

OPENBSD

STEPHAN ECKNER

INHALTSVERZEICHNIS

1. Disclaimer	2
2. Grundkonfiguration	3
2.1. Installation von CDROM	3
2.2. Packaging Tools	3
2.3. CVS	3
2.4. Kernelparameter und -kompilierung	3
2.5. Kompilierung von Systemkomponenten	4
2.6. Konfiguration — Wo steht was	4
2.7. Networking	4
2.8. Systemstart und -stop	6
3. Firewalling	7
3.1. Kernelparameter	7
3.2. Systemkonfiguration	7
3.3. ipf	7
3.4. ipmon	9
3.5. ipnat	9
3.6. ipf und Bridging	9
4. VPN	11
4.1. IPSec	11
4.2. Kernelparameter	11
4.3. Systemkonfiguration	11
4.4. isakmpd	12
4.5. VPN durch Firewalls	12

1. DISCLAIMER

Dieser Text soll dokumentieren, wofür ich in bisherigen Kundenprojekten OpenBSD eingesetzt habe und wie ich die Systeme konfiguriert habe. Ich habe nicht die Absicht, die vorhandene sehr gute und ausführliche Dokumentation für OpenBSD noch einmal neu zu schreiben. Vielmehr soll aus dem Text hervorgehen, was ich in den oben genannten Projekten umgesetzt habe und was nicht, so dass der Leser wisse, was er mich fragen kann und was nicht :). Ausserdem werde ich an den entsprechenden Stellen versuchen, möglichst viele Literaturhinweise zu geben auf die Dokumente, die mir am besten weitergeholfen haben.

2. GRUNDKONFIGURATION

2.1. Installation von CDROM. TBD

2.2. Packaging Tools. TBD

2.3. **CVS.** Siehe auch <http://www.openbsd.org/anoncv.html> und die **cv(1)** Manual Page. Folgendermaßen macht man einen CVS Checkout des gesamten Systems (in diesem Fall Version 2.7) von unserem lokalen CVS-Server:

```
# export CVS_RSH="/usr/bin/ssh"
# export CVSROOT="<username>@cvs:/cvs/OpenBSD"
# cd /usr
# cvs checkout -P -rOPENBSD_2_7 src
```

Falls man nur die Kernelquellen benötigt, gibt man stattdessen folgendes ein:

```
# cvs checkout -P -rOPENBSD_2_7 src/sys
```

Allerdings sollte man nicht davon ausgehen, dass mit einem aktualisierten Kernel noch alles so funktioniert wie vorher. Bestimmte Systemkomponenten, die besonders eng mit dem Kernel zusammenarbeiten, sollten im Zweifel gleich mitkompiliert werden (das betrifft zum Beispiel den `isakmpd`, die OpenBSD eigene IKE Implementierung (s.a. Abschnitt 4.4)).

2.4. **Kernelparameter und –kompilierung.** Siehe [Kapitel 5](#) des OpenBSD FAQ.

Die Kernelparameter sind bei OpenBSD auf zwei Dateien verteilt: Die architekturunabhängigen Parameter befinden sich in dem Verzeichnis `/usr/src/sys/conf`, die architekturabhängigen in `/usr/src/sys/arch/i386/conf/` (für andere Architekturen entsprechend). In beiden Verzeichnissen befinden sich bereits diverse Konfigurationen, von denen sich `GENERIC` als Ausgangsbasis fuer eine eigene Konfiguration eignen. Dabei geht man folgendermassen vor: Zuerst ändert man den Namen der beiden Konfigurationsdateien in z.B. `CUSTOM`:

```
# cd /usr/src/sys/conf
# cp GENERIC CUSTOM
# cd ../arch/i386/conf/
# cp GENERIC CUSTOM
```

Nun muss man noch am Anfang der Datei `/usr/src/sys/arch/i386/config/CUSTOM` das `include` Statement anpassen:

```
include "../../../conf/CUSTOM"
```

2.4.1. *Kernelparameter.* Siehe [Kapitel 5.2](#) des OpenBSD FAQ und die [options\(4\)](#) Manual Page.

Folgende Parameter müssen gesetzt werden, um das Kernel Debugging zu ermöglichen (aktiviert wird es dann durch den Eintrag `debug.on.panic=0` in `/etc/sysctl.conf`, s. Abschnitt 2.6.2):

```
option          DBB          # in kernel debugger
makeoptions     DEBUG="-g"    # compile full symbol table
option          DIAGNOSTIC   # internal consistency checks
```

Auf die für Firewalls bzw VPN's notwendigen Kernelparameter wird in den Abschnitten 3.1 und 4.2 eingegangen.

2.4.2. *Kompilierung*. Siehe [Kapitel 5.3](#) des OpenBSD FAQ.

Der OpenBSD Kernel (in diesem Fall mit der Konfiguration namens `CUSTOM`) kompiliert sich folgendermaßen:

```
# cd /usr/src/sys/arch/i386/conf/
# config CUSTOM
# cd ../compile/CUSTOM
# make depend
# make
```

Um den frischgebackenen Kernel zu installieren, muss man ihn einfach nur nach `/` kopieren, ein expliziter Aufruf eines Boot-Loaders wie bei Linux ist nicht erforderlich:

```
# cp /bsd /bsd.old
# mv bsd /
```

2.5. **Kompilierung von Systemkomponenten**. Im Verzeichnis `/usr/src` befindet sich eine Verzeichnisstruktur, in der zu jedem auf dem System vorhandenen Binary ein entsprechendes Unterverzeichnis existiert, in welchem sich der Quellcode des Binaries befindet. So findet man z.B. den Quellcode des Programmes `/sbin/isakmpd` in dem Verzeichnis `/usr/src/sbin/isakmpd/`. Die Kompilierung des Programmes erfolgt in folgenden Schritten:

```
# cd /usr/src/sbin/isakmpd/
# make obj
# make
# make install
```

2.6. **Konfiguration — Wo steht was**. Siehe auch die [afterboot\(8\)](#) Manual Page.

Auf die für Firewalls und VPN's notwendigen Parameter wird in den Kapiteln [3.2](#) und [4.3](#) eingegangen.

2.6.1. */etc/rc.conf*. In der Datei `/etc/rc.conf` wird festgelegt, welche Dienste beim Booten mit welchen Parametern gestartet werden. Die Einträge sind in der Regel von der Form `<name>_flags="<value>"`, wobei `<name>` der Name des Dienstes ist und `<value>` entweder `YES`, `NO`, oder die Parameter, mit denen der Dienst gestartet wird. Beispiel:

```
sendmail_flags="-q30m"
named_flags=NO
sshd=YES
```

2.6.2. */etc/sysctl.conf*. In der Datei `/etc/sysctl.conf` werden Einstellungen wie IP-Forwarding oder Swap-Encryption vorgenommen. Die Einträge sind standardmässig durch ein `#` auskommentiert. Um Änderungen der `sysctl.conf` wirksam werden zu lassen, ohne das System neu zu starten, benutzt man direkt den Befehl [sysctl\(8\)](#), also z.B.:

```
sysctl -w net.inet.ip.forwarding=1
TBD: sysctl -w dbb.panic=0
```

2.7. **Networking**. Siehe [Kapitel 6.1](#) des OpenBSD FAQ.

Netzwerk Interfaces unter OpenBSD tragen treiberspezifische Namen, also z.B. `x10` für die erste 3Com 3C9xx Karte, `x11` für die zweite, `ne0` für die erste ne2000 Karte usw. Die Charakteristika der Treiber sind in den entsprechenden Manal Pages beschrieben, z.B. [xl\(4\)](#). Die **Boot-Konfiguration** der Interfaces wird in den Dateien `/etc/hostname.if` eingetragen:

```
# cat /etc/hostname.xl0
inet 10.0.0.23 255.255.255.0 NONE
```

Die Broadcast Adresse wird dabei aus der IP-Adresse und der Netzmaske automatisch berechnet. Durch Einfügen eines Ausrufezeichens zu Beginn einer Zeile können Shell-Kommandos aufgerufen werden. So können z.B. **statische Routen** konfiguriert werden:

```
# cat /etc/hostname.xl1
inet 10.0.0.24 255.255.255.0 NONE
!route add -net 10.0.1.0 -netmask 255.255.255.0 10.0.0.1
!route add -net 10.0.2.0 -netmask 255.255.255.0 10.0.0.1
```

Aliases für die Interfaces werden ab Version 2.7 ebenfalls in der Datei `/etc/hostname.*` eingetragen:

```
# cat /etc/hostname.xl2
inet 10.0.0.25 255.255.255.0 NONE
inet alias 192.168.0.1 255.255.255.0 NONE
```

Das **Default Gateway** wird in der Datei `/etc/mygate` eingetragen:

```
# cat /etc/mygate
10.0.0.254
```

Der **Hostname** wird in der Datei `/etc/myname` eingetragen:

```
# cat /etc/myname
foobar
```

Nameserver werden (wie bei Linux auch) in die Datei `/etc/resolv.conf` geschrieben:

```
# cat /etc/resolv.conf
search yourdomain.com
nameserver 10.0.0.1
nameserver 10.0.0.254
lookup file bind
```

Änderungen in diesen Konfigurationsdateien werden durch Aufruf des Skriptes **netstart** wirksam:

```
# sh /etc/netstart
```

Die Syntax der Befehle **ifconfig**, **route** und **netstat** unterscheidet sich geringfügig von der bei Linux gebräuchlichen:

Linux:

```
ifconfig
ifconfig eth0 1.0.0.2 netmask 255.0.0.0 up
route -n
route add default gw 1.0.0.1
route add -net 2.0.0.0 netmask 255.0.0.0 gw 1.0.0.1
netstat -rn
```

OpenBSD:

```
ifconfig -a
ifconfig xl0 1.0.0.2 netmask 255.0.0.0 up
route -n show
route add default 1.0.0.1
route add -net 2.0.0.0 -netmask 255.0.0.0 1.0.0.1
netstat -rn
```

Zur schnellen Überprüfung bei Routing-Problemen eignet sich folgender Befehl:

```
# route get <Zieladresse>
```

2.8. Systemstart und -stop. Systemstart und -stop unter OpenBSD wird von dem Shellskript `/etc/rc` gesteuert (s. auch die Manual Page [rc\(8\)](#)). Aus diesem Skript heraus werden die Shellskripte `/etc/rc.securelevel`, `/etc/netstart` und `/etc/rc.shutdown` aufgerufen. Das Verhalten des Skriptes kann durch die in der Datei `/etc/rc.conf` und `/etc/rc.local` gesetzten Variablen beeinflusst werden.

TBD: was zu den Securelevels schreiben.

3. FIREWALLING

3.1. **Kernelparameter.** Folgende Kernelparameter müssen für eine Firewall gesetzt werden (Siehe auch [Kapitel 5.2](#) des OpenBSD FAQ und die [options\(4\)](#) Manual Page):

```
option          GATEWAY
option          NMBCLUSTERS=8192
option          NKMEMCLUSTERS=8192
option          MAX_KMAP=120
option          MAX_KMAPENT=6000
option          INET
option          INET6

option          IPFILTER
option          IPFILTER_LOG
option          IPFILTER_DEFAULT_BLOCK

pseudo-device  loop 2
pseudo-device  bpfiler 8
```

3.2. Systemkonfiguration.

3.2.1. *rc.conf*. Folgende Parameter müssen für eine Firewall gesetzt werden (siehe auch die Manual Page [rc.conf\(8\)](#)):

```
ipfilter=YES
ipnat=YES
ipfilter_rules=/etc/ipf.rules
ipnat_rules=/etc/ipnat.rules
ipmon_flags="-Ds -o I"
syslogd_flags=
```

3.2.2. *sysctl.conf*. Folgende Parameter müssen für eine Firewall gesetzt werden (siehe auch die Manual Page [sysctl.conf\(8\)](#)):

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

Weitere Einstellungsmöglichkeiten findet man im [ipf HOWTO](#).

Falls die Firewall kein Packet-Forwarding sondern Bridging machen soll, braucht sie natürlich kein forwarding und obige Option kann auskommentiert bleiben. Siehe dazu Abschnitt [3.6](#).

3.3. **ipf.** Das OpenBSD [ipf HOWTO](#) ist ein hervorragend geschriebene Einführung in die **ipf**-Syntax und in Firewalls im Allgemeinen.

3.3.1. *Konfigurationsdateien und Kommandozeilenparameter.* Das Kommando zum managen der Firewall unter OpenBSD lautet **ipf**. Die Firewallregeln befinden sich in der Datei `/etc/ipf.rules` ¹. Änderungen der Firewallregeln werden durch den Befehl

```
# ipf -Fa -f /etc/ipf.rules -E
```

wirksam. Durch den Befehl **ipf -Fa** können die Firewallregeln gelöscht werden. Ist das geschehen, so greift die durch die Kerneloption `IPFILTER_DEFAULT_BLOCK` definierte Default Policy. Ist diese gesetzt, so muss die Firewall explizit durch den Aufruf von

¹Falls nicht in `/etc/rc.config` die Variable `ipfilter_rules` auf einen anderen Wert gesetzt wird

```
# ipf pass in from any to any
# ipf pass out from any to any
```

freigeschaltet werden². Eine Liste der Firewallregeln kann man sich durch den Befehl

```
# ipfstat -hion
```

anzeigen lassen.

3.3.2. *Syntax*. Die Syntax von **ipf** ist in der Manual Page **ipf(5)** beschrieben³. **ipf** ist ein dynamischer Paketfilter. Im Gegensatz zu **ipchains**, bei dem je eine Regel für den Verbindungsaufbau und eine für die zurückkommende Antwort generiert werden muss, muss bei **ipf** nur die Regel für den Verbindungsaufbau definiert werden. Alle weiteren Pakete, die zu der aufgebauten Verbindung gehören⁴, werden dann automatisch durchgelassen. In folgendem Beispiel soll die Firewall das interne Netz 10.0.0.0/8 auf ihre externe Adresse 1.2.3.4 maskieren (siehe dazu auch 3.5) und Verbindungen auf Port 80 vom internen Netz nach aussen zulassen:

```
Internal NW 10.0.0.0/8 - 10.0.0.1 FW 1.2.3.4 - WWW
# ipchains -A input -i eth1 -p tcp -s 10.0.0.0/8 1024:65535
-d any/0 80 -j ACCEPT
# ipchains -A output -i eth0 -p tcp -s 1.2.3.4 1024:65535
-d any/0 80 -j ACCEPT
# ipchains -A input -i eth0 -p tcp ! -y -s any/0 80
-d 1.2.3.4 1024:65535 -j ACCEPT
# ipchains -A output -i eth1 -p tcp ! -y -s any/0 80
-d 10.0.0.0/8 1024:65535 -j ACCEPT
# ipchains -A forward -i eth0 -s 10.0.0.0/8 -j MASK
```

entspricht:

```
# ipf pass in quick on x11 proto tcp from 10.0.0.0/8 port > 1024
to any port 80 flags S keep state
# ipf pass out quick on x10 proto tcp from 1.2.3.4 port > 1024
to any port 80 flags S keep state
# ipnat map x10 10.0.0.0/8 -> 1.2.3.4/32 portmap tcp 1025:65000
```

Obwohl sich im Falle von UDP oder ICMP nicht wirklich von einem Verbindungszustand reden lässt, ist die Syntax von **ipf** wie im Fall von TCP⁵ (siehe auch die Abschnitte über **UDP** und **ICMP** im **ipf HOWTO**. ICMP Pakete von innen nach außen werden folgendermaßen ermöglicht:

```
# ipf pass in quick on x11 proto icmp from 10.0.0.0/8 to any keep
state
```

²Es bietet sich an, diese beiden Zeilen in z.B. die Datei `/root/fw_off` zu schreiben, so dass die Firewall im Zweifel mittels `# ipf -Fa -f /root/fw_off` freigeschaltet werden kann.

³Ein Muss für alle Freunde der Backus-Nauer-Form.

⁴Genau genommen natürlich nur die Pakete, die den Anschein haben, zu der Verbindung zu gehören. Bei TCP Verbindungen also die mit der richtigen IP Adresse, dem richtigen TCP Port, den richtigen TCP-Flags und der richtigen Sequenznummer. Bei UDP und ICMP Paketen, bei denen es keinen durch Flags definierten und durch Sequenznummern gesicherten Verbindungszustand gibt, werden alle Pakete mit der richtigen IP Adresse und Port Nummer durchgelassen, sofern sie innerhalb eines Timeouts (TBD: wieviel, wie kann man den setzen) eintreffen.

⁵Die Syntax von **iptables** unterscheidet hingegen zwischen `ESTABLISHED` und `RELATED`

```
# ipf pass out quick on xl0 proto icmp from 1.2.3.4 to any keep
state
```

Im Gegensatz zu **ipchains** können in der Konfigurationsdatei von **ipf** *keine* Variablen benutzt werden. In dem Verzeichnis `/usr/share/ipf/` befinden sich in den Dateien TBD Beispielkonfigurationen für `/etc/ipf.rules` und `/etc/ipnat.rules`.

3.3.3. *must-have ipf-Regeln.* Folgende Regeln filtern Pakete mit gesetzten IP Optionen (z.B. Source Routing) und stark fragmentierte Pakete. Sie sollten am Anfang jeder Filterliste stehen:

```
# ipf block quick in log all with ipopts
# ipf block quick in log all with short
```

3.4. **ipmon.** Siehe auch die Manual Page [ipmon\(8\)](#) und den entsprechenden [Abschnitt über ipmon](#) im **ipf-HOWTO**.

3.5. **ipnat.** Siehe auch die Manual Pages [ipnat\(8\)](#) und [ipnat\(5\)](#) sowie den [Abschnitt über ipnat](#) im **ipf-HOWTO**. Die Regeln für Network Address Translation, Masquerading und Port-Forwarding befinden sich in der Datei `/etc/ipnat.rules`⁶. Änderungen der NAT-Regeln werden durch den Befehl

```
# ipnat -CF -f /etc/ipnat.rules
```

wirksam gemacht.

In dem Verzeichnis `/usr/share/ipf/` befinden sich in den Dateien `nat.1` und `nat.2` Beispielkonfigurationen für `/etc/ipnat.rules`.

3.6. **ipf und Bridging.**

3.6.1. *Vorüberlegungen.* Siehe auch den entsprechenden Abschnitt des [ipf-HOWTO](#).

Es gibt prinzipiell zwei verschiedene Situationen, in denen Firewalls zum Einsatz kommen:

- (1) Der Schutz eines internen Netzes, also von Rechnern, die aus dem Internet nicht erreichbar sein müssen/sollen,
- (2) Der Schutz von Servern, die aus dem Internet erreichbar sein müssen.

Im ersten Fall ist es extrem sinnvoll, dem internen Netz private IP Adressen zu geben, da diese im Internet nicht geroutet werden, und damit die Rechner nicht von einer anderen Stelle des Internet aus erreichbar sind. Aus dem selben Grund muss bei einem Verbindungsaufbau eines internen Rechners zu einem externen Server die IP Source Adresse auf die (offizielle) IP Adresse des externen Interfaces des Gateways umgesetzt werden. Wird auf einem solchen Gateway eine Firewall aufgesetzt, so muss sie also NAT⁷ bzw. Masquerading⁸ durchführen. Soll die Firewall jedoch Server schützen, die aus dem Internet erreichbar sein sollen, so gibt es keinen Grund dafür, sie NAT machen zu lassen⁹. Es ist vielmehr völlig ausreichend, wenn die Firewall den ein- und ausgehenden IP Verkehr filtert, die IP Source und Destination Adressen braucht sie dabei nicht verändern. Dann braucht sie aber auch keine eigene IP Adresse, was den weiteren Vorteil mit sich bringt, dass die Firewall nicht mehr direkt

⁶Falls nicht in `/etc/rc.config` die Variable `ipnat_rules` auf einen anderen Wert gesetzt wird

⁷Network Address Translation

⁸Spezialfall des NAT, bei dem alle internen Adressen auf genau eine externe Adresse umgesetzt werden

⁹Es sei denn, man möchte unterschiedliche Services auf physikalisch unterschiedlichen Maschinen unter genau einer externen Adresse via Portforwarding erreichbar machen.

angegriffen werden kann¹⁰!

Mit anderen Worten: Eine Bridging-Firewall.

Typischerweise tritt sowohl der erste als auch der zweite Fall gleichzeitig auf: wenn z.B. ein internes Netz ans Internet angeschlossen werden, und der Mailserver von draussen erreichbar sein soll.

Sowohl mit OpenBSD als auch mit Linux ist es möglich, zwei oder mehr Netzwerkkarten zu einer Bridge zusammenzuschalten. Im Gegensatz zu OpenBSD funktioniert jedoch *unter Linux dann das Firewalling nicht mehr*.

3.6.2. *Bridgingspezifische Konfiguration und Komandozeilenparameter.* Folgender Kernelparameter muss zusätzlich für Bridging Support gesetzt werden (Siehe auch die [options\(4\) Manual Page](#)):

```
pseudo-device    bridge 2
```

In `rc.conf` müssen zu denen in Abschnitt 3.2.1 erwähnten *keine* weiteren Parameter gesetzt werden.

In `sysctl.conf` muss die Option `net.inet.ip.forwarding=1` auskommentiert bleiben.

Welche Interfaces zu einer Bridge verbunden werden, wird in der Datei `bridgename.if` festgelegt (siehe auch die Manual Page [bridgename.if](#)).

```
# cat /etc/bridbename.bridge0
add x10
add x11
up
```

Gleichzeitig muss in den Dateien `/etc/hostname.x10` und `/etc/hostname.x11` das Wort `up` eingetragen werden:

```
# cat /etc/hostname.x10
up
# cat /etc/hostname.x11
up
```

Laut dem [ipf-HOWTO](#) sind im Bridging Modus die `in` und `out` Keywords nicht unterstützt. Ich habe noch nicht nachgeprüft, ob sich das mit dem Release von OpenBSD 2.8 inzwischen geändert hat ...

3.6.3. *Remote-Administration von gebridgeten Firewalls.* Ein Rechner ohne IP Adresse lässt sich schlecht remote administrieren. Möchte man dennoch eine Remote Administration, so gibt man dem entsprechenden (gebridgeten) Interface einfach zusätzlich eine IP-Adresse. Auf dieser IP-Adresse kann man die Firewall dann problemlos ansprechen. (Am besten nimmt man natürlich das interne Interface)

¹⁰Angenommen, ein Angreifer versucht einen Buffer Overflow im IP Stack aus, um von der Firewall aus eine Telnet Verbindung zu seinem auf einem hohen Port laufendem Telnet-Server aufzubauen, so ist der Aufbau der Verbindung schon deshalb unmöglich, weil die Pakete keine IP Source Adresse hätten.

4. VPN

4.1. IPSec.

4.1.1. *Allgemeines.* Eine sehr ausführliche Beschreibung von IPSec unter OpenBSD findet man in [Kapitel 13](#) des OpenBSD FAQ's. Die Abschnitte über die manuelle Konfiguration von IPSec Verbindungen (13.5 und 13.6) oder die Konfiguration von `photurisd` (13.7) kann man dabei getrost überspringen, wenn man vorhat, `isakmpd` zu benutzen. Sehr zu empfehlen ist auch der Artikel [ISAKMP and IPSec in the VPN environment](#) von Patrick Ethier.

IPSec ist der von der [IETF](#) entwickelte Standard¹¹ für die Verschlüsselung von Datenverkehr auf IP-Ebene.

4.1.2. *SA und SPD.*

4.1.3. *Tunnel-Mode versus Transport-Mode.*

4.1.4. *AH und ESP.*

4.1.5. *ISAKMP.*

4.2. **Kernelparameter.** Folgende Kernelparameter müssen für ein VPN-Rechner gesetzt werden (Siehe auch [Kapitel 5.2](#) des OpenBSD FAQ und die [options\(4\)](#) Manual Page):

```
option          CRYPTO
option          UVM_SWAP_ENCRYPT
option          IPSEC
option          INET
pseudo-device   loop 2
pseudo-device   bpfiler 8
pseudo-device   enc 4
```

4.3. Systemkonfiguration.

4.3.1. *rc.conf.* Folgende Parameter müssen für ein VPN mit `isakmpd` gesetzt werden (siehe auch die Manual Page [rc.conf\(8\)](#)):

```
isakmpd_flags=""
```

¹¹Im einzelnen handelt es sich dabei um die folgenden RFC's:

[RFC 2401](#) Security Architecture for the Internet Protocol
[RFC 2402](#) IP Authentication Header (AH)
[RFC 2403](#) The Use of HMAC-MD5-96 within ESP and AH
[RFC 2404](#) The Use of HMAC-SHA-1-96 within ESP and AH
[RFC 2405](#) The ESP DES-CBC Cipher Algorithm
[RFC 2406](#) IP Encapsulating Security Payload (ESP)
[RFC 2407](#) The Internet Security Domain of Interpretation for ISAKMP
[RFC 2408](#) Internet Security Association and Key Management Protokoll (ISAKMP)
[RFC 2409](#) The Internet Key Exchange (IKE)
[RFC 2410](#) The NULL Encryption Algorithm and Its Use with IPsec
[RFC 2411](#) IP Security Document Roadmap
[RFX 2412](#) The OAKLEY Key Determination Protocol

4.3.2. *sysctl.conf*. Folgende Parameter müssen für ein VPN gesetzt werden (siehe auch die Manual Page [sysctl.conf\(8\)](#)):

```
net.inet.ip.forwarding=1
net.inet.esp.enable=1
net.inet.ah.enable=1
vm.swapencrypt=1
net.inet.ip.ipsec-acl=1
```

4.4. **isakmpd**. Unter OpenBSD existieren zur Zeit zwei Implementierungen von ISAKMP: *isakmpd* und *photurisd*. Zur Zeit ist der *isakmpd* der Ausgereiftere von beiden. Daher werde ich ab jetzt nur noch über ihn reden.

Eine Beschreibung von *isakmpd* und seiner Konfigurationsdateien findet man in [Abschnitt 13.8](#) des OpenBSD FAQ's. Besser gelungen ist allerdings der Artikel [How to set up a basic VPN between two OpenBSD gateways using ISAKMPD](#) von Patrick Ethier. Bei der Konfiguration meines ersten OpenBSD basierten VPN habe ich mich fast ausschliesslich an diesem Paper orientiert.

4.4.1. *Konfigurationsdateien und Komandozeilenparameter*. OpenBSD bringt mehrere Beispielkonfigurationen mit, die in das Verzeichnis `/etc/isakmpd/` kopiert werden müssen:

```
# cp /usr/share/ipsec/isakmpd/policy /etc/isakmpd/isakmpd.policy
# cp /usr/share/ipsec/isakmpd/VPN-east.conf /etc/isakmpd/isakmpd.conf
# chmod 600 isakmpd.policy
# chmod 600 isakmpd.conf
```

In der Datei *isakmpd.policy* müssen daraufhin die folgenden beiden Zeilen gelöscht werden:

```
$OpenBSD: policy,v 1.3 1999/11/18 21:45:52 angelos Exp $
$EOM: policy,v 1.3 1999/08/26 11:51:37 niklas Exp $
```

4.5. VPN durch Firewalls.

4.5.1. Topologie.

4.5.2. *Firewallregeln*. Folgende Firewallregeln erlauben **isakmpd** die Aushandlung der Verbindungsparameter durch die Firewall:

```
#pass in quick on xl2 proto udp from 62.159.77.123 port = 500 to
194.140.99.144 port = 500 keep state
# pass out quick on xl0 proto udp from 62.159.77.123 port = 500
to 194.140.99.144 port = 500 keep state
# pass in quick on xl0 proto udp from 194.140.99.144 port = 500
to 62.159.77.123 port = 500 keep state
# pass out quick on xl2 proto udp from 194.140.99.144 port = 500
to 62.159.77.123 port = 500 keep state
```

Folgende Firewallregeln erlauben eine esp-verschlüsselte Verbindung durch die Firewall:

```
# pass in quick on xl2 proto esp from 62.159.77.123 to 194.140.99.144
# pass out quick on xl0 proto esp from 62.159.77.123 to 194.140.99.144
# pass in quick on xl0 proto esp from 194.140.99.144 to 62.159.77.123
# pass out quick on xl2 proto esp from 194.140.99.144 to 62.159.77.123
```

E-mail address: `stephan@eckner.org`