IPsec mit OpenBSD und X.509-Zertifikaten als sichere Alternative zu WEP

Waldemar Brodkorb Tim Kornau

26-12-2002

Inhaltsverzeichnis

1 Einleitung

IPsec ermöglicht eine verschlüsselte und authentifizierte Verbindung zwischen verschiedenen Computer-Systemen die untereinander via TCP/IP kommunizieren. Im folgenden werden wir detailliert beschreiben, wie dieses Protokoll genutzt werden kann, um einen sicheren, ausgereiften Ersatz zu WEP - Wired Equivalent Privacy für Wireless Networks zu bieten.

Im Vorfeld benötigen wir eine CA - Certification Authority, die die Vertrauenswürdigkeit der verwendeten X.509-Zertifikate sichern wird. Danach beschreiben wir die Konfiguration des OpenBSD [1] IPsec Gateways und der Clientsysteme unter diversen Betriebssystemen.

Die Dokumentation ist sehr praxisorientiert gehalten und verweist auf viele weiterführende Dokumente, die die Theorie erklären oder als Befehlsreferenz dienen.

Aktuelle Versionen dieses Dokumentes werden unter folgender Seite veröffentlicht:
http://www.openbsd.de
Die meisten Konfigurationsdateien sind über folgende URL herunterladbar:
http://www.openbsd.de/ipsec/
Die Verwendung dieser Dateien als Vorlage ist empfehlenswert, weil schwer nachvollziehbare Fehler beim Rauskopieren aus diesem Dokument, vermieden werden. Hier ist auch die Historie des Dokumentes auffindbar, weil wir Korrekturen und Erweiterungen regelmäßig einpflegen.

2 Danksagung

Herzlichen Dank an Philipp Buehler für seine konstruktive Kritik und seine helfenden Hinweise bei der Verbesserung dieses Dokumentes. Dank an Nils Beck und Thomas Küpper für die Erstellung, Tests und Erweiterung der Appleskripte. Weiterer Dank an Andreas Steffen für eine schnelle Hilfe bei einem FreeS/WAN Problem. Ebenfalls dankbar sind wir Silvia Brodkorb für das Korrektur lesen dieses Textes.

3 Warum nicht WEP?

WEP ist unsicher! [2] Es werden nur symmetrische Schlüssel (shared secrets) zur Authentifizierung verwendet. WEP ist aufgrund einer schlechten Implementation des RC4 Codecs für Verschlüsselungen anfällig für Angriffe. Im Gegensatz dazu vereinfachen Public-Key-Verfahren die Verteilung von Authentifizierungsdaten. IPsec ist standardisiert, weit verbreitet und getestet. Es gibt freie und kommerzielle IPsec-Implementierungen für die meisten Betriebssysteme.

4 Grundlagen zu VPN/IPsec

Ein Verständnis für die Grundlagen zu VPN und IPsec vereinfacht die Installation, Konfiguration und das Lösen von Problemen bei einem Nachbauversuch. Als Vorlage für diese Sektion diente "VPNs and IPsec Demystified"[3].

4.1 VPN Grundlagen

Ein VPN - Virtuelles Privates Netzwerk - erlaubt es, gesicherte, verschlüsselte Verbindungen über ein unsicheres Netzwerk herzustellen. Der Zugriff auf angebotene Ressourcen (Fileserver, Mailserver, ...) im entfernten Netzwerk ist in Verbindung mit einem VPN weiterhin wie gewohnt möglich.

Eine VPN Verbindung besteht aus einer Punkt-zu-Punkt Verbindung. Die Verbindung zwischen den beiden Punkten über ein unsicheres Netz wird als Tunnel bezeichnet. Die beiden Enden dieser Punkt-zu-Punkt Verbindung sind die Peers. Die Peers sind für die Verschlüsselung der Daten beim Eintritt in den Tunnel und für die Entschlüsselung beim Austritt aus dem Tunnel zuständig.

Obwohl ein VPN Tunnel immer nur zwischen zwei Peers etabliert wird, kann ein Peer auch für mehrere andere Peers Tunnel aufbauen. So besteht die Möglichkeit, dass ein Peer für ein ganzes Netzwerk die Ver- und Entschlüsselung übernimmt. Wenn diese Situation eintrifft, wird dieser VPN Peer als VPN Gateway bezeichnet. Das Netzwerk, das sich hinter dem VPN Gateway befindet wird als "Encryption Domain" bezeichnet. Gründe für ein solches Gateway sind die vereinfachte Administration der "Security Policy" sowie ein reduzierter administrativer Aufwand bei der Einrichtung. Man bedenke, dass ohne VPN Gateway, für jede Verbindung, die ein Client initiiert, eine VPN Verbindung zu diesem Rechner im LAN aufgemacht werden müßte.

Innerhalb einer "Encryption Domain" findet keine Verschlüsselung statt. Der Grund hierfür, ist dass dieser Teil des Netzwerks als sicher angesehen wird und unter der Kontrolle des Administrators steht. Wohingegen das Internet als unsicher gilt und ausserhalb der Kontrolle des Administrators liegt.

Wenn zwei VPN Gateways für den Zusammenschluss von zwei Netzen benutzt werden, findet der Tunnel Modus Anwendung. Das bedeutet, das das komplette IP Paket verschlüsselt, mit einem neuen IP Header versehen, versendet wird. Dieser neue IP Header enthält nur die IP Adressen der jeweiligen VPN Gateways und nicht der Maschinen in den jeweiligen "Encryption Domains". Dies führt zu einer erhöhten Sicherheit, weil nur die IP Adressen der Gateways in einem Paket Sniffer auftauchen.

Wenn nur die Daten hinter dem jeweiligen VPN Gateway von Interesse sind und nicht die Daten die auf dem jeweiligen Gateway selber vorhanden sind, wird der Tunnel Modus verwendet. Sind gesicherte Verbindungen mit dem VPN Gateway selbst erwünscht, wird der Transport Modus verwendet. Der Transport Modus verändert das IP Paket anders als der Tunnel Modus, wodurch eine Verbindung zwischen den beiden Gateways möglich wird. Die folgenden Client-Beschreibungen verwenden nur den Tunnel Modus, Verbindungen zu dem Gateway oder zwischen unterschiedlichen Clients im Wireless LAN werden nicht durch ein VPN gesichert.

Unabhängig von der VPN Software die benutzt wird, teilen alle VPN's die folgenden Attribute.

- Beide Peers authentifizieren sich vor dem Tunnelaufbau untereinander, um sicherzustellen, das die verschlüsselten Daten zum richtigen Peer gesendet werden.
- Beide Peers benötigen eine vorkonfigurierte "Policy", die die Protokolle für die spätere Verschlüsselung und Authentifizierung festlegt und die Integrität der Daten sicherstellt.
- Die beiden Peers vergleichen ihre Policy's untereinander und versuchen ein von beiden Seiten akzeptiertes Protokoll auszuhandeln. Schlägt dieser Handel fehl, kommt kein Tunnel zustande.
- Ist die Policy verglichen und abgesprochen, wird ein Schlüssel erzeugt, der von einem symmetrischen Algorithmus für die Ver- und Entschlüsselung der Daten benutzt wird.

4.2 IPsec Grundlagen

Der IPsec Standard wurde entwickelt, um das IP Protokoll sicherer zu machen. Dies wird durch Hinzufügen von zusätzlichen IP Paketheadern erreicht, die auch "Encapsulations" genannt werden. Wenn man sich weitergehend mit IPsec beschäftigen möchte, sind die folgenden RFC's ein guter Startpunkt. Sie sind auf der Seite http://www.rfc-editor.org/ zu finden.

- RFC 2401 IPsec
- RFC 2402 AH
- RFC 2406 ESP
- RFC 2409 IKE

AH ist das "Authentication Header" Protokoll. Es bringt Integrität der IP Pakete aufgrund von Checksummen und prüft, dass keine der Bits innerhalb eines Pakets während der Übertragung verändert worden sind. Die genaue Beschreibung welcher, Teil des IP Pakets verschlüsselt ist und wo der AH Header genau zu finden ist, hängt sehr stark vom Verschlüsselungs-Modus ab und wird in der RFC näher beschrieben. Ein Problem mit AH ist, dass Verbindungen in Kombination mit NAT - Network Address Translation - nicht funktionieren, da die IP Checksumme durch das Hinzufügen eines neuen Headers verändert wird und die AH Checksumme nicht mehr mit der originalen Checksumme übereinstimmt. AH ist nur für die Intregrität der Pakete zuständig und hat nichts mit der Verschlüsselung des Paketinhaltes zu tun.

ESP - "Encapsulating Security Payload" stellt sowohl Integrität als auch Datenverschlüsselung zur Verfügung. Im Transport Modus, der für die Verbindung zwischen zwei VPN Gateways zuständig ist, wird der ESP Header zwischen den originalen unverschlüsselten IP Header und den TCP oder UDP Header eingefügt. Im Tunnel Modus, der benutzt wird, um zwei "Encryption Domains" miteinander über zwei

VPN Gateways zu verbinden, wird der ESP Header zwischen den neuen IP Header und das Original geschrieben. Dies resultiert dann in einem vollständig verschlüsselten IP Paket.

Beide der oben genannten Protokolle fügen einen neuen Header zu einem IP Paket hinzu. Aus diesem Grund sind eigene Protokoll-ID's registriert, und ermöglichen die Identifizierung des IP Paketes als AH oder ESP Paket. AH hat die Protokoll-ID 51 und ESP die Protokoll-ID 50, im Vergleich dazu hat TCP die Protokoll-ID 6 und UDP 17. Die Protokoll-ID ist für die Konfiguration eines Paketfilters notwendig.

Das dritte Protokoll, das IPsec nutzt, ist IKE - "Internet Key Exchange". Dieses Protokoll wird für den automatisierten Schlüsselaustausch zwischen den beiden VPN Peers benutzt. Das manuelle Generieren der Schlüssel für eine Verbindung ist ebenfalls möglich. Der administrative Aufwand bei grossen Installationen ist jedoch nicht vertretbar. IKE benutzt UDP Port 500 für den automatischen Schlüsselaustausch.

Die SA - "Security Association" ist bei IPsec die Bezeichnung für eine Verbindung. Ist eine VPN Verbindung initialisiert, wird eine SA für jedes Protokoll (AH und ESP) erzeugt. SA's werden unidirektional erzeugt, so dass für jede logische Verbindung zwei SA's notwendig sind.

Jeder Peer ist in der Lage mehrere Tunnels aufzubauen. Daraus resultierend haben SA's eineindeutige Nummern. Mit diesen Nummern können die SA's der jeweiligen Gegenstelle zugeordnet werden. Diese Nummer wird SPI - "Security Parameter Index" genannt.

SA's sind in einer Datenbank gespeichert, die als SAD - "Security Association Database" bezeichnet wird. Jeder IPsec Peer hat ausser dieser Datenbank noch eine eigene zweite Datenbank, die SPD - "Security Policy Database". Diese zweite Datenbank enthält die vorkonfigurierten Policy's auf dem Peer. Die meisten IPsec Implementationen erlauben das Anlegen von mehreren Policy's und damit die Auswahl zwischen den unterschiedlichen Algorithmen.

Die Konfigurationsoptionen innerhalb der Policy:

1. Symmetrischer Algorithmus für das Verschlüsseln und Entschlüsseln der Daten
2. Kryptographische Checksumme zur Sicherstellung der Datenintegrität
3. Methode der Authentifizierung
4. Tunnel oder Transport Modus
5. Diffie-Hellman Gruppe
6. Anzahl der Authentifizierungsversuche mit dem Peer
7. Wechselfrequenz der Schlüssel für Ver- und Entschlüsselung
8. Aktivierung von PFS - Perfect Forward Secrecy
9. AH, ESP oder Beides

Bei der Erstellung einer Policy wird eine geordnete Liste von Algorithmen und weiteren Parametern definiert. Der erste Eintrag in dieser Liste der auf beiden Peers übereinstimmt, wird übernommen. Die Policy ist ein sehr kritischer Punkt beim Aufbau eines VPN. Stimmt ein Faktor bei einem der beiden Peers nicht überein, wird kein Tunnel aufgebaut.

Etablieren und Betreiben eines VPN Tunnels erfolgt in zwei Phasen. In der ersten Phase tauschen die beiden Peers die Authentifizierungsmethode, den Verschlüsselungsalgorithmus, den Hashalgorithmus und die Diffie-Hellmann Gruppen aus. Ausserdem authentifizieren sie ihre gegenseitige Identität. Diese Authentifizierung kann entweder im so genannten "Agressive Mode" mit 3 unverschlüsselten Paketen oder im so genannten "Main Mode" mit 6 unverschlüsselten Paketen zustande kommen. Ist dieser Teil des Prozesses abgeschlossen, wird eine so genannte "IKE SA" etabliert und mit der Phase 2 fortgesetzt.

In der Phase 2 wird das Schlüsselmaterial generiert und die beiden Peers einigen sich auf die zu benutzende Policy. Dieser Modus wird "Quick Mode" genannt. Der Unterschied zum "Main Mode", besteht darin, dass der "Quick Mode" nur zustande kommen kann, wenn der "Main Mode" mit Erfolg abgeschlossen worden ist. und alle Pakete in diesem Modus verschlüsselt sind. Das Suchen nach Fehlern in der Konfiguration der Phase 2 ist aufgrund der Verschlüsselung komplizierter. Phase 1 und Phase 2 sind in ihrer Konfiguration voneinander unabhängig. Isakmpd, der IKE Dämon unter OpenBSD, bietet sehr gute Debugfunktionalitäten, die später weitergehend erläutert werden. Ist die Phase 2 fehlerlos und vollständig durchlaufen, wird eine "IPsec SA" etabliert und der Tunnel steht.

Der Aufbau eines Tunnels erfolgt, wenn ein IP-Paket zum ersten Mal bei einem Peer auftrifft mit dem Ziel einer anderen "Encryption Domain". Dann wird der Peer die Phase 1 mit der Gegenstelle, die zu der entsprechenden "Encryption Domain" gehört, einleiten. Haben beide Peers den Tunnel erfolgreich aufgebaut, bleibt er empfangsbereit. Nach Ablauf einer vorkonfigurierten Zeitspanne erfolgt eine Reauthentifizierung und ein erneuter Vergleich der Policy's. Das gleiche gilt analog für die Verschlüsselung und Schlüsselmaterialien. Diese Zeit wird als "IPsec SA" Lebenszeit bezeichnet. Typische Lebenszeiten für "IKE SA's" sind 24 Stunden, für "IPsec SA's" 60 Minuten.

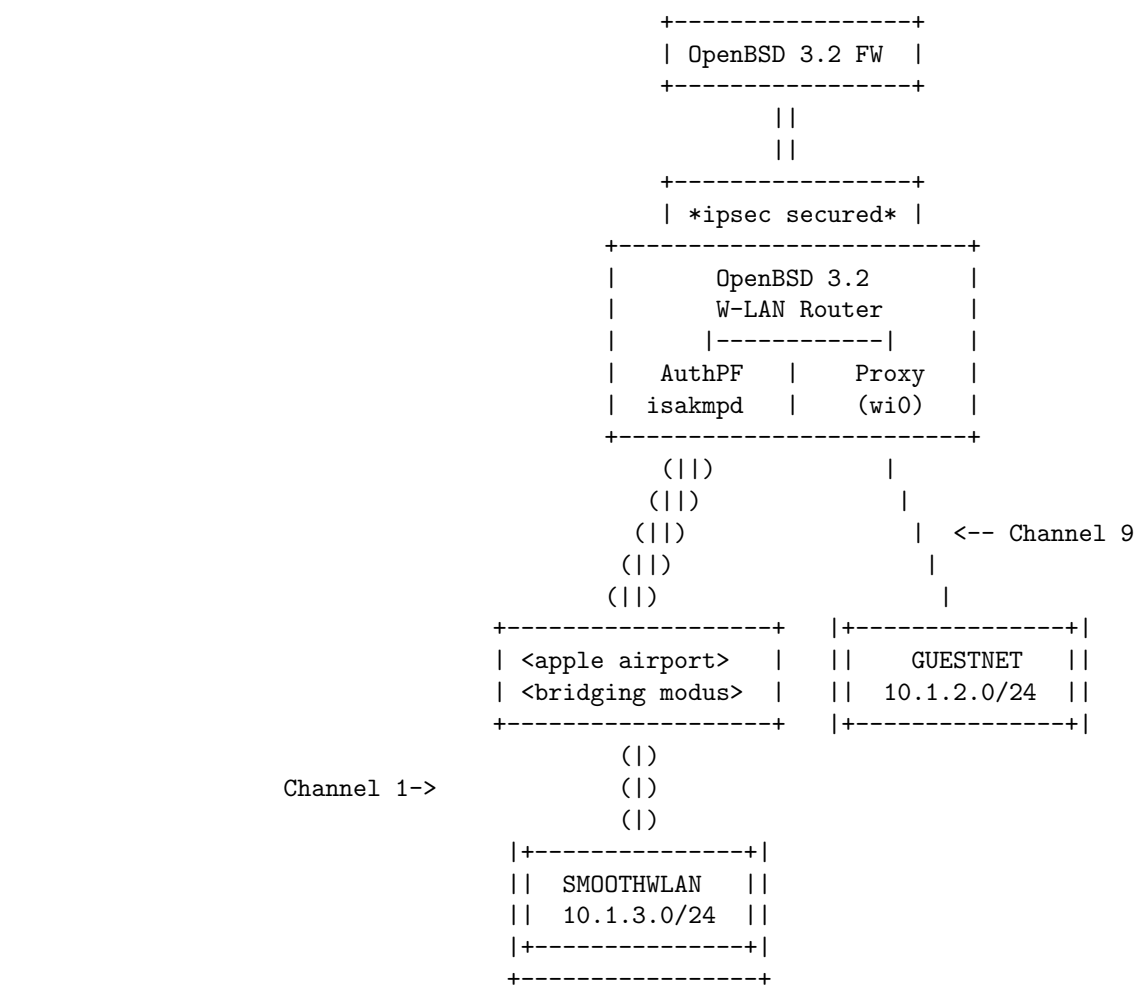
Probleme können dann auftreten, wenn die Lebenszeiten der Tunnels auf beiden Seiten unterschiedlich lang sind oder die Phase 1 Lebenszeit kürzer ist als die Phase 2 Lebenszeit. Bei solchen Problemen kann es sein, dass der Tunnel am Anfang zustandekommt, allerdings aufgrund der unterschiedlichen Lebenszeiten nach der ersten Neukonfiguration kein Tunnel mehr zustande kommen kann. Ein weiteres Fehlerindiz ist, wenn der Tunnel anfängt zu stocken. Hierfür ist meistens eine zu kurze Phase 1 Lebenszeit verantwortlich.

5 Netzstruktur und Hardware

5.1 Netzstruktur

Unser Wireless LAN (Local Area Network) besteht aus zwei unterschiedlichen Zugangspunkten und IP Subnetzen. Das "offene" Netz ist für Gäste, das "sichere" Netz

für Bewohner. In beiden Netzen werden die Netzclienten automatisch per DHCP - Dynamic Host Configuration Protokoll - konfiguriert. Zentrale Komponente bei diesem Aufbau ist unser OpenBSD Wireless LAN Gateway. Im “offenen” Netz werden IP-Adressen an jeden verteilt, im “sicheren” Netz nur an bekannte MAC-Adressen. Dies bietet eine zusätzliche Sicherheitsschicht, da ein potentieller Angreifer erst den IP-Raum ermitteln muß. Im “offenen” Netz wird eine Wireless-LAN-Karte mit PCI-Adapter als Zugang verwendet. Im “sicheren” Netz kommt eine im Bridging-Modus betriebene Apple Airbasestation zum Einsatz, die per Crosslink-Kabel mit dem OpenBSD Wireless LAN Gateway verbunden ist.



Legende:

|| <--- Wired Netzwerk
| <--- Wireless Netzwerk
() <--- IPsec gesichertes Netzwerk

5.2 Offenes Netz

Auf dem OpenBSD Router für das Wireless LAN sorgt ein transparenter Squid Proxy für eine nominelle Sicherheit vor Verbindungen die aus dem unsicheren Netzwerk ins Internet oder LAN gehen. Eine Trafficbeschränkung mit `altqd(8)` ist in Planung, um die vorhandene Bandbreite für Gäste einzuschränken.

Squid kann über das Portssystem (`ports(7)`) von OpenBSD nachinstalliert werden:

```
# cd /usr/ports/www/squid
# env FLAVORS=transparent make install
```

Die folgenden Änderungen an der mitgelieferten, kommentierten Beispielkonfiguration (squid.conf) sind mindestens erforderlich, um in Verbindung mit `pf(4)` die Zwangsproxy Funktion zu realisieren.

```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy  on
httpd_accel_uses_host_header on
...
...
acl our_networks src 10.1.2.0/24
http_access allow our_networks
```

In der `pf.conf(8)` muss folgende Regel aktiviert werden:

```
rdr on wi0 proto tcp from any to any port 80 -> 127.0.0.1 port 3128
pass in quick on wi0 proto tcp from 10.1.2.0/24 to 127.0.0.1/32 port 3128
```

Dadurch werden alle ankommenden Pakete auf dem Gastnetzinterface auf den lokal lauschenden Squid-Dämon umgeleitet. In unserem Gastnetz ist zur Zeit nur HTTP durch den Proxy, DNS und DHCP erlaubt (siehe Paketfilterkonfiguration 8.3).

5.3 IPsec gesichertes Netz

In unserem durch IPsec gesicherten Netz, verwenden wir diverse Programme von OpenBSD die für die Netzwerksicherheit zuständig sind. Der IPsec Stack (`ipsec(4)`) unter OpenBSD wird durch `isakmpd(8)` konfiguriert, für die Firewall setzen wir `pf(4)` und `authpf(8)` in Kombination ein. Der Einsatz von `systrace(1)` und `chroot(8)` zur Abhärtung der nach außen erreichbaren Dämonen ist in Planung.

Die Konfiguration von pf (`pf.conf(5)`), authpf (`authpf(8)`) und isakmpd (`isakmpd.conf(5)`, sowie die Erstellung der x509-Zertifikate wird in den späteren Sektionen näher besprochen.

5.4 Wireless LAN Hardware

Die Wireless Hardware, die wir einsetzen, besteht aus folgenden Komponenten die nicht wahlweise austauschbar sind:

Für das Vorhaben eignet sich eine Wireless LAN Karte von SMC mit Prism II Chipsatz hervorragend, da sie mit den folgenden Einstellungen (`wicontrol(8)`) bis zu 11 Mbit/s an Übertragungsgeschwindigkeit leisten kann:

```
inet 10.1.2.1 255.255.255.0
!wicontrol \$if -c 1 -p 6 -t 11 -f 9 -n guestnet -s guestnet
```

Da die Spezifikation des Prism II Chipsatzes bekannt ist, kann mit Wireless LAN Karten, die diesen Chipsatz verwenden, der HostAP-Modus des wi-Treibers verwendet werden und die Funktionalität einer Basestation nachgebildet werden. Sollte dies nicht auf Anhieb funktionieren (ioctl-Fehlermeldung des Kernels), führt ein Firmwareupdate der Karte meist zum Erfolg.

Für die Seite, die an das durch IPsec gesicherte Netzwerk angeschlossen ist, verwenden wir eine PCI Netzwerkkarte, die direkt an eine Apple Air Port Basestation angeschlossen ist. In der Apple Air Port Basestation befindet sich eine Lucent Orinoco Karte.

Auf der Client Seite haben wir folgende Karten erfolgreich getestet:

- Prism2 basierte Karte von EMC
- Prism2 basierte Karte von Compaq
- Lucent Orinoco Karte von Lucent
- Lucent Orinoco kompatible Karte, die als Einbau in ein Apple Notebook verwendet wird

6 Certification Authority

Für die Erstellung aller Zertifikate wird die freie Krypto-Software openssl [4] verwendet. Der Rechner, der als CA fungiert, ist ein alter Intel 486 Laptop ohne Netzwerkkarte mit installiertem OpenBSD. Der Transfer der Zertifikate erfolgt über Disketten. Der private Schlüssel der CA ist nur auf der Festplatte des CA Rechners gespeichert und durch eine Passphrase geschützt.

Im Vorfeld kann die Konfigurationsdatei openssl.cnf (Beispiel siehe Anhang) editiert werden, um sinnvolle Voreinstellungen vorzunehmen und Schreibarbeit bei der Erstellung der Zertifikate zu sparen.

Vor der Erstellung sollte das korrekte Datum auf dem System eingestellt sein.

Als erstes wird ein RSA Schlüssel mit einer Bitlänge von 2048 generiert und mit einer Passphrase geschützt:

```
# openssl genrsa -des3 -out /etc/ssl/private/ca.key 2048
```

Als nächstes wird die Nachfrage zur Eigensignierung erzeugt. Bei der Wahl des Distinguished Name (DN) sollte auf das Attribut email verzichtet werden. Als "Common Name" empfiehlt sich CA:

```
# openssl req -new -key /etc/ssl/private/ca.key -out /tmp/ca.csr
```

Jetzt erfolgt die Signierung mit Hinzunahme von zusätzlichen Attributen:

```
# openssl x509 -req -days 999 -in /tmp/ca.csr \
   -signkey /etc/ssl/private/ca.key -extfile /etc/ssl/x509v3.cnf \
   -extensions x509v3_CA -out /etc/ssl/ca.crt
```

Anschauen und Verifizieren:

```
# openssl x509 -text -noout -in ca.crt
```

Der DN im Subject Attribut wird später für die Erstellung der isakmpd.policy auf dem OpenBSD Gateway und Client benötigt. Viele nützliche Hinweise und umfangreiche Dokumentation zu openssl in deutsch, sind im OpenSSL-Handbuch des DFN-CERT [5] zu finden.

7 Erstellung der X.509v3 Zertifikate

Die nach der folgenden Anweisung erstellten Schlüssel und Zertifikate können analog für das Gateway und alle Clientsysteme verwendet werden. Der String "foobar" sollte zur Übersichtlichkeit durch den Hostnamen des Systems ersetzt werden. Die Erstellung der Zertifikate kann wahlweise auf der CA oder auf dem Zielsystem selbst erfolgen.

Erzeugung des privaten RSA-Schlüssels mit einer Bitlänge von 2048:

```
# openssl genrsa -out foobar.key 2048
```

Signierungswunsch erzeugen:

```
# openssl req -new -key foobar.key -out foobar.csr
```

Signierung durch die CA und Hinzufügen des Attributes "Subject Alternative Name":

```
# openssl x509 -req -days 999 -in foobar.csr -CA /etc/ssl/ca.crt \
   -CAkey /etc/ssl/private/ca.key -CAcreateserial -out foobar.crt
```

```
# certpatch -t fqdn -i hostname.domain.tld \
   -k /etc/ssl/private/ca.key foobar.crt foobar.crt
```

Falls auf dem System, auf dem die Zertifikate erstellt werden, kein certpatch zur Verfügung steht, gibt es noch einen weiteren Weg, das Zertifikat zu signieren und das Attribut "Subject Alternative Name" hinzuzufügen (`isakmpd(8)`). Dafür wird die openssl-Konfigurationsdatei x509v3.cnf (siehe Anhang) benötigt.

```
# export CERTFQDN=hostname.domain.tld
# openssl x509 -req -days 365 -in foobar.csr -CA /etc/ssl/ca.crt \
   -CAkey /etc/ssl/private/ca.key -CAcreateserial \
   -extfile /etc/ssl/x509v3.cnf -extensions x509v3_FQDN \
   -out foobar.crt
```

Verifikation des signierten Zertifikates:

```
# openssl verify -CAfile ca.crt foobar.crt
```

Anschauen und Überprüfen des "Subject Alternative Name":

```
# openssl x509 -text -noout -in foobar.crt
```

Das Attribut "Subject Alternative Name" im X.509v3 Zertifikat dient beim Schlüsselaustausch über IKE [6] als Identifikation der Kommunikationspartner und muß in der Konfiguration des entsprechenden Dämonen angegeben werden. Es besteht die Auswahl an diversen Identifikationstypen (FQDN, UFQDN, IPv4, IPv6, ASN1). Im Beispiel wurde FQDN - Fully Qualified Domain Name - als Identifikationstyp gewählt.

8 Konfiguration des OpenBSD Gateways

Die Konfiguration des IPsec-Gateways erfolgt in drei Schritten: Zuerst werden die Schlüssel und Zertifikate installiert, dann wird der IKE Key Management Dämon (`isakmpd`) konfiguriert, danach der Paketfilter (`pf`) konfiguriert und aktiviert.

8.1 Installation der Schlüssel und Zertifikate

Nach der Erstellung der Schlüssel und Zertifikate, wie in 7 beschrieben, müssen der private Schlüssel nach /etc/isakmpd/private kopiert werden und die Rechte angepasst werden, so dass nur der Systemadministrator root Leserechte erhält. Das von der CA signierte Zertifikat muß nach /etc/isakmpd/certs kopiert werden und das CA-Zertifikat nach /etc/isakmpd/ca.

```
# ls -laR /etc/isakmpd
```

```
drwxr-xr-x   7 root  wheel   512 Dec  2 22:35 .
drwxr-xr-x  25 root  wheel  2048 Nov 12 22:01 ..
drwxr-xr-x   2 root  wheel   512 Oct  4 03:33 ca
drwxr-xr-x   2 root  wheel   512 Nov 27 08:40 certs
drwxr-xr-x   2 root  wheel   512 Aug 25 19:40 crls
-rw-------   1 root  wheel   884 Dec  3 00:16 isakmpd.conf
-rw-------   1 root  wheel   254 Oct  5 22:06 isakmpd.policy
drwxr-xr-x   2 root  wheel   512 Oct  4 03:33 keynote
drwx------   2 root  wheel   512 Oct  4 03:33 private

/etc/isakmpd/ca:
total 8
drwxr-xr-x  2 root  wheel   512 Oct  4 03:33 .
drwxr-xr-x  7 root  wheel   512 Dec  2 22:35 ..
-rw-r--r--  1 root  wheel  1168 Oct  5 23:34 ca.crt

/etc/isakmpd/certs:
total 12
drwxr-xr-x  2 root  wheel   512 Nov 27 08:40 .
drwxr-xr-x  7 root  wheel   512 Dec  2 22:35 ..
-rw-r--r--  1 root  wheel  1172 Oct  5 23:45 wlan.crt

/etc/isakmpd/crls:
total 4
drwxr-xr-x  2 root  wheel  512 Aug 25 19:40 .
drwxr-xr-x  7 root  wheel  512 Dec  2 22:35 ..

/etc/isakmpd/keynote:
total 4
drwxr-xr-x  2 root  wheel  512 Oct  4 03:33 .
drwxr-xr-x  7 root  wheel  512 Dec  2 22:35 ..

/etc/isakmpd/private:
total 8
drwx------  2 root  wheel   512 Oct  4 03:33 .
drwxr-xr-x  7 root  wheel   512 Dec  2 22:35 ..
-r--------  1 root  wheel  1679 Oct  5 22:58 local.key
```

8.2 Konfiguration des isakmpd-Dämons

Die Konfiguration des isakmpd-Dämonen besteht aus zwei Konfigurationsdateien, der isakmpd.conf und der Policy isakmpd.policy.
isakmpd.conf:

```
[General]
Listen-on= 10.1.3.1

[Phase 1]
Default= SECUREWLAN-1
```

```
[SECUREWLAN-1]
Phase= 1
Transport= udp
Configuration= Default-main-mode
ID= WLAN

[WLAN]
ID-type= FQDN
Name= wlan.mydomain.privat

[Default-main-mode]
DOI= IPSEC
EXCHANGE_TYPE= ID_PROT
Transforms= 3DES-SHA-GRP2-RSA_SIG,BLF-SHA-GRP2-RSA_SIG

[Phase 2]
Passive-connections= SECUREWLAN-2

[SECUREWLAN-2]
Phase= 2
ISAKMP-peer= SECUREWLAN-1
Configuration= Default-quick-mode
Local-ID= wlan-router

[wlan-router]
ID-type= IPV4_ADDR_SUBNET
Network= 0.0.0.0
Netmask= 0.0.0.0

[Default-quick-mode]
DOI= IPSEC
EXCHANGE_TYPE= QUICK_MODE
Suites= QM-ESP-AES-SHA-PFS-GRP2-SUITE,QM-ESP-3DES-SHA-PFS-GRP2-SUITE
```

isakmpd.policy:

```
KeyNote-Version: 2
Comment: Authentication based on CA certificates
Authorizer:     "POLICY"
Licensees:      "CA"

Authorizer:     "CA"
Licensees:      "DN:/C=DE/ST=NRW/L=Bonn/O=Unix WG/CN=CA"
Conditions:     app_domain == "IPsec policy"
&& esp_present == "yes" -> "true";
```

Beide Dateien dürfen nur für root les- und schreibbar sein. Der Dämon wird permanent über die Variable isakmpd_flags in der /etc/rc.conf aktiviert. Um den Dämon zu starten, muß als root folgender Befehl ausgeführt werden:

```
# isakmpd
```

Beim ersten Start empfiehlt es sich, den Dämon im debug-Modus zu starten und zu sehen, ob alles fehlerfrei konfiguriert wurde:

```
# isakmpd -d -DA=70
```

Das sollte etwa folgende Ausgabe liefern:

```
....
Default log_debug_cmd: log level changed from 0 to 70 for class 9
Misc 60 connection_record_passive: passive connection "SECUREWLAN-2" added
Plcy 30 policy_init: initializing
Cryp 40 x509_read_from_dir: reading certs from /etc/isakmpd/ca/
Cryp 60 x509_read_from_dir: reading certificate ca.crt
Cryp 40 x509_read_from_dir: reading certs from /etc/isakmpd/certs/
Cryp 60 x509_read_from_dir: reading certificate wlan.crt
Cryp 70 x509_hash_enter: cert 0x116800 added to bucket 16
Cryp 70 x509_hash_enter: cert 0x116800 added to bucket 53
Cryp 40 x509_read_crls_from_dir: reading CRLs from /etc/isakmpd/crls/
Trpt 70 transport_add: adding 0x117200
Trpt 70 transport_add: adding 0x1172c0
Trpt 70 transport_add: adding 0x117300
```

Beide Zertifikate wurden erfolgreich eingelesen und eine passive Verbindung vorbereitet, der Dämon lauscht jetzt auf UDP-Port 500 und wartet auf Verbindungen von Clientsystemen.

Sollte diese Ausgabe nicht erscheinen, müssen die Zertifikate und Konfigurationsdateien noch einmal sorgfältig überprüft werden.

Viele nützliche und ausführliche Hinweise zur Konfiguration von IPsec unter OpenBSD sind in der englischen FAQ [7] des OpenBSD-Projektes zu finden.

8.3 Konfiguration des Paketfilters

Die TCP/IP-Konfiguration in unserem Netz erfolgt über DHCP. Die Administration der Systeme über Secure Shell. Hier unsere pf.conf, die wir zum Schutz unseres Wireless-LAN Gateways nutzen:

```
# See pf.conf(5) for syntax and examples
#
#########################################################################
#Grundkonfiguration
#########################################################################
#Variablen
#########################################################################
```

```
wlan_base = "rl1"
tcp_services_wlan_base = "{ ssh,domain }"
udp_services_wlan_base = "{ domain,67 }"

wlan_nic = "wi0"
tcp_services_wlan_nic = "{ http,domain }"
udp_services_wlan_nic = "{ domain,67 }"

trusted_networks = "{ 10.1.4.0/24,10.1.3.0/24,10.1.2.0/24,10.1.1.0/24 }"
internal = "rl0"

scrub in all no-df
scrub out all no-df

# transparent proxy
rdr on $wlan_nic proto tcp from any to any port 80 -> 127.0.0.1 port 3128

# first block everything on every interface
block in log all
block out log all

# antispoof rules
antispoof for lo0
antispoof for $wlan_base
antispoof for $wlan_nic
antispoof for $internal

block in from no-route to any
block in quick on $internal from any to 255.255.255.255

pass in quick on lo0 all
pass out quick on lo0 all

pass in quick on enc0 all
pass out quick on enc0 all

#############################################################################
#Interne Connection
#############################################################################

pass in quick on $internal all
pass out quick on $internal all


#############################################################################
#Wavelan Card Intern
#############################################################################
# transparent proxy
pass in quick on $wlan_nic proto tcp from 10.1.2.0/24 to 127.0.0.1/32 port 3128

pass in quick on $wlan_nic proto tcp from 10.1.2.0/24 to any \
   port $tcp_services_wlan_nic keep state
```

```
pass in quick on $wlan_nic proto udp from 10.1.2.0/24 to any \
   port $udp_services_wlan_nic keep state

pass out quick on $wlan_nic inet proto { tcp, udp } all keep state
pass out quick on $wlan_nic inet6 proto { tcp, udp } all keep state

pass out quick on $wlan_nic inet proto icmp all icmp-type 8 code 0 keep state
pass in quick on $wlan_nic inet proto icmp all icmp-type 8 code 0 keep state

#########################################################################
#Wavelan AP
#########################################################################

pass in quick on $wlan_base proto tcp from 10.1.3.0/24 to \
   $trusted_networks port $tcp_services_wlan_base keep state

pass in quick on $wlan_base proto udp from 10.1.3.0/24 to \
   any port $udp_services_wlan_base keep state

pass out quick on $wlan_base inet proto { tcp, udp } all keep state

pass out quick on $wlan_base inet proto icmp all icmp-type 8 code 0 keep state
pass in quick on $wlan_base inet proto icmp all icmp-type 8 code 0 keep state
```

Die permanente Aktivierung des Paketfilters erfolgt über die Variable pf in der rc.conf. Zur Laufzeit kann pf mittels

```
 # pfctl -e -f /etc/pf.conf
```

aktiviert werden.

Eine weitere Funktion, die OpenBSD in diesem Zusammenhang bietet, ist die Nutzung eines auf Benutzerebene funktionierenden Zusatzes zu `pf(4)`. Dieser Zusatz heißt `authpf(8)`. Die Idee hinter authpf ist, dass auf Benutzerebene bestimmte Filterregeln dem Paketfilter nach erfolgreicher Authentifizierung über `ssh(1)` dynamisch hinzugefügt werden und während der aktiven SSH-Verbindung bestehen bleiben.

authpf arbeitet in diesem Fall mit ssh zusammen und hat eine gesonderte Shell in der Passwortdatenbank (`passwd(1)`). Zum Schutz eines mit root-Rechten laufenden Dämonen gegen einen potentiellen Angriff eignet sich authpf hervorragend. Wir schützen uns damit aktiv vor Denial of Service-Attacken, da der isakmpd-Dämon nur mit authentifizierten Benutzern kommuniziert.

Zur Konfiguration:

Da in unserem Netzwerk jeder Wireless LAN Benutzer, der IPsec verwendet, ebenfalls eine normale Shell auf dem Gateway besitzt, verwenden wir einen zentralen Benutzeraccount “auth”, der als Shell “authpf” hat und eine “authorized_keys” `ssh(1)` mit allen öffentlichen SSH-Schlüsseln. Denkbar ist aber auch, jeden Benutzer separat anzulegen, um die Möglichkeit der Regulierung des Zugriffs über die möglichen Fähigkeiten von authpf zu nutzen.

In der Datei /etc/authpf/authpf.allow stehen die Benutzer, denen ein Zugriff auf die Shell authpf gestattet ist. Inhalt von authpf.allow:

```
auth
```

In der Datei /etc/authpf/authpf.conf steht die Anweisung, wie neue Regeln dem bestehenden pf-Regelwerk hinzugefügt werden sollen.

```
rule_action=tail
```

In der Datei /etc/authpf.rules werden die neuen pf-Regeln definiert:

```
if="rl1"
ipsec_gw="10.1.3.1"
pass in quick on $if proto udp from $user_ip/32 to $ipsec_gw/32 port = isakmp
keep state
pass out quick on $if proto udp from $ipsec_gw/32 to $user_ip/32 port = isakmp
keep state
pass in quick on $if proto esp from $user_ip/32 to $ipsec_gw/32
pass out quick on $if proto esp from $ipsec_gw/32 to $user_ip/32
```

Es besteht die Möglichkeit, eine auth.message und eine auth.problem anzulegen, die bei einer erfolgreichen oder nicht erfolgreichen Authentifizierung in der Konsole des Benutzers als Hinweis ausgegeben wird.

9 Konfiguration eines OpenBSD Clients

OpenBSD eignet sich auch hervorragend als sicheres Betriebssystem für den Desktopbereich. In dieser Sektion beschreiben wir die Konfiguration eines OpenBSD Clients.

9.1 Installation der Schlüssel und Zertifikate

Nach der Erstellung der Schlüssel und Zertifikate, wie in 7 beschrieben, muß der private Schlüssel nach /etc/isakmpd/private kopiert werden und die Rechte angepasst werden, so dass der private Schlüssel nur von root lesbar ist. Das eigene Zertifikat muß nach /etc/isakmpd/certs, das CA Zertifikat nach /etc/isakmpd/ca kopiert werden.

9.2 Konfiguration von isakmpd

Die Konfiguration von isakmpd erfolgt in ähnlicher Weise wie die des Gateways. Die isakmpd.policy unterscheidet sich nicht. In der isakmpd.conf kann man sich für eine Kombination von Algorithmen entscheiden. Aufgrund von Performanceunterschieden zwischen den diversen Modi wurde folgende Kombination gewählt:

isakmpd.conf:

```
[General]
Listen-on= 10.1.3.2

[Phase 1]
10.1.3.1= WLAN-REMOTE

[WLAN-REMOTE]
Phase= 1
Transport= udp
Configuration= Default-main-mode
Address= 10.1.3.1
Local-address= 10.1.3.2
ID= LAPTOP

[LAPTOP]
ID-type= FQDN
Name= client.mydomain.privat

[Default-main-mode]
DOI= IPSEC
EXCHANGE_TYPE= ID_PROT
Transforms= BLF-SHA-GRP2-RSA_SIG

[Phase 2]
Connections= SECUREWLAN-GATEWAY

[SECUREWLAN-GATEWAY]
Phase= 2
ISAKMP-peer= WLAN-REMOTE
Configuration= Default-quick-mode
Local-ID= extern-laptop
Remote-ID= wlan-router

[extern-laptop]
ID-type= IPV4_ADDR
Address= 10.1.3.2

[wlan-router]
ID-type= IPV4_ADDR_SUBNET
Network= 0.0.0.0
Netmask= 0.0.0.0

[Default-quick-mode]
DOI= IPSEC
EXCHANGE_TYPE= QUICK_MODE
Suites= QM-ESP-AES-SHA-PFS-GRP2-SUITE
```

isakmpd.policy:

```
KeyNote-Version: 2
Comment: Authentication based on CA certificates
Authorizer:     "POLICY"
Licensees:      "CA"

Authorizer:     "CA"
Licensees:      "DN:/C=DE/ST=NRW/L=Bonn/O=Unix WG/CN=CA"
Conditions:     app_domain == "IPsec policy"
&& esp_present == "yes" -> "true";
```

Bei Verwendung von authpf muß jetzt eine SSH-Verbindung zum Wireless LAN Gateway aufgebaut werden und während der Benutzung des IPsec Tunnels auch aktiv bleiben:

```
$ ssh auth@wlan
```

Dann kann der isakmpd-Dämon gestartet werden:

```
# isakmpd
```

Jetzt sollte jeglicher Traffic durch die aufgebauten unidirektionalen Tunnel verlaufen. Überprüfen läßt sich der erfolgreiche Aufbau der Security Association's mit `netstat(1)`:

```
# netstat -rn -f encap
```

über das Kernfilesystem (`mount_kernfs(8)`) von OpenBSD

```
# mkdir /kern ; mount -t kernfs /kern /kern
# cat /kern/ipsec
```

oder mit `tcpdump(8)`.

```
# tcpdump -i wi0
```

In der Skriptesektion (Sektion 13) des Dokumentes findet sich eine Möglichkeit, den Aufbau des Tunnels zu automatisieren.

10 Konfiguration eines Linux Clients

Der Linux Kernel hat leider keinen integrierten IPsec-Support, allerdings gibt es durch das FreeS/WAN-Projekt [8] eine freie Lösung dafür. Als erstes muß der Kernel gepatcht und mit IPsec kompiliert werden. Der FreeS/WAN Superpatch [10] ist sehr zu empfehlen, da er den X.509-Patch [9] und Patches, um einen IPsec-Tunnel sorgfältig abzubauen, integriert hat.

10.1 Installation der Schlüssel und Zertifikate

Die Zertifikate werden in folgende Verzeichnisse abgelegt:
Privater Schlüssel:
/etc/ipsec.d/private/client.key
Zertifikat:
/etc/ipsec.d/client.crt
CA-Zertifikat:
/etc/ipsec.d/cacerts/ca.crt
Der private Schlüssel sollte vor unberechtigten Lesezugriff geschützt werden.

```
 # ls -laR /etc/ipsec.d/
/etc/ipsec.d/:
insgesamt 36
drwxr-xr-x    6 root     root         4096 2002-12-15 12:16 .
drwxr-xr-x   95 root     root         8192 2003-01-11 17:18 ..
drwxr-xr-x    2 root     root         4096 2002-12-16 10:03 cacerts
drwxr-xr-x    2 root     root         4096 2002-10-05 23:55 crls
drwx------    2 root     root         4096 2002-12-15 12:16 private
-rw-r--r--    1 root     root         1216 2002-10-07 09:16 client.crt

/etc/ipsec.d/cacerts:
insgesamt 16
drwxr-xr-x    2 root     root         4096 2002-12-16 10:03 .
drwxr-xr-x    6 root     root         4096 2002-12-15 12:16 ..
-rw-------    1 root     root         1168 2002-10-07 09:17 ca.crt

/etc/ipsec.d/crls:
insgesamt 8
drwxr-xr-x    2 root     root         4096 2002-10-05 23:55 .
drwxr-xr-x    6 root     root         4096 2002-12-15 12:16 ..

/etc/ipsec.d/private:
insgesamt 16
drwx------    2 root     root         4096 2002-12-15 12:16 .
drwxr-xr-x    6 root     root         4096 2002-12-15 12:16 ..
-r--------    1 root     root         1679 2002-10-07 08:39 client.key
```

10.2 Konfiguration von FreeS/WAN

Die Konfiguration von FreeS/WAN besteht aus zwei Dateien, ipsec.conf:

```
# /etc/ipsec.conf - FreeS/WAN IPsec configuration file
# basic configuration
config setup
    interfaces=%defaultroute
    klipsdebug=none
    plutodebug=none
    plutoload=%search
    plutostart=%search
```

```
    uniqueids=yes

conn %default
    keyingtries=3
    disablearrivalcheck=no
    authby=rsasig

conn wlan
    left=10.1.3.2
    leftcert=client.crt
    leftid=@client.mydomain.privat
    rightid=@wlan.mydomain.privat
    right=10.1.3.1
    rightsubnet=0.0.0.0/0
    rightrsasigkey=%cert
    type=tunnel
    auto=start
    keyexchange=ike
```

und ipsec.secrets:

```
: RSA client.key
```

Über die beigelieferten Start- und Stoppskripte kann der Tunnel auf- und abgebaut werden. Bei Verwendung von authpf nicht vergessen, vorher die ssh Verbindung zum Wireless LAN Gateway aufzubauen.

```
 # /etc/init.d/ipsec start
```

In der Skriptesektion (Sektion 13) des Dokumentes findet sich eine Möglichkeit, den Aufbau des Tunnels zu automatisieren.

11 Konfiguration eines MacOS X Clients

Seit MacOS X 10.2 (Codename Jaguar) ist der KAME [12] IPsec Stack in das Betriebssystem integriert worden. Leider wird weder eine vollständige Dokumentation zu allen Komponenten mitgeliefert noch eine graphische Oberfläche zur Konfiguration. Die Konfiguration besteht aus der Erstellung der Zertifikate und den beiden Konfigurationsdateien für setkey (`setkey(1)`) und racoon (`racoon.conf(5)`).

11.1 Installation der Schlüssel und Zertifikate

Nach der Erstellung der Schlüssel und Zertifikate, wie in Sektion 7 beschrieben, werden die Dateien passend im Filesystem plaziert und bestimmte Zugriffsrechte eingestellt. Der private Schlüssel des Clients und die Zertifikate der CA und des Clients werden nach /etc/certs kopiert. Die Benutzerrechte des privaten Schlüssel müssen angepasst werden.

```
% sudo mkdir /etc/certs
% sudo cp client.key client.crt ca.crt /etc/certs
% sudo chown -R 0.0 /etc/certs
% sudo chmod 400 /etc/certs/client.key
```

11.2 Konfiguration von KAME

Die Konfigurationsdatei (/etc/ipsec.conf) für setkey legt fest, welcher Traffic über den Tunnel geleitet werden soll:

ipsec.conf:

```
spdadd 10.1.3.2/32 0.0.0.0/0 any -P out ipsec
 esp/tunnel/10.1.3.2-10.1.3.1/require;
spdadd 0.0.0.0/0 10.1.3.2/32 any -P in ipsec
 esp/tunnel/10.1.3.1-10.1.3.2/require;
```

Racoon ist der IKE-Dämon unter MacOS X 10.2 und wird über /etc/racoon/racoon.conf konfiguriert.

racoon.conf:

```
path certificate "/etc/certs";
log info;

remote anonymous
{
  exchange_mode main;
  doi ipsec_doi;
  certificate_type x509 "client.crt" "client.key";
  my_identifier fqdn "client.mydomain.privat";
  peers_identifier fqdn "wlan.mydomain.privat";
  initial_contact on;
  situation identity_only;
  proposal_check strict;
  nonce_size 16;
  support_mip6 on;

  proposal {
    encryption_algorithm 3des;
    hash_algorithm sha1;
    authentication_method rsasig;
    dh_group 2;
  }
}

sainfo anonymous
{
  pfs_group 2;
  encryption_algorithm aes;
```

```
  authentication_algorithm hmac_sha1;
  compression_algorithm deflate;
}
```

Nach dem Aufbau der SSH Verbindung zum Wireless LAN Gateway wird zuerst setkey aufgerufen:

```
% sudo setkey -f /etc/ipsec.conf
```

Danach muß der IKE Dämon gestartet werden:

```
% sudo racoon -f /etc/racoon/racoon.conf
```

In der Skriptesektion (Sektion 13) des Dokumentes findet sich eine Möglichkeit, den Aufbau des Tunnels zu automatisieren.

12 Konfiguration eines Windows 2000 Clients

Bei der Benutzung von einem Windows Client ist zu beachten, dass nach Möglichkeit alle Patches und Service Packs von M$ installiert wurden. Windows 2000 hat einen integrierten IPsec Stack, allerdings wird im folgenden die Konfiguration der Software SSH Sentinel [15] beschrieben, da die Installation und Konfiguration einfacher ist. (**Die Software ist nicht kostenfrei benutzbar!**)

12.1 Installation der Schlüssel und Zertifikate

Nach der Erstellung der Schlüssel und Zertifikate, wie in Sektion 7 beschrieben, wird ein Schlüsselbund im pkcs12-Format benötigt:

```
# openssl pkcs12 -export -certfile ca.crt -in client.crt \
    -inkey client.key -out client.p12
```

Der erzeugte pkcs12-Schlüsselbund beinhaltet sowohl den privaten Schlüssel und das Zertifikat für den Client als auch das CA Zertifikat. Dieser kann nach der Installation von SSH Sentinel importiert werden.

12.2 Konfiguration von SSH Sentinel

Wir werden keine Bilder einfügen, da im Internet illustrierte Beschreibungen [11] zu finden sind. Wenn SSH Sentinel installiert wird, wird ein Schlüssel erzeugt, der ingnoriert werden kann.

Nach dem Import des pkcs12-Schlüsselbundes kann unter VPN Connections eine neue Verbindung angelegt werden. Diese sollte wie in der OpenBSD Client Configuration mit dem “Quick Mode” Transform AES-SHA1 und “Main Mode” Transform Blowfish-SHA1 konfiguriert werden.

Die Group sollte auf Group 2 (1024 Bit Key) gestellt werden, damit sie mit der Konfiguration des Gateways übereinstimmt. Die Option PFS kann in unserem Beispiel aktiviert werden. In das Feld des zu benutzenden Keys wird der eingebundene CA Key ausgewählt.

Nachdem die Verbindung eingerichtet wurde, kann als Gegenstellen IP-Adresse die IP-Adresse des Gateways eintragen werden. Als Netzwerk der Gegenstelle wird das Netzwerk 0.0.0.0 eingetragen. Nachdem die Konfiguration abgeschlossen ist, kann man über Select VPN die Konfiguration auswählen und starten. In der Option Audit kann das Log der IPsec Verbindung angesehen werden und mögliche Probleme ähnlich wie mit der Option -D von isakmpd analysiert werden.

12.3 SSH Tools für authpf

Für die Verwendung von authpf wird ein SSH Client benötigt. Putty und Co. [16] sind frei erhältlich und leicht zu installieren. Man sollte mindestens folgende Tools installieren:

1. Putty (Free Win32 SSH Client)
2. Puttygen (SSH Key Generation Tool für Win32)
3. Pageant (SSH KeyAgent)

Nachdem diese Werkzeuge installiert worden sind, sollte ein Schlüsselpaar erzeugt werden, um sich als Benutzer “auth” auf dem Gateway einzuloggen. Es besteht aber auch die Möglichkeit ein von OpenSSH erzeugtes Schlüsselpaar zu verwenden bzw. zu konvertieren. (`ssh-keygen(1)`) Vor der Anmeldung per SSH muß der öffentliche Schlüssel auf dem Wireless LAN Gateway installiert werden.

13 Skripte

Eine sehr nützliche Vereinfachung bei der Verwendung dieser vielen Sicherheitsmechanismen ist der Einsatz von kleinen Skripten. Für die Skripte unter Unix / Linux wird das Softwarepaket Xdialog benötigt. Folgende Varianten können als Grundlage für eigene Kreationen genutzt werden:

13.1 OpenBSD und Blackbox

Variante mit Blackbox unter OpenBSD:

```
~/.xinitrc:
ssh-agent /bin/sh -c "ssh-add < /dev/null ; /usr/local/bin/ipsec.sh";

ipsec.sh:
#!/bin/sh
export PASS=$(Xdialog --stdout --center --password --inputbox "IPsec enablen?"
8
 35)
if [ \$PASS ];then
  aterm -geometry 36x3 -e ssh -t auth@wlan &
  sleep 3
  echo \$PASS|sudo -S /sbin/isakmpd
  unset PASS
  /sbin/pfctl -f /etc/pf-ipsec.conf
  /sbin/pfctl -e
  exec blackbox
else
  exec blackbox
fi
```

13.2 Linux und Windowmaker

Variante mit Windowmaker unter Linux:

```
~/GNUstep/autostart:
~/bin/ipsec.sh &
/usr/bin/ssh-add /home/wbx/.ssh/id_dsa < /dev/null &

ipsec.sh:
#!/bin/sh

Xdialog --stdout --center --yesno "IPsec aktivieren?" 8 35
if [ $? -eq 0 ]; then
  aterm -geometry 40x10 -e ssh -t auth@wlan &
  sleep 3
  sudo /etc/init.d/ipsec start
  sleep 2
fi

/etc/sudoers:
wbx ALL=NOPASSWD: /etc/init.d/ipsec
```

13.3 MacOS X und Aqua

Mit zwei Shellskripten und einem Appleskript kann der Vorgang des Tunnelauf- und abbaus automatisiert werden.

/usr/local/bin/start_vpn:

```
#!/bin/sh
sudo setkey -FP
sudo setkey -f /etc/ipsec.conf
sudo racoon -f /etc/racoon/racoon.conf
```

/usr/local/bin/stop_vpn:

```
#!/bin/sh
pid=‘ps ax | awk ’/racoon/ { print \$1 }’ | head -1‘
sudo kill $pid
sudo setkey -FD
sudo setkey -FP
```

Apple Skript (nicht paste&copy nutzen!):

```
on run
    tell application "Finder"
     activate
       display dialog "IPSEC Verbindung auf- oder abbauen?" buttons \
            {"abbrechen", "abbauen", "aufbauen"} default button "aufbauen"
         set myResult to (button returned of result)
end tell

    if myResult is "aufbauen" then
       do shell script
           "/usr/local/bin/start_vpn" with administrator privileges
       tell application "Finder"
        activate
          display dialog "Tunnel erfolgreich aufgebaut" buttons {"OK"}

    end tell

    else if myResult is "abbauen" then
        do shell script
            "/usr/local/bin/stop_vpn" with administrator privileges
        tell application "Finder"
          activate
            display dialog "VPN Tunnel gestoppt!" buttons {"OK"}
        end tell
    end if
end run
```

14 Analysemöglichkeiten und Probleme

14.1 Tcpdump

Mit tcpdump lassen sich die Pakete auf ihrem Weg vom ersten bis zum letzten Netzwerkinterface des Senders und dann wieder vom ersten bis zum letzten Netzwerkinterface des Empfängers loggen. Mit -s kann die Größe der eingefangenen

Pakete festgelegt werden, so dass Phase 1 (Main Mode) der Kommunikation zum Aufbau der Tunnel komplett eingefangen werden kann. Die geloggten Pakete lassen sich dann bequem über Ethereal[17] analysieren.

14.2 Zeit

Die korrekt eingestellte Systemzeit auf allen Systemen ist Grundvoraussetzung für ein funktionierendes, IPsec gesichertes, Wireless LAN. Als kleine Anekdote:
Als mein ibook aus der Reparatur wiederkam, wunderte ich mich mindestens 2-3 Stunden, warum mein Tunnel nicht mehr aufgebaut werden konnte, um dann festzustellen, dass die Systemzeit auf irgendwas um 1961 eingestellt war und das Zertifikat ungültig war ....

14.3 SA und Tunnel entfernen

Sehr hilfreich in der Testphase ist der Befehl:

```
ipsecadm flush
```

Dieser Befehl auf dem Gateway ausgeführt, entfernt alle Security Associations und Tunnelverbindungen.

Für den Fall das nur eine Tunnelverbindung enfernt werden soll, hier ein Shellskript, welches als Parameter die IP-Adresse des Clients benötigt:

```
#!/bin/sh
#
# q&d shellscript to flush a ipsec connection
#

if [ ! $1 ];then
  echo "no ip address given"
  exit 1
fi

SRC=$1

# get spi's
mount |grep kernfs 2>&1 > /dev/null

if [ $? != 0 ];then
  echo "you have to mount kernfs first, man mount_kernfs"
  exit 1
fi

SPIDST=`grep "Destination.*${SRC}," /kern/ipsec|awk '{ print $3 }'|sed 's/,$//'`
SPISRC=`grep -B 3 "Source.*${SRC}" /kern/ipsec|awk '/SPI/ { print $3 }'| sed 's/,$//'`
DST=`grep -B 3 "Source.*${SRC}" /kern/ipsec|awk '/SPI/ { print $6 }'| sed 's/,$//'`
```

```
# flush SA
ipsecadm delspi -spi $SPISRC -dst $DST
ipsecadm delspi -spi $SPIDST -dst $SRC
# flush flows
ipsecadm flow -delete -src $SRC -addr 0.0.0.0 0.0.0.0 $SRC 255.255.255.255 -out
ipsecadm flow -delete -dst $SRC -addr $SRC 255.255.255.255 0.0.0.0 0.0.0.0 -in
```

14.4 MTU-Problem

Um diverse Probleme mit NFS und langsamen Verbindungen von anderen Protokollen wie HTTP zu vermeiden, muß auf allen Gateways, Wireless-LAN-Gateway als auch Internet-Gateway, der Paketfilter so konfiguriert werden, dass alle Pakete normalisiert, defragmentiert und das dont-fragment bit zurückgesetzt wird. Dies erfolgt bei OpenBSD's Paketfilter pf über folgende Anweisungen in der /etc/pf.conf:

```
scrub in all no-df
scrub out all no-df
```

14.5 Probleme mit NFS - Network File System

NFS scheint nicht mittels UDP über IPsec zu funktionieren. Es sollte TCP verwendet werden. Sowohl Linux als auch OpenBSD Clients unterstützen NFS über TCP. (`mount(1)`)

14.6 Probleme mit dem Tunnelabbau bei Verwendung von authpf

Auf Clientseite muß vor dem Beenden der SSH-Verbindung der Tunnel abgebaut werden, sonst besteht bis zum Auslaufen des Timeouts eine logische Verbindung zwischen Client und Gateway, so dass vom Client aus keine Verbindungen nach außen mehr möglich sind. Letzter Ausweg siehe Punkt 14.3.

Internet-Quellen

[1] *OpenBSD Projektseite*
http://www.openbsd.org/

[2] *802.11 Security Vulnerabilities*
http://www.cs.umd.edu/~waa/wireless.html

[3] *VPNs and IPsec Demystified*
http://www.oreillynet.com/pub/bsd/2002/12/12/FreeBSD_Basics.html

[4] *OpenSSL Projektseite*
http://www.openssl.org/

[5] *OpenSSL Handbuch des DFN-CERT*
http://www.dfn-pca.de/certify/ssl/handbuch/ossl095/ossl095.html

[6] *RFC2409 - The Internet Key Exchange*
ftp://ftp.rfc-editor.org/in-notes/rfc2409.txt

[7] *OpenBSD IPsec FAQ*
http://www.openbsd.org/faq/faq13.html

[8] *FreeS/WAN Projektseite*
http://www.freeswan.org

[9] *X.509 Patches für FreeS/WAN*
http://www.strongsec.com/freeswan/

[10] *FreeS/WAN Superpatch*
http://www.freeswan.ca

[11] *OpenBSD IPsec Clientkonfigurationen*
http://www.allard.nu/openbsd/

[12] *KAME Projektseite*
http://www.kame.net

[13] *KAME Configuration Hints*
http://www.kame.net/newsletter/20001119/

[14] *KAME mit X.509 Zertifikaten*
http://www.kame.net/newsletter/20001119b/

[15] *SSH Sentinel*
http://www.ssh.com/products/security/sentinel/

[16] *Putty - Free Win32 SSH Client*
http://www.chiark.greenend.org.uk/~sgtatham/putty/

[17] *Ethereal - Netzwerkanalysetool*
http://www.ethereal.com

x509v3.cnf:

```
# default settings
CERTPATHLEN = 1
CERTUSAGE = digitalSignature,keyCertSign
CERTIP = 0.0.0.0
CERTFQDN = nohost.nodomain
CERTUFQDN = nobody@nohost.nodomain

# This section should be referenced when building an x509v3 CA
# Certificate.
# The default path length and the key usage can be overriden
# modified by setting the CERTPATHLEN and CERTUSAGE environment
# variables.
[x509v3_CA]
basicConstraints=critical,CA:true,pathlen:$ENV::CERTPATHLEN
keyUsage=$ENV::CERTUSAGE

# This section should be referenced to add an IP Address
# as an alternate subject name, needed by isakmpd
# The address must be provided in the CERTIP environment variable
[x509v3_IPAddr]
subjectAltName=IP:$ENV::CERTIP

# This section should be referenced to add a FQDN hostname
# as an alternate subject name, needed by isakmpd
# The address must be provided in the CERTFQDN environment variable
[x509v3_FQDN]
subjectAltName=DNS:$ENV::CERTFQDN

# This section should be referenced to add a UFQDN hostname (email)
# as an alternate subject name, needed by isakmpd
# The address must be provided in the CERTUFQDN environment variable
[x509v3_UFQDN]
subjectAltName=email:$ENV::CERTUFQDN
```

openssl.cnf:

```
#
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

RANDFILE = /dev/arandom

#####################################################################
[ req ]
default_bits = 2048
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
```

```
[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = DE
countryName_min = 2
countryName_max = 2

stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = NRW

localityName = Locality Name (eg, city)
localityName_default = Bonn

0.organizationName = Organization Name (eg, company)
0.organizationName_default = Unix WG

commonName = Common Name (eg, fully qualified host name)
commonName_max = 64

emailAddress = Email Address
emailAddress_max = 64

[ req_attributes ]
challengePassword = A challenge password
challengePassword_min = 4
challengePassword_max = 20
```