OpenVPN ist eine Virtual-Private-Network-Software, die auf dem bewährten Verschlüsselungsprotokoll SSL/TLS aufbaut, welches auch für die Verschlüsselung von Webbrowser-Sitzungen im E-Commerce eingesetzt wird.

Die Vorteile von OpenVPN gegenüber anderen Lösungen liegen in der (relativ) einfachen Konfiguration und der Verfügbarkeit für zahlreiche Plattformen (u.a. Linux, Solaris, versch. BSDs, Mac OS X und Microsoft Windows) bei gleichzeitig sehr guter Sicherheit.

# Installation

Mit folgendem Befehl wird openvpn installiert

```
sudo apt-get install openvpn
```

Als erstes sollte man die Beispielkonfiguration und das Verzeichnis zur Schlüsselerzeugung an einen geeigneten Ort entpacken:

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

# Server

## Schlüssel und Zertifikate generieren

Hinweis:

Im folgenden Abschnitt wird beschrieben, wie man die Zertifikate auf dem Server als Root im Verzeichnis */etc/openvpn/easy-rsa2* erstellt. Man kann diese *Certificate Authority* (CA) aber auch problemlos auf irgendeinem anderen Rechner und/oder an einer anderen Stelle im Dateisystem erstellen, wozu man dann auch keine Root-Rechte benötigt, wenn man als normaler Benutzer Schreibrechte besitzt. Man muss sich dann auch nicht immer auf dem Server einloggen, wenn man einen neuen Schlüssel für einen neuen Benutzer erstellen möchte.

Nach dem Erzeugen muss man dann nur noch die jeweiligen Schlüssel und Zertifikate auf die entsprechenden Rechner kopieren, und zwar auf jeden Rechner (Server und Clients) seine eigene *.key*- und *.crt*-Datei sowie die globale *ca.crt*. Der Server benötigt auch noch die Diffie-Hellman-Datei *dh1024.key*. Der CA-Schlüssel *ca.key* wird dagegen nur innerhalb der CA benötigt und muss nicht transferiert werden.

Die gegenseitige Authentifizierung zwischen Server und Client findet bei OpenVPN über kryptografische Schlüssel und Zertifikate statt, die als erstes erstellt werden müssen. Dazu wechselt

man zuerst in das Verzeichnis */etc/openvpn/easy-rsa2/* und editiert dort die Datei *vars* mit Root-Rechten.

In der *vars* findet man am Ende folgende Einträge:

```
export KEY_COUNTRY=DE
export KEY_PROVINCE=Bayern
export KEY_CITY=Bad Neustadt
export KEY_ORG=”Vpntest”
export KEY_EMAIL=”onlyspam@myhomepage.net”
```

Diese Einträge müssen ggf. auf die eigenen Verhältnisse angepasst werden.

Aufgrund eines Fehlers wird manchmal das Unterverzeichnis für die Keys nicht erstellt. Man sollte dies darum von Hand nachholen:

```
sudo mkdir keys
```

Danach muss die oben angepasste Datei *vars* in die Umgebungsvariablen aufgenommen werden:

```
source ./vars
```

Es erscheint eine Warnmeldung, worauf mit den folgenden Skript-Aufrufen das Master-Zertifikat und der Master-Schlüssel erstellt werden:

```
sudo -E ./clean-all
 sudo -E ./build-ca
```

Jetzt muss noch das Zertifikat und der Schlüssel für den Server erstellt werden:

```
sudo -E ./build-key-server server
```

Wichtig dabei ist, dass bei „Common Name“ der Name eingegeben werden sollte, mit dem auf den Server später zugegriffen wird. Das kann z.B. auch der DynDNS-Name sein. Das „Challenge Password“ kann leer gelassen werden. Um die Datenbank zu aktualisieren, muss danach zweimal mit y bestätigt werden.

Anschließend müssen nun die Schlüssel für die Benutzer angelegt werden:

</file> sudo -E ./build-key ersterclient sudo -E ./build-key zweiterclient sudo -E ./build-key dritterclient </file>

Das geschieht nach dem gleichen Prinzip wie gerade eben beim Server, jedoch muss nun bei „*Common Name*“ jeweils der Name des Clients eingetragen werden.

Hinweis:

Benötigt man zu einem späteren Zeitpunkt weitere Zertifikate, muss zuerst die vars als root (sudo -s) erneut gesourcet werden. Danach können wie gewohnt neue Zertifikate mittels ./build-key erstellt werden.

Jetzt müssen noch die Diffie-Hellman-Parameter generiert werden. Diese sind nötig, um kryptografische Schlüssel sicher über unsichere Kanäle auszuhandeln. Das geschieht mit

```
sudo -E ./build-dh
```

Die Erstellung kann je nach System einige Zeit dauern.

Wenn die Erstellung erfolgreich gewesen ist, liegen nun alle benötigten Dateien im Verzeichnis */etc/openvpn/easy-rsa2/keys/*. Die Dateien mit der Endung *.key* sind die geheimen Schlüssel, die nur auf dem entsprechenden Rechner, zu dem sie gehören, gespeichert werden sollten. Die *.crt*-Dateien sind die Zertifikate, die nicht geheim gehalten werden müssen. Die Client-Schlüssel und -Zertifikate müssen nun auf die Clients transferiert werden, *server.key* und *ca.key* bleiben wo sie sind.

Außerdem muss jeder Client noch die Datei *ca.crt* erhalten, damit er den Server einwandfrei identifizieren kann. Zusätzlich muss noch darauf geachtet werden, dass die Dateien nie im ASCII-Modus übertragen werden. Dies kann dazu führen, dass die Datei nicht mehr entschlüsselt werden kann und somit ein Verbinden mit dem openVPN-Server nicht möglich ist. Fehlermeldung: „`Error: private key password verification failed`“. Am besten umgeht man das Problem, indem man alle zu übertragenden Dateien in ein *.tar*- oder *.rar*-Archiv packt.

## Konfiguration

Nun muss noch die Server-Konfigurationsdatei */etc/openvpn/server.conf* angepasst werden. Wichtig ist, dass die Pfade zu den Schlüsseln angepasst werden. Alle anderen Einstellungen sind schon ganz brauchbar.

```
ca ./easy-rsa2/keys/ca.crt
cert ./easy-rsa2/keys/server.crt
key ./easy-rsa2/keys/server.key    # Diese Datei geheim halten.
dh ./easy-rsa2/keys/dh1024.pem     # Diffie-Hellman-Parameter
```

Zur Verbesserung der Sicherheit sollte man den Daemon unter einer unprivilegierten Benutzerkennung laufen lassen, indem man folgende Zeilen aktiviert:

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

Noch besser ist es, hier nicht auf die Kennung `nobody/nogroup` zurückzugreifen, sondern eine eigene spezialisierte `openvpn/openvpn`-Identität zu schaffen, wobei man die Shell auf „`/bin/false`“ setzen kann.

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

Wer den Server in einem privaten LAN stehen hat, muss noch „Port-Forwarding“ auf seinem Router aktivieren. OpenVPN nutzt standardmäßig den Port 1194 (UDP), der auf die interne IP-Adresse des VPN-Servers weitergeleitet werden muss. Wer einen Linux-Router betreibt, kann dafür das nathelper-Skript verwenden. Besitzer eines Hardware-Routers sollten bei Bedarf dessen Betriebsanleitung oder

die Webseiten des Router-Herstellers zu Rate ziehen.

Jetzt kann der Server folgendermaßen gestartet werden:

```
sudo /etc/init.d/openvpn restart
```

Falls das nicht erfolgreich ist und folgender Fehler in */var/log/daemon.log* bzw. */var/log/syslog* steht (häufig bei virtualisierten Systemen der Fall):

```
openvpn[...]: Note: Cannot open TUN/TAP dev /dev/net/tun: Permission denied
(errno=13)
openvpn[...]: Note: Attempting fallback to kernel 2.2 TUN/TAP interface
openvpn[...]: Cannot allocate TUN/TAP dev dynamically
```

hilft eventuell der Abschnitt [#Probleme Probleme] weiter.

### LAN einbeziehen

Um dem Client nicht nur den Server selber, sondern auch das LAN über das VPN zugänglich zu machen, muss nochmal die *server.conf* bearbeitet werden. Dort wird im betreffenden Bereich folgendes eingetragen:

```
push "route 192.168.2.0 255.255.255.0"
```

Der hier genannte IP-Bereich muss natürlich durch den des serverseitigen LANs ersetzt werden. Außerdem muss noch das IP-Forwarding am Server aktiviert werden:

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

Um diese Änderung permanent zu machen, kann man sie in die Datei */etc/sysctl.conf* eintragen:

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

Wenn der OpenVPN-Daemon nicht auf dem „Default-Gateway“ des lokalen Netzes läuft, so muss auf letzterem (dem Default-Gateway/Router) noch eine Route erstellt werden, die den OpenVPN-Server als Gateway für das VPN festlegt. Handelt es sich dabei um einen Linux-Rechner, so lautet der Befehl wie folgt und kann bei Bedarf (ohne „`sudo`“) in die Datei */etc/rc.local* eingetragen werden:

```
sudo route add -net 10.8.0.0 netmask 255.255.255.0 gw vpn.server.i.p
```

Weiterhin müssen bei einer evtl. vorhandenen Firewall Regeln (bei Iptables in der „Forward-chain“) eingerichtet werden, die die Kommunikation erlauben.

## Client-Konfiguration

Zuerst muss auch auf den Clients das *openvpn*-Paket installiert werden, wie weiter oben beschrieben. Statt der Datei *server.conf* wird hier aber logischerweise die */etc/openvpn/client.conf* editiert, die

vorher mit

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

in das entsprechende Verzeichnis kopiert wurde.

Dort ändert man die IP-Adresse, unter der der Server erreichbar ist. Wer keine statische IP zur Verfügung hat, kann dort auch seine dyndns-Adresse angeben.

```
remote ich.dyndns.org
```

Die Pfade der einzelnen *ca*, *cert*, usw. müssen hier nicht unbedingt angepasst werden, wenn die einzelnen Dateien auf dem Client im gleichen Ordner liegen wie die **client.conf** und die voreingestellten Dateinamen besitzen. Natürlich müssen sie erstmal vom Server herüberkopiert worden sein.

Die Authentifizierung des OpenVPN-Servers ist für einen ersten Test nicht notwendig, für die spätere Benutzung aus Sicherheitsgründen aber empfohlen. Dazu wird eine der Methoden, die auf der Seite http://openvpn.net/howto.html beschrieben ist, verwendet . Bei einem aktuellen Ubuntu (ab Hardy) mit OpenVPN-Version ab 2.1 funktioniert:

```
remote-cert-tls server
```

Andernfalls bekommt man bei jeder Verbindung eine Warnung („WARNING: No server certificate verification method has been enabled. See [http://openvpn.net/howto.html#mitm http://openvpn.net/howto.html#mitm] for more info.“)

# Ubuntu

Hinweis:

Ab Ubuntu 7.04 Feisty Fawn bietet sich die komfortable Nutzung der Network-Manager/VPN Plugins an.

Auch hier muss das VPN jetzt neu gestartet werden:

```
sudo /etc/init.d/openvpn restart
```

Wer die Verbindung nicht bei jedem Systemstart automatisch aufbauen lassen will, muss folgende Zeile in der Datei */etc/default/openvpn* aktivieren:

```
AUTOSTART="none"
```

In diesem Fall empfiehlt es sich, die Konfiguration im eigenen Homeverzeichnis anzulegen. Dann kann man die Verbindung bei Bedarf auch mit einfachen Benutzerrechten über folgenden Befehl starten:

```
openvpn ~/pfad/zu/client.conf
```

# Microsoft Windows

OpenVPN-Pakete für Windows sind über die folgende URL erhältlich: http://openvpn.se/download.html. Nach der Installation müssen dann folgende Dateien, die vorhin erstellt wurden, in den Ordner *C:\Programme\OpenVPN\config* kopiert werden:

```
client.ovpn
clientX.crt
clientX.key
ca.crt
```

Wie zu erkennen ist, muss die *client.conf*-Datei zur Verwendung mit der Windows-GUI die Endung *.ovpn* haben.

Nun mit der rechten Maustaste auf das Icon von OpenVPN klicken und „Connect“ auswählen. Wenn alles richtig gemacht wurde, sollte die Verbindung nun funktionieren.

# OpenVPN über HTTP-Proxy

Sitzt man nicht direkt an einem Internetzugang (z.b. im Büro oder beim Kunden), so hat man oft keine Möglichkeit über Port 1194 ins Internet zu kommen. Üblicherweise ist aber der Port 443 (HTTPS) in jedem Netz freigeschaltet. Um vom Client über den HTTPS-Proxy ins Internet und zum heimischen OpenVPN-Server zu gelangen, müssen folgende Konfigurationen gemacht werden:

## server.conf

```
# 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 443
;port 1194

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

Auf dem Server wird in der *server.conf* auf den Port 443 umgeschaltet. Zudem wird hier auf das Protokoll TCP umgeschaltet, was nicht zwingend ist, aber bei der Benutzung eines HTTP-Proxys weniger Probleme verursacht.

### client.conf

```
# 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 1080
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]
```

Auf dem Client wird in der *client.conf* die Zeile „`http-proxy 192.168.4.1 1080`“ hinzugefügt. Wobei die Addresse des Proxy „`192.168.4.1`“ und der Port „`1080`“ mit den effektiven Angaben ersetzt werden müssen. Weitere Informationen zur Konfiguration von HTTP-Proxys sind unter [http://openvpn.net/howto.html#http http://openvpn.net/howto.html#http] verfügbar.

## Probleme

### Virtualisierte Systeme

Viele Provider bieten mittlerweile günstige virtualisierte Server an (auch VPS oder VServer). Bestimmte Funktionen lassen sich auf solchen Servern nur nutzen, wenn die entsprechenden Fähigkeiten auch in den Kernel des Host-Systems kompiliert wurden, auf dem die VPS läuft. Das gilt insbesondere für TUN/TAP und iptables. Bei vielen Providern genügt es, wenn man in einem Support-Ticket kurz erklärt, dass man TUN/TAP oder iptables nutzen möchte und sie wechseln den Kernel aus.

Funktioniert TUN/TAP dann immer noch nicht, muss man das Gerät noch selbst erstellen:

```
sudo mkdir -p /dev/net
sudo mknod /dev/net/tun c 10 200
sudo chmod 600 /dev/net/tun
```

From:
https://wiki.da-checka.de/ - **PSwiki**

Permanent link:
**https://wiki.da-checka.de/doku.php/wiki/dienste/openvpn?rev=1298471995**

Last update: **2011/02/23 15:39**