OpenBSD Gateway Cluster

Carp, pfsync, pf, sasyncd:
Firewall und IPSec Failover unter OpenBSD

15. September 2005

Markus Wernig

Firewall Grundlagen:
- Sessions
- Stateful inspection
- IPSec

Das Problem: Ausfall
- IP-Adressen
- Active-passive
- Synchronisation

Die Lösung:
- carp
- pfsync
- sasyncd

IP „Session“

Was ist eine „Session“?

Im Firewall-Kontext bezeichnet eine „Session“ einen Satz von **konstanten Parametern,** die zusammengehörige Pakete („Datagramme“) identifizieren. Sie sind in verschiedenen „Layern“ des Paketes gespeichert:

Source IP
Destination IP
Protokoll
} IP Header

Source Port
Destination Port
} TCP/UDP Header

Darüber hinaus gibt es **variable Parameter** wie TTL, TCP-Flags etc.

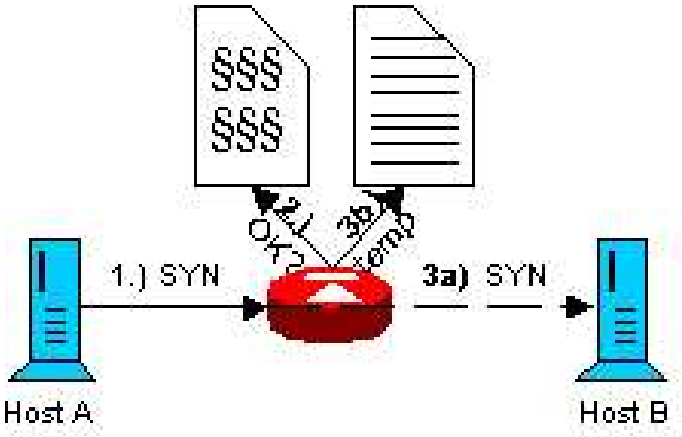
Stateful Inspection

Was ist Stateful Inspection?

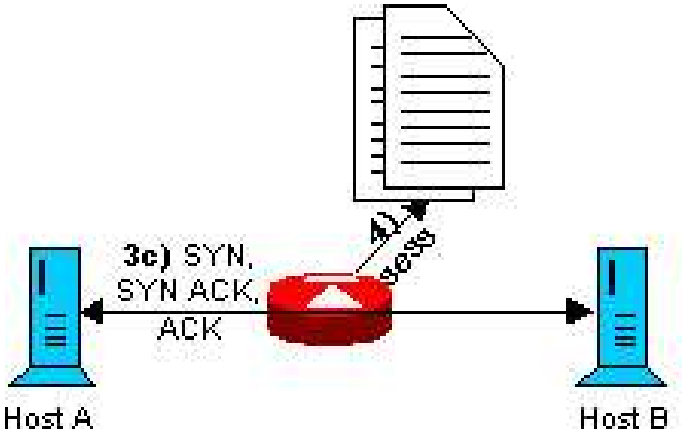
Stateful Inspection bezeichnet die Fähigkeit moderner Firewalls, sich den augenblicklichen Zustand („State“) einer Verbindung („Session“) zu merken und Pakete danach zu beurteilen, ob sie diesem Zustand entsprechen. Dazu führen die Firewalls sog. „State tables“.

Vorteile:
Weniger Rules – es muss nicht für jedes Paket eine eigene Rule geschrieben werden
Höhere Performance – die meisten Pakete müssen nur mehr gegen die State table geprüft werden.
Höhere Sicherheit – Verkehrsströme können auf ihre Protokollkonsistenz geprüft werden, eingeschleuste Pakete werden erkannt.

Stateful Inspection

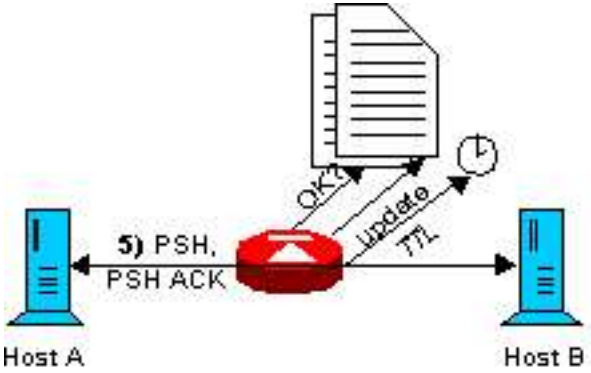


IP A, IP B, TCP	Port 21, SYN	NULL

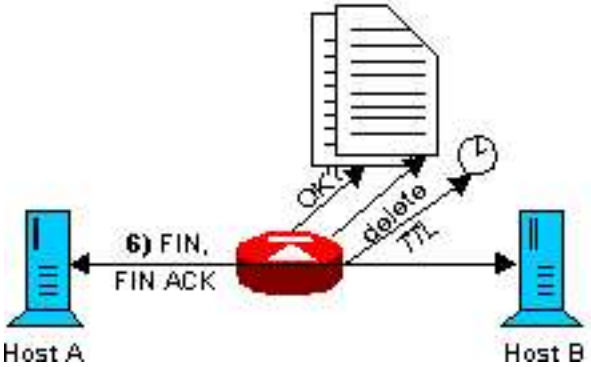


IP A, IP B, TCP	Port 21, SYN/ACK	NULL

Stateful Inspection



IP A, IP B, TCP	Port 21, PSH/ACK	user\n passwd\n RETR /testfile\n



IP A, IP B, TCP	Port 21, FIN/ACK	NULL

IPSec: Verschlüsselung und Enkapsulierung

Die IPSec-Protokolle verpacken IP-Pakete beim Sender in größere IP-Pakete (Enkapsulierung) und verschlüsseln diese dann. Beim Empfänger wird nach der Entschlüsselung das „äußere“ Paket entfernt und nur das „innere“ weitergeleitet.

Die für die Ver- und Entschlüsselung und En-/Dekapsulierung nötigen Parameter (Schlüssel, IP, etc.) werden zu Beginn ausgehandelt (IKE) und danach im als sog. „Security Associations“ (SAs) im Kernel-Memory gespeichert.

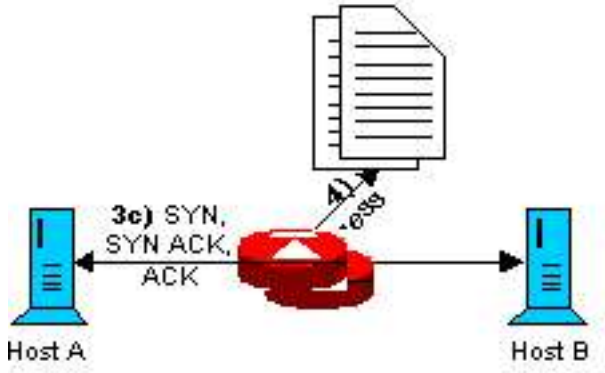
Die verschlüsselten Pakete werden über IP transportiert und bilden zusammen wieder eine -> IP Session.

IPSec selbst führt eigene Counter u.ä. und bildet eigene Sessions.

Ausfall einer Firewall

Problem: IP-Adressen

Um Ausfällen vorzubeugen, werden Firewalls als Paare (Cluster) betrieben.



IP A, IP B, TCP	Port 21, SYN/ACK	NULL

Problem: Jede Firewall hat eine IP-Adresse mit dazugehörender MAC-Adresse, an die benachbarte Systeme ihre Pakete schicken.
Wie gelangen diese Adressen auf die Backup-Firewall, wenn die aktive Firewall ausfällt?

Ausfall einer Firewall

Problem: Active-passive

Während und nach einer Failover-Situation ist es nicht immer leicht festzustellen, welche Firewall gerade aktiv ist und welche Backup. Der Fluss der Pakete und die Stateful inspection setzen aber für die meisten Protokolle voraus, dass es einen eindeutig definierten Pfad gibt und der für Hin- und Rückweg gleich ist.
Wie wird in einem Firewall-Cluster das augenblicklich aktive System bestimmt?

Ausfall einer Firewall

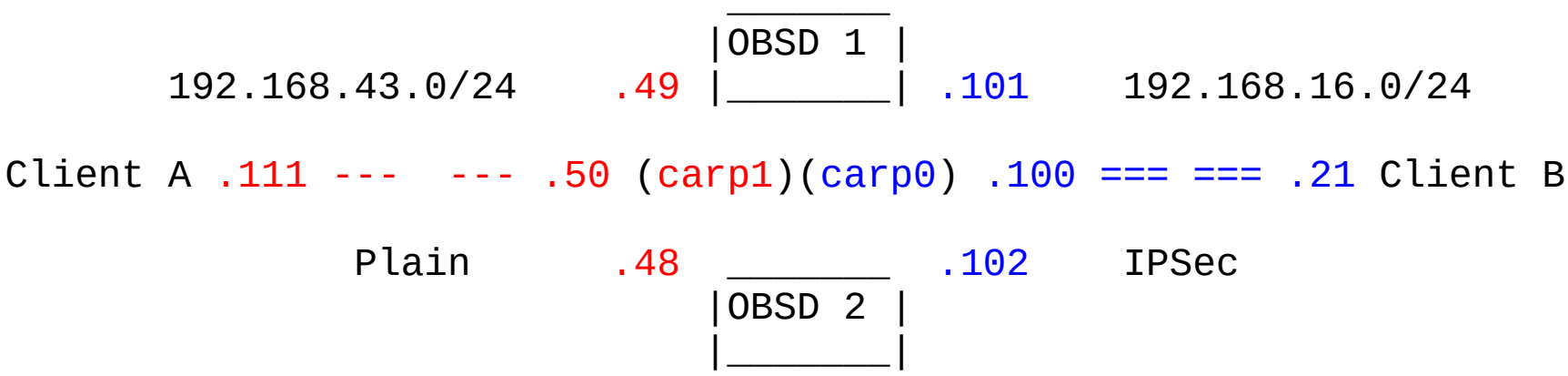
Problem: Synchronisation

State table
Wenn die Backup-Firewall ihren Betrieb aufnimmt, muss sie über alle auf der aktiven Firewall offenen Sessions Bescheid wissen. Pakete, die sie keiner offenen Session zuordnen kann, werden verworfen, geNATete Pakete nicht mehr erkannt ...
Wie kommt die Information aus der State table der aktiven auf die Backup-Firewall?

IPSec
Die zur Ver-/Entschlüsselung nötigen SAs werden nur auf der aktiven Firewall gespeichert, wo der Tunnel aufgebaut wurde. Diese führt auch die IPSec-eigenen Counter (Replay etc.)
Wie kommen die SAs von der aktiven auf die Backup-Firewall?

Netzwerk-Layout

Das Layout des Test-Netzwerks



CARP: Common Address Redundancy Protocol

Das CARP ist ein Protokoll, das zwischen mehreren Systemen in einem LAN einen „Master“ ermittelt und diesem eine virtuelle IP-Adresse zuteilt. Diese Adresse wird von benachbarten Systemen als Gateway verwendet. Fällt der Master aus, übernimmt ein Slave die IP-Adresse.

Jedes System hat eine fixe Priority, die es beim Start via Multicast an alle Systeme im LAN schickt („Advertisment“).

Empfängt ein System ein Advertisment mit einer besseren Priority, geht es in den BACKUP Status und schickt keine weiteren Adv.

Kommen in einem bestimmten Zeitraum („Hello interval“) keine neuen Master Adv. mehr herein, schickt jedes System wieder ein Adv., bis ein neuer Master gefunden ist.

CARP: Konfiguration externes Interface

Master:

```
/etc/hostname.carp0
   inet 192.168.16.100 255.255.255.0 192.168.16.255 \
   vhid 1 pass brabbel carpdev fxp0 advskew 1 \
   description external
```

Slave:

```
/etc/hostname.carp0
   inet 192.168.16.100 255.255.255.0 192.168.16.255 \
   vhid 1 pass brabbel carpdev fxp0 advskew 3 \
   description external
```

CARP: Konfiguration internes Interface

Master:

```
/etc/hostname.carp1
   inet 192.168.43.50 255.255.255.0 192.168.43.255 \
   vhid 2 pass brabbel carpdev rl0 advskew 1 \
   description internal
```

Slave:

```
/etc/hostname.carp1
   inet 192.168.43.50 255.255.255.0 192.168.43.255 \
   vhid 2 pass brabbel carpdev rl0 advskew 3 \
   description internal
```

Interface-Konfiguration

BSD: Interface-Konfiguration

```
/etc/hostname.fxp0
   inet 192.168.16.101 255.255.255.0 NONE

/etc/hostname.rl0
   inet 192.168.43.49 255.255.255.0 NONE
   inet alias 192.168.43.200 255.255.255.255

# sh /etc/netstart
```

Fragen?

```
# man hostname.if
```

pfsync: Synchronisation von Kernel State tables

pfsync stellt ein sog. „Pseudo-Device“ zur Verfügung, über das jeder Host Änderungen in seiner State table weitergibt. Es gibt keinen Master und Slave, jeder Host führt die selbe State table.

```
# pfctl -s state
all ipencap 192.168.16.100 <- 192.168.16.21       NO_TRAFFIC:SINGLE
all tcp 192.168.16.101:22 <- 192.168.16.21:47854  ESTABLISHED:ESTABLISHED
all tcp 192.168.16.102:22 <- 192.168.16.21:59771  ESTABLISHED:ESTABLISHED
all tcp 192.168.43.111:22 <- 192.168.16.21:56954  ESTABLISHED:ESTABLISHED
all tcp 192.168.43.111:22 <- 192.168.16.21:47287  ESTABLISHED:ESTABLISHED
all tcp 192.168.16.102:500 <- 192.168.16.101:6816 ESTABLISHED:ESTABLISHED
all tcp 192.168.16.102:28977 -> 192.168.16.101:500ESTABLISHED:ESTABLISHED
all tcp 192.168.16.101:500 <- 192.168.16.102:28977ESTABLISHED:ESTABLISHED
all tcp 192.168.16.101:6816 -> 192.168.16.102:500 ESTABLISHED:ESTABLISHED
all tcp 192.168.16.21:56954 -> 192.168.43.111:22  ESTABLISHED:ESTABLISHED
all tcp 192.168.16.21:47287 -> 192.168.43.111:22  ESTABLISHED:ESTABLISHED
```

Dieses Pseudo-Device kann dann direkt an ein physikalisches oder an ein weiteres Pseudo-Device (enc*N* bei IPSec) gebunden werden.

pfsync: Konfiguration des Synchronisations-Interface

Master+Slave:

```
/etc/hostname.pfsync0
   syncdev fxp0

oder mit IPSec (und entsprechender Konfiguration):

/etc/hostname.pfsync0
```

Master

```
   syncpeer 192.168.16.102 syncdev enc1
```

Slave

```
   syncpeer 192.168.16.101 syncdev enc1

# man pfsync
```

Interface-Konfiguration

```
# ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
        groups: lo
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x6
fxp0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        lladdr 00:60:94:6b:45:a9
        groups: egress
        media: Ethernet autoselect (100baseTX full-duplex)
        status: active
        inet 192.168.16.101 netmask 0xffffff00 broadcast 192.168.16.255
        inet6 fe80::260:94ff:fe6b:45a9%fxp0 prefixlen 64 scopeid 0x1
rl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        lladdr 00:c0:26:26:90:33
        media: Ethernet autoselect (10baseT)
        status: active
        inet 192.168.43.49 netmask 0xffffff00 broadcast 192.168.43.255
        inet6 fe80::2c0:26ff:fe26:9033%rl0 prefixlen 64 scopeid 0x2
pflog0: flags=141<UP,RUNNING,PROMISC> mtu 33224
pfsync0: flags=0<> mtu 1348
        pfsync: syncdev: fxp0 syncpeer: 224.0.0.240 maxupd: 128
enc0: flags=0<> mtu 1536
carp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        description: external
        carp: MASTER carpdev fxp0 vhid 1 advbase 1 advskew 1
        groups: carp
        inet 192.168.16.100 netmask 0xffffff00 broadcast 192.168.16.255
carp1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        description: internal
        carp: MASTER carpdev rl0 vhid 2 advbase 1 advskew 1
        groups: carp
        inet 192.168.43.50 netmask 0xffffff00 broadcast 192.168.43.255
```

pf: *der* Paketfilter

pf, der BSD packet filter, ist eine wahre Fundgrube an Optionen.
Alles in allem eine der umfangreichsten Firewalls auf dem „Markt“.
Besonderheit: „Last match wins“, keine Chains/Policies, stateful

```
/etc/pf.conf
    ext_if="fxp0"
    int_if="rl0"
    carp0_if="192.168.16.100"
    scrub in
    nat on $ext_if from 192.168.43.111 -> $carp0_if
    block in log
    pass out keep state
    pass quick on { lo pfsync0 }
    pass quick on { $ext_if $int_if } proto carp
    pass quick on { $ext_if } proto pfsync
    pass in on $ext_if proto udp to (carp0) port 500 keep state
    pass in on $ext_if proto tcp to ($ext_if) port 500 keep state
    pass in on $ext_if proto esp to (carp0) keep state
    pass in on enc0 proto ipencap from ($ext_if:network) to $carp0_if keep state
    pass in on enc0 from ($ext_if:network) to 192.168.43.111 keep state
    pass in on $int_if from 192.168.43.111 to any keep state
    pass in on $ext_if proto tcp to ($ext_if) port ssh keep state
```

sasyncd: Synchronisation von IPSec SAs

sasyncd funktioniert ähnlich wie pfsync und verwendet das selbe Pseudo-Device (pfsync*N*). Er kennt jedoch einen Master -und Slave-Status, den er vom jeweiligen CARP-Device übernimmt.

Der Master wartet auf Anfragen der Slaves und gibt einen Snapshot der Kernel IPSec SAs weiter.

Die Kommunikation von `sasyncd` ist immer AES-verschlüsselt (tcp/500). Keys werden von Hand auf beiden Nodes installiert.

```
# openssl rand 32 | perl -pe 's/./unpack("H1",$&)/ges'
```

erzeugt einen 256bit AES Key.

Die `sasyncd` im Cluster kommunizieren ständig miteinander. Der Master-Slave-Status ist also jederzeit bekannt.

Known bugs: Resynchronisation des Master nach Reboot.

sasyncd: Konfiguration

`/etc/sasyncd.conf`

Master:

```
carp interface carp0 interval 1
sharedkey fb76feb53a48d8dbce6a48ad836e748e
peer 192.168.16.102
```

Slave:

```
carp interface carp0 interval 1
sharedkey fb76feb53a48d8dbce6a48ad836e748e
peer 192.168.16.101
```

isakmpd: Key exchange und flow setup

```
/etc/isakpmd/isakmpd.conf
/etc/isakpmd/isakmpd.policy

# man vpn
# man ipsec
# man isakmpd{|.conf|.policy}
```

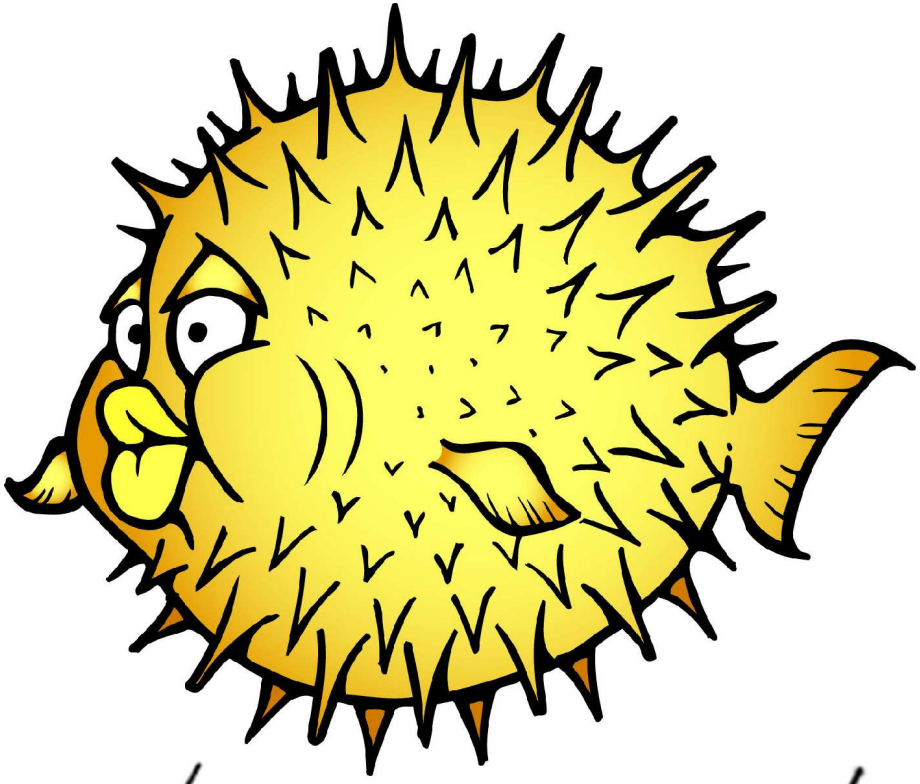
oder

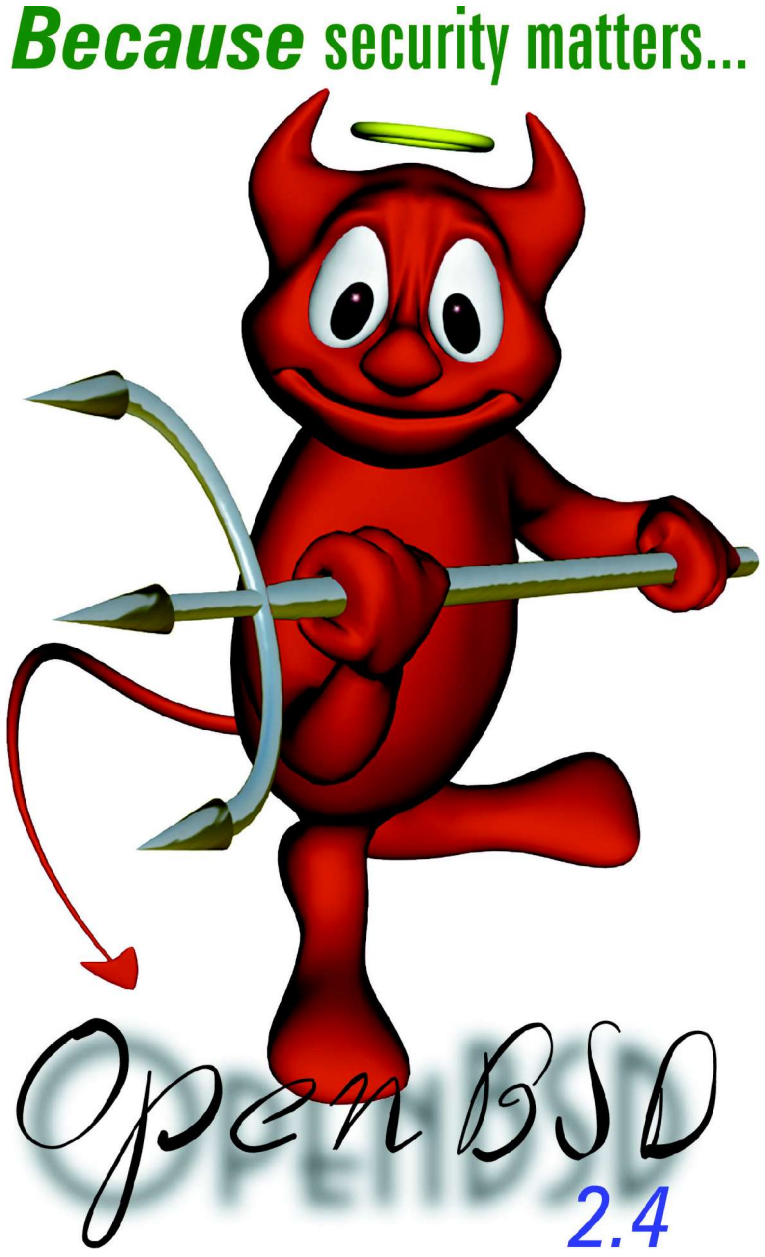
```
# man ipsecadm
# man ipsecctl
```

Ressourcen

Fundgruben:

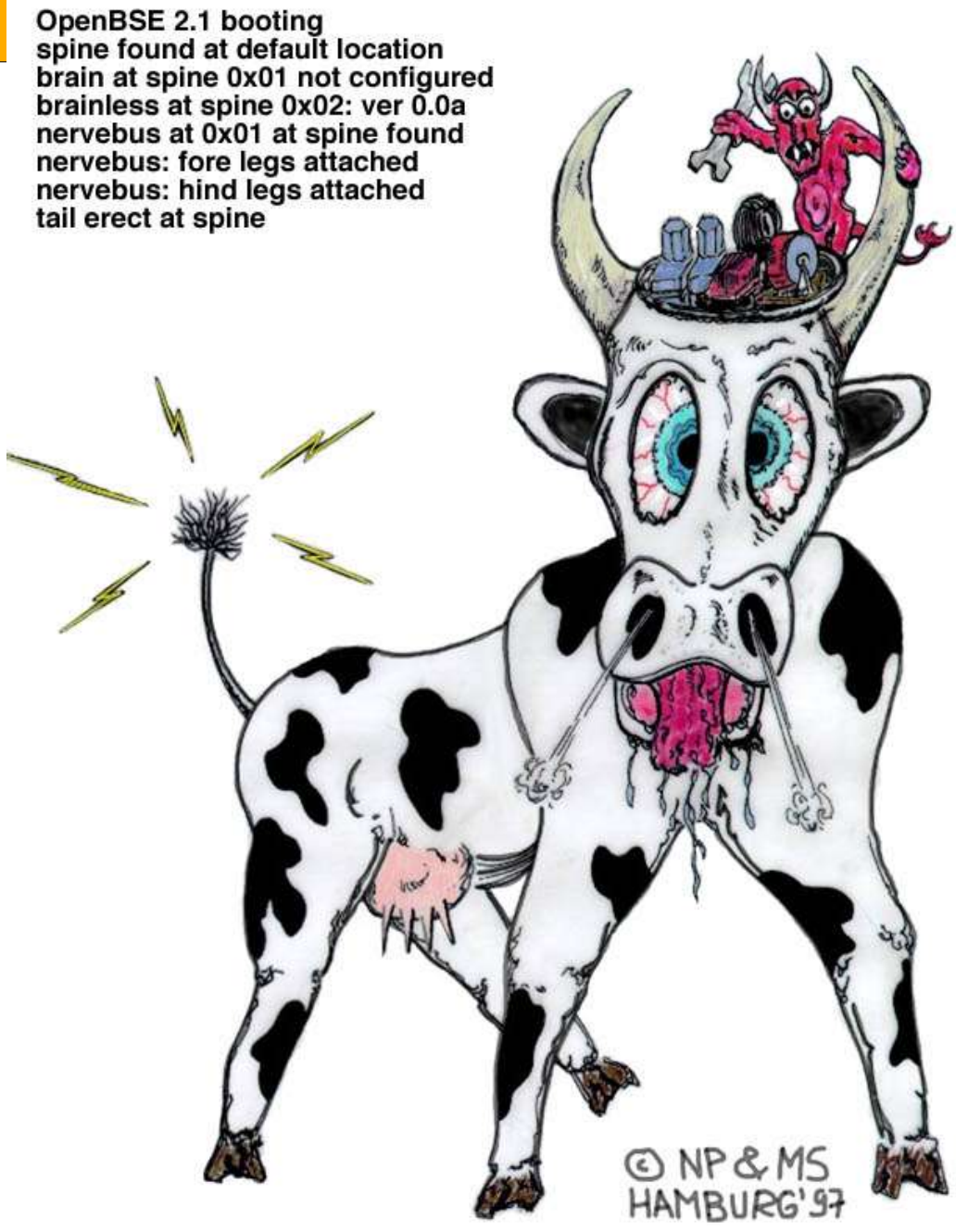
http://www.openbsd.org
misc@openbsd.org
http://marc.theaimsgroup.com/?l=openbsd-misc


Strong crypto


Because security matters...
OpenBSD
2.4

LugBE

Linux User Group Bern



... and, yes:
Security can be funny.

So long