OpenVPN

Christian Barthel, christian@barthel-connect.de,

Bildquelle: Luciano Caputo, 'Tubi', http://www.flickr.com/photos/lucianocaputo/2235576177/sizes/z/, 14. Januar 2008

Inhaltsverzeichnis

1 VPN

1.1 Über VPN

Virtuelle Private Netze (engl.: virtual private network) - kurz VPN - beschreibt die Technik, einen sicheren und geschützten Datenverkehr über ein öffentliches, unsicheres Netzwerk zu ermöglichen. Speziell über das Internet, also dem weltweitem Datennetz, ist es für kleine, mittel und große Organisationen sehr interessant, ein virtuelles Weitverkehrsnetzwerk bereitzustellen. Das Ziel ist die Anbindung von Niederlassungen, Verkaufsablegern, Heimarbeitsplätzen, Geschäftspartnern oder Beratern an das lokale Firmennetz. Da über diesen Weg unter anderem sensible Daten ausgetauscht werden können, ist es umso wichtiger, dieses Verkehrsnetz durch Verschlüsselungstechniken abzusichern.

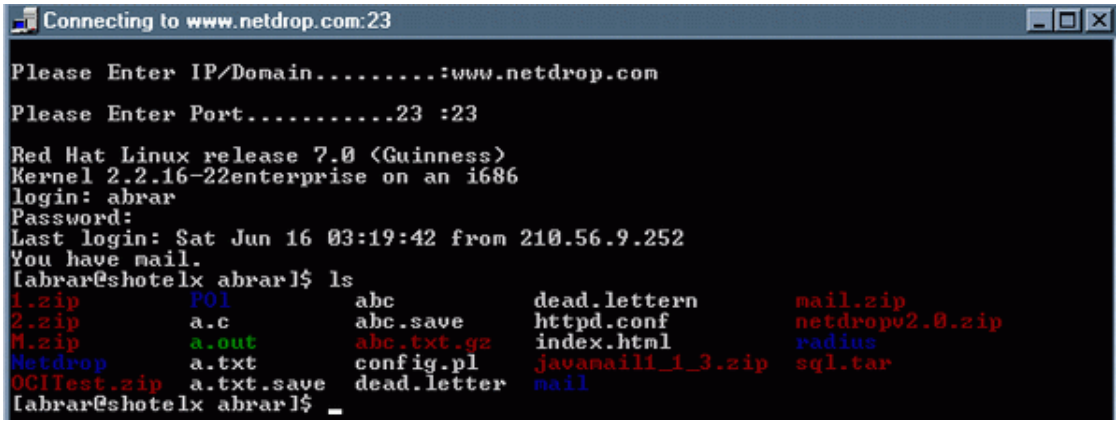
Wurden früher häufig sogenannte 'Standleitungen' zwischen großen Niederlassungen gelegt, so bedeutet diese Variante doch enorm hohe Kosten. Diese Kosten konnten nur von großen bis sehr international ausgelegten Firmen getragen werden. Hier hat VPN einen großen Vorteil: Es nutzt ein verhältnismäßig billiges Netzwerk, um somit eine Datenverbindung zwischen entfernten Netzen zu ermöglichen.

1.2 Fernzugriff und VPN alternativen

In unserer heutigen, vernetzten Internetwelt spielt der Fernzugriff auf Ressourcen eine wichtige Rolle. Bereits in der Vergangenheit gab es mehr oder weniger sichere Protokolle, um einen Fernzugriff auf entfernte Daten und Informationen zu ermöglichen. Wichtige werden hier nun kurz genannt:

1.2.1 Telnet

Telnet ist ein traditioneller Konsolenzugriff auf ein Datenverarbeitungsgerät. Dies kann beispielsweise ein Switch, ein Server oder aber auch ein Industrie-PC sein. Telnet sendet Daten im Klartext. Daher sind die übertragenen Passwörter natürlich sehr schnell ausspioniert. Von der heutigen Verwendung wird dringend abgearten. Maximal in sicheren Netzwerkumgebung kann dieser Dienst noch benutzt werdne, z. B. um Swich-Konfigurationen automatisiert zu sichern. Der Telnet-Dämon öffnet Port 23.



1.2.2 File transfer protocol - FTP

Wie der name bereits vermuten lässt, ist FTP für den Datenaustausch geschaffen worden. FTP stellt Daten zum Download bereit, und bietet Möglichkeiten, Daten hochzuladen. Dies geschieht, wie bei allen älteren Protokollen, im Klartext. Trotzdem möchte ich Ihnen FTP empfehlen, wenn Sie zum Beispiel Daten zur Verfügung stellen, die jeder sehen kann (öffentliche FTP-Server). Für sensible Daten sollten Sie jedoch ein anderes Protokoll wählen. FTP arbeitet auf Port 21.

1.2.3 Secure shell - SSH

SSH ermöglicht unter anderem den sicheren Konsolenzugriff auf entfernte Geräte. Jedoch kann SSH viel mehr. SFTP stellt einen vollwertigern Ersatz zu FTP dar. SCP - secure copy - ermöglicht ebenfalls einen sicheren Datenaustausch mit Befehlskommandos. Aber die Featureliste von SSH ist noch wesentlich länger. Über SSH können unsichere Protokolle getunnelt werden. VNC oder X11 Applikationen können somit sicher über das Netzwerk transportiert werden. SSH nutzt dazu ein kryptografisches Verfahren, und bietet so maximalen Datenschutz. SSH benutzt Port 22.

1.2.4 Sonstiges

Das war nun ein kleiner Ausschnitt. Weiterhin gibt es natürlich noch viele verschiedene Protokolle, wie z. B. R-Dienste, WebDAV und HTTP Konsorten, auch klassiche LAN Protokolle können über das Internet getunnelt werden: Samba und NFS.

1.3 Arten von VPNs

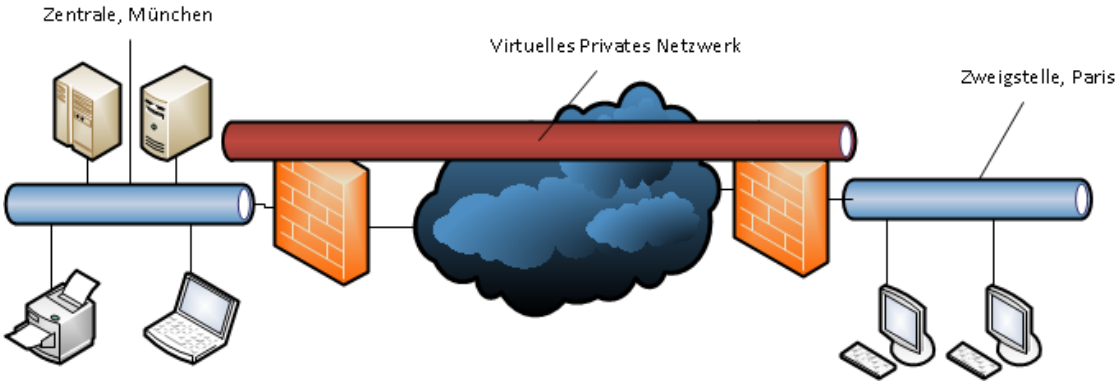
Grundsätzlich kann man in der Praxis 3 unterschiedliche Konzepte unterscheiden:

- Site-to-Site
- Host-to-Site, auch Remote-Access

- Host-to-Host

1.3.1 Site-to-Site

Diese Methode schließt zwei oder mehrere, geografisch getrennte Netzwerke zu einem privaten, logischen IP-Netz zusammen.

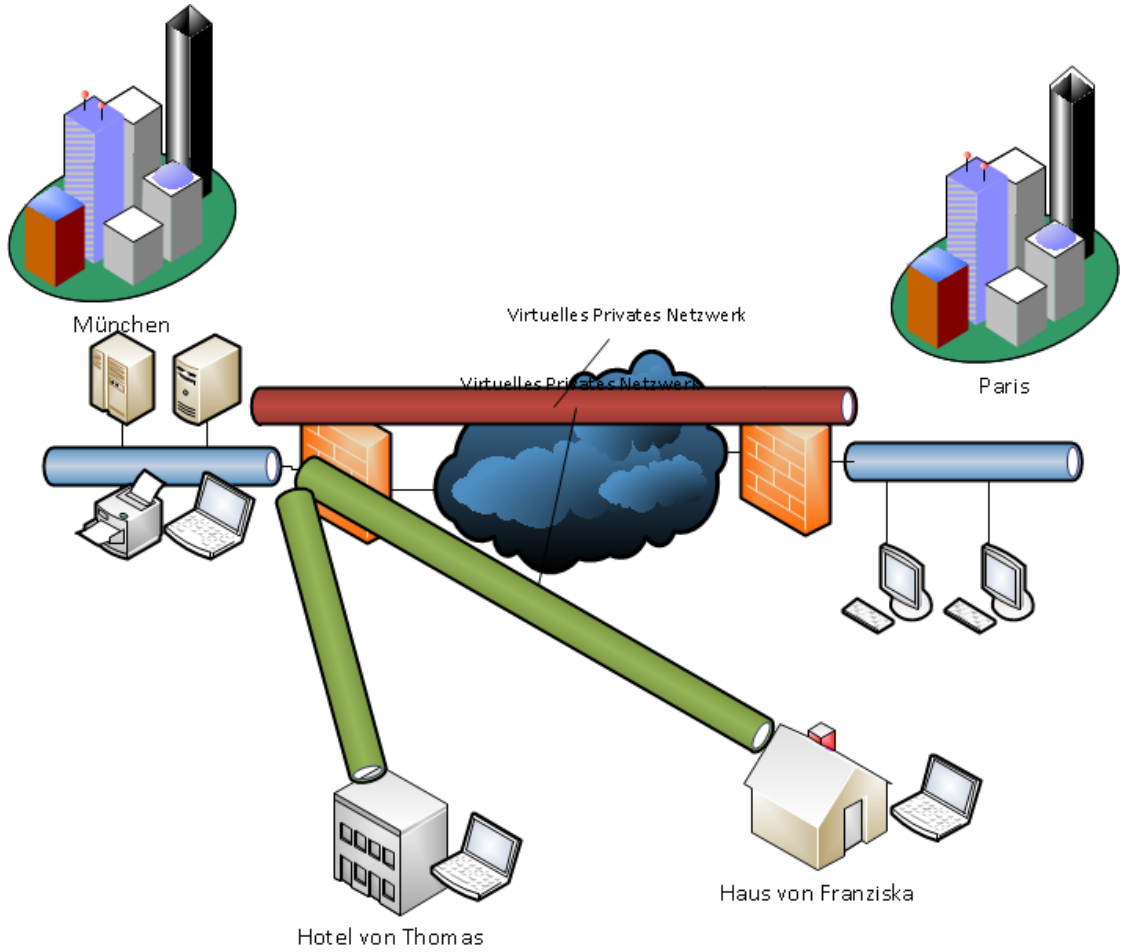


Fallbeispiel: Eine international tätige Organisation verbindet eine Verkaufsstelle in Frankreich, Paris, mit ihrer Firmenzentrale in Deutschland, München. Diese Art des

VPN ist eine Erweiterung des klassischen Weitverkersnetzes (z. B. Frame Relay). Bei Site-to-Site VPNs wird der Datenverkehr über VPN-Gateways in das Internet geleitet. Meist handelt es sich um einen Router oder eine Firewall. Das VPN Gateway verpackt die gesamten Datenpakete in ein Rahmenprotokoll, z. B. Generic Routing Encapsulation (GRE) und sendet die Daten über einen Tunnel. Das Gateway am anderen Standort entfernt die Verschlüsselung und leitet die Daten zum Empfänger im lokalen Netz (LAN).

1.3.2 Host-to-Site

Hierbei handelt es sich um die Verbindung eines einzelnen Clients in das Firmennetz. Über einen lokalen Internet Service Provider, einen Hot-Spot (offene WLan-Quellen), ein GSM-Netz oder ähnliches kann das mobile Endgerät sich in das Firmennetz gelangen.

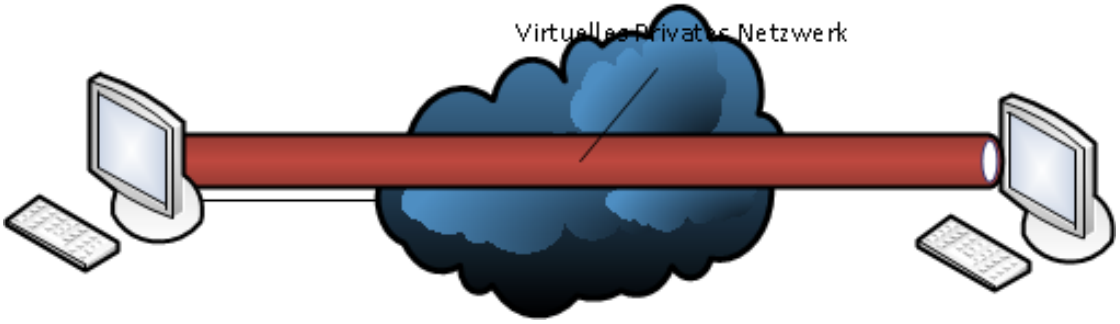


Fallbeispiel 1: Eine Mitarbeiterin, Petra, die von zu Hause aus arbeitet, kann somit auf alle Ressourcen des Firmennetzwerk sicher zugreifen.

Fallbeispiel 2: Thomas Müller, Verkaufsleiter einer großen Versicherung, kann sich unterwegs bei Kunden oder im Hotel in das Firmennetzwerk verbinden. Er hat somit Zugriff auf wichtige Ressourcen, beispielsweise dem Versicherungsdatenbankserver mit 14 Terabyte, der wichtige Prozentzahlen ausgibt.

1.3.3 Host-to-Host

Diese Variante des VPNs kommt wesentlich seltener vor. Dabei werden zwei Rechner direkt mittels einem VPN-Tunnel verbunden.



Fallbeispiel 1: Franz, der häufig noch Probleme im Umgang mit seiner Online-Banking-Software hat, wird mittels VPN von einem Supportmitarbeiter unterstützt.

Fallbeispiel 2: Der Server Paris benötigt von der 'Bundeszentrale für Finanzen' Daten, für komplexe Gehaltsabrechnungen. Dabei wird ein Tunnel aufgebaut, um die sensiblen Daten sicher zu transportieren.

1.4 Eigenschaften sicherer Verbindungen

Das Internet stellt ein unsicheres Übertragunsmedium dar. Niemand kann bestimmen oder weiß, über welche Routen seine Datenpakete fließen. Vormittags kann das Routing über Backbones von China gehen, nachmittags über Zentralserver in den USA. Pakete können dabei sehr leicht aufgezeichnet und mitprotokolliert werden. Handelt es sich dabei um ein Klartextprotokoll wie beispielsweise FTP, HTTP oder Telnet, so kann jeder die Daten ohne viel Mühe ausspähen. Das Passwortknacken wird somit zum Kinderspiel!
Mit VPN sollen Verbindungen über das Internet hergestellt werden. Die Sicherheit der übertragenen Daten spielen dabei eine primäre Rolle dar. VPN bietet Mechanismen für folgende Sicherheitsaspekte:

- Datenvertraulichkeit
- Datenintegrität
- Authentifizierumg

1.4.1 Datenvertraulichkeit

Die erste Anstelle eines sicheren Datenaustausches ist es, den Schutz der Daten vor Lauschangriffen nicht autorisierter dritter Personen zu gewährleisten. Sendet beispielsweise Petra vertrauliche Personaldaten über das VPN-Netzwerk, so muss sichergestellt sein, dass kein Unbefugter Zugang zu diesen Informationen bekommt. Dies wird erreicht, indem der Datenverkehr verschlüsselt wird.

1.4.2 Datenintegrität

Beim herkömmlichen Datenaustausch im Internet kann der Empfänger nicht gewährleisten, dass die Daten, die er erhalten hat, tatsächlich so gesendet wurden. Aufgrund

der unterschiedlichen Routen und Netzverbindungen kann nicht bestimmt werden, wer tatsächlich Zugriff auf die Daten hat. Ein dritter könnte somit die Originalnachrichten verschwinden lassen und gefälschte Datenpakete an den Empfänger senden. Es wäre fatal, wenn der Verkaufsleiter Thomas Müller beim Kunden falsche Preise bekommen würde, die eventuell ein Vertragsabkommen verhinden könnten. Die Datenintegrität stellt sicher, dass die empfangenen Daten auch tatsächlich so versendet wurden. VPN benutzt zu diesem Zweck Hash-Verfahren, zur Prüfung auf Korrektheit und Vollständigkeit

1.4.3 Authentifizierung

Die Authentifizierung stellt sicher, dass die Informationen und Daten auch tatsächlich aus der Quelle stammen, die vorgegeben wird. Es werden unterschiedliche Techniken verwendet, um Benutzer eindeutig zu identifizieren. Wichtigstes Merkmal sind Kennwörter, digitale Zertifikate, biometrische Nutzererkennung.

2 OpenVPN

2.1 Infos zu OpenVPN

OpenVPN ist eine VPN-Lösung für Linux, Windows und etlichen weiteren Betriebssystemen. Es handelt sich um eine freie Software, die unter der GPL (GNU General Public License) steht. Dies hat etliche Vorteile:

- darf ohne Einschränkungen eingesetzt werden, egal ob privat oder kommerziell
- der Quellcode kann eingesehen und angepasst werden

2.2 Verschlüsselung von OpenVPN

Die Verschlüsselung und Authentifizierung der Daten erfolgt durch kryptografische Verfahren auf Basis von TLS [1]. TLS ist ein Verschlüsselungsprotokoll zur Datenübertragung. Da OpenVPN TLS nutzt, kann IPSec nicht benutzt werden. Für die Authentifizierung können statische Schüssel oder Zertifikate benutzt werden.

statische Schlüssel sind einfacher in der Handhabung, sind allerdings nicht so sicher wie Zertifikate. Außerdem wir es sehr schwierig, bei einer großen Benutzeranzahl die statischen Schlüssel zu pflegen.

Zertifikate basieren auf dem asymmetrischen, Public-Key Verfahren. Um Zertifikate nutzen zu können, benötigt man eine Zertifizierungsstelle. Sie können eine kommerzielle, anerkannte Zertifizierungsstelle verwenden; es ist aber auch möglich, eine eigene Zertifizierungsstelle aufzumachen.

[1] Transport Layer Security, der Nachfolger von SSL (Secure Sockets Layer)

2.3 Verbindungsarten

OpenVPN ermöglicht 2 unterschiedliche Verbindungsarten, wie die lokalen Netze mittels VPN zusammenknüpft werden.

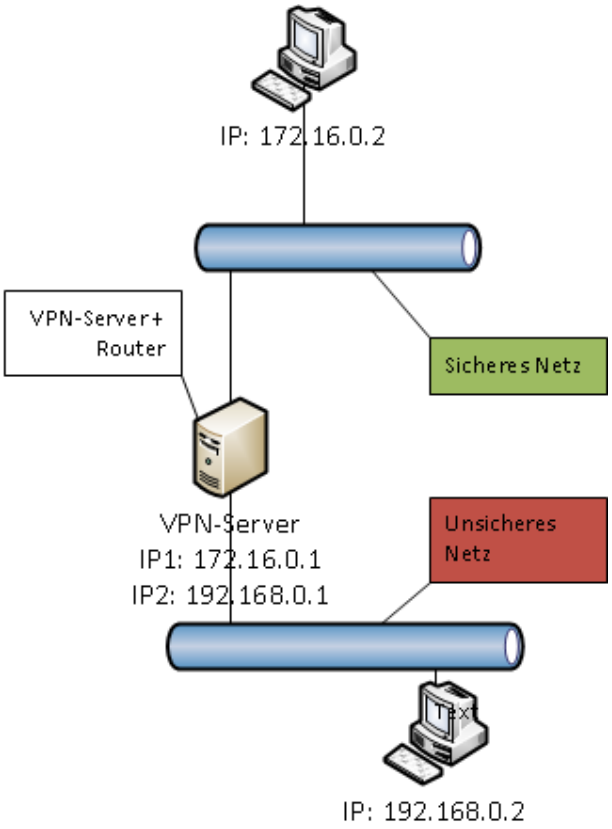
Routing-Modus virtuelle Tun-Devices ermöglichen die Verbindung zwischen zwei Gegenstellen. Dabei basiert es auf dem Prinzip des Routings.

Bridging-Modus Im Bridging-Modus werden 2 Netzwerkkarten miteinander verbunden. Dies wird meist über TAP-Devices gesteuert.

2.4 OpenVPN installieren und konfigurieren

2.4.1 Szenario

Die nächsten Zeilen beschäftigen sich mit der Installation von OpenVPN. Dabei verwendete ich ein Debian Lenny System. Zunächst noch ein Überblick der vorhandenen Netzwerkinfrastruktur:



2.4.2 Installation

Zunächst muss auf Ihrem Server OpenVPN isntalliert werden. Dies kann über die Paketverwaltung von Debian erledigt werden. Des weiteren ist es möglich, die Installation auch mittels manueller Übersetzung einzurichten.

```
$ apt-get install openvpn
```

Sofern Sie den Bridging-Modus verwenden wollen, müssen Sie noch das Paket `bridge-utils` installieren. Meine Anleitung bezieht sich auf den Routing-Modus, deshalb installiere ich dieses Paket nicht.

Um die Konfiguration zu erleichtern bekommen wir mit der Paketinstallation von openvpn bereits sehr viel Beispieldateien und Konfigurationsanleitungen. Diese sollten nun an die richtigen Stellen kopiert und entpackt werden.

```
$ cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /etc/openvpn/
$ gunzip /etc/openvpn/server.conf.gz
$ cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0 /etc/openvpn/easy-rsa2
```

2.4.3 Schlüssel und Zertifikate erstellen

Die gegenseitige Authentifizierung zwischen VPN-Server und VPN-Client findet über kryptografische Schlüssel und Zertifikate statt. Diese müssen nun erstellt werden. Im Verzeichnis `/etc/opvenvpn/easy-rsa2/` befindet sich nun eine Datei **vars**. Hier muss man nun die gewünschten Zertifikatsinformationen anpassen.

```
export KEY_COUNTRY="DE"
export KEY_PROVINCE="CY"
export KEY_CITY="Musterhausen"
export KEY_ORG="barthelconnect"
export KEY_EMAIL="christian@barthel-connect.de"
```

Die oben genannten Variablen dürften weitestgehend selbsterklärend sein. `KEY_COUNTRY`, `KEY_PROVINCE` und `KEY_CITY` beschreiben die geografischen Gegebenheiten. Bei Org kann der Organisationsname angegeben werden. Bei E-Mail eignet sich eine Service-Adresse der Firma. Des weiteren kann in dieser Datei noch etliches mehr eingestellt werden. Z. B. kann die `key_size` angepasst werden. Dabei handelt es sich um die Länge der Schlüssel. Standardmäßig ist dies auf 1024 eingestellt. `KEY_EXPIRE` gibt das Ablaufdatum der Zertifikate an.

Hat man nun die Datei `vars` angepasst, so muss das folgende Verzeichnis erstellt werden. Der zweite Befehl nimmt die Daten der Datei `vars` und erstellt die gewünschten Umgebungsvariablen:

```
$ mkdir /etc/openvpn/easy-rsa2/keys
$ source ./vars
```

Überspringen Sie die Warnmeldung und erstellen Sie danach das Master-Zertifikat und den Master-Schlüssel:

```
$ ./clean-all
$ ./build-ca
```

Server-Zertifikat

Nun kann damit begonnen werden, den Schlüssel und das Zertifikat für den Server anzulegen. Als Parameter der Anweisung `build-key-server` sollte der Servername gewählt werden. Da mein Hostname 'vienna' ist, übergebe ich diesen:

```
$ ./build-key-server vienna
```

Sollte Ihr Server unter einem anderem Namen aus dem Internet erreicht werden, z. B. einem DynDNS Name oder ähnliches, so muss dieser Name verwendet werden. Das 'Challenge Password' kann leer gelassen werden. Mittels zweimaliger Bestitigung der Taste Y wird nun die Datenbank aktualisiert.

Client-Zertifikate

Anschließend muss man für jeden VPN Client ebenfalls ein Zertifikat erstellen. Dieser Vorgang muss für jeden separaten Client ausgeführt werden. Ähnlich wie bei dem Serverzertifikat muss dem Befehl `build-key` der Hostname des Clients übergeben werden.

```
$ ./build-key client01
```

Sofern später weitere VPN Clients erstellt werden sollen, so geschieht dies auf dem gleichen Weg. Allerdings muss die Datei `vars` noch einmal 'gesourcet' werden.

Diffie-Hellman-Parameter

Als letztes muss der Diffie-Hellman-Parameter generiert werden. Diese werden verwendet, um kryptografische Schlüssel sicher über unsichere Kanäle auszutauschen.

```
$ ./build-dh
```

Keys im Überblick

Hat man nun alle Keys erzeugt, so sollte das ganze in etwa so aussehen:

```
vienna:/etc/openvpn/easy-rsa2/keys# ls -alh
total 88K
drwx------ 2 root root 4.0K 2010-12-21 09:17 .
drwxr-xr-x 3 root root 4.0K 2010-12-21 09:11 ..
-rw-r--r-- 1 root root 4.0K 2010-12-21 09:12 01.pem
-rw-r--r-- 1 root root 3.9K 2010-12-21 09:13 02.pem
```

```
-rw-r--r-- 1 root root 1.3K 2010-12-21 09:11 ca.crt
-rw------- 1 root root  887 2010-12-21 09:11 ca.key
-rw-r--r-- 1 root root 3.9K 2010-12-21 09:13 client01.crt
-rw-r--r-- 1 root root  696 2010-12-21 09:13 client01.csr
-rw------- 1 root root  887 2010-12-21 09:13 client01.key
-rw-r--r-- 1 root root  245 2010-12-21 09:13 dh1024.pem
-rw-r--r-- 1 root root  246 2010-12-21 09:13 index.txt
-rw-r--r-- 1 root root   20 2010-12-21 09:13 index.txt.attr
-rw-r--r-- 1 root root   21 2010-12-21 09:12 index.txt.attr.old
-rw-r--r-- 1 root root  122 2010-12-21 09:12 index.txt.old
-rw-r--r-- 1 root root    3 2010-12-21 09:13 serial
-rw-r--r-- 1 root root    3 2010-12-21 09:12 serial.old
-rw-r--r-- 1 root root 4.0K 2010-12-21 09:12 vienna.crt
-rw-r--r-- 1 root root  692 2010-12-21 09:12 vienna.csr
-rw------- 1 root root  887 2010-12-21 09:12 vienna.key
```

Die Datiendung **.key** signalisiert die geheimen Schlüssel. Server und Client besitzen dabei jeweils einen davon. Diese Key-Datei sollte nur auf dem entsprechenden Gerät gespeichert werden. Beispielsweise sollte nunt die Datei 'client01.key' auf dem Rechner 'client01' verschoben werden. Die **.crt**-Dateien sind Zertifikate, die öffentlich sind. Der **ca.key** befindet sich ebenfalls auf dem VPN-Server und sollte nicht auf die Client-Geräte transferiert werden.

Um nun die Dateien an unseren 'client01' übergeben zu können, ist es Empfehlenswert, diese in einem TAR-Archiv zu verpacken. Folgende Dateien müssen in das Archiv:

- client01.crt
- client01.key
- ca.crt

```
tar cfv client01.tar client01.crt client01.key ca.crt
```

Statischer Schlüssel

Sofern Sie sich für das Verfahren der symmetrischen Verschlüsselung entscheiden, benötigen Sie nur exakt einen Schlüssel. Dieser eine Schlüssel muss auf dem Server und allen Clients verteilt werden.

Vorteile:

- einfaches Setup, vorallem gegenüber den Zertifikaten/Public-Key-Verfahren
- Es ist keine X509 PKI-Infrastruktur notwendig

Nachteile:

- unflexibel und wenig skalierbar

- Sobald der Key geknackt wird, kann der gesamte Datenverkehr mitgelesen werden
- der Geheimschlüssel muss in Plaintext-Form auf jedem VPN-Knoten vorbehalten werden.
- Schlüsselaustausch ist sehr problematisch

2.4.4 Server-Konfiguration

Bevor nun der VPN-Server in Betrieb gehen kann, müssen noch diverse Pfäde angepasst werden. In der Datei `/etc/openvpn/server.conf` werden folgende Zeilen geändert:

```
ca ./easy-rsa2/keys/ca.crt
cert ./easy-rsa2/keys/vienna.crt
key ./easy-rsa2/keys/vienna.key
dh ./easy-rsa2/keys/dh1024.pem
```

Die `server.conf`-Datei ist sehr ausführlich dokumentiert und deshalb etwas länger. Die Keypfäde finden Sie etwa ab Zeile 70.

Außerdem ist es empfehlenswert, die Berechtigungen des OpenVPN-Benutzers zu reduzieren. Sie können folgende Zeilen, die standardmäßig auskommentiert sind, aktivieren, indem Sie das Semikolon vor `user` und `group` entfernen.

```
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nogroup
```

Sie können auch einen eigenen Benutzer für den OpenVPN-Dienst erstellen. Dabei legen Sie zunächst in der Systemverwaltung eine eigene, spezialisierte Identität an (z. B. openvpn) und setzen die Login-Shell auf '/bin/false'.

```
sudo adduser --system --no-create-home --disabled-login openvpn
sudo addgroup --system --no-create-home --disabled-login openvpn
```

Sofern der Server bei Ihnen eventuell in einem privatem Netzwerk (LAN) steht, müssen Sie noch eventuell Port-Forwarding am Router aktivieren. OpenVPN nutzt, sofern Sie nicht etwas anderes eingestellt haben, den Port 1194 und als Transportprotokoll UDP.

Der Server kann nun gestartet werden:

```
$ /etc/init.d/openvpn stop
$ /etc/init.d/openvpn start
```

Sollte der Server nicht wie gewünscht starte, so springen Sie bitte in das Kapitel 'Logging'. Hier werden Troubleshooting-Tipps und Logdateien erklärt.

Prüfen und kontrollieren

Um zu prüfen, ob der Server korrekt gestartet wurde können wir uns zunächst einen Auszug aus der Prozesstabelle ausgeben lassen (gefiltert nach OpenVPN):

```
vienna:~# ps aux | grep openvpn
root      4526  0.0  0.7   6176  3952 ?        Ss   20:07   0:00 /usr/sbin/openvpn
--writepid /var/run/openvpn.server.pid --daemon ovpn-server
--cd /etc/openvpn --config /etc/openvpn/server.conf
```

Hier sehen wir also schon, mit welchen Parametern OpenVPN gestartet wurde. Als nächstes können wir uns noch kurz ansehen, ob auch der Port 1194 erfolgreich geöffnet wurde:

```
vienna:~# netstat -panu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address   PID/Program name
udp        0      0 0.0.0.0:1194    0.0.0.0:*         4526/openvpn
```

Auch hier sieht es soweit ganz gut aus. Den virtuellen Netzadapter (/dev/tun) können Sie mittels ifconfig genauer unter die Lupe nehmen:

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.1  P-t-P:10.8.0.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:19 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1228 (1.1 KiB)  TX bytes:1090 (1.0 KiB)
```

Der OpenVPN-Server hat die IP-Adresse 10.8.0.1.

Proxys und Firewalls umgehen

Häufig kommt es vor, dass ihre Benutzer nicht direkt mit dem Internet verbunden sind. Meist wählt man sich über Proxy, Gatways mit Firewalls oder anderen Geräten in das Internet ein. Dabei kann es vorkommen, dass der Port 1194 geblockt wird, und es nicht möglich ist, eine VPN-Verbindung herzustellen. Für solche Zwecke ist es sehr praktisch, wenn Sie in der `server.conf` die Portangabe auf 443 setzen. Dies ist der HTTPS-Port, der im Netz weitestgehend benutzbar sein sollte. Theoretisch könnte auch jeder andere Port gewählt werden (z. B. 80), praktisch wird dies aber in Firewall- und Proxyumgebungen nicht funktionieren, da diese die Benutzung auf HTTP CONNECT limitieren.

Die Einstellungen der Server-Konfiguration sehen dann so aus:

```
port 443
;port 1194

# TCP or UDP server?
proto tcp
;proto udp
```

Bitte vergessen Sie auch nicht, dass Sie dann in der `client.conf` die gleichen Port-Einstellungen setzen müssen:

```
# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
proto tcp
;proto udp

[...]
# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here.  See the man page
# if your proxy server requires
# authentication.
http-proxy 192.168.4.1 3129
```

Hier trägt der Proxyserver des Clients die IP 192.168.4.1. Außerdem wird dieser Proxy auf Port 3129 erreicht. Diese Einstellungen können natürlich in jedem Netzwerk, in dem sich der Client befindet, verschieden sein!

Routing-Konfiguration

In der bisherigen Konfiguration ist es nur möglich, den VPN-Server zu erreichen. Alle ausgehenden Dienste dieses Servers können abgerufen werden, es ist allerdings nicht möglich, weitere Clients über den VPN-Server zu erreichen. Konkret heits das nun, dass der Client aus Szenario 2.4.1 nur den VPN-Server sicher erreichen kann. Das zweite LAN - 172.16.0.0 des VPN-Servers, ist nicht erreichbar .Um den Clients auch dieses LAN zugänglich zu machen, muss eine weitere Route in der `server.conf` Datei eingepflegt werden.

```
push "route 172.16.0.0 255.255.255.0"
```

Diese Route wird dem Client mitgeteilt, sobald er sich mit dem VPN-Server verbindet. Der Client weiß dann, dass er das Netzwerk 172.16.0.0 über den VPN-Server erreichen kann.

Standardmäßig sind Debian-Systeme so eingestellt, dass Sie den IP-Verkehr nicht routen. Dies muss noch nachträglich umgestellt werden, damit aus unserem VPN-Server tatsächlich ein VPN-Router/Gateway wird:

```
$ sysctl -w net/ipv4/ip_forward=1
```

Um das ganze permanent auch nach einem Systemstart zu aktivieren, kann die Datei `/etc7sysctl.conf` wie folgt editiert werden:

```
net.ipv4.ip_forward=1
```

Sofern Ihre Netzwerkkonfiguration von meiner abweicht, müssen Sie gegebenfalls noch weitere Routen hinzufügen. Beachten Sie, dass das Routing hier eine wichtige Rolle spielt und eine häufige Fehlerquelle darstellt.

Konfiguration mit statischen Schlüsseln

Sofern Sie im vorigen Abschnitt lediglich einen privaten Schlüssel erstellt haben, können Sie folgende Serverkonfiguration verwenden:

```
server.conf:
dev tun
ifconfig 10.8.0.1 10.8.0.2
secret static.key
```

Log-Files

Standardmäßig werden die Log-Nachrichten unter Debian an 'syslog' gesendet. Diese Datei finden Sie unter `/var/log/syslog`. Hier sehen Sie einen sehr detaillierten Beispielauszug

```
Dec 24 17:32:14 vienna ovpn-server[9976]: client01/192.168.0.2:35290 TLS: soft reset sec=0 bytes=487312/0 pkts=2695/0
Dec 24 17:32:14 vienna ovpn-server[9976]: client01/192.168.0.2:35290 VERIFY OK: depth=1,
/C=DE/ST=BY/L=Tittmoning/O=barthelconnect/CN=barthelconnect_CA/emailAddress=christian@barthel-connect.de
Dec 24 17:32:14 vienna ovpn-server[9976]: client01/192.168.0.2:35290 VERIFY OK: depth=0,
/C=DE/ST=BY/L=Tittmoning/O=barthelconnect/CN=client01/emailAddress=christian@barthel-connect.de
Dec 24 17:32:15 vienna ovpn-server[9976]: client01/192.168.0.2:35290 Data Channel Encrypt: Cipher 'BF-CBC' initialized with 128 bit key
Dec 24 17:32:15 vienna ovpn-server[9976]: client01/192.168.0.2:35290 Data Channel Encrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
Dec 24 17:32:15 vienna ovpn-server[9976]: client01/192.168.0.2:35290 Data Channel Decrypt: Cipher 'BF-CBC' initialized with 128 bit key
Dec 24 17:32:15 vienna ovpn-server[9976]: client01/192.168.0.2:35290 Data Channel Decrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
Dec 24 17:32:15 vienna ovpn-server[9976]: client01/192.168.0.2:35290 Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 1024 bit RSA
Dec 24 17:37:19 vienna ovpn-server[9976]: read UDPv4 [EHOSTUNREACH]: No route to host (code=113)
Dec 24 17:37:29 vienna ovpn-server[9976]: read UDPv4 [EHOSTUNREACH]: No route to host (code=113)
Dec 24 17:37:39 vienna ovpn-server[9976]: read UDPv4 [EHOSTUNREACH]: No route to host (code=113)
Dec 24 17:37:40 vienna kernel: [37380.125980] VMCIUtil: Updating context id from 0xf2c02e70 to 0xf2c02e70 on event 0.
Dec 25 08:42:07 vienna ovpn-server[9976]: client01/192.168.0.2:35290 [client01] Inactivity timeout (--ping-restart), restarting
Dec 25 08:42:07 vienna ovpn-server[9976]: client01/192.168.0.2:35290 SIGUSR1[soft,ping-restart] received, client-instance restarting
Dec 25 08:43:21 vienna ovpn-server[9976]: MULTI: multi_create_instance called
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Re-using SSL/TLS context
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 LZO compression initialized
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Control Channel MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Data Channel MTU parms [ L:1542 D:1450 EF:42 EB:135 ET:0 EL:0 AF:3/1 ]
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Local Options hash (VER=V4): '530fdded'
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Expected Remote Options hash (VER=V4): '41690919'
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 TLS: Initial packet from 192.168.0.2:60837, sid=9c69e022 6ddad51f
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 VERIFY OK: depth=1,
/C=DE/ST=BY/L=Tittmoning/O=barthelconnect/CN=barthelconnect_CA/emailAddress=christian@barthel-connect.de
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 VERIFY OK: depth=0,
/C=DE/ST=BY/L=Tittmoning/O=barthelconnect/CN=client01/emailAddress=christian@barthel-connect.de
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Data Channel Encrypt: Cipher 'BF-CBC' initialized with 128 bit key
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Data Channel Encrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
```

```
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Data Channel Decrypt: Cipher 'BF-CBC' initialized with 128 bit key
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Data Channel Decrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 1024 bit RSA
Dec 25 08:43:21 vienna ovpn-server[9976]: 192.168.0.2:60837 [client01] Peer Connection Initiated with 192.168.0.2:60837
Dec 25 08:43:21 vienna ovpn-server[9976]: client01/192.168.0.2:60837 MULTI: Learn: 10.8.0.6 -> client01/192.168.0.2:60837
Dec 25 08:43:21 vienna ovpn-server[9976]: client01/192.168.0.2:60837 MULTI: primary virtual IP for client01/192.168.0.2:60837: 10.8.0.6
Dec 25 08:43:24 vienna ovpn-server[9976]: client01/192.168.0.2:60837 PUSH: Received control message: 'PUSH_REQUEST'
Dec 25 08:43:24 vienna ovpn-server[9976]: client01/192.168.0.2:60837 SENT CONTROL [client01]:
'PUSH_REPLY,route 172.16.0.0 255.255.255.0,route 10.8.0.1,topology net30,ping 10,ping-restart 120,ifconfig 10.8.0.6 10.8.0.5' (status=1)
```

Sofern Sie nicht nach `/var/log/syslog` schreiben möchten, können Sie dies in der Datei `server.conf` angeben. Kommentieren Sie die Zeile aus, wenn Sie eine eigene 'openvpn.log'-Datei haben wollen:

```
# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it.  Use one
# or the other (but not both).
;log         openvpn.log
;log-append  openvpn.log
```

Per Default gibt es auch eine `openvpn-status.log`-Datei, die die aktuellen Client-Verbindungen auflistet. Die Datei befindet sich standardmäßig unter `/etc/openvpn/openvpn-stats.l`
In der server.conf können Sie ebenfalls den Pfad zu dieser Datei anpassen:

```
# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log
```

Beispiel output dieser Datei:

```
OpenVPN CLIENT LIST
Updated,Sat Dec 25 08:56:17 2010
Common Name,Real Address,Bytes Received,Bytes Sent,Connected Since
client01,192.168.0.2:60837,8191,9527,Sat Dec 25 08:43:21 2010
ROUTING TABLE
Virtual Address,Common Name,Real Address,Last Ref
10.8.0.6,client01,192.168.0.2:60837,Sat Dec 25 08:43:21 2010
GLOBAL STATS
Max bcast/mcast queue length,0
END
```

Wir sehen aktuell eine Verbindung, deren tatsächliche IP-Adresse die 192.168.0.2 ist. Virtuell wurde diesem client die 10.8.0.6 zugewiesen.

Management-Interface aktivieren

OpenVPN besitzt ein Kommandozeilenmanagementprogramm, mit dem man zur Laufzeit unterschiedliche Einstellungen und Eigenschaften setzen kann. Dies ist standardmäßig deaktivert. Um es zu aktivieren müssen Sie folgende Zeile in Ihrer `server.conf` Datei hinzufügen

```
management localhost 7505
```

Das bedeutet nun, dass das Management-Interface nur über die lokale Netzwerkschnittstelle auf Port 7505 zur Verfügung steht. Da es sich unter anderem um Telnet handelt, sollte dies auch lokal behalten werden. Eine Verbindung zu dieser Konsole kann wie folgt hergestellt werden:

```
vienna:/var/log# telnet localhost 7505
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
>INFO:OpenVPN Management Interface Version 1 -- type 'help' for more info
help
Management Interface for OpenVPN 2.1_rc11 i486-pc-linux-gnu [SSL] [LZO2] [EPOLL] [PKCS11] built
Commands:
auth-retry t           : Auth failure retry mode (none,interact,nointeract).
bytecount n            : Show bytes in/out, update every n secs (0=off).
echo [on|off] [N|all]  : Like log, but only show messages in echo buffer.
exit|quit              : Close management session.
forget-passwords       : Forget passwords entered so far.
help                   : Print this message.
hold [on|off|release]  : Set/show hold flag to on/off state, or
                         release current hold and start tunnel.
kill cn                : Kill the client instance(s) having common name cn.
kill IP:port           : Kill the client instance connecting from IP:port.
log [on|off] [N|all]   : Turn on/off realtime log display
                         + show last N lines or 'all' for entire history.
mute [n]               : Set log mute level to n, or show level if n is absent.
needok type action     : Enter confirmation for NEED-OK request of 'type',
                         where action = 'ok' or 'cancel'.
needstr type action    : Enter confirmation for NEED-STR request of 'type',
                         where action is reply string.
net                    : (Windows only) Show network info and routing table.
password type p        : Enter password p for a queried OpenVPN password.
pkcs11-id-count        : Get number of available PKCS#11 identities.
pkcs11-id-get index    : Get PKCS#11 identity at index.
client-auth CID KID    : Authenticate client-id/key-id CID/KID (MULTILINE)
client-auth-nt CID KID : Authenticate client-id/key-id CID/KID
```

```
client-deny CID KID R  : Deny auth client-id/key-id CID/KID with reason text R
client-kill CID        : Kill client instance CID
client-pf CID          : Define packet filter for client CID (MULTILINE)
signal s               : Send signal s to daemon,
                         s = SIGHUP|SIGTERM|SIGUSR1|SIGUSR2.
state [on|off] [N|all] : Like log, but show state history.
status [n]             : Show current daemon status info using format #n.
test n                 : Produce n lines of output for testing/debugging.
username type u        : Enter username u for a queried OpenVPN username.
verb [n]               : Set log verbosity level to n, or show if n is absent.
version                : Show current version number.
END
```

Einen Befehl absenden (listet aktuelle Client-Verbindungen auf)

```
status
OpenVPN CLIENT LIST
Updated,Sat Dec 25 09:16:48 2010
Common Name,Real Address,Bytes Received,Bytes Sent,Connected Since
client01,192.168.0.2:35839,4852,6188,Sat Dec 25 09:14:30 2010
ROUTING TABLE
Virtual Address,Common Name,Real Address,Last Ref
10.8.0.6,client01,192.168.0.2:35839,Sat Dec 25 09:14:30 2010
GLOBAL STATS
Max bcast/mcast queue length,0
END
```

2.5 Client-Konfiguration

Da nun der Server gestartet ist, können wir nun eine sichere VPN-Verbindung herstellen. Dazu muss zunächst auch auf dem Client das Paket `openvpn` an. Möchte man die grafische VPN-Oberfläche des 'gnome-network-managers' nutzen, so kann ebenfalls noch das Paket `network-manager-openvpn-gnome` und `network-manager-openvpn` installiert werden.

```
$ apt-get install openvpn
```

Nach der Installation kann man sich nun wieder eine Beispiel-Konfiguration in das Verzeichnis `/etc/openvpn` kopieren. Dabei handelt es sich allerdings nicht um die `server.conf` sondern um die Datei für Client-Geräte, nämlich `client.conf`

```
$ cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/openvpn
```

In der Datei `/etc/openvpn/client.conf` müssen dann zunächst wieder die Pfadangaben zu den richtigen Zertifikaten angegeben werden. In meiner Konfiguration habe ich

sämtliche Zertifikate, die ich zuvor in ein TAR-Archiv verpackt habe, nach `/etc/openvpn/` extrahiert.

```
client01 ~ # ls -alh /etc/openvpn/
total 36K
drwxr-xr-x   2 root root 4.0K 2010-12-21 10:17 .
drwxr-xr-x 146 root root  12K 2010-12-24 08:43 ..
-rw-r--r--   1 root root 1.3K 2010-12-21 10:17 ca.crt
-rw-r--r--   1 root root 3.9K 2010-12-21 10:17 client01.crt
-rw-------   1 root root  887 2010-12-21 10:17 client01.key
-rw-r--r--   1 root root 3.4K 2010-12-21 10:16 client.conf
-rwxr-xr-x   1 root root 1.4K 2010-07-12 15:52 update-resolv-conf

client01 ~ # cat client.conf
# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.

ca ca.crt
cert client01.crt
key client01.key
```

Des weiteren muss noch der VPN-Server angepasset werden. Dies geschieht ebenfalls in der Datei `client.conf`, nämlich in der Zeile

```
# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote 192.168.0.1 1194
```

Ich habe die direkte IP-Adresse des VPN-Servers angegeben, Sie können aber natürlich auch einen DNS-Namen verwenden, der auf den Server zeigt.

Nun kann bereits der Client Verbindung aufnehmen. Starten Sie dazu einfach `openvpn` wie folgt:

```
$ /etc/init.d/openvpn stop
$ /etc/init.d/openvpn start
```

Prüfen und kontrollieren

Nach einer kurzen Prüfung sehen wir, dass der Client sich erfolgreich verbunden hat:

```
client01 ~ # ps aux | grep vpn
root      3952  0.1  0.5   5312  09:26   0:00 /usr/sbin/openvpn --writepid
/var/run/openvpn.client.pid --daemon ovpn-client --status
/var/run/openvpn.client.status 10 --cd /etc/openvpn
--config /etc/openvpn/client.conf --script-security 2
```

Auch die Überprüfung der Netzwerkkonfiguration sollte ein neues `tun0`-Device auflisten:

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.8.0.6  P-t-P:10.8.0.5  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  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:100
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Um nun den Server mittels VPN eine ICMP-Message (PING-Paket) zu senden, müssen wir dessen IP-Adresse, die 10.8.0.1 verwenden:

```
client01 ~ # ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_req=1 ttl=64 time=0.976 ms
64 bytes from 10.8.0.1: icmp_req=2 ttl=64 time=1.25 ms
64 bytes from 10.8.0.1: icmp_req=3 ttl=64 time=0.673 ms
^C
--- 10.8.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.673/0.966/1.251/0.238 ms
```

Auch dieser Test funktioniert!

Noch ein Tipp: Sofern Sie VPN nicht bei jedem Systemstart automatisch starten lassen wollen, können Sie in der Datei `/etc/default/openvpn` folgende Zeile aktivieren:

```
AUTOSTART="none"
```

Sofern Sie dann zur gewünschten Zeit die VPN Verbindung herstellen möchten, können Sie einfach als Administrator folgenden Befehl absetzen

```
$ openvpn /etc/openvpn/client.conf
```

Sofern Sie Benutzer ohne ROOT-Rechte haben, ist es sinnvoll, die VPN-Konfigurationsdaten in einem Homeverzeichnis zu speichern. Beispielsweise kann ein Verzeichnis `/home/username/.openvpn` erstellt werden, um einem normalen Benutzer Zugriff auf die Dateien zu ermöglichen. Gestartet wird dann wie folgt:

```
client01 ~# openvpn /home/username/.openvpn/client.conf
```

Client-Konfiguration

Wie schon bei der Server-Konfiguration folgt hier noch der Abschnitt, falls Sie nur einen statischen Schlüssel verwenden wollen.

```
remote serveradresse.de
dev tun
ifconfig 10.8.0.2 10.8.0.1
secret static.key
```

2.5.1 Test der Security

Um nun den tatsächlichen Sicherheitsgewinn zu testen, habe ich zusätzlich einen FTP-Server auf meinem VPN-Server isntalliert. Es handelt sich um den `vsftpd`-Server. Dieser dient hauptsächlich zum Testen. FTP ist ein Klartext-Protokoll. Der gesamte Verkehr erfolgt ohne Verschlüsselung und kann somit mitgelesen werden. Mit einem geeignetem Programm, zum Beispiel Wireshark, dsniff oder tcpdump, können Sie diesen Netzwerkverkehr mitprotokollierenn. Sehr schön kann man in der Abbildung die Zeile, die das

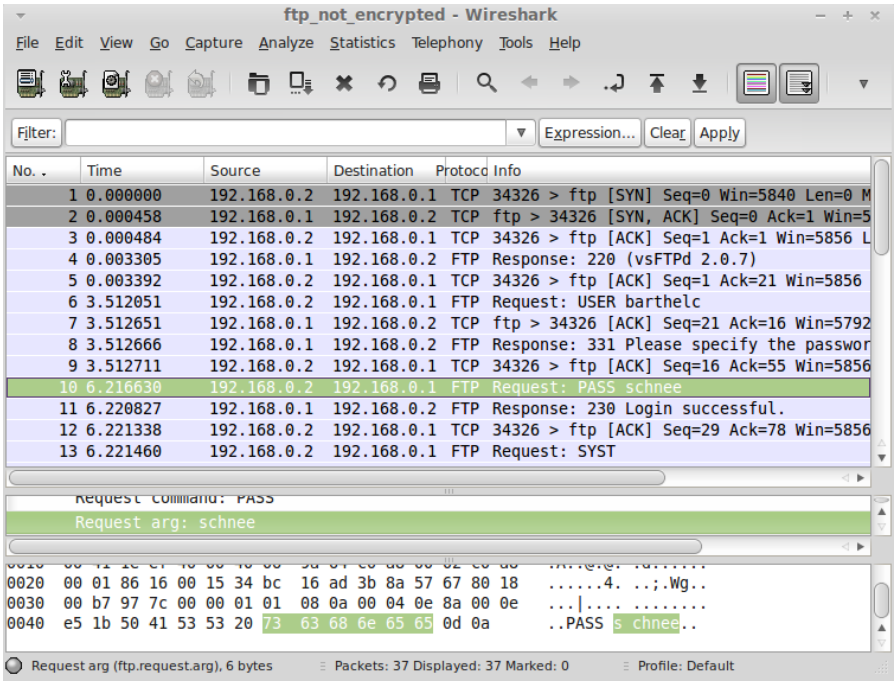


Abbildung 2.1: Wireshark, unverschlüsselter Datenverkehr

Passwort enthält, auslesen. Auch die kompletten transferierten Daten sind natürlich öffentlich einsehbar. Sehen wir uns nun das ganze auf einem verschlüsselten Weg an.
Wie Sie in der Abbildung sehen, sehen Sie nicht mehr den detaillierten Netzwerkverkehr von FTP. Obwohl hier exakt die selben Befehle abgesendet wurden, können wir nur

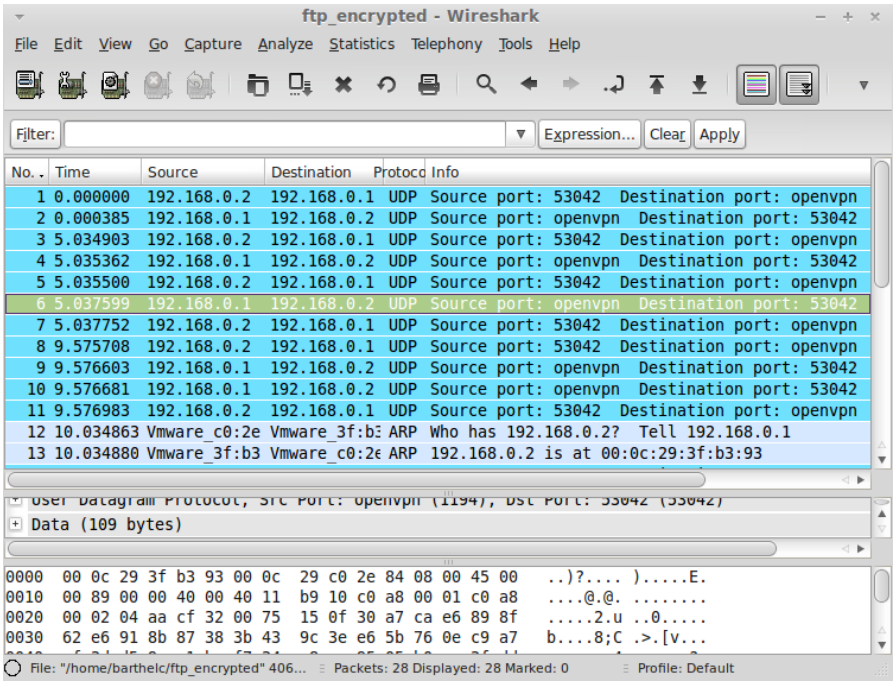


Abbildung 2.2: Wireshark, verschlüsselter FTP-Datenverkehr

noch aus den Portangaben herauslesen, dass es sich um openvpn-Pakete handeln muss. Benutzername, Kennwörter oder sonstige übertragene Daten sind nun verschlüsselt und können nicht ohne weiteres gelesen werden.

2.5.2 GUI - network-manager-openvpn

Unter Gnome kann ebenfalls der `network-manager` verwendet werden, um VPN-Verbindungen zu verwalten. Dabei muss zunächst mittels apt-get folgendes Paket installiert werden:

```
$ apt-get install network-manager-openvpn network-manager-openvpn-gnome openvpn
```

In der Abbildung sehen Sie nochmals die Dateien, die benötigt werden, um einen VPN-Client einzurichten.

Um nun den Network-Manager einzurichten, klicken Sie auf das Symbol und wählen ‘configure VPN...’ aus (Bild). In dem folgendem Dialogfenster klicken Sie auf ‘Add’.

Im nächsten Dialog müssen Sie nun folgende Einstellungen tätigen.

Connection name Der Name der Verbindung. Dieser kann willkürlich gewählt werden, z. B. vpncompany oder vpnhome.

Gateway Die IP-Adresse oder der DNS-Name, unterdem der Ziel-VPN-Server erreichbar ist.

Type Muss auf ‘Certificates (TLS)’ eingestellt werden.

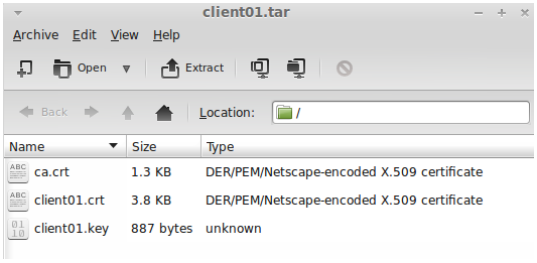


Abbildung 2.3: VPN-Client-Dateien

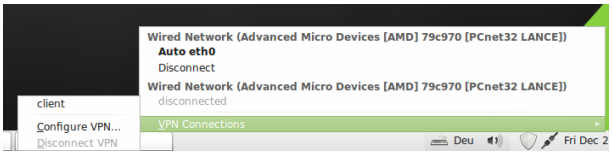


Abbildung 2.4: Network-Manager

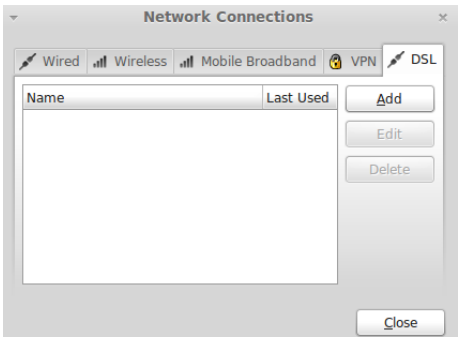


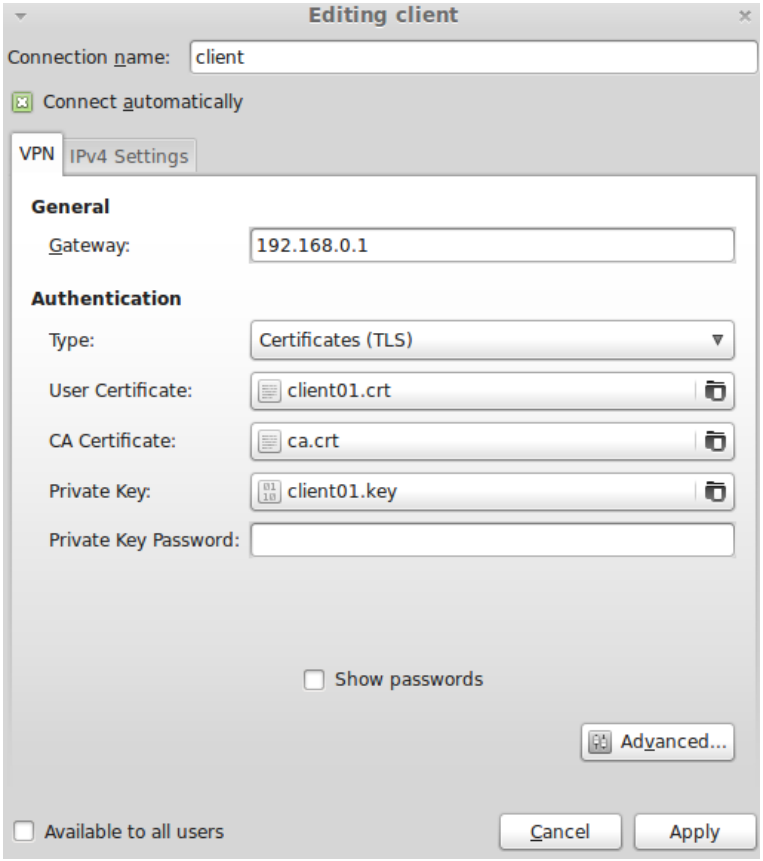
Abbildung 2.5: Hier muss nun eine neue VPN-Verbindung eingerichtet werden

User Certificate Das Zertifikat, dass wir zuvor für den client erstellt haben. In meinem Fall `client01.crt`.

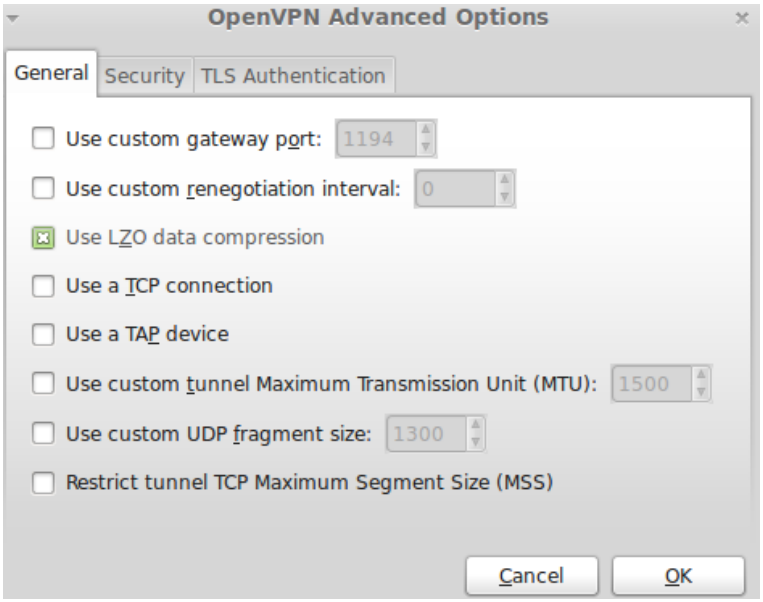
CA Certificate Hier muss die Datei 'ca.crt' angegeben werden (Pfad)

Private Key Die Datei 'client01.key' beschreibt den privaten Schlüssel.

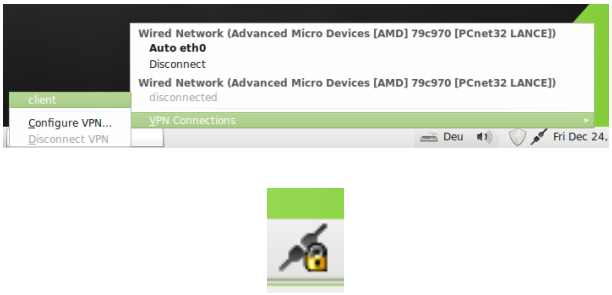
Die kompletten Einstellungen sehen Sie in folgender Abbildung.



Um eine erfolgreiche Verbindung aufzubauen, muss noch unter den Advanced-Settings die Einstellung 'Use LZO data compression' aktiviert werden.



Sofern alle Angaben passen, können Sie mit einem Klick auf die VPN-Verbindung im network-manager die Verbindung aufbauen.



2.5.3 Microsoft Windows

OpenVPN exisitert auch für Windows-Systeme. Sie können sogar einen VPN-Server komplett mit Microsoft Windows realisieren. Im folgenden möchte ich nur kurz eine Client-Konfiugration mit Windows XP vorstellen.
Die benötigte Installationsdatei kann unter `http://openvpn.se/download.html` heruntergeladen werden. Wählen sie dabei 'Applicatin only' aus, um die VPN-Client-Komponente zu bekommen. Nachdem Sie den Installationsanweisungen gefolgt sind, können sie in das Verzeichnis `C:\\Programme\OpenVPN\config\` die Client-Zertifikate sowie da CA-Zertifikat kopieren. Wichtig ist, dass die Datei `client.conf` umbenannt wird, nach `client.ovpn`. Ist dies geschen, so kann mit der rechten Maustaste auf das OpenVPN mittels 'Connect' die Verbindung hergestellt werden. Sofern Sie die VPN-Verbindung beim Systemstart herstellen wollen, können Sie mittels `services.msc` OpenVPN auch als Dienst beim Systemstart automatisch ausführen lassen.

2.5.4 Troubleshooting-Tipps

2.6 Meine Beispieldateien

Folgende Dateien sollten an diesen Orten aufbewahrt werden:

- Server
  - ca.crt
  - ca.key
  - servername.crt
  - servername.key
  - servername.csr
  - dh1024.pem
  - server.conf
- pro Client
  - ca.crt
  - client01.crt
  - client01.key
  - client.conf

2.6.1 server.conf

```
################################################
# Sample OpenVPN 2.0 config file for           #
# multi-client server.                         #
#                                              #
# This file is for the server side             #
# of a many-clients <-> one-server             #
# OpenVPN configuration.                       #
#                                              #
# OpenVPN also supports                        #
# single-machine <-> single-machine            #
# configurations (See the Examples page        #
# on the web site for more info).              #
#                                              #
# This config should work on Windows           #
# or Linux/BSD systems.  Remember on           #
# Windows to quote pathnames and use           #
# double backslashes, e.g.:                    #
# "C:\\Program Files\\OpenVPN\\config\\foo.key" #
#                                              #
```

```
# Comments are preceded with '#' or ';'         #
################################################

# Which local IP address should OpenVPN
# listen on? (optional)
;local a.b.c.d

# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one.  You will need to
# open up this port on your firewall.
port 1194

# TCP or UDP server?
;proto tcp
proto udp

# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an ethernet tunnel.
# Use "dev tap0" if you are ethernet bridging
# and have precreated a tap0 virtual interface
# and bridged it with your ethernet interface.
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one.  On XP SP2 or higher,
# you may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
```

```
# (cert), and private key (key).  Each client
# and the server must have their own cert and
# key file.  The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys.  Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca ./easy-rsa2/keys/ca.crt
cert ./easy-rsa2/keys/vienna.crt
key ./easy-rsa2/keys/vienna.key  # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
#   openssl dhparam -out dh1024.pem 1024
# Substitute 2048 for 1024 if you are using
# 2048 bit keys.
dh ./easy-rsa2/keys/dh1024.pem

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
# associations in this file.  If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt

# Configure server mode for ethernet bridging.
# You must first use your OS's bridging capability
# to bridge the TAP interface with the ethernet
```

```
# NIC interface.  Then you must manually set the
# IP/netmask on the bridge interface, here we
# assume 10.8.0.4/255.255.255.0.  Finally we
# must set aside an IP range in this subnet
# (start=10.8.0.50 end=10.8.0.100) to allocate
# to connecting clients.  Leave this line commented
# out unless you are ethernet bridging.
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100

# Configure server mode for ethernet bridging
# using a DHCP-proxy, where clients talk
# to the OpenVPN server-side DHCP server
# to receive their IP address allocation
# and DNS server addresses.  You must first use
# your OS's bridging capability to bridge the TAP
# interface with the ethernet NIC interface.
# Note: this mode only works on clients (such as
# Windows), where the client-side TAP adapter is
# bound to a DHCP client.
;server-bridge

# Push routes to the client to allow it
# to reach other private subnets behind
# the server.  Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
push "route 172.16.0.0 255.255.255.0"
;push "route 192.168.10.0 255.255.255.0"
;push "route 192.168.20.0 255.255.255.0"

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files (see man page for more info).

# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
```

```
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
#   iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN.  This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
#   ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
# of clients.  There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
#     group, and firewall the TUN/TAP interface
#     for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
#     modify the firewall in response to access
#     from different clients.  See man
#     page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# or bridge the TUN/TAP interface to the internet
# in order for this to work properly).
;push "redirect-gateway def1 bypass-dhcp"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses.  CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
# The addresses below refer to the public
# DNS servers provided by opendns.com.
```

```
;push "dhcp-option DNS 208.67.222.222"
;push "dhcp-option DNS 208.67.220.220"

# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
;client-to-client

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names.  This is recommended
# only for testing purposes.  For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
```

```
;tls-auth ta.key 0 # This file is secret

# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC        # Blowfish (default)
;cipher AES-128-CBC   # AES
;cipher DES-EDE3-CBC  # Triple-DES

# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nogroup

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it.  Use one
```

```
# or the other (but not both).
;log         openvpn.log
;log-append  openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3

# Silence repeating messages.  At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20
```

2.6.2 client.conf

```
################################################
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server.     #
#                                            #
# This configuration can be used by multiple #
# clients, however each client should have   #
# its own cert and key files.                #
#                                            #
# On Windows, you might want to rename this  #
# file so it has a .ovpn extension           #
################################################

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun
```

```
# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel
# if you have more than one.  On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
;dev-node MyTap

# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
;proto tcp
proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote 192.168.0.1 1194
;remote my-server-2 1194

# Choose a random host from the remote
# list for load-balancing.  Otherwise
# try hosts in the order specified.
;remote-random

# Keep trying indefinitely to resolve the
# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
;user nobody
;group nogroup

# Try to preserve some state across restarts.
persist-key
persist-tun

# If you are connecting through an
```

```
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here.  See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca ca.crt
cert client01.crt
key client01.key

# Verify server certificate by checking
# that the certicate has the nsCertType
# field set to "server".  This is an
# important precaution to protect against
# a potential attack discussed here:
#  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the nsCertType
# field set to "server".  The build-key-server
# script in the easy-rsa folder will do this.
ns-cert-type server

# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
;cipher x
```

```
# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
comp-lzo

# Set log file verbosity.
verb 3

# Silence repeating messages
;mute 20
```

2.6.3 server.conf für statische Schlüssel

```
dev tun
ifconfig 10.8.0.1 10.8.0.2
secret static.key
```

2.6.4 client.conf für statische Schlüssel

```
remote servername.de
dev tun
ifconfig 10.8.0.2 10.8.0.1
secret static.key
```