

Projektarbeit

Schaffung der VPN-Lösungen mittels IPSec-Protokoll und Auswertung der Leistung.

Implementierung eines VPN zwischen Cisco-PIX & Linux-Gateway sowie Leistungsauswertung.

Von Eno Vaso
info@eno-vaso.de



Abnahme durch:
Ingo Schoof
stellv. Fachbereichsleiter und Infrastrukturbetreuer

Inhaltsverzeichnis

1 Einleitung	4
2 Mitteln	5
3 VPN kurz erklärt	5
4 Das IPSec-Protokoll	6
4.1.1 Security Associations (SA).....	6
4.1.2 Transportmodus.....	6
4.1.3 Tunnelmodus.....	7
4.1.4 AH (Authentication Header).....	7
4.1.5 ESP (Encapsulating Security Payload).....	8
4.1.6 IKE (Internet Key Exchange).....	10
5 Der Versuchsaufbau im Site-to-Site VPN Architektur	11
5.1 Site-to-Site VPN zwischen Linux-Gateways.....	11
5.2 Site-to-Site VPN zwischen PIX- und Linux-Gateway.....	12
6 Installation und Konfiguration von FreeS/Wan.....	13
6.1 Linux FreeS/WAN.....	13
6.2 Installation von FreeS/Wan.....	13
6.3 Konfiguration von FreeS/Wan.....	14
6.3.1 Konfiguration von /etc/ipsec.conf.....	14
6.3.2 Konfiguration von /etc/ipsec.secrets.....	17
7 Die Konfiguration von Cisco-PIX.....	20
7.1 Zugriff auf Cisco-PIX über das Terminalprogramm Kermit.....	20
7.2 Die Grundkonfiguration von PIX.....	20
7.2.1 Die wichtigsten Einstellungen für eine VPN-Verbindung.....	21
7.3 Die Konfiguration sichern.....	22

8 Starten und Testen der Verbindung	24
8.1 Starten und Testen der Verbindung zwischen den Linux-Gateways.....	24
8.2 Starten und Testen der Verbindung auf den PIX-Gateway.....	27
9 Mitschnitt eines Verbindungsaufbaus	31
9.1 Vor dem Aufbau des Tunnels.....	31
9.2 Nach dem Aufbau des Tunnels.....	32
10 Die Leistungsmessung	35
10.1 Auswertung der Leistung zwischen den Linux-Gateways.....	36
10.1.1 Leistung bei einem 200 MHz CPU / 128 MB RAM.....	36
10.1.2 Leistung bei einem 500 MHz CPU / 128 MB RAM.....	36
10.1.3 Leistung bei einem 1.6 GHz CPU / 512 MB RAM.....	37
10.1.4 Grafischer Darstellung der Rechner-Leistung.....	37
10.2 Auswertung der Leistung zwischen den Linux- und PIX-Gateway.....	39
11 Quellen.....	40

1. Einleitung

In der Kommunikationszeit ist der so genannte Virtual Privat Network (VPN) für den gesicherten Datentransfer ein Muss. Es gibt jedoch viele VPN-Lösungen auf dem Markt mit unterschiedlichen Kosten. Die vorliegende Projektarbeit beschreibt im Schnelldurchgang, wie ein VPN zwischen Linux-Rechnern, sowie Linux-Rechnern und Cisco-Routern installiert und konfiguriert werden kann. Besonderes interessant ist die Implementierung eines VPN zwischen Cisco PIX und Linux-Gateways mit FreeS/WAN. Der eigentliche Schwerpunkt dieser Arbeit ist jedoch die Messung der Leistung einer VPN-Verbindung auf unterschiedlichen Rechnern und der Vergleich mit der Leistung von unverschlüsseltem Datentransfer.

Realisierung der VPN, wird mit Hilfe des IPSec-Protokolls, welches durch eine gewisse Standardisierung eine Interoperabilität zwischen verschiedenen Systemen möglich macht. Mit Linux FreeS/WAN und CISCO existieren zwei Implementierungen dieses Standards. Hier sollte es nun möglich sein, eine VPN-Verbindung zwischen beiden Systemen zu installieren und die Leistung zu prüfen.

Eine weitere Projektarbeit über den Aufbau eines Virtual Privat Network in Site-to-Site-Architektur unter Verwendung des IPSec-Protokolls und den Einsatz von Windows 2000 Server sowie die Implementierung von Firewallkonzepten findet man auf meiner Internetseite:

<http://www.eno-vaso.de/project.htm>

Dort wird das IPSec-Protokoll etwas ausführlicher behandelt.

2. Mitteln

Für den Versuch zur Verfügung gestellte Mitteln

Software:

OS Linux SuSE 8.2 für die PCs

IOS Cisco Ver. 6.3

FreeS/Wan

Hardware:

Für den Server-Betrieb

2 x PCs 200 MHz CPU, 128 MB RAM, 2x 3Com Netzwerkkarten BTX 10/100 Mbps

2 x PCs 500 MHz CPU, 128 MB RAM, 2x 3Com Netzwerkkarten BTX 10/100 Mbps

2 x PCs 1.6 GHz CPU, 512 MB RAM, 2x 3Com Netzwerkkarten BTX 10/100 Mbps

1 x CISCO PIX-506E, 300 MHz CPU, 32 MB RAM

1 x PC 667 MHz CPU, 128 MB RAM, 2x 3Com Netzwerkkarten BTX 10/100 Mbps

Für den Client-Betrieb

2 x PCs 1.6 GHz CPU, 256 MB RAM, 2x 3Com Netzwerkkarten BTX 10/100 Mbps

3. VPN kurz erklärt

Was ist ein 'Virtual Private Network'?

Ein Virtual Private Network ist ein Kommunikationsnetzwerk, zu dem der Zugang kontrolliert ist und in dem Verbindungen nur zwischen Partnern einer geschlossenen, definierten Gruppe möglich sind. Ein VPN verbindet lokale Netze (Intranet) oder einzelne Rechner über öffentliche Netze (z. B. Internet) miteinander.

Der Ausdruck "privat" bedeutet, dass bei der Verbindung eine Sicherheit geschaffen wird, die mit einem geschützten lokalen Netz vergleichbar ist. Obwohl die Rechner räumlich voneinander getrennt und nur über ein unsicheres Netz miteinander verbunden sind, ist eine virtuelle private Verbindung, vergleichbar mit einem lokalen Netz, entstanden.

4. Das IPSec-Protokoll

Das Internet Security Protokoll (IPSec), entwickelt und verwaltet von der IETF- Internet Engineering Task Force, definiert keine speziellen Verschlüsselungsalgorithmen oder Schlüsselaustauschverfahren, sondern liefert den Rahmen für eine modulare Sicherheitsstruktur. Die grundlegende Idee besteht darin, jedes einzelne Datenpaket vor Verfälschung zu schützen und / oder zu verschlüsseln.

Folgende Aufgaben müssen bewerkstelligt werden:

- Authentifikation der Gesprächspartner (Ich muss definitiv wissen, wer mein Kommunikationspartner ist!)
- Integrität der Informationen (Sicherheit, dass die Informationen wirklich von meinem Kommunikationspartner stammen)
- Verschlüsselung der Informationen (Niemand darf die Informationen mitlesen)
- Maßnahmen gegen Replay-Angriffe (Es muss unmöglich sein, die Informationen auf ihrem Weg zu manipulieren)
- Schlüssel Management

4.1.1 Security Associations (SA)

Security Associations sind ein fundamentaler Bestandteil der IP-Sicherheitsarchitektur. Eine SA beschreibt eine unidirektionale "Verbindung" bezüglich ihrer Sicherheitseigenschaften. Sie stellt eine Art Vertrag über einzuhaltende Sicherheitsfunktionen für den Datenverkehr dieser Verbindung dar. Hier wird also zum Beispiel festgelegt, welches Authentisierungs- und Verschlüsselungsverfahren zur Anwendung kommen soll. Das IPSec-geschützte Paket trägt keine direkte Information in sich, welches Verschlüsselungsverfahren mit welchem Schlüssel gerade eingesetzt wird. Auf diese, zur Paketbearbeitung jedoch zwingend notwendige Information zeigt der SPI (Security Parameter Index) im Header. Der SPI ist Bestandteil einer Security Association. Ein weiterer Bestandteil ist das Protokoll unter Verwendung von sogenannten Transport- oder Tunnelmodi. Als Protokolle werden AH (Authentication Header), ESP (Encapsulating Security Payload) und IKE (Internet Key Exchange) verwendet. Im Folgenden soll nun auf die verschiedenen Modi und Protokolle ein wenig näher eingegangen werden.

4.1.2 Transportmodus

In diesem Modus bleibt der ursprüngliche IP-Header erhalten. Je nachdem, ob AH oder ESP angewendet wird, sind die Daten entweder nur authentisiert oder authentisiert und verschlüsselt. Der Vorteil ist ein günstiges Verhältnis von Nutzdaten zu IPSec-Header. Dies bedeutet weniger Rechenaufwand und kann z. B. bei Echtzeit-Anwendungen von Bedeutung sein.

4.1.3 Tunnelmodus

Hier findet das sogenannte Tunneling Anwendung, d. h. das zu übertragende Datenpaket wird komplett eingepackt und mit einem neuen IP-Header versehen. Zusätzlich ist eine Verschlüsselung mit ESP möglich.

Vorteil: die Quell- und Zieladressen sind nicht erkennbar, d. h. die Identität der Kommunikationspartner bleibt anonym. Ein weiterer großer Vorteil ist das Senden von Datenpaketen durch ein öffentliches TCP/IP-Netz bei Benutzung privater nicht routingfähiger IP-Adressen, da diese sich in den Nutzdaten des neuen Datenpakets befinden. Somit kann man also mit nicht routingfähigen Adressen, Datenpakete durch ein öffentliches TCP/IP-Netz senden.

4.1.4 AH (Authentication Header)

Mit AH wird (verbindungslos) die Integrität der Daten gesichert und die Herkunft authentisiert. Sowohl die Daten des Payloads als auch Teile des Headers werden gesichert. AH im Tunnel Mode sichert das gesamte getunnelte Paket sowie Headerdaten des IP-Tunnels. AH im Transport Mode sichert nur den Payload sowie Teile des Headers.

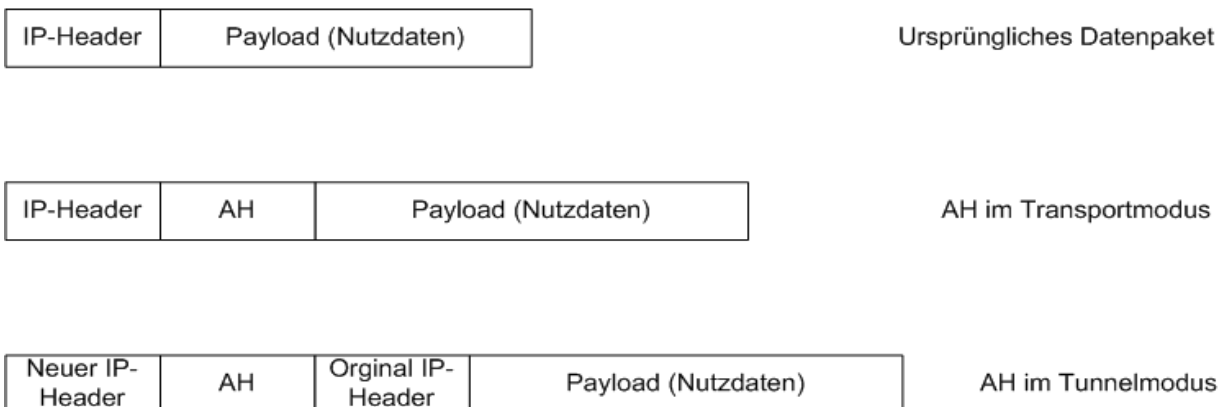
Erreicht wird dies mit dem Erzeugen eines sogenannten Hashwertes. Er stellt eine Art "Quersumme" der Daten dar. Diese "Quersumme" ist je nach Hashverfahren ein Ergebnis fester Länge (z. B. 160 Bit). Dabei spielt es keine Rolle, welche Größe die Daten haben. Dieses Hashergebnis ist vergleichbar mit einem Fingerabdruck. Werden nun die Daten verändert und wiederum ein Hashwert gebildet, wird ein anderes Hashergebnis die Folge sein. Dies ist ein Zeichen dafür, dass die Daten verändert wurden.

Es stehen zwei Hashalgorithmen zur Verfügung:

HMAC-MD5, erzeugt einen Hashwert mit 128 Bit Länge

HMAC-SHA, erzeugt einen Hashwert mit 160 Bit Länge

Darstellung von AH im Transport- und Tunnelmodus:



Aufbau des Authentication Header:

Next Header	Payload Length	Reserved
Security Parameter Index (SPI)		
Sequence Number Field		
Authentication Data (variable)		

Next Header:	Typ des Payloads (UDP/TCP)
Payload Length:	Länge des AH-Headers
Reserved:	Für zukünftige Anwendungen reserviert
Security Parameter Index SPI:	Zeiger auf die Security Association (SA)
Sequence Number:	Schutz gegen Reply-Angriffe
Authentication Data:	Hashwert, Länge abhängig vom Hashalgorithmus

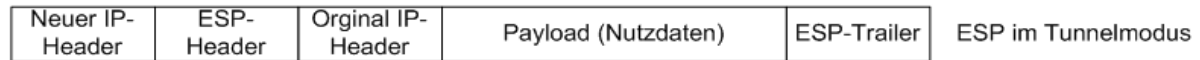
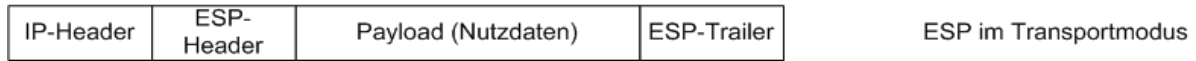
4.1.5 ESP (Encapsulating Security Payload)

ESP bietet zusätzlich zu Authentizität und Integrität eine Verschlüsselungskomponente, wodurch Vertraulichkeit der gesendeten Information erreicht wird. Damit werden die Pakete vor unberechtigtem Lesen geschützt.

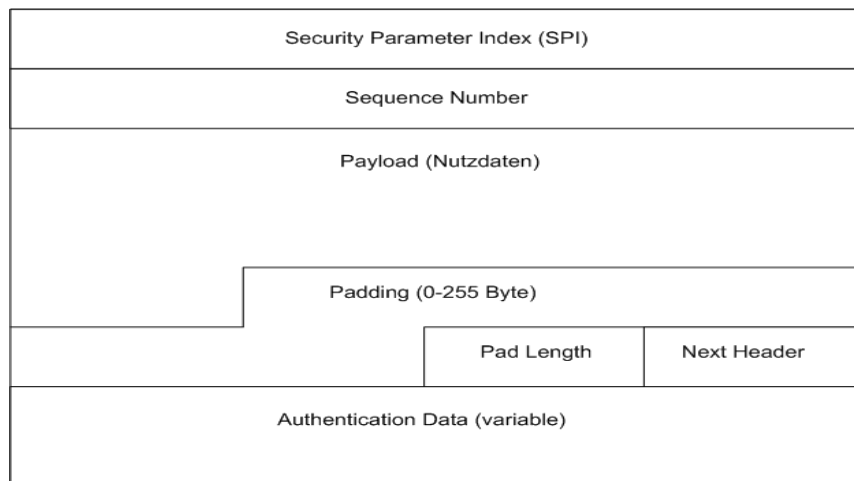
Folgende Verschlüsselungsalgorithmen können benutzt werden:

DES_CBC (RFC2405) Data Encryption Standard_Cypher Block Chaining
IDEA (RFC 2451) International Data Encryption Standard
Blowfish (RFC 2451)
3DES (RFC 2451) Triple Data Encryption Standard
CAST_128 (RFC 2451)

Darstellung von ESP im Transport- und Tunnelmodus:



Aufbau eines ESP-Datenpaketes:



- Security Parameter Index SPI: Zeiger auf die Security Association (SA)
- Sequence Number: Schutz gegen Reply-Angriffe
- Payload: Verschlüsselte Nutzdaten
- Padding: Füll-Bytes, um eine definierte Länge zu erreichen (Abhängig vom verwendeten Verschlüsselungsalgorithmus)
- Pad Length: Länge des Padding Feldes
- Next Header: Typ des Payloads (UDP/TCP)
- Authentication Data: Hashwert, Länge abhängig vom Hashalgorithmus

4.1.6 IKE (Internet Key Exchange)

AH und ESP sind nicht die einzigen Komponenten von IPSec. Zur sicheren Kommunikation müssen die beteiligten Parteien die Schlüssel vereinbaren, die während der Kommunikation verwendet werden. Außerdem muss entschieden werden, welche Algorithmen zur Verschlüsselung und Authentifizierung benutzt werden sollen. Das IKE-Protokoll (früher ISAKMP/Oakley) ermöglicht die Authentifizierung, steuert den Schlüsselaustausch und verwaltet die Sicherheitsmaßnahmen -> es werden die schon weiter oben beschriebenen Security Associations erzeugt.

Bevor nun also eine IPSec-Verbindung etabliert ist, kümmert sich das IKE-Protokoll um all diese Dinge.

Zur Authentifizierung, also zur Sicherstellung, dass man mit dem gewünschten Partner kommuniziert, sind folgende Verfahren möglich:

Authentifikation mit Pre-Shared-Key (ein geheimer gemeinsamer Schlüssel)

Authentifikation mit Digitaler Signatur

Authentifikation mit Public-Key-Verschlüsselung

In IKE werden zwei Phasen abgearbeitet:

Phase 1 -> Main Mode oder Aggressive-Mode

Phase 2 -> Quick Mode

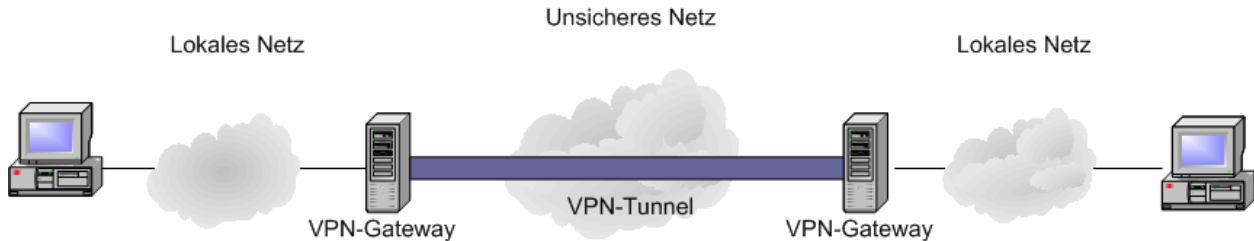
Phase 1 erzeugt eine IKE-SA, also diejenige Security Association, welche zuständig ist für die Verschlüsselung des IKE-Datenpakets selbst. Phase 2 erzeugt die IPSec-SA, die auf die Nutzdaten angewandt wird.

Schlüsselerzeugung und -austausch spielt eine entscheidende Rolle. Je länger ein Schlüssel gültig ist, desto größer ist die Gefahr, dass ein Schlüssel geknackt wird. Dieser Gefahr begegnet man mit dem Konzept der perfect forward secrecy. Dabei werden die Schlüssel sehr häufig gewechselt.

Wichtiger Bestandteil des IKE ist das Diffie-Hellman-Verfahren.

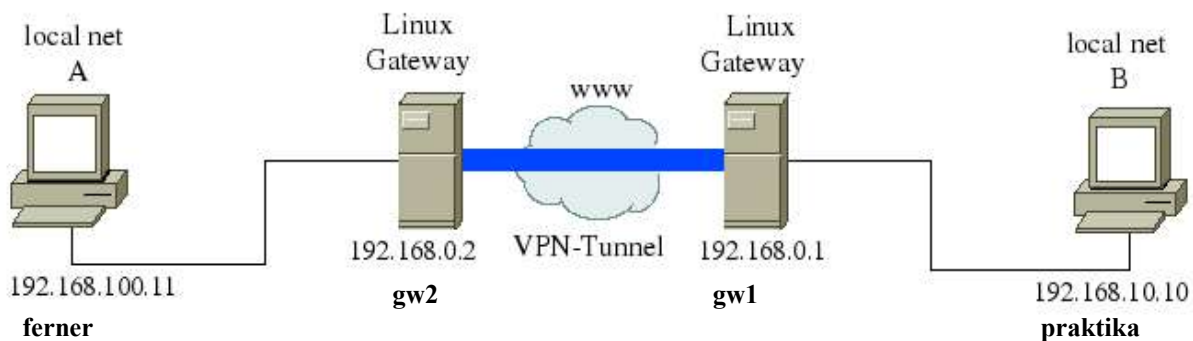
5. Der Versuchsaufbau im Site-to-Site VPN Architektur

Hier sind ganze Intranets über VPN-Gateways miteinander verbunden. Die Daten werden nur zwischen den Gateways verschlüsselt transportiert. Auf den Rechnern im LAN muss keine spezielle Software installiert sein, für sie ist der Vorgang völlig transparent.



5.1 Site-to-Site VPN zwischen Linux-Gateways

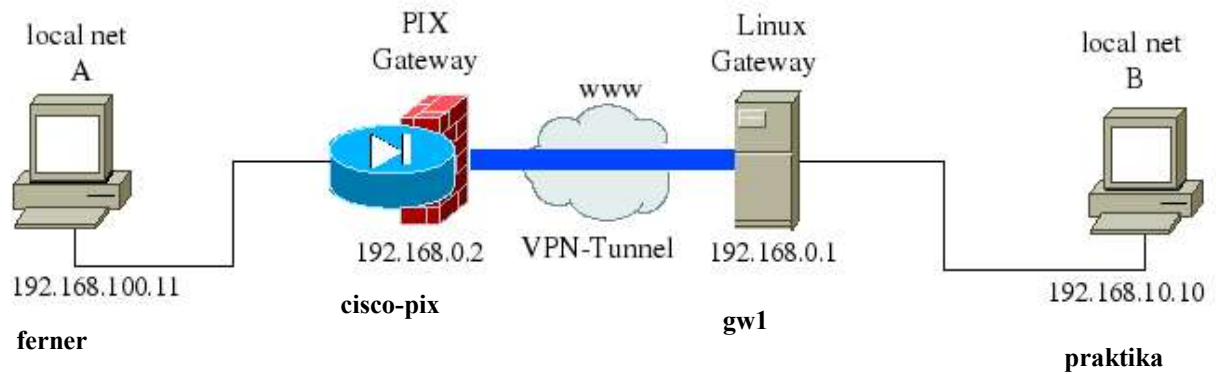
In dieser Arbeit werden die VPN-Gateways ständig durch Rechner mit unterschiedlicher Leistung ausgetauscht um die verschlüsselte Leistung zu messen, wobei die Site-to-Site Architektur erhalten bleibt. Die Tatsache, dass die IP-Adressen von Gateways nicht wirklich den im Internet routbaren IP-Adressen entsprechen, sondern aus dem privaten Bereich der C-Netzklasse ausgewählt wurden, hat einfach damit zu tun, dass zu Simulationszwecken Crossover-Kabel verwendet wurde. Im wirklichen Leben sollten die IP-Adressen angepasst werden. Beispielsweise an Stelle von 192.168.0.1 sollte eine gültige IP-Adresse wie 217.212.67.1 angegeben werden.



In diesem Projekt bekommt das Linux-Gateway mit der IP-Adresse 192.168.0.1 den Hostnamen „**gw1**“ und das mit der IP-Adresse 192.168.0.2 den Hostnamen „**gw2**“. Der Host mit der IP-Adresse 192.168.10.10 trägt den Namen „**praktika**“ und der Host mit der IP-Adresse 192.168.100.11 den Namen „**ferner**“.

5.2 Site-to-Site VPN zwischen PIX- und Linux-Gateway

Die unten aufgeführte Grafik zeigt, wie die zwei Intranets über zwei VPN-Gateways verbunden werden, wobei eine der Gateways der PIX -Firewall von Cisco ist. Dadurch ist es hier möglich, das Lokalnetz A für andere Internetverbindungen zusätzlich durch den Firewall zu schützen.



6. Installation und Konfiguration von FreeS/Wan

6.1 Linux FreeS/WAN

FreeS/WAN stellt eine Implementation des IPSec-Standards unter Linux dar und erlaubt somit VPN-Funktionalität unter diesem Betriebssystem. Dieses Paket unterliegt der GNU General Public License und ist daher kostenlos zu verwenden. Hier sollen nun kurz die Bestandteile von FreeS/WAN vorgestellt werden:

KLIPS (Kernel IP SEC Support)

Zuständig für:

- Verschlüsselung und Paketauthentifikation
- Erstellen der ESP- und AH-Header ausgehender Pakete
- Interpretation der Header eingehender Pakete
- Prüfung aller Nicht-IPSec-Pakete

Pluto:

- beinhaltet IKE und ist somit für die Security Associations zuständig

Authentifizierungsmethoden:

- Authentifikation mit Pre-Shared-Key
- Authentifikation mit Public-Key-Verschlüsselung basierend auf RSA-Algorithmus
- Opportunistic Encryption via secure DNS (RSA basierend)
- X.509-Zertifikate (Patch)

Verschlüsselungsalgorithmen:

- 3DES

Datenintegrität:

- HMAC-MD5
- HMAC-SHA1

6.2 Installation von FreeS/Wan

Bei der Linux SuSE 8.2 Version ist das Paket freeswan-1.99_0.9.23-20.i586.rpm bereits enthalten. Es unterstützt allerdings kein CA-Authentifikation (Certification Authority). Ist eine Authentifikation über x.509-Zertifikate erwünscht, dann sollte der patch x509patch-x.y (in unserem Fall x509patch-0.9.37-freeswan-1.99.tar.gz) eingespielt und anschließend die Kernelkompilierung durchgeführt werden.

```
praktika:/ # rpm -ihv freeswan-1.99_0.9.23-20.i586.rpm
```

Ist die Installation erfolgreich, erhält man die zwei wichtigsten Konfigurationsdateien

```
/etc/ipsec.conf  
/etc/ipsec.secret
```

sowie die Ordner

```
/etc/ipsec.d/  
/etc/ipsec.d/cacerts/  
/etc/ipsec.d/crls/  
/etc/ipsec.d/private/
```

Unterhalb dieses Ordners werden die Zertifikate abgelegt. Weitere Einzelheiten über Installation und Zertifikate findet man unterhalb `/usr/share/doc/packages/freeswan/`.

6.3 Konfiguration von FreeS/Wan

Die Konfiguration erfolgt hauptsächlich über zwei Dateien:

```
/etc/ipsec.conf  
/etc/ipsec.secrets
```

6.3.1 Konfiguration von `/etc/ipsec.conf`

Die Datei `/etc/ipsec.conf` dient dazu, Verbindungsvereinbarungen zwischen den Kommunikationspartnern festzulegen. Sie ist in mehrere Abschnitte aufgeteilt. Im ersten Abschnitt wird FreeS/WAN allgemein konfiguriert. Im zweiten Abschnitt werden Default-Werte, die für jede Verbindung gelten sollen, gesetzt. Nachfolgend werden die einzelnen Verbindungen konfiguriert. Jede Verbindung bildet einen neuen Abschnitt in dieser Datei. Die grundsätzliche Struktur der Datei ist vorhanden. Die Eintragungen müssen, wie dargestellt, hinzugefügt werden:

```
# /etc/ipsec.conf - FreeS/WAN IPsec configuration file  
#  
# More elaborate and more varied sample configurations can be found  
# in FreeS/WAN's doc/examples file, and in the HTML documentation.  
  
# basic configuration  
config setup  
    # THIS SETTING MUST BE CORRECT or almost nothing will work;  
    # %defaultroute is okay for most simple cases.  
    interfaces="ipsec0=eth0"  
    # Debug-logging controls: "none" for (almost) none, "all" for lots.  
    klipsdebug=none  
    plutodebug=none  
    # Use auto= parameters in conn descriptions to control startup actions.  
    plutoload=%search  
    plutostart=%search  
    # Close down old connection when new one using same ID shows up.  
    uniqueids=yes  
    # Enable NAT-Traversal  
    #nat_traversal=yes  
  
# defaults for subsequent connection descriptions  
# (these defaults will soon go away)  
conn %default  
    keyingtries=0  
    disablearrivalcheck=no  
    authby=rsasig  
    lefttrsasigkey=%dnsondemand  
    righttrsasigkey=%dnsondemand  
  
# connection description for opportunistic encryption  
# (requires KEY record in your DNS reverse map; see doc/opportunism.howto)  
conn me-to-anyone  
    left=%defaultroute
```

```

right=%opportunistic
keylife=1h
rekey=no
# for initiator only OE, uncomment and uncomment this
# after putting your key in your forward map
#leftid=@myhostname.example.com
# uncomment this next line to enable it
#auto=route

# VPN connection gw1 <=> gw2 (beide linux-gateways)
conn gw1-gw2
# Left security gateway, subnet behind it, next hop toward right.
left=192.168.0.1
leftsubnet=192.168.10.0/24
leftrsasigkey=0sAQOaw68mLcJBWMKqWWuTukfvFZI3zmoeRkanPVMXqPjPUAPgplLu34Ez3bXEf6DsfpuhWksHSPmjbqULX+fc
9rJRCYJXnXgpRnKBkvvORj2eu5HbmwzewG7XoaxCY6SZRwNAfHrts0en8cclhSCuNYF5VcG68WXeZGUDurmMDQJEGsTY/UyRqjRG
/loYERE5qAijAJqcX8LRMAWbndpI3ufCWFfp1jA24UzU5rw63LfwNQn6Md+zm4OujHTF8T3XSo6kaNEAeC74qTodfirtUn921CFLV
GpPuxHqM9lG50p6fF2Z7asaEHkEjPFZ3feNj/X+mYMMzstWUSS2li+eF3+uf
#leftnexthop=
# Right security gateway, subnet behind it, next hop toward left.
right=192.168.0.2
rightsubnet=192.168.100.0/24
rightrsasigkey=0sAQN2XHBb9UfLaxvgEFY/LaCVH/8CyI6YkPukGVTRiUhy9ejdAXB1VoRoLHbaAW12Wj2MwUls+S5uhJaB3
r1Q1pZH/wB2jIAZ3yNJAu60V3BNNFCVoJpK62w3vz1joJiMMSeWf6oT2qK2vFNQmTVgois4a7k/Dq2OBIfwA6LvlUbgQaB2unxbu
3tpQohzSwzMQGBSLRkqIcmxCEzqsxKPKPFKJ2R54CTmh41Z+X81qC1P8703QLGgXxYeBMVP7vXPPAlwE/wdeUu+gPGpcXpxE9ts0
bzDRpUkMsOnwf0zxnalHE/+1r4zP5cpW64NzCtzs2AGItN8WX+wGEGbwKrLM9
#rightnexthop=
# To authorize this connection, but not actually start it,at startup,
# uncomment this.
auto=add
#auto=route
# Select allowed cipher/hash algorithms
#ike=aes128-sha-modp1536,aes128-sha-modp1024,aes128-md5-modp1536,aes128-md5-
modp1024,3des-sha-modp1536,3des-sha-modp1024,3des-md5-modp1536,3des-md5-modp1024
esp=3des-md5-96
keylife=30m
type=tunnel

# VPN zwischen cisco-pix <=> gw1
conn cisco-pix
# Left security gateway, subnet behind it, next hop toward right.
keyexchange=ike
type=tunnel
auth=esp
# authby=secret bei ältere Freeswan-Versionen steht hier „psk“ anstatt „secret“
authby=secret
left=192.168.0.1
leftsubnet=192.168.10.0/24
#leftnexthop=
# Right security gateway, subnet behind it, next hop toward left.
right=192.168.0.2
rightsubnet=192.168.100.0/24
#rightnexthop=
# To authorize this connection, but not actually start it,at startup,
# uncomment this.
auto=add
#auto=route
# Select allowed cipher/hash algorithms
esp=3des
keylife=30m
# Ende der Konfiguration

```

Diese Konfiguration, die in unserem Beispiel der Gateway „gw1“ entspricht, muss bei dem Gateway „gw2“ wiederholt werden. Im Prinzip kann sogar die Konfigurationsdatei von „gw1“ auf den „gw2“ kopiert werden. Was allerdings bei „gw1“ mit „left.“ bezeichnet wird, entspricht bei „gw2“ nun dem Eintrag „right.“.

Weiter unten sind auch andere mögliche relevante Einträge im /etc/ipsec.conf und deren Erklärung.

interfaces

das Interface das IPSec verwenden soll.

forwardcontrol

soll nach dem IPSec-Start setup IP forwarding gestartet und vor dem Ende gestoppt werden?

([default] no, yes)

syslog

syslog-Einstellungen für startup/shutdown log messages. ([default] daemon.error)

klipsdebug

KLIPS Debugging Output. ([default] none, all)

plutodebug

Pluto Debugging Output. ([default] none, all)

dumpdir

in welchem Directory sollen dump cores erlaubt werden?

manualstart

welche Manuelle-Verbindung soll beim Start gestartet werden?

pluto

soll Pluto gestartet werden? (no, yes)

plutoload

welche Verbindung soll beim Start in die Pluto DB geladen werden?

plutostart

welche Verbindung soll beim Start initialisiert werden?

plutowait

soll beim Start auf jede plutostart-Initialisierung gewartet werden? ([default] yes, no)

Relevante Einträge für die automatische und die manuelle Schlüsselverteilung:

type

bezeichnet den Typ der Verbindung (tunnel, transport)

auto

welche Operation automatisch beim Start von IPsec ausgeführt werden soll (add, start, ignore)

left

gibt die IP-Adresse des linken Partners an (aaa.bbb.ccc.ddd)

leftsubnet

hier steht das private Subnetz des linken Partners (network/netmask)

leftnexthop

next-hop Gateway IP-Adresse für den linken Partner ([default] right)

leftupdown

welches updown Skript ausgeführt werden soll, wenn der Status der Verbindung sich ändert ([default] ipsec_updown).

leftfirewall

setzt der linke Partner Forwarding-Firewalling (beinhaltet Masquerading) ein? (yes, no)

Relevante Einträge für die automatische Schlüsselverteilung:

keyexchange

gibt die Methode des Schlüsselaustausches an (ike)

auth

Authentifikation mittels ESP-Verschlüsselung oder separat mithilfe des AH-Protokolls (esp, ah)

authby

wie sich die Sicherheits-Gateways authentifizieren sollen. Möglich ist hier *shared secrets* oder *rsasig* für eine RSA digital signatures ([default] secret, rsasig)

leftid

wie der linke Partner zur Authentifikation identifiziert werden soll. ([default] left, aaa.bbb.ccc.ddd)

leftrsasigkey

der öffentliche Schlüssel für RSA signature authentication (im RFC 2537 Format)

pfs

ob Perfect Forward Secrecy für die Verbindung erwünscht ist. Mit PFS ist bei einem Verlust des Geheimnisses die Sicherheit der Verbindung nicht gefährdet. (yes, no)

keylife

wie lange die SAs und die Schlüssel verwendet werden sollen. ([default] 8.0h, maximum 24h)

rekeymargin

wie lange Pluto versuchen soll, eine (gewechselte) SAs Verbindung aufzubauen, bevor abgebrochen werden soll. ([default] 9m, (s. keylife))

keyingtries

wieviele Versuche gemacht werden sollen, um eine Verbindung aufzubauen. ([default] 3, (ganze Zahl))

ikelifetime

wie lange die Verbindung zum anderen Key-Management Daemon höchstens bestehen soll, bevor sie neu aufgebaut werden soll.

Relevante Einträge für die manuelle Schlüsselverteilung:

spi

SPI Nummer für die Verbindung. (0xhex)

spibas

Basis Nummer für SPIs. (0xhex0)

esp

eingesetzter ESP-Algorithmus. ([default] (not to use), 3des-md5-96)

espenchkey

ESP Encryption Key

espauthkey

ESP Authentication Key

espreplay_window

ESP-Replay-Window-Einstellungen. Nur relevant beim Einsatz von ESP Authentication. (0,...,64)

leftespsi

zu benützte SPI für die ESP SA

ah

AH Authentication Algorithmus. ([default] (not to use), hmac-md5-96)

ahkey

AH Authentication Key.

leftahspi

Security Parameter Index der AH-Security-Assoziation.

6.3.2 Konfiguration von /etc/ipsec.secrets

In der Datei **/etc/ipsec.secrets** sind die Authentifikationsinformationen, also die Schlüssel gespeichert, die für die definierten Verbindungen notwendig sind. Bei der Installation von FreeS/WAN werden automatisch Schlüssel erzeugt. Für das Pre-Shared-Secrets-Verfahren ist es notwendig, dass beide Gateways den gleichen Verbindungsschlüssel verwenden.

In unserem Beispiel findet die Authentifikation mit Hilfe der bei der Installation automatisch erzeugter RSA-Signature. Der Einfachheit halber habe ich für die Authentifizierung mit CISCO-PIX das PSK-Verfahren angewandt und einen sehr kurzen Schlüssel "SECRET" gewählt, der natürlich

nicht für die Praxis zu empfehlen ist. Maximal ist eine Länge von 256 Bit möglich.

```
# This file holds shared secrets or RSA private keys for inter-Pluto
# authentication. See ipsec_pluto(8) manpage, and HTML documentation.

192.168.0.1 192.168.0.2: PSK "SECRET"

# RSA private key for this host, authenticating it to any other host
# which knows the public part. Suitable public keys, for ipsec.conf, DNS,
# or configuration of other implementations, can be extracted conveniently
# with "ipsec showhostkey".

: RSA {
  # RSA 2048 bits gw1 Mon Nov 10 14:22:27 2003
  # for signatures only, UNSAFE FOR ENCRYPTION
  #pubkey=0sAQOaw68mLcJBWMKqWWuTukfvFZI3zmoeRkanPVMXqPjPUAPgplLu34Ez3bXEf6DsfpuhWksHSPmjbqULX+
fc9rJRCYJXnXgpRnKBkvORj2eu5HbmwzewG7XoaxCY6SZRwNAfHrts0en8cclhSCuNYF5VcG68WXeZGUDurmMDQJEGsTY/UyRqj
RG/loYERE5qAIjAjqcX8LRMAWBndpI3ufCWFfpljA24UzU5rw63LfwNQN6Md+zm4OujHTF8T3XSo6kaNEAeC74qTodfrtUn921CF
LVGpPuxHqM9lG50p6fF2Z7asaEHkEjPFZ3feNj/X+mYMMzstWUSS2li+eF3+uf
  #IN KEY 0x4200 4 1
AQOaw68mLcJBWMKqWWuTukfvFZI3zmoeRkanPVMXqPjPUAPgplLu34Ez3bXEf6DsfpuhWksHSPmjbqULX+fc9rJRCYJXnXgpRnKB
kvvORj2eu5HbmwzewG7XoaxCY6SZRwNAfHrts0en8cclhSCuNYF5VcG68WXeZGUDurmMDQJEGsTY/UyRqjRG/loYERE5qAIjAjqc
X8LRMAWBndpI3ufCWFfpljA24UzU5rw63LfwNQN6Md+zm4OujHTF8T3XSo6kaNEAeC74qTodfrtUn921CFLVGpPuxHqM9lG50p6f
F2Z7asaEHkEjPFZ3feNj/X+mYMMzstWUSS2li+eF3+uf
  # (0x4200 = auth-only host-level, 4 = IPSec, 1 = RSA)
  Modulus:
0x9ac3af262dc24158c2aa596b93ba47ef159237ce6a1e4646a73d5317a8f8cf5003e0a652eedf8133ddb5c47fa0ec7e9ba1
5a4b0748f9a36ea50b5fe7dcf6b2510982579d782946728192fbce463d9ebb91db9b0cdec06ed7a1ac4263a4994703407c7a
edb347a7f1c7258520ae35817955c1baf165de646503bab98c0d02441ac4d8fd4611aa3446fe5398111139a80223009a9c5f
c2d13005819dda48dee7c25857e9963036e14cd4e6bc3adcb7d69d037a31dfb39b83ae8c74c5f13dd74a8ea468d100782ef8
a93a1d7ebb549fddb50852d51a93eec47a8cf651b9d29e9f17667b6ac6841e41233c56777de363fd7fa660c333b13594492d
a58be785dfeb9f
  PublicExponent: 0x03
  # everything after this point is secret
  PrivateExponent:
0x19cb47dbb24b0ae42071b991edf4615283985ea267050bb67134e32e9c2977e2ab501bb87d2540334f9e4b6a9ad21519f0
39b72be17ef09270d73aa6a4d3c862d6eb0e9a3eb18bbdc04329f7b65f9a74984f448225201279459cb5bb46198bd5e014bf
279de146a84dbb96301d08eae98e4af47d90fa6610d5f47442022b0b5988331f5fb79b1b59a04d3d65a9b32016d97e6275b2
51f01192db0537830d9031d28d3e23b585eb4b6d2cc811ad40fcbfffd1244af8b24d42624efcc694cf6ad9157fc21377e76
906172a1f6df3587047203398355d15adb61e1a44d80980dfb50efb61f0db89f1e9fbbb1bd4cb29a445930ac25255c45424c
dd45006382587b
  Primel:
0xeb31bfc2b933a7b1cfc7e2441ba513ea24a5d4fa1a12e2e4f6440014da346b5b61e06a4ee05ce2e2e2f7eacb1588528b84
512d4677a9c2203c9ba1dea4c6fe9d9fd352508103091cd2de9d904bb8f0207c56405057daac05d8ac053678004bef828061
f50f5dc7bc267efe1bbc4743d8b6f428e6bb835eb617cc0613f57b5caf
  Prime2:
0xa87481450ad3e87b6cbc456afb61d38ee56476e017c44de19e1b7e785c591b3c07280a710eb67aa562e22105b7a985497f
3a96fd1eaae4956afb8548214283df9c8595e8aa35282a1e8f1561cda5c379dfe93bb6392bf750690072b2570dff43fe5cc4
17226cbeaf57714ef25aed1c0cd848916e1588960548a9e7d13b567c11
  Exponent1:
0x9ccbd52c7b77c521352fec2d67c3629c186e8dfc1161ec98a42d55633c22f23cebeaf189eae89741eca547320e5ae1b258
361e2efa712c157dbd16946dd9ff13bfe236e056020613373f13b587d0a015a83980358fe71d593b1d58cefaaadd4a570041
4e0a3e852819afebd282f829079f81b447d023f240fdd5962a3a7931f
  Exponent2:
0x704dab835c8d45a79dd2d8f1fcebe25f43984f400fd833ebbecfefa83b677d5a1ab1a0b479a718ec96c0ae7a71038654
dlb9fe147498639ca7ae30162c57ea68590e9b1c23701c145f639689192cfbea9b7d24261d4f8af0aaf7218f5eaa2d543dd8
0f6c487f1f8fa0df4c3c9e12b33adb0b9eb905b958db1be0d239a80b
  Coefficient:
0xe4249170527f7e7db0454d70394e7c034bf431f6641d87e7ef76b24253df9c57d545b8f933ce72ce39fe1dca487a92db61
a2df4dab099e686df8d1f1913295e59b46fec86acb2242866aadcc3f24026911fa8e9678c24d18a3ce05b1baf936cc7460b
766ea5313529041fc83c30e8e4d87ec6aaa131de78f8a9261b7c9159a4
}
# do not change the indenting of that "}"
# ENDE
```

Der einzige Eintrag der manuell hinzugefügt wird, ist der PSK-Eintrag, der nur für die VPN-Verbindung zwischen „gw1“ und Cisco-PIX benötigt wird. Der folgende Befehl zeigt den Publik-Schlüssel von „gw1“ an.

```
gw1:/etc # ipsec showhostkey
; RSA 2048 bits gw1 Mon Nov 10 14:22:27 2003
gw1. IN KEY 0x4200 4 1
```

```
AQOaw68mLcJBWmkqWwuTukfvFZI3zmoeRkanPVMXqPjPUAPgplLu34Ez3bXEf6DsfpuhWksHSPmjbqULX+fc9rJRCYJXnXgpRnKB
kvvORj2eu5HbmwzewG7XoaxCY6SZRwNAfHrts0en8cclhSCuNYF5VcG68WXeZGUDurmMDQJEGsTY/UYRqjRG/1OYERE5qAIjAJqc
X8LRMAWBndpI3ufCWFfpljA24UzU5rw63LfwNQN6Md+zm4OujHTF8T3XSo6kaNEAeC74qTodfrtUn921CFLVGpPuxHqM91G50p6f
F2Z7asaEHkEjPFZ3feNj/X+mYMMzstWUSS2li+eF3+uf
```

Der folgende Befehl listet alle bekannten Public-Schlüssel auf.

```
gw1:~ # ipsec auto --listpubkeys
000
000 List of Public Keys:
000
000 Nov 20 08:58:39 2003, 2048 RSA Key AQN2XHBb9, until --:--:-- ---- ok (expires never)
000      ID_IPV4_ADDR '192.168.0.2'
000 Nov 20 08:58:39 2003, 2048 RSA Key AQOaw68mL, until --:--:-- ---- ok (expires never)
000      ID_IPV4_ADDR '192.168.0.1'
```

Als nächster Schritt ist unbedingt die Dateirechte von ipsec.secret zu überprüfen.
Sie müssen auf rw- --- --- gesetzt sein, damit nur root sie lesen und editieren kann.

Achtung!

Wenn es jemandem Fremden möglich ist diese Datei zu lesen, muss er nur noch die IP-Adresse des eigentlichen Kommunikationspartners vortäuschen. Er bekommt dann kompletten Zugriff auf das zu schützende und verschlüsselte Netzwerk. Dies muss unbedingt vermieden werden.

Damit wäre FreeS/WAN für meine VPN-Verbindung konfiguriert.

7. Die Konfiguration von Cisco-PIX

7.1 Zugriff auf Cisco-PIX über das Terminalprogramm Kermit

Zum Aufbau einer VPN-Verbindung mit Hilfe der IPSec-Protokoll sollte der PIX entsprechend konfiguriert werden. In unserem Fall werde ich den Cisco PIX-506E IOS Ver. 6.3 als VPN-Gateway verwenden. Die Konfiguration kann über einem Terminal-, SSH- oder aber auch eine Telnet-Verbindung erfolgen. Die letzten zwei sollten allerdings vorher schon konfiguriert worden sein, um einen Zugriff auf Cisco-PIX zu ermöglichen.

In der Regel erfolgt der erstmalige Zugriff über einen Terminal, da der PIX keine IP-Adresse besitzt. Dafür braucht man einen Rechner, ein Terminalkabel, einen freien COM-Port auf dem Rechner und ein Terminalprogramm. Linux bietet eine Konsolenanwendung namens Kermit. Diese Anwendung erfüllt die Aufgabe eines Terminalprogramms.

Unten habe ich ein Verbindungsbeispiel mittels Kermit über den COM-Port 1 (/dev/ttyS1) eingefügt.

```
eno@praktika:~> kermit
C-Kermit 8.0.206, 24 Oct 2002, for Linux
  Copyright (C) 1985, 2002,
  Trustees of Columbia University in the City of New York.
Type ? or HELP for help.
Linux Kermit>set line /dev/ttyS1
Linux Kermit>set carrier-watch off
Linux Kermit>connect
Connecting to /dev/ttyS1, speed 9600
  Escape character: Ctrl-\ (ASCII 28, FS): enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
-----
```

An diese Stelle einfach „ENTER“ drücken und schon ist man mit Cisco-PIX verbunden. Der nächste Schritt ist nun in den privilegierten Modus einzuloggen.

```
cisco-pix>
cisco-pix> enable
Password:*****
cisco-pix#
cisco-pix# conf t
cisco-pix(config)#
```

7.2 Die Grundkonfiguration von PIX

Dem PIX wird nun eine Host- und Domainname vergeben. Anschließend generiert man einen RSA-Schlüssel (für spätere ssh-Zugriffe unbedingt erforderlich) und mit „show ca mypub rsa“ lässt man ihn zeigen.

```
cisco-pix(config)#hostname cisco-pix
cisco-pix(config)#domain-name praktika
cisco-pix(config)#ca generate rsa key 2048
cisco-pix(config)#sh ca mypub rsa
% Key pair was generated at: 02:34:37 UTC Nov 17 2003
Key name: cisco-pix.praktika
Usage: General Purpose Key
Key Data:
 30820122 300d0609 2a864886 f70d0101 01050003 82010f00 3082010a 02820101
```

```

009fb7cf 79ad225f 19d79cd8 689e7881 40439f98 12c6d959 25536bdf e376c7a6
9ae04b82 872e98bc 04dc554e 44fff2f51 b163bc23 e7be5e9d de8c838d 7dc8282e
65039556 b7ec8bdc 36bd4b55 05e4be28 12f79bc9 c24ef8c3 495b15cd 912ed0ed
9083cc29 16677b34 6e5bc792 32114744 48f313df d3138fea fb3f520e f506bf47
dc3b51c6 27a5901c 318d2260 29fc1cc4 6b9acc0c 4b695ba8 476f9097 e0a4d5d8
5b405c64 1b6dca96 352f9241 1800c74d d9a385f6 a5dd317e 342ec348 e2faf375
1d507276 a4d806c6 4c8c66d4 cadcc480 2ce613ae a9bd12e9 882d354c a9e19636
4561d569 002e375e f69f2ccd c5a7dff6 ae215036 5e065814 c70be4bd 10d26417
5b020301 0001

```

Nun sollte man damit anfangen dem PIX IP-Adressen für outside und inside zu vergeben sowie die notwendigen Routen einzurichten.

```

cisco-pix(config)#ip address outside 192.168.0.2 255.255.255.0
cisco-pix(config)#ip address inside 192.168.100.2 255.255.255.0
cisco-pix(config)#route outside 0.0.0.0 0.0.0.0 192.168.0.1 1
cisco-pix(config)#route outside 192.168.10.0 255.255.255.0 192.168.0.1 1

```

7.2.1 Die wichtigsten Einstellungen für eine VPN-Verbindung

Und nun auf einem Blick die wichtigsten Einträge, die für den Schlüsselaustausch und die Verschlüsselung notwendig sind.

```

isakmp enable outside
isakmp key SECRET address 192.168.0.1 netmask 255.255.255.255
isakmp policy 10 authentication pre-share
isakmp policy 10 encryption 3des
isakmp policy 10 hash md5
isakmp policy 10 group 2
isakmp policy 10 lifetime 14400
crypto ipsec transform-set my_vpn esp-3des esp-sha-hmac
crypto dynamic-map vpn-Dyn 10 set peer 192.168.0.2 192.168.0.1
crypto dynamic-map vpn-Dyn 10 set transform-set my_vpn
crypto dynamic-map vpn-Dyn 10 set security-association lifetime seconds 3600 kilobytes 400000
crypto map vpn-Map 10 ipsec-isakmp
crypto map vpn-Map 10 match address vpn_acl
crypto map vpn-Map 10 set peer 192.168.0.1
crypto map vpn-Map 10 set transform-set my_vpn
crypto map vpn-Map 10 set security-association lifetime seconds 3600 kilobytes 400000
crypto map vpn-Map 20 ipsec-isakmp dynamic vpn-Dyn
crypto map vpn-Map interface outside
access-list vpn_acl permit ip 192.168.100.0 255.255.255.0 192.168.10.0 255.255.255.0
sysopt connection permit-ipsec
global (outside) 1 192.168.0.1 netmask 255.255.255.0
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
static (inside,outside) 192.168.100.0 192.168.100.0 netmask 255.255.255.0 0 0

```

Ein wichtiger Eintrag ist der von „isakmp identity“. Hier müsste die Identität so eingetragen werden, wie sie in der Konfigurationsdatei /etc/ipsec.secret von „gw1“ (192.168.0.1 **192.168.0.2**: PSK "SECRET") eingegeben ist. Anstelle von IP-Adressen könnte aber auch der Hostname stehen.

```

cisco-pix(config)#isakmp identity address 192.168.0.2

```

Weitere Einstellungen sind für den Zugriff über das Netzwerk notwendig.

```

telnet 192.168.0.0 255.255.255.0 outside
telnet 192.168.100.0 255.255.255.0 inside
telnet timeout 60

```

```
ssh 192.168.0.0 255.255.255.0 outside
ssh 192.168.100.0 255.255.255.0 inside
ssh timeout 60
ca identity pix-firewall.ba-pankow.verwalt-berlin.de 192.168.0.2:/cgi-bin
ca configure pix-firewall.ba-pankow.verwalt-berlin.de ra 1 0
```

Damit wäre Cisco-PIX für meine VPN-Verbindung konfiguriert. Um die Einstellungen zu sichern, damit beim rebooten des Systems beibehalten werden, geht man folgendermaßen vor:

7.3 Die Konfiguration sichern

Einstellung lokal sichern

```
cisco-pix(config)# write mem
Building configuration...
Cryptochecksum: d64215ae 193cf23f 382b615e f8b6900c [OK]
```

Einstellung auf entferntem tftp-server sichern

```
cisco-pix(config)#tftp-server outside 192.168.10.10 pix.conf
cisco-pix(config)write net
Building configuration...
TFTP write 'pix.conf' at 192.168.10.10 on interface 0 [OK]
```

Mit „show run“ lässt man die aktuelle Konfiguration zeigen.

```
cisco-pix(config)# sh run
: Saved
:
PIX Version 6.3(1)
interface ethernet0 auto
interface ethernet1 auto
nameif ethernet0 outside security0
nameif ethernet1 inside security100
enable password 8Ry2YjIyt7RRXU24 encrypted
passwd LH0DyfA6eIOQUwWc encrypted
hostname cisco-pix
domain-name praktika
fixup protocol ftp 21
fixup protocol h323 h225 1720
fixup protocol h323 ras 1718-1719
fixup protocol http 80
fixup protocol ils 389
fixup protocol rsh 514
fixup protocol rtsp 554
fixup protocol sip 5060
fixup protocol sip udp 5060
fixup protocol skinny 2000
fixup protocol smtp 25
fixup protocol sqlnet 1521
names
access-list vpn_acl permit ip 192.168.100.0 255.255.255.0 192.168.10.0 255.255.255.0
pager lines 37
logging console debugging
logging history debugging
mtu outside 1500
mtu inside 1500
ip address outside 192.168.0.2 255.255.255.0
ip address inside 192.168.100.2 255.255.255.0
ip audit info action alarm
ip audit attack action alarm
pdm location 192.168.10.0 255.255.255.0 outside
pdm history enable
arp timeout 14400
```

```

global (outside) 1 192.168.0.1 netmask 255.255.255.0
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
static (inside,outside) 192.168.100.0 192.168.100.0 netmask 255.255.255.0 0 0
route outside 0.0.0.0 0.0.0.0 192.168.0.1 1
route outside 192.168.10.0 255.255.255.0 192.168.0.1 1
timeout xlate 3:00:00
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 rpc 0:10:00 h225 1:00:00
timeout h323 0:05:00 mgcp 0:05:00 sip 0:30:00 sip_media 0:02:00
timeout uauth 0:05:00 absolute
aaa-server TACACS+ protocol tacacs+
aaa-server RADIUS protocol radius
aaa-server LOCAL protocol local
no snmp-server location
no snmp-server contact
snmp-server community public
no snmp-server enable traps
tftp-server outside 192.168.10.10 pix.conf
floodguard enable
sysopt connection permit-ipsec
crypto ipsec transform-set my_vpn esp-3des esp-sha-hmac
crypto dynamic-map vpn-Dyn 10 set peer 192.168.0.2 192.168.0.1
crypto dynamic-map vpn-Dyn 10 set transform-set my_vpn
crypto dynamic-map vpn-Dyn 10 set security-association lifetime seconds 3600 kilobytes 400000
crypto map vpn-Map 10 ipsec-isakmp
crypto map vpn-Map 10 match address vpn_acl
crypto map vpn-Map 10 set peer 192.168.0.1
crypto map vpn-Map 10 set transform-set my_vpn
crypto map vpn-Map 10 set security-association lifetime seconds 3600 kilobytes 400000
crypto map vpn-Map 20 ipsec-isakmp dynamic vpn-Dyn
crypto map vpn-Map interface outside
isakmp enable outside
isakmp key ***** address 192.168.0.1 netmask 255.255.255.255
isakmp identity address
isakmp policy 10 authentication pre-share
isakmp policy 10 encryption 3des
isakmp policy 10 hash md5
isakmp policy 10 group 2
isakmp policy 10 lifetime 14400
ca identity pix-firewall.ba-pankow.verwalt-berlin.de 192.168.0.2:/cgi-bin
ca configure pix-firewall.ba-pankow.verwalt-berlin.de ra 1 0
telnet 192.168.0.0 255.255.255.0 outside
telnet 192.168.100.0 255.255.255.0 inside
telnet timeout 60
ssh 192.168.0.0 255.255.255.0 outside
ssh 192.168.100.0 255.255.255.0 inside
ssh timeout 60
console timeout 0
terminal width 80
Cryptochecksum:d64215ae193cf23f382b615ef8b6900c
: end

```

Die an einem entfernten TFTP-Server gespeicherte Konfigurationsdatei „pix.conf“ dient dem Backup im Falle einer Misskonfiguration des PIX. Mit dem Befehl „conf net“ wird die Konfiguration wieder von entferntem Rechner zurückgeholt. An dieser Stelle hat CISCO eine Klasse Dienst geleistet, um einen völlig nackten PIX mit einem einzigen Schritt zu konfigurieren. Man braucht nur eine Textdatei manuell mit den richtigen Parametern zu editieren, die Textdatei auf den Wurzeln des TFTP-Servers zu speichern und dann mittels „conf net“-Befehl auf den Cisco-PIX zu laden.

Schon hat man für eine VPN-Verbindung ein fertig konfiguriertes Cisco-Gerät.

8. Starten und Testen der Verbindung

8.1 Starten und Testen der Verbindung zwischen den Linux-Gateways.

Nachdem nun alle Rechner konfiguriert sind, ist ein Verbindungsaufbau und ein Test der Verbindung möglich. Folgende Schritte sind durchzuführen:

Als erstens wird die Routing-Tabelle auf der Gateways überprüft. Hier ein Beispiel aus der Gateway „gw1“ Routing-Tabelle:

```
gw1:/# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.0.0      0.0.0.0         255.255.255.0   U        0      0      0 eth0
192.168.10.0     0.0.0.0         255.255.255.0   U        0      0      0 eth1
0.0.0.0          192.168.0.1     0.0.0.0         UG       0      0      0 eth0
```

Dieser Schritt wird auch beim „gw2“ durchgeführt (für die VPN-Verbindung Linux<==>Linux). Hier ein Mitschnitt aus der PIX-Routing-Tabelle:

```
cisco-pix(config)# sh route
  outside 0.0.0.0 0.0.0.0 192.168.0.1 1 OTHER static
  outside 192.168.0.0 255.255.255.0 192.168.0.2 1 CONNECT static
  outside 192.168.10.0 255.255.255.0 192.168.0.1 1 OTHER static
  inside 192.168.100.0 255.255.255.0 192.168.100.2 1 CONNECT static
```

Es wird nun FreeS/Wan gestartet und anschließend überprüft, ob die gewünschten Parameter korrekt installiert und gestartet sind.

```
gw1:/# ipsec setup start
ipsec_setup: Starting FreeS/WAN IPsec 1.99...
ipsec_setup: ipsec ipsec_3des ipsec_md5 ipsec_shal ipsec_aes
ipsec_setup: done
```

```
gw1:~ # ipsec verify
Checking your system to see if IPsec was installed and started correctly
Version check and ipsec on-path [OK]
Checking for KLIPS support in kernel [OK]
Checking for RSA private key (/etc/ipsec.secrets) [OK]
Checking that pluto is running [OK]
DNS checks.
Looking for forward key for gw1 [OK]
Does the machine have at least one non-private address [OK]
```

Hier wird der VPN-Verbindungsaufbau zwischen Cisco-PIX und „gw1“ (Linux-Gateway) dargestellt. Für einen VPN-Verbindungsaufbau zwischen „gw1“ und „gw2“ wird anstelle von „cisco-pix“ der Name der Verbindung vergeben so wie diese in der Konfigurationsdatei /etc/ipsec.conf beschrieben ist (in unserem Fall „gw1-gw2“).

```

gw1:~ # ipsec auto --up cisco-pix
104 "cisco-pix" #1: STATE_MAIN_I1: initiate
106 "cisco-pix" #1: STATE_MAIN_I2: sent MI2, expecting MR2
003 "cisco-pix" #1: ignoring Vendor ID payload [XAUTH]
003 "cisco-pix" #1: ignoring Vendor ID payload [Dead Peer Detection]
003 "cisco-pix" #1: ignoring Vendor ID payload [Cisco-Unity]
003 "cisco-pix" #1: ignoring Vendor ID payload [cde5c15ca8bcfc6a...]
108 "cisco-pix" #1: STATE_MAIN_I3: sent MI3, expecting MR3
004 "cisco-pix" #1: STATE_MAIN_I4: ISAKMP SA established
112 "cisco-pix" #2: STATE_QUICK_I1: initiate
003 "cisco-pix" #2: ignoring informational payload, type IPSEC_RESPONDER_LIFETIME
004 "cisco-pix" #2: STATE_QUICK_I2: sent QI2, IPsec SA established

```

Ist der Verbindungsaufbau erfolgreich, ist die zusätzliche virtuelle Schnittstelle ipsec0, abgebildet auf der IP-Adresse 192.168.0.1, hinzugekommen. Diese kann man mit dem Befehl „ifconfig“ prüfen.

```

gw1:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:10:5A:1C:1D:08
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::210:5aff:fe1c:1d08/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:125258 errors:0 dropped:0 overruns:0 frame:0
          TX packets:222974 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:15152316 (14.4 Mb)  TX bytes:291205829 (277.7 Mb)
          Interrupt:5 Base address:0xdc00

eth1      Link encap:Ethernet  HWaddr 00:50:04:63:BE:2E
          inet addr:192.168.10.1  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::250:4ff:fe63:be2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:251648 errors:0 dropped:0 overruns:0 frame:0
          TX packets:137725 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:307307868 (293.0 Mb)  TX bytes:11268605 (10.7 Mb)
          Interrupt:10 Base address:0xe400

ipsec0    Link encap:IPIP Tunnel  HWaddr
          inet addr:192.168.0.1  Mask:255.255.255.0
          UP RUNNING NOARP  MTU:16260  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:3859 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3859 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:356198 (347.8 Kb)  TX bytes:356198 (347.8 Kb)

```

Nun wird die Routing-Tabelle erneut überprüft.

```

gw1:~ # route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.100.0    192.168.0.2     255.255.255.0  UG      0      0      0 ipsec0
192.168.0.0      0.0.0.0         255.255.255.0  U        0      0      0 eth0
192.168.0.0      0.0.0.0         255.255.255.0  U        0      0      0 ipsec0
192.168.10.0     0.0.0.0         255.255.255.0  U        0      0      0 eth1
0.0.0.0          192.168.0.1     0.0.0.0        UG      0      0      0 eth0

```

Zu den Standardrouten (Netze, die direkt über die Netzwerkkarten erreichbar sind -> hier: 192.168.0.0 und 192.168.10.0) sind zwei zusätzliche Einträge mit dem ipsec0-Interface

hinzugekommen.

Ziel: 192.168.100.0 -> Dieses Netz ist jetzt ausschließlich über IPsec erreichbar.

Ziel: 192.168.0.0 -> Das Netz, in dem sich der Tunnel befindet

Hier noch zwei weitere Befehle die den Status der VPN-Verbindung anzeigen.

ipsec look

```
gw1:~ # ipsec look
gw1 Thu Nov 27 13:39:48 CET 2003
019216810000024:0:192.168.10.0/24:0 -> 192.168.100.0/24:0 => tun0x1002@192.168.0.2:0 (19)
ipsec0->eth0 mtu=16260(1443)->1500
esp0xbe35ad71@192.168.0.2 ESP_3DES_HMAC_SHA1: dir=out src=192.168.0.1 iv_bits=64bits
iv=0xaa00af3fcf40dfc8 ooowin=64 seq=19 alen=160 aklen=160 eklen=192 life(c,s,h)=bytes(2584,0,0)
addtime(567,0,0)usetime(513,0,0)packets(19,0,0) idle=495
esp0xc15f5191@192.168.0.1 ESP_3DES_HMAC_SHA1: dir=in src=192.168.0.2 iv_bits=64bits
iv=0x64bf6b2f56f05e20 ooowin=64 seq=19 bit=0x7fffff alen=160 aklen=160 eklen=192 life(c,s,h)=bytes
(1976,0,0)addtime(567,0,0)usetime(513,0,0)packets(19,0,0) idle=495
tun0x1001@192.168.0.1 IPIP: dir=in src=192.168.0.2 policy=192.168.100.0/24->192.168.10.0/24
flags=0x8<> life(c,s,h)=bytes(1976,0,0)addtime(567,0,0)usetime(513,0,0)packets(19,0,0) idle=495
tun0x1002@192.168.0.2 IPIP: dir=out src=192.168.0.1 life(c,s,h)=bytes(1976,0,0)addtime(567,0,0)
usetime(513,0,0)packets(19,0,0) idle=495
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 192.168.0.1 0.0.0.0 UG 0 0 0 eth0
192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 ipsec0
192.168.100.0 192.168.0.2 255.255.255.0 UG 0 0 0 ipsec0
```

ipsec auto --status

```
gw1:~ # ipsec auto --status
000 interface ipsec0/eth0 192.168.0.1
000
000 algorithm ESP encrypt: id=3, name=ESP_3DES, ivlen=64, keysize=168, keydeflen=128
000 algorithm ESP encrypt: id=12, name=ESP_AES, ivlen=128, keysize=256, keydeflen=128
000 algorithm ESP auth attr: id=1, name=AUTH_ALGORITHM_HMAC_MD5, keysize=128, keydeflen=128
000 algorithm ESP auth attr: id=2, name=AUTH_ALGORITHM_HMAC_SHA1, keysize=160, keydeflen=160
000
000 algorithm IKE encrypt: id=65289, name=OAKLEY_SSH_PRIVATE_65289, blocksize=16, keydeflen=128
000 algorithm IKE encrypt: id=65005, name=OAKLEY_TWOFISH_CBC, blocksize=16, keydeflen=128
000 algorithm IKE encrypt: id=3, name=OAKLEY_BLOWFISH_CBC, blocksize=8, keydeflen=128
000 algorithm IKE encrypt: id=7, name=OAKLEY_AES_CBC, blocksize=16, keydeflen=128
000 algorithm IKE encrypt: id=5, name=OAKLEY_3DES_CBC, blocksize=8, keydeflen=192
000 algorithm IKE hash: id=6, name=OAKLEY_SHA2_512, hashsize=64
000 algorithm IKE hash: id=4, name=OAKLEY_SHA2_256, hashsize=32
000 algorithm IKE hash: id=2, name=OAKLEY_SHA, hashsize=20
000 algorithm IKE hash: id=1, name=OAKLEY_MD5, hashsize=16
000 algorithm IKE dh group: id=2, name=OAKLEY_GROUP_MODP1024, bits=1024
000 algorithm IKE dh group: id=5, name=OAKLEY_GROUP_MODP1536 (extension), bits=1536
000 algorithm IKE dh group: id=42048, name=OAKLEY_GROUP_MODP2048 (extension), bits=2048
000 algorithm IKE dh group: id=43072, name=OAKLEY_GROUP_MODP3072 (extension), bits=3072
000 algorithm IKE dh group: id=44096, name=OAKLEY_GROUP_MODP4096 (extension), bits=4096
000
000 stats db_ops.c: {curr_cnt, total_cnt, maxsz} :context={0,713,36} trans={0,713,144} attrs=
{0,713,160}
000
000 "cisco-pix": 192.168.10.0/24===192.168.0.1..192.168.0.2===192.168.100.0/24
000 "cisco-pix": ike_life: 3600s; ipsec_life: 1800s; rekey_margin: 540s; rekey_fuzz: 100%;
keyingtries: 0
000 "cisco-pix": policy: PSK+ENCRYPT+TUNNEL+PFS; interface: eth0; erouted
000 "cisco-pix": newest ISAKMP SA: #711; newest IPsec SA: #714; eroute owner: #714
000 "cisco-pix": IKE algorithms wanted: 5_000-1-5, 5_000-2-5, 5_000-1-2, 5_000-2-2, flags=-strict
000 "cisco-pix": IKE algorithms found: 5_192-1_128-5, 5_192-2_160-5, 5_192-1_128-2, 5_192-2_160-
2,
000 "cisco-pix": IKE algorithm newest: 3DES_CBC_192-MD5-MODP1024
000 "cisco-pix": ESP algorithms wanted: 3_000-1, 3_000-2, flags=-strict
000 "cisco-pix": ESP algorithms loaded: 3_168-1_128, 3_168-2_160,
000 "cisco-pix": ESP algorithm newest: 3DES_0-HMAC_SHA1; pfs_group=<Phase1>
```

```

000 "gw1-gw2": 192.168.10.0/24===192.168.0.1...192.168.0.2===192.168.100.0/24
000 "gw1-gw2": ike_life: 3600s; ipsec_life: 1800s; rekey_margin: 540s; rekey_fuzz: 100%;
keyingtries: 0
000 "gw1-gw2": policy: RSASIG+ENCRYPT+TUNNEL+PFS; interface: eth0; unrouted
000 "gw1-gw2": newest ISAKMP SA: #0; newest IPsec SA: #0; eroute owner: #0
000 "gw1-gw2": IKE algorithms wanted: 7_128-2-5, 7_128-2-2, 7_128-1-5, 7_128-1-2, 5_000-2-5,
5_000-2-2, 5_000-1-5, 5_000-1-2, flags=-strict
000 "gw1-gw2": IKE algorithms found: 7_128-2_160-5, 7_128-2_160-2, 7_128-1_128-5, 7_128-1_128-2,
5_192-2_160-5, 5_192-2_160-2, 5_192-1_128-5, 5_192-1_128-2,
000 "gw1-gw2": ESP algorithms wanted: 12_128-2, 12_128-1, 3_000-2, 3_000-1, flags=-strict
000 "gw1-gw2": ESP algorithms loaded: 12_128-2_160, 12_128-1_128, 3_168-2_160, 3_168-1_128,
000
000 #714: "cisco-pix" STATE_QUICK_I2 (sent QI2, IPsec SA established); EVENT_SA_REPLACE in 1086s;
newest IPSEC; eroute owner
000 #714: "cisco-pix" esp.c1ae6c22@192.168.0.2 esp.4eeb936f@192.168.0.1 tun.141a@192.168.0.2
tun.1419@192.168.0.1
000 #711: "cisco-pix" STATE_MAIN_I4 (ISAKMP SA established); EVENT_SA_REPLACE in 35s; newest ISAKMP
000
gw1:~

```

Der folgende Befehl baut die Verbindung wieder ab:

```
gw1:~ # ipsec auto --down cisco-pix
```

8.2 Starten und Testen der Verbindung auf den PIX-Gateway.

Auf den Cisco-PIX sind die Einstellungen sofort wirksam. Das bedeutet, dass es nicht erforderlich (oder auch möglich) ist, separat einen manuellen Verbindungsstart zu veranlassen. Der PIX wartet auf die Anfragen (udp-protocol port 500) und nachdem die Authentifizierung erfolgreich ist, baut er den Tunnel selbstständig auf. Es können jedoch mehrere Tunnel parallel aufgebaut werden. Die max. Tunnelanzahl ist von den Cisco-Geräten abhängig.

Die Überprüfung des Tunnelaufbaus erfolgt mit Hilfe der folgenden Befehle:

debug crypto isakmp

```

cisco-pix(config)# debug crypto isakmp

crypto_isakmp_process_block:src:192.168.0.1, dest:192.168.0.2 spt:500 dpt:500
OAK_MM exchange
ISAKMP (0): processing SA payload. message ID = 0

ISAKMP (0): Checking ISAKMP transform 0 against priority 10 policy
ISAKMP:   life type in seconds
ISAKMP:   life duration (basic) of 3600
ISAKMP:   encryption 3DES-CBC
ISAKMP:   hash MD5
ISAKMP:   auth pre-share
ISAKMP:   default group 5
ISAKMP (0): atts are not acceptable. Next payload is 3
ISAKMP (0): Checking ISAKMP transform 1 against priority 10 policy
ISAKMP:   life type in seconds
ISAKMP:   life duration (basic) of 3600
ISAKMP:   encryption 3DES-CBC
ISAKMP:   hash SHA
ISAKMP:   auth pre-share
ISAKMP:   default group 5
ISAKMP (0): atts are not acceptable. Next payload is 3
ISAKMP (0): Checking ISAKMP transform 2 against priority 10 policy
ISAKMP:   life type in seconds
ISAKMP:   life duration (basic) of 3600

```

```

ISAKMP:      encryption 3DES-CBC
ISAKMP:      hash MD5
ISAKMP:      auth pre-share
ISAKMP:      default group 2
ISAKMP (0):  atts are acceptable. Next payload is 3
ISAKMP (0):  SA is doing pre-shared key authentication using id type ID_IPV4_ADDR
return status is IKMP_NO_ERROR
crypto_isakmp_process_block:src:192.168.0.1, dest:192.168.0.2 spt:500 dpt:500
OAK_MM exchange
ISAKMP (0):  processing KE payload. message ID = 0
ISAKMP (0):  processing NONCE payload. message ID = 0
return status is IKMP_NO_ERROR
crypto_isakmp_process_block:src:192.168.0.1, dest:192.168.0.2 spt:500 dpt:500
OAK_MM exchange
ISAKMP (0):  processing ID payload. message ID = 0
ISAKMP (0):  processing HASH payload. message ID = 0
ISAKMP (0):  SA has been authenticated
ISAKMP (0):  ID payload
      next-payload : 8
      type          : 1
      protocol      : 17
      port          : 500
      length        : 8
ISAKMP (0):  Total payload length: 12
return status is IKMP_NO_ERROR
ISAKMP (0):  sending INITIAL_CONTACT notify
ISAKMP (0):  sending NOTIFY message 24578 protocol 1
VPN Peer: ISAKMP: Added new peer: ip:192.168.0.1/500 Total VPN Peers:1
VPN Peer: ISAKMP: Peer ip:192.168.0.1/500 Ref cnt incremented to:1 Total VPN Peers:1
crypto_isakmp_process_block:src:192.168.0.1, dest:192.168.0.2 spt:500 dpt:500
OAK_QM exchange
oakley_process_quick_mode:
OAK_QM_IDLE
ISAKMP (0):  processing SA payload. message ID = 3062273838
ISAKMP : Checking IPsec proposal 0
ISAKMP: transform 0, ESP_3DES
ISAKMP:  attributes in transform:
ISAKMP:  group is 2
ISAKMP:  encaps is 1
ISAKMP:  SA life type in seconds
ISAKMP:  SA life duration (basic) of 1800
ISAKMP:  authenticator is HMAC-MD5
ISAKMP (0):  atts not acceptable. Next payload is 3
ISAKMP: transform 1, ESP_3DES
ISAKMP:  attributes in transform:
ISAKMP:  group is 2
ISAKMP:  encaps is 1
ISAKMP:  SA life type in seconds
ISAKMP:  SA life duration (basic) of 1800
ISAKMP:  authenticator is HMAC-SHA
ISAKMP (0):  atts are acceptable.
ISAKMP (0):  processing NONCE payload. message ID = 3062273838
ISAKMP (0):  processing KE payload. message ID = 3062273838
ISAKMP (0):  processing ID payload. message ID = 3062273838
ISAKMP (0):  ID_IPV4_ADDR_SUBNET src 192.168.10.0/255.255.255.0 prot 0 port 0
ISAKMP (0):  processing ID payload. message ID = 3062273838
ISAKMP (0):  ID_IPV4_ADDR_SUBNET dst 192.168.100.0/255.255.255.0 prot 0 port 0
return status is IKMP_NO_ERROR
crypto_isakmp_process_block:src:192.168.0.1, dest:192.168.0.2 spt:500 dpt:500
OAK_QM exchange
oakley_process_quick_mode:
OAK_QM_AUTH_AWAIT
ISAKMP (0): Creating IPsec SAs
      inbound SA from      192.168.0.1 to      192.168.0.2 (proxy  192.168.10.0 to
192.168.100.0)
      has spi 2045573164 and conn_id 5 and flags 25
      lifetime of 1800 seconds
      outbound SA from     192.168.0.2 to     192.168.0.1 (proxy  192.168.100.0 to
192.168.10.0)
      has spi 1284333619 and conn_id 6 and flags 25
      lifetime of 1800 seconds
VPN Peer: IPSEC: Peer ip:192.168.0.1/500 Ref cnt incremented to:2 Total VPN Peers:1
VPN Peer: IPSEC: Peer ip:192.168.0.1/500 Ref cnt incremented to:3 Total VPN Peers:1
return status is IKMP_NO_ERROR

```

debug crypto ipsec

```

cisco-pix(config)#debug crypto ipsec
cisco-pix(config)# IPSEC(validate_proposal): transform proposal (prot 3, trans 3, hmac_alg 1) not
supported
IPSEC(validate_proposal_request): proposal part #1,
  (key eng. msg.) dest= 192.168.0.2, src= 192.168.0.1,
  dest_proxy= 192.168.100.0/255.255.255.0/0/0 (type=4),
  src_proxy= 192.168.10.0/255.255.255.0/0/0 (type=4),
  protocol= ESP, transform= esp-3des esp-sha-hmac ,
  lifedur= 0s and 0kb,
  spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x24
IPSEC(key_engine): got a queue event...
IPSEC(spi_response): getting spi 0xbe35ad71(3191188849) for SA
  from 192.168.0.1 to 192.168.0.2 for prot 3
IPSEC(key_engine): got a queue event...
IPSEC(initialize sas): ,
  (key eng. msg.) dest= 192.168.0.2, src= 192.168.0.1,
  dest_proxy= 192.168.100.0/255.255.255.0/0/0 (type=4),
  src_proxy= 192.168.10.0/255.255.255.0/0/0 (type=4),
  protocol= ESP, transform= esp-3des esp-sha-hmac ,
  lifedur= 1800s and 0kb,
  spi= 0xbe35ad71(3191188849), conn_id= 5, keysize= 0, flags= 0x25
IPSEC(initialize sas): ,
  (key eng. msg.) src= 192.168.0.2, dest= 192.168.0.1,
  src_proxy= 192.168.100.0/255.255.255.0/0/0 (type=4),
  dest_proxy= 192.168.10.0/255.255.255.0/0/0 (type=4),
  protocol= ESP, transform= esp-3des esp-sha-hmac ,
  lifedur= 1800s and 0kb,
  spi= 0xc15f5191(3244249489), conn_id= 6, keysize= 0, flags= 0x25

```

show crypto ipsec sa

```

cisco-pix(config)# show crypto ipsec sa

interface: outside
  Crypto map tag: vpn-Map, local addr. 192.168.0.2

local ident (addr/mask/prot/port): (192.168.100.0/255.255.255.0/0/0)
remote ident (addr/mask/prot/port): (192.168.10.0/255.255.255.0/0/0)
current peer: 192.168.0.1:500
  PERMIT, flags={origin_is_acl,}
  #pkts encaps: 736, #pkts encrypt: 736, #pkts digest 736
  #pkts decaps: 734, #pkts decrypt: 734, #pkts verify 734
  #pkts compressed: 0, #pkts decompressed: 0
  #pkts not compressed: 0, #pkts compr. failed: 0, #pkts decompress failed: 0
  #send errors 7, #rcv errors 0

  local crypto endpt.: 192.168.0.2, remote crypto endpt.: 192.168.0.1
  path mtu 1500, ipsec overhead 56, media mtu 1500
  current outbound spi: 4eeb9164

inbound esp sas:
  spi: 0x6562525d(1700942429)
  transform: esp-3des esp-sha-hmac ,
  in use settings ={Tunnel, }
  slot: 0, conn id: 4, crypto map: vpn-Map
  sa timing: remaining key lifetime (k/sec): (399920/1413)
  IV size: 8 bytes
  replay detection support: Y

inbound ah sas:

inbound pcp sas:

outbound esp sas:
  spi: 0x4eeb9164(1324061028)
  transform: esp-3des esp-sha-hmac ,
  in use settings ={Tunnel, }
  slot: 0, conn id: 3, crypto map: vpn-Map
  sa timing: remaining key lifetime (k/sec): (399942/1413)
  IV size: 8 bytes
  replay detection support: Y

outbound ah sas:

```

outbound pcp sas:

sh crypto map

```
cisco-pix(config)# sh crypto map

Crypto Map: "vpn-Map" interfaces: { outside }

Crypto Map "vpn-Map" 10 ipsec-isakmp
  Peer = 192.168.0.1
  access-list vpn_acl; 1 elements
  access-list vpn_acl line 1 permit ip 192.168.100.0 255.255.255.0 192.168.10.0 255.255.255.0
(hitcnt=6064)
  Current peer: 192.168.0.1
  Security association lifetime: 400000 kilobytes/3600 seconds
  PFS (Y/N): N
  Transform sets={ my_vpn, }

Crypto Map "vpn-Map" 20 ipsec-isakmp
  Dynamic map template tag: vpn-Dyn

Crypto Map "vpn-Map" 30 ipsec-isakmp
  Peer = 192.168.0.1
  access-list dynacl2; 1 elements
  access-list dynacl2 line 1 permit ip 192.168.100.0 255.255.255.0 192.168.10.0 255.255.255.0
(hitcnt=0)
  dynamic (created from dynamic map vpn-Dyn/10)
  Current peer: 192.168.0.1
  Security association lifetime: 400000 kilobytes/1800 seconds
  PFS (Y/N): Y
  DH group: group2
  Transform sets={ my_vpn, }
cisco-pix(config)#
```

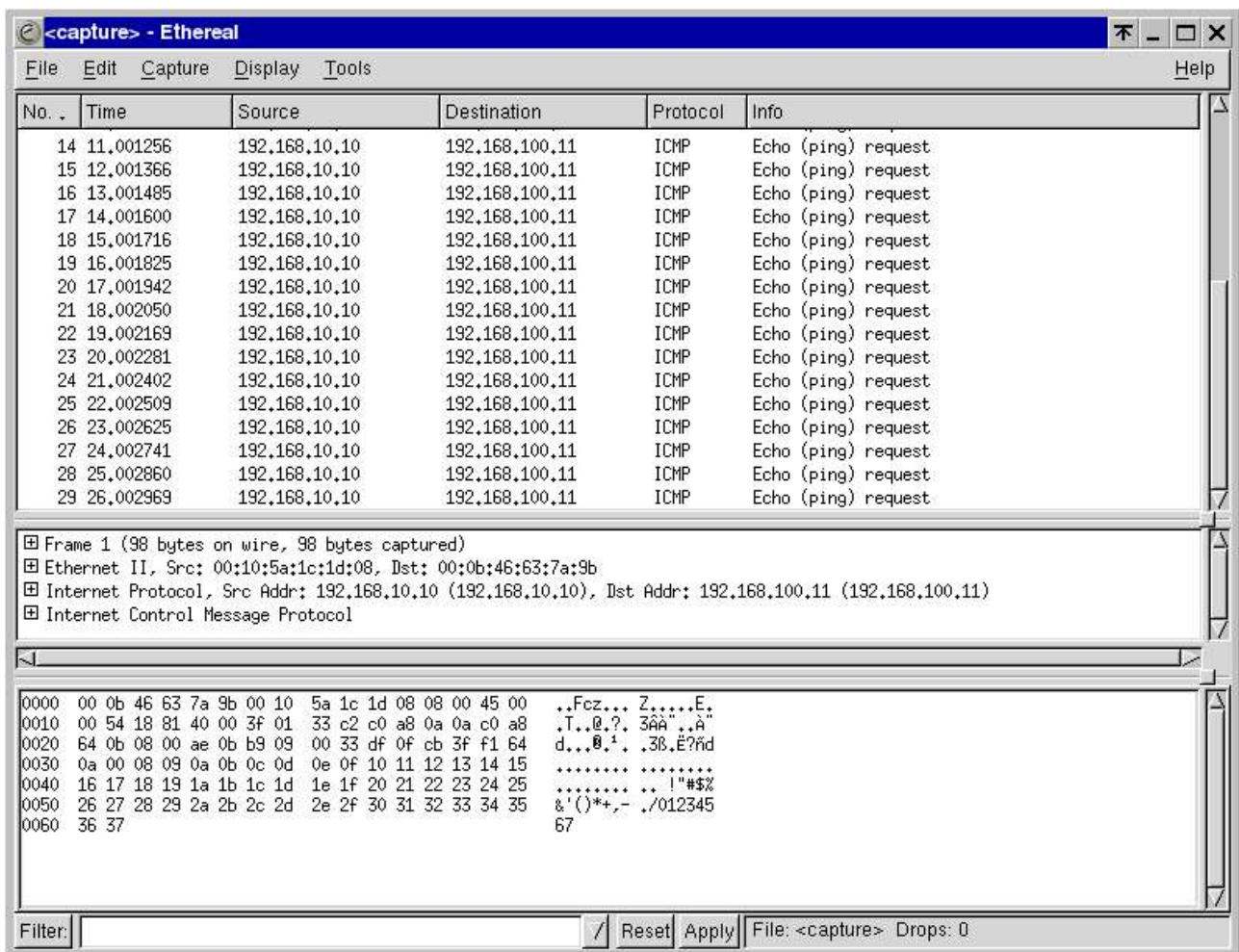
9. Mitschnitt eines Verbindungsaufbaus

Durch einen sogenannten Netzwerk-Sniffer ist es möglich, den Datenverkehr im Netzwerk mitzuschneiden und zu analysieren. Somit ist eine gewisse Analyse der IPsec-Pakete, die zwischen Lokalem-Netz A und Lokalem-Netz B unterwegs sind, möglich.

Weiter unten habe ich eine Darstellung des Sniffers Ethereal eingefügt. Zu sehen ist ein Verbindungsaufbau mit anschließendem Austausch von ICMP-Paketen (Ping zwischen Lokalem-Netz B und Lokalem-Netz A) vor und nach dem Starten der verschlüsselten Verbindung.

9.1 Vor dem Aufbau des Tunnels

```
eno@praktika:~> ping 192.168.100.11
```



The screenshot shows the Ethereal network sniffer interface. The main window displays a list of captured packets. The first 29 packets are ICMP Echo (ping) requests from 192.168.10.10 to 192.168.100.11. Below the list, the details for the first frame (98 bytes on wire, 98 bytes captured) are shown, including Ethernet II, Internet Protocol, and Internet Control Message Protocol layers. The packet bytes are displayed in hexadecimal and ASCII format.

No.	Time	Source	Destination	Protocol	Info
14	11.001256	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
15	12.001366	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
16	13.001485	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
17	14.001600	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
18	15.001716	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
19	16.001825	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
20	17.001942	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
21	18.002050	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
22	19.002169	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
23	20.002281	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
24	21.002402	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
25	22.002509	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
26	23.002625	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
27	24.002741	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
28	25.002860	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request
29	26.002969	192.168.10.10	192.168.100.11	ICMP	Echo (ping) request

Frame 1 (98 bytes on wire, 98 bytes captured)
Ethernet II, Src: 00:10:5a:1c:1d:08, Dst: 00:0b:46:63:7a:9b
Internet Protocol, Src Addr: 192.168.10.10 (192.168.10.10), Dst Addr: 192.168.100.11 (192.168.100.11)
Internet Control Message Protocol

```
0000  00 0b 46 63 7a 9b 00 10 5a 1c 1d 08 08 00 45 00  ..Fcz... Z.....E.  
0010  00 54 18 81 40 00 3f 01 33 c2 c0 a8 0a 0a c0 a8  .T..@.? 3AA" ..A"  
0020  64 0b 08 00 ae 0b b9 09 00 33 df 0f cb 3f f1 64  d...@.1. .3@.É?fid  
0030  0a 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15  .....  
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25  ..... ..!"#$%  
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35  &'()*+,-./012345  
0060  36 37 67
```

9.2 Nach dem Aufbau des Tunnels

eno@praktika:~> ping 192.168.100.11

The screenshot shows the Wireshark interface with a packet capture of IKE-ESP traffic. The main pane displays a list of 18 packets. The first 10 packets are ISAKMP, and the last 8 are ESP. The packet details pane shows the structure of the first packet: Ethernet II, ARP (request), and a hex dump of the raw data.

No.	Time	Source	Destination	Protocol	Info
3	0.000143	192.168.0.1	192.168.0.2	ISAKMP	Identity Protection (Main Mode)
4	0.078034	192.168.0.2	192.168.0.1	ISAKMP	Identity Protection (Main Mode)
5	0.094266	192.168.0.1	192.168.0.2	ISAKMP	Identity Protection (Main Mode)
6	0.173751	192.168.0.2	192.168.0.1	ISAKMP	Identity Protection (Main Mode)
7	0.189231	192.168.0.1	192.168.0.2	ISAKMP	Identity Protection (Main Mode)
8	0.189586	192.168.0.2	192.168.0.1	ISAKMP	Identity Protection (Main Mode)
9	0.189732	192.168.0.2	192.168.0.1	ISAKMP	Informational
10	0.206727	192.168.0.1	192.168.0.2	ISAKMP	Quick Mode
11	0.365092	192.168.0.2	192.168.0.1	ISAKMP	Quick Mode
12	0.499377	192.168.0.1	192.168.0.2	ISAKMP	Quick Mode
13	14.431099	192.168.0.1	192.168.0.2	ESP	ESP (SPI=0x395f348e)
14	14.432138	192.168.0.2	192.168.0.1	ESP	ESP (SPI=0x78f3b55b)
15	15.436188	192.168.0.1	192.168.0.2	ESP	ESP (SPI=0x395f348e)
16	15.436878	192.168.0.2	192.168.0.1	ESP	ESP (SPI=0x78f3b55b)
17	16.446304	192.168.0.1	192.168.0.2	ESP	ESP (SPI=0x395f348e)
18	16.446967	192.168.0.2	192.168.0.1	ESP	ESP (SPI=0x78f3b55b)

Packet 1 details:

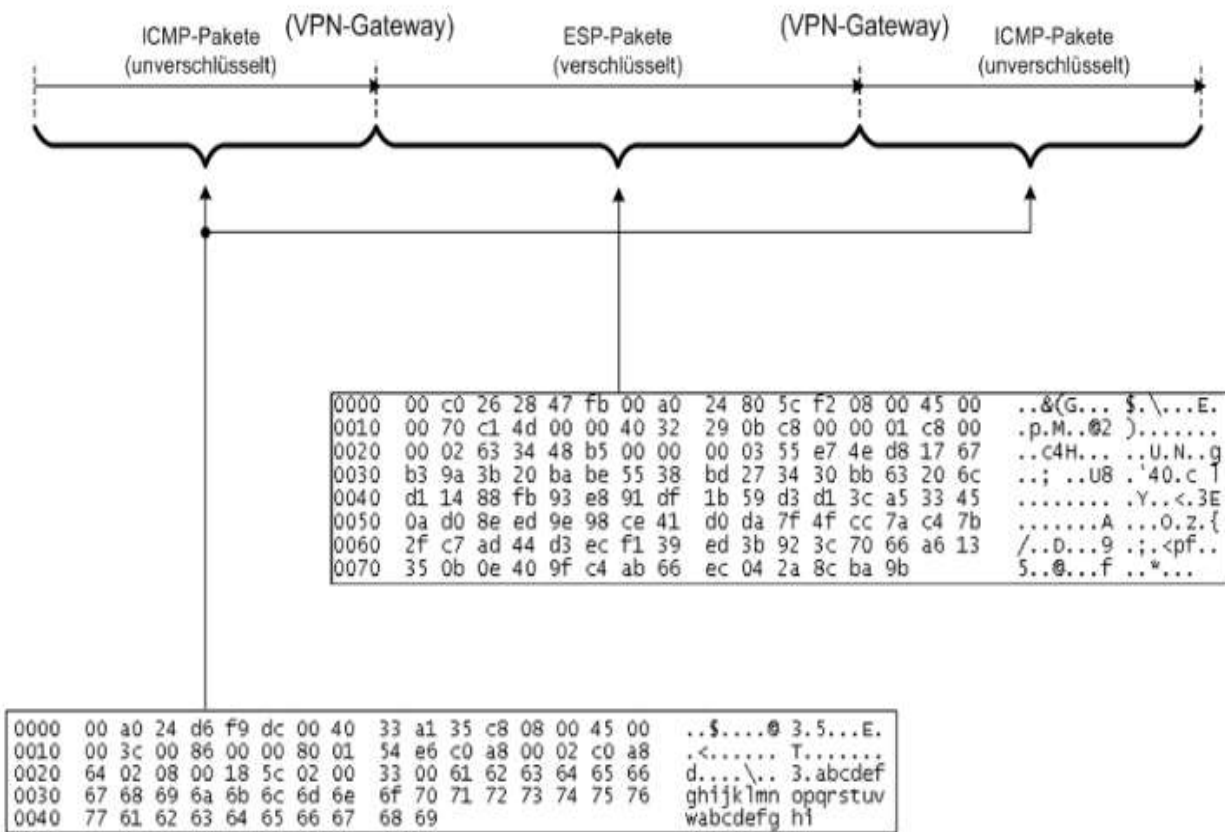
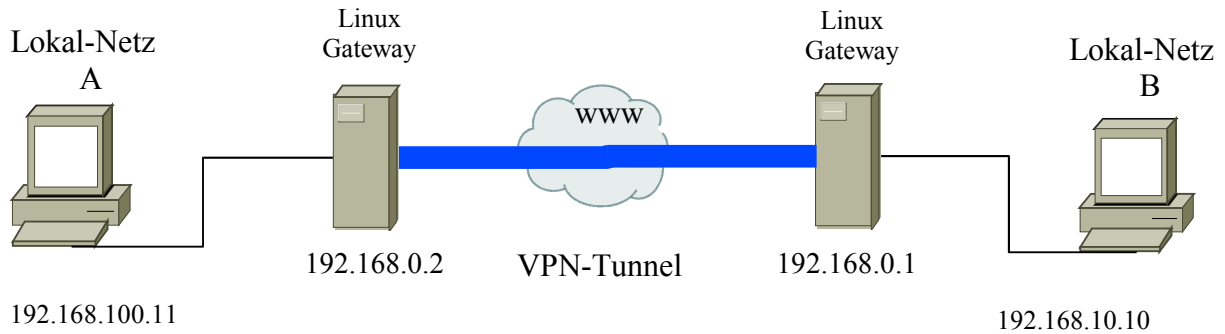
- Frame 1 (42 bytes on wire (42 bytes captured) on interface eth0)
- Ethernet II, Src: 00:10:5a:1c:1d:08, Dst: ff:ff:ff:ff:ff:ff
- Address Resolution Protocol (request)

```

0000  ff ff ff ff ff ff 00 10 5a 1c 1d 08 08 06 00 01  00000000.. Z.....
0010  08 00 06 04 00 01 00 10 5a 1c 1d 08 c0 a8 00 01  ..... Z...A" ..
0020  00 00 00 00 00 00 c0 a8 00 02  .....A" ..
  
```

Wie schon im theoretischen Teil beschrieben, erfolgt vor der Etablierung eines Tunnels der Austausch von IKE-(ISAKMP/Oakley)-Paketen. Es sind insgesamt 10 Pakete, wobei die ersten 6 zum Main-Mode und die restlichen 4 zum Quick-Mode gehören. Nach Beendigung des Quick-Modes ist dann der eigentliche Austausch der verschlüsselten ESP-Datenpakete möglich. Hier sei erwähnt, dass die eigentlichen Quell- und Zieladressen der Datenverbindung (Lokalen-Netz **A** mit 192.168.0.2 und Lokalen-Netz **B** mit 192.168.0.1) nicht zu erkennen sind.

Die folgende Darstellung gibt einen Überblick über den Paketfluss. Hier ist zum Beispiel auch der Größenunterschied zwischen einem unverschlüsselten ICMP-Paket und einem ESP-Paket (welches das ICMP-Paket in verschlüsselter Form enthält) erkennbar. Das ESP-Paket ist gegenüber dem ICMP-Paket um ungefähr 70% größer.



Die folgende Darstellung gibt noch einmal einen Überblick über den Paketfluß im Tunnel während des normalen Betriebes. Die wirklichen Quell- bzw. Zieladressen sind hier nicht zu erkennen. Anstatt dessen sieht man nur die IP-Adressen der Gateways. Das Transfer-Protokoll ist nun ESP. Das Sammeln von Paketen über Ethereal erfolgt über den Button „Capture“. Möchte man die gesammelten Pakete auswerten, dann bietet sich unter „Tools“ die Menü „Summary“ eine Hilfe.

The screenshot shows the Ethereal interface with a packet capture window and a summary window.

Packet Capture Window:

No.	Time	Source	Destination	Protocol	Info
4237	1205,723788	192.168,0,2	192.168,0,1	ESP	ESP (SPI=0x4eeb9165)
4238	1205,724099	192.168,0,1	192.168,0,2	ESP	ESP (SPI=0x27aa92f4)
4239	1206,723720	192.168,0,2	192.168,0,1	ESP	ESP (SPI=0x4eeb9165)
4240	1206,724052	192.168,0,1	192.168,0,2	ESP	ESP (SPI=0x27aa92f4)
4241	1207,723660	192.168,0,2	192.168,0,1	ESP	ESP (SPI=0x4eeb9165)
4242	1207,723961	192.168,0,1	192.168,0,2	ESP	ESP (SPI=0x27aa92f4)
4243	1208,723640	192.168,0,2	192.168,0,1	ESP	ESP (SPI=0x4eeb9165)
4244	1208,723938	192.168,0,1	192.168,0,2	ESP	ESP (SPI=0x27aa92f4)
4245	1209,723551	192.168,0,2	192.168,0,1	ESP	ESP (SPI=0x4eeb9165)
4246	1209,723864	192.168,0,1	192.168,0,2	ESP	ESP (SPI=0x27aa92f4)
4247	1210,723488	192.168,0,2	192.168,0,1	ESP	ESP (SPI=0x4eeb9165)
4248	1210,723802	192.168,0,1	192.168,0,2	ESP	ESP (SPI=0x27aa92f4)
4249	1211,723406	192.168,0,2	192.168,0,1	ESP	ESP (SPI=0x4eeb9165)
4250	1211,723718	192.168,0,1	192.168,0,2	ESP	ESP (SPI=0x27aa92f4)
4251	1212,723347	192.168,0,2	192.168,0,1	ESP	ESP (SPI=0x4eeb9165)

Frame 1 (150 bytes on wire, 150 bytes captured)

- Ethernet II, Src: 00:10:5a:1c:1d:08, Dst: 00:0b:46:63:7a:9b
- Internet Protocol, Src Addr: 192.168.0.1 (192.168.0.1), Dst Addr: 192.168.0.2 (192.168.0.2)
- Encapsulating Security Payload

Hex Dump:

```

0000  00 0b 46 63 7a 9b 00 10 5a 1c 1d 08 08 00 45 00  ..Fcz... Z.....E.
0010  00 88 2a 31 00 00 40 32 ce bf c0 a8 00 01 c0 a8  ..*1..02 f0A" ..A"
0020  00 02 65 62 5d 00 00 00 31 cb 4e fd 2e 5a 91     ..ebR].. 1EMg.Z.
0030  75 50 aa 03 6c e2 75 a8 c5 8d 63 c4 04 4e e9 0c  uP=,1au" Á,cÁ,Né.
0040  e1 94 ab fa 61 f7 d8 42 07 92 c1 96 41 73 dc 7a  á.«úa+0B ..Á,AsÚz
0050  bb f2 e6 ae 62 e7 56 5d bc 08 72 4d 37 49 d1 f1  »0a0bcV] 4.rM7INñ
0060  89 c2 08 28 2f 39 04 50 a3 98 18 e5 4a 22 4e b6  .Á.(/9.P f..ãJ"NM
0070  e1 13 e7 18 84 87 73 8b ea 37 68 06 a3 e3 eb 96  á.c...s. è7h.fää.
0080  63 38 65 a5 ca cd 39 ea 11 b0 4e ca ec 83 b4 b1  c8e#ÉI9é ."NÉi. ±
0090  e9 c6 5a 7a 06 e0                                     éKZz.â
  
```

Ethereal: Summary Window:

- Name: /tmp/etherXXXXX7IHQt
- Length: 24
- Format: libpcap (tcpdump, Ethereal, etc.)
- Snapshot length: 65535
- Data:
 - Elapsed time: 21.498 seconds
 - Between first and last packet: 21.498 seconds
 - Packet count: 28
 - Filtered packet count: 0
 - Marked packet count: 0
 - Avg. packets/sec: 1.302
 - Dropped packets: 0
 - Bytes of traffic: 4518
 - Avg. bytes/sec: 210.163
 - Avg. Mbit/sec: 0.002
- Capture:
 - Interface: eth0
 - Display filter: none
 - Capture filter: none

10. Die Leistungsmessung

Für die Messung der Transfer-Leistung in verschlüsselter Form, habe ich Daten vom Lokal-Netz B nach Lokal-Netz A mit Hilfe der Befehlen „wget“ und „ftp“ transferiert und die Ausgabe dieser Programme bewertet. Gleichzeitig habe ich den Sniffer Ethereal laufen lassen, um die Ausgaben der Programme miteinander zu vergleichen. Anschließend habe ich den Vorgang des Datentransfers in unverschlüsselter Form wiederholt und das Ergebnis der Ausgaben verglichen.

Das unten dargestellte Beispiel zeigt die Ausgabe des Datentransfers von A nach B über „ftp“, indem vorher ein Tunnel zwischen Cisco-PIX <==> Linux-Gateway aufgebaut worden war.

```
eno@praktika:~> ftp 192.168.100.11
Connected to 192.168.100.11.
220 (vsFTPd 1.1.3)
Name (192.168.100.11:eno): ftp
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.

ftp> ls
229 Entering Extended Passive Mode (|||32822|)
150 Opening BINARY mode data connection for '/bin/ls'.
total 79546
drwxr-xr-x  2 root  root           96 Sep 15 12:05 bin
drwxr-xr-x  2 root  root           72 Sep 15 12:05 dev
drwxr-xr-x  2 root  root           96 Sep 15 12:05 etc
drwxr-xr-x  2 root  root          280 Sep 15 12:05 lib
drwxr-xr-x  2 root  root          104 Sep 15 12:05 msgs
drwxr-xr-x  2 root  root           48 Nov 14 10:48 pix
drwxr-xr-x 24 root  root          680 Oct 21 11:14 pub
-rw-r--r--  1 root  root    81373141 Oct 21 11:17 pub.tgz
drwxr-xr-x  3 root  root           72 Sep 15 12:05 usr
226 Transfer complete.

ftp> get pub.tgz
local: pub.tgz remote: pub.tgz
229 Entering Extended Passive Mode (|||32824|)
150 Opening BINARY mode data connection for 'pub.tgz' (81373141 bytes).
100% |*****| 79465 KB 1.67
MB/s 00:00 ETA
226 Transfer complete.
81373141 bytes received in 00:46 (1.67 MB/s)
```

Da aber die Datei „pub.tgz“ als eine einzelne Datei transferiert wurde, ist das Ergebnis des Datentransfers als optimal zu bewerten. Beim Transfer mehrerer Dateien ist die verschlüsselte Leistung entsprechend geringer. Um den Transfer mehrerer Dateien zu überwachen, eignet sich am besten „wget“. Hier ein Beispiel, in dem der komplette Inhalt des Ordners „pub“ samt Unterordner kopiert wird.

```
eno@praktika:~> wget -r ftp://192.168.100.11/pub -o output.txt
```

Die Ausgabe wird hier in die Datei „output.txt“ gespeichert und ähnlich wie beim „ftp“ ausgewertet. Die kompletten Ergebnisse habe ich in den weiter unten gezeigten Tabellen dokumentiert.

10.1 Auswertung der Leistung zwischen den Linux-Gateways

Wie ich oben bereits erwähnt habe, wurden für die Tests verschiedene Rechner mit unterschiedlicher Leistung verwendet. Den Leistungsunterschied zwischen verschlüsseltem und unverschlüsseltem Datentransfer - je nach Rechnerleistung - wird in den Tabellen unten aufgeführt.

10.1.1 Leistung bei einem 200 MHz CPU / 128 MB RAM

(site - to - site) 200 MHz CPU / 128 MB RAM
Ausschnitt aus der Ethereal-Bewertung getestet mit der Datei linux-praxis.de/ 12 MB

protocol	ESP (wget)	TCP (wget)
Enlapsed time	310,963 sec	165,442 sec
Packet count	28252	25224
Avg. packets/sec	90,85	152,59
Bytes of traffic	15882172	13562248
Avg. bytes/sec	51074,48	81975,72
Avg. Mbit/sec	0,41	0,66

Ausschnitt aus der Ethereal-Bewertung getestet mit der Datei pub.tgz / 78 MB

Protocol	ESP (ftp get)	TCP (ftp get)
Enlapsed time	3734,956 sec	821,815 sec
Packet count	144491	118994
Avg. packets/sec	38,69	144,79
Bytes of traffic	135544866	102784184
Avg. bytes/sec	35755,4	125069,81
Avg. Mbit/sec	0,29	1

10.1.2 Leistung bei einem 500 MHz CPU / 128 MB RAM

(site - to - site) 500 MHz CPU / 128 MB RAM
Berechnung aus der Bewertung der Output-Datei von „wget“
Transfer der Dateien aus der Verzeichnis srv/ftp/pub/ 176 MB

protocol	ESP (wget)	TCP (wget)
Enlapsed time	38,013 sec	35,289 sec
Transfer MB/sec	4,631	5,02
Transfer Mbit/sec	37,05	40,22

Bewertung der Ausgabe des ftp-Transfers.
getestet mit der Datei pub.tgz / 78 MB

protocol	ESP (ftp get)	TCP (ftp get)
Enlapsed time	12,127 sec	9,589 sec
Transfer MB/sec	6,44	8,125
Transfer Mbit/sec	51,6	65

10.1.3 Leistung bei einem 1.6 GHz CPU / 512 MB RAM

(site - to - site) 1.6 GHz CPU / 512 MB RAM
Ausschnitt aus der Ethereal-Bewertung getestet mit der Datei linux-praxis.de/ 12 MB

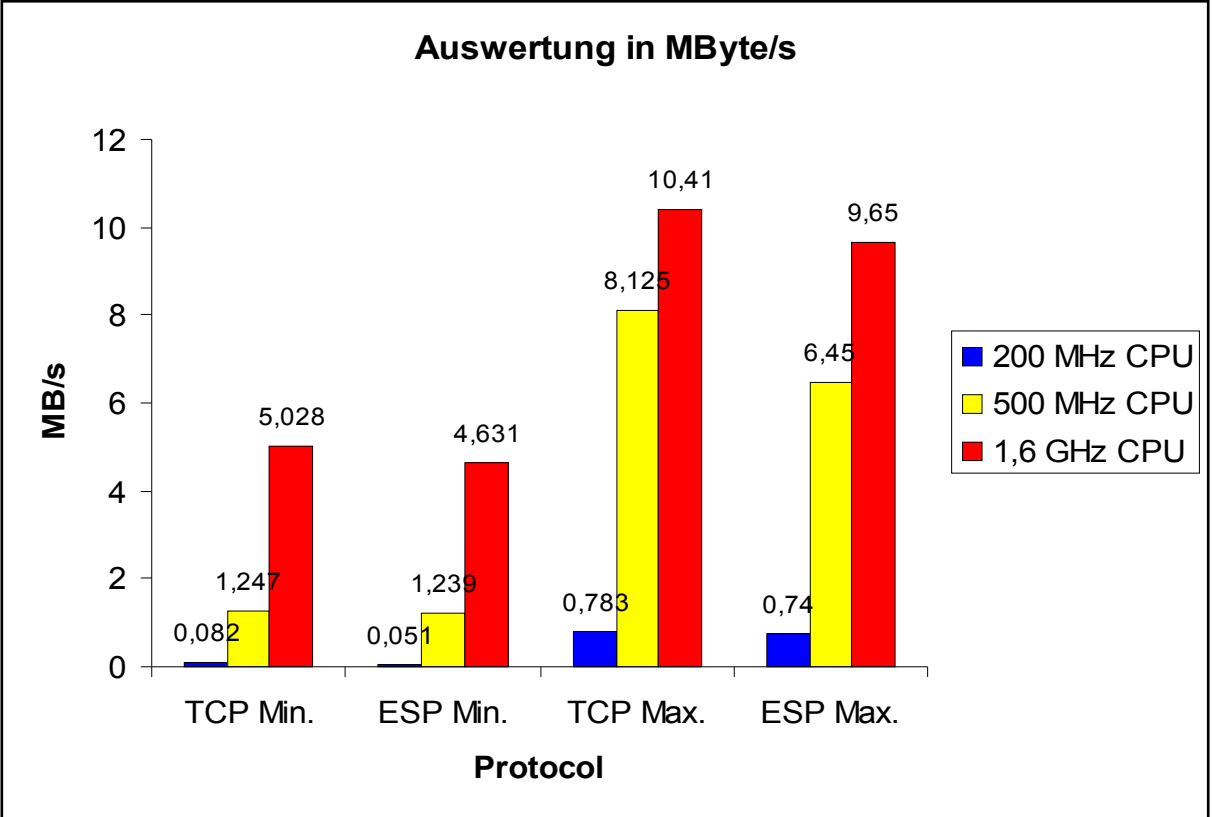
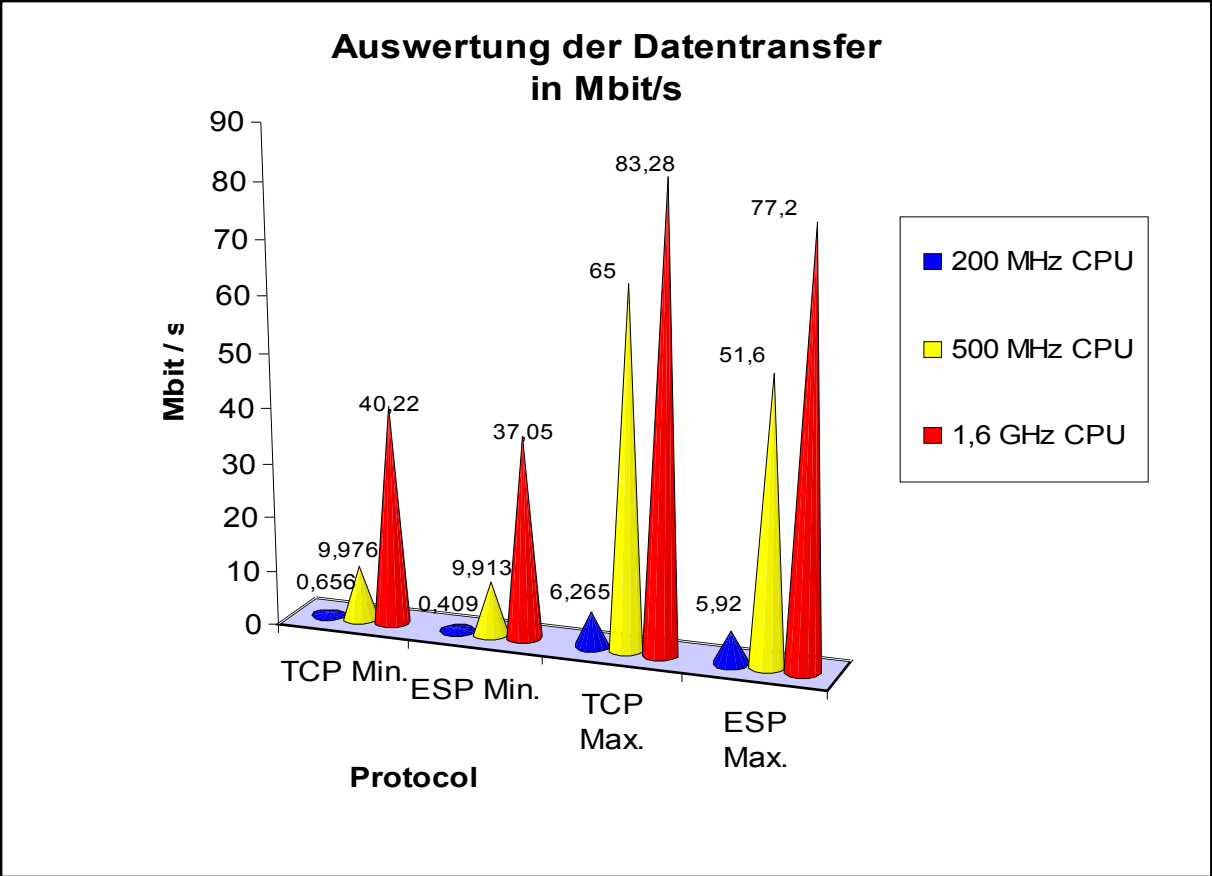
protocol	ESP (wget)	TCP (wget)
Enlapsed time	8,013 sec	7,989 sec
Packet count	21976	19720
Avg. packets/sec	2132,89	2094,44
Bytes of traffic	12767104	11014793
Avg. bytes/sec	1239117,81	1098774,04
Avg. Mbit/sec	9,91	9,98

Ausschnitt aus der Ethereal-Bewertung getestet mit der Datei pub.tgz / 78 MB

protocol	ESP (ftp get)	TCP (ftp get)
Enlapsed time	8,127 sec	7,01 sec
Packet count	53544	47774
Avg. packets/sec	6588,01	7148,87
Bytes of traffic	824773141	813773141
Avg. bytes/sec	9531311,48	9974384
Avg. Mbit/sec	77,02	83,28

10.1.4 Grafischer Darstellung der Rechner-Leistung

Die unteren Diagramme stellen den Leistungsunterschied noch mal grafisch dar. Das beste Ergebnis des Datentransfers habe ich als „MAX“ und den Datentransfer im ungünstigen Fall als „MIN“ bezeichnet. Wie auch aus der Tabelle deutlich wird, ist bei dem schnellsten Rechner nur 7.2 % Leistungsunterschied vorhanden. Der mittlere Rechner dagegen weist einen Leistungsverlust von 20.6 % auf.



10.2 Auswertung der Leistung zwischen den Linux- und PIX-Gateway

VPN (site - to - site) Cisco <====>FreeSwan (Linux)
 CISCO PIX-506E, 32 MB RAM, CPU Pentium II 300 Mhz (Ver. 6.3)
 Linux SuSE 8.2, 128 MB RAM, CPU Pentium III 667 MHz

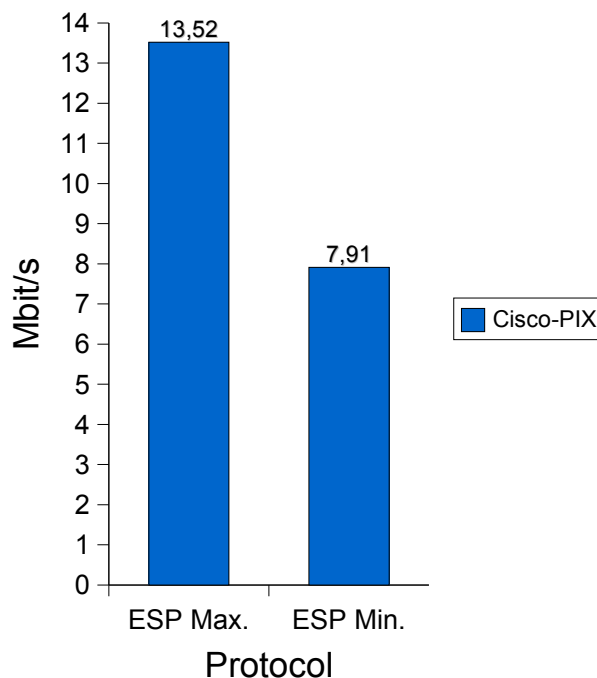
Berechnung aus der Bewertung der Output-Datei von „wget“
 Transfer der Dateien aus der Verzeichnis srv/ftp/pub/ 175 MB

protocol	ESP (wget)
Enlapsed time	177 sec
Transfer MB/sec	0,9887
Transfer Mbit/sec	7,91

Bewertung der Ausgabe des ftp-Transfers.
 getestet mit der Datei pub.tgz / 78 MB

protocol	ESP (ftp get)
Enlapsed time	45 sec
Transfer MB/sec	1,69
Transfer Mbit/sec	13,52

Auswertung in Mbit/s



11 Quellen

Literatur

- [1] Kai Martius, "Sicherheitsmanagement in TCP/IP-Netzen", Vieweg, 2000
ISBN 3-528-05725-4
- [2] Kai Fuhrberg, "Internet-Sicherheit", Hanser, 2000,
ISBN 3-446-21333-3
- [3] IPSec-RFCs 2401 bis 2409 (Request For Comment)

Links

- [4] FreeS/WAN Projekt:
<http://www.freeswan.org>
- [5] CISCO PIX 500 SERIES FIREWALLS
<http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/index.html>
- [6] VPNlabs:
<http://www.vpnlabs.org>
- [7] Projektarbeit VPN
<http://www.eno-vaso.de/project.html>

Newsgroups

- [8] Diskussionsthema VPN:
comp.dcom.vpn