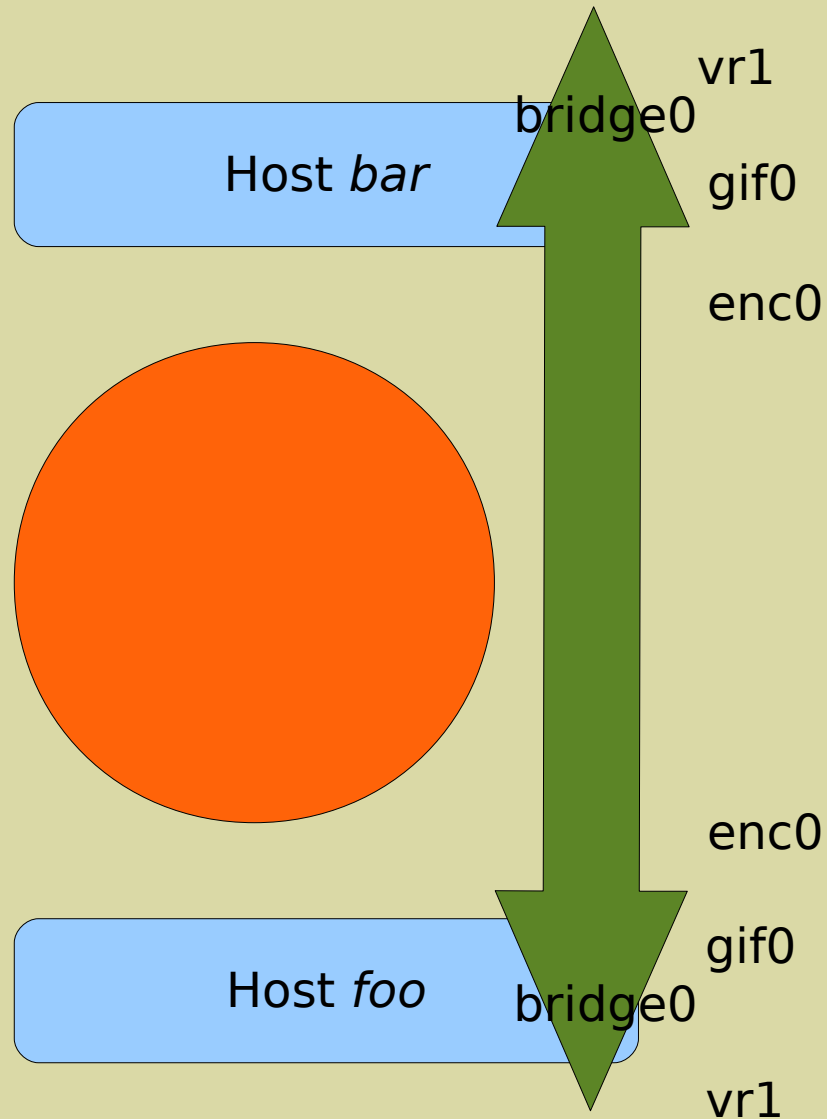
*/etc/ipsec.conf*

ike esp from 10.0.0.1 to 10.0.2.1

- isakmpd starten
  - `isakmpd -k`
- `ipsecctl -f /etc/ipsec.conf`



# Simple Bridge-Setup mit IPsec



# ALTQ - Bandbreitenbegrenzung

- ALTQ seit OpenBSD 3.0 enthalten
  - Seit 3.3 in PF integriert
- Als Default greift der *FIFO* Scheduler
  
- OpenBSD unterstützt desweiteren noch
  - Class Based Queueing
  - Priority Queueing



# Class Based Queueing

- Bandbreite wird in Klassen unterteilt
- Jeder Klasse wird Netzwerkverkehr zugeteilt
  - Source / Destination IP
  - Protokoll
- Jede Klasse bekommt eine mögliche Bandbreite und Priorität zugewiesen

Root Queue (2Mbps)

NetworkA (1Mbps)

ssh (100Kbps)

ftp (900Kbps, borrow)

NetworkB (1Mbps)



# Priority Queueing

- Jedes Interface bekommt Queues zugewiesen
- Jede Queue hat eine Priorität
  - Höhere Priorität wird zuerst abgearbeitet
  - Planung notwendig!
- Keine Verschachtelung von Queues möglich

Root Queue (2Mbps)

Queue A (priority 1)

Queue B (priority 2)

Queue C (priority 3)



# ALTQ Beispiel – Priority Queueing

```
altq on $int_if priq bandwidth 100% queue { std, ssh, sip }  
queue std on $int_if priority 1 priq(default)  
queue ssh on $int_if priority 10 priq  
queue sip on $int_if priority 11 priq
```

```
pass in on $int_if proto { tcp udp } from any  
  to { $int_ips 134.106.146.208 134.106.146.209 } port 22 queue ssh  
pass in on $int_if proto { tcp udp } from $sip_clients to any  
  queue sip  
pass in on $int_if proto { tcp udp } from any to any port  
  $allowed_ports queue std
```





# Redundanz mittels CARP

- Seit OpenBSD 3.5 enthalten
  - *Common Address Redundancy Protocol*
- Redundanzprotokoll für
  - pf mittels pfsync
  - IPsec mittels sasyncd
- Ermöglicht komplettes Failover
  - Master, Slave
  - Master, Master
  - Auch Redundanzverbände möglich



# Redundanz mittels CARP (2)

- Seit OpenBSD 4.5 unicast möglich
- *net.inet.carp.preempt=1* setzen
- Ggf. *net.inet.carp.log=1* setzen



# CARP Beispiel

- */etc/hostname.carpN*

```
inet 10.1.2.3 255.255.255.0 NONE vhid 2  
      carpdev vr0 pass keep_me_secret
```

- */etc/hostname.pfsync0*

```
up syncdev vr3
```

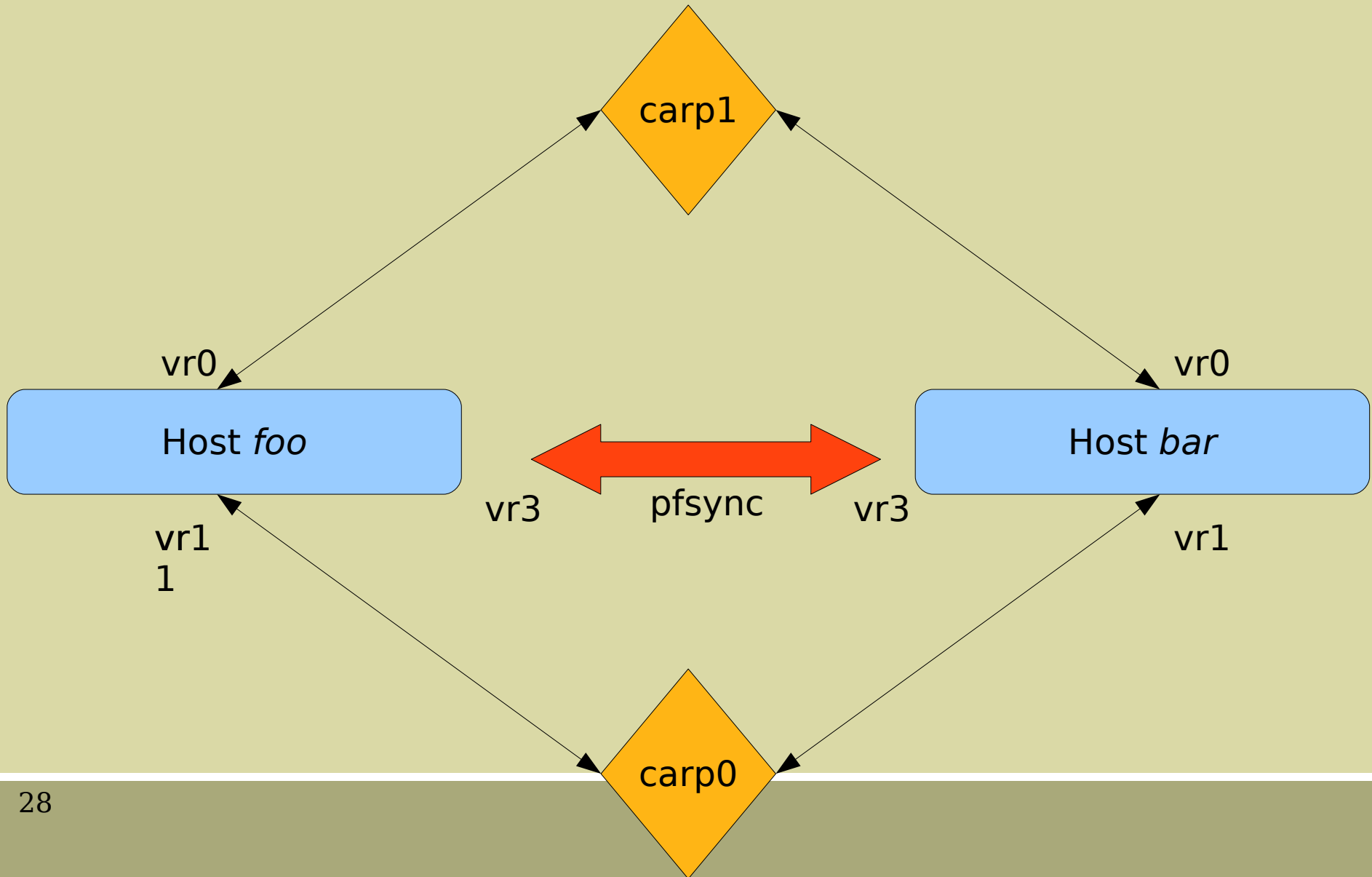
- Wichtig: im Paketfilter entsprechend zulassen

```
pass quick proto carp
```

```
pass quick on $sync_if proto pfsync
```



# CARP Beispiel(2)





# OpenBSD im VoIP Einsatz

- */usr/ports/telephony/*
- In OpenBSD 4.5
  - Asterisk 1.4.22.2
- Asterisk 1.6.0.3 in Arbeit
  - sthen@
- Bisher noch keine Ports:
  - Opensips
  - Yate



# Ressourcen

- <http://www.openbsd.org/faq/>
- Insbesondere die PF-FAQ
  - <http://www.openbsd.org/faq/pf/index.html>
- OpenBSD man-pages
- “The book of pf” - Peter Hansteen
  - <http://nostarch.com/pf.htm>
- <http://blog.bytemine.net/>



# Die berühmte (fast) letzte Seite

- Vielen Dank an Stefan für die Einladung und die Organisation!
- Die Weiterentwicklung von OpenBSD wird am besten durch Spenden an das Projekt unterstützt
- Auch wenn CDs kaufen fast 'out' ist, die CD Releases unterstützen das Projekt  
...und kommen mit Comic und Lied :-)





# Vielen Dank für die Aufmerksamkeit!

kronlage@bytemine.net

bytemine  
Marie-Curie-Str. 1  
26129 Oldenburg

info@bytemine.net  
<http://www.bytemine.net>  
+49-441-309197-0

