OpenVPN – Server und Client konfigurieren

OpenVPN bietet eine tolle Möglichkeit, ein verschlüsselte VPN-Verbindung aufzubauen. Das Programm dient dabei sowohl als Server- als auch als Client-Applikation. Die Software ist OpenSource und steht für Linux, OpenBSD, Microsoft Windows und MAC OS X zur Verfügung.

Da in den meisten Fällen wahrscheinlich Linux als Server-OS und Windows an den Client-Maschinen eingesetzt wird, beschreibt diese Anleitung die Konfiguration eines VPN-Servers unter Linux sowie das Einrichten der Client-Zugänge mit Windows-Systemen. Die Konfigurationen dürften sich zu den anderen Systemen jedoch überhaupt nicht bzw. kaum unterscheiden.

Grundlagen des OpenVPN-Tunnels

Grundlage des OpenVPN-Tunnels ist das openvpn-Programm, welches sowohl auf Server- als auch auf Clientseite auf einem UDP-Port läuft (Standard 5000) und mit Hilfe des Tun/Tap-Treibers eine virtuelle Netzwerkschnittstelle anlegt, welche jeweils ein Ende des Tunnels darstellt. Im Gegensatz zum PPP, welches ähnlich arbeitet, kann der gesamte Netzwerkverkehr auf Grundlage von OpenSSL verschlüsselt werden.
Zur Authentifizierung von Server und Client können zwei Verfahren verwendet werden:

- Shared-Secret-Authentifizierung (auch Static-Key-Authentifizierung)
- Zertifikatbasierte Authentifizierung

Shared-Secret-Authentifizierung

Die Shared-Secret-Authentifizierung basiert auf einem symmetrischen Verschlüsselungsverfahren. Dabei müssen die Kommunikationspartner den gemeinsamen Schlüssel (shared key) zuvor über einen sicheren Kanal austauschen. Der gesamte Tunnel-Netzwerkverkehr wird dann mit diesem Schlüssel verschlüsselt und kann von jedem, der ebenfalls den Schlüssel besitzt, entschlüsselt werden. Daher muss der Schlüssel bei Kompromittierung auf allen beteiligten Systemen ausgetauscht werden.

Zuerst wird ein Key generiert:

openvpn --genkey --secret static.key

Dieser Schlüssel wird am Client- und Server-Rechner für die Authentifizierung benötigt.

Angenommen, der Server hat die reale IP-Adresse *192.168.0.1* und der Client die IP-Adresse *192.168.0.2* kann man folgendermaßen eine erste Testverbindung aufbauen:

Server:
openvpn --dev tun0 --remote 192.168.0.2 --ifconfig 192.168.10.1 192.168.10.2 --secret static.key

Client:
openvpn --dev tun0 --remote 192.168.0.1 --ifconfig 192.168.10.2 192.168.10.1 --secret static.key

Die Rechner sollten nun untereinander kommunizieren können. Alternativ könnte man eine Konfigurations-Datei auf dem Server bzw. Client erstellen und dann mit

openvpn --config config.ovpn

die Verbindung herstellen.

Zertifikatbasierte Authentifizierung

Die zertifikatbasierte Authentifizierung basiert auf einem asymmetrischen Verschlüsselungsverfahren, über welches ein sicherer Kanal erzeugt wird. Da die asymmetrische Verschlüsselung erheblich langsamer im Vergleich zur symmetrischen Verschlüsselung ist, wird ein temporärer symmetrischer Zufallsschlüssel von einem Kommunikationspartner erzeugt und über den sicheren Kanal an sein Gegenüber geschickt. Bei Kompromittierung braucht dann nur noch das entsprechende Zertifikat gesperrt werden. Allerdings ist für das Aufsetzen der Public-Key-Infrastruktur ein größerer Konfigurationsaufwand erforderlich.

Für kleine Netzwerke kann der Peer-to-Peer-Modus (mit Shared-Secret-Authentifizierung) von OpenVPN verwendet werden. Der Administrationsaufwand steigt bei dieser Realisierung mit der Größe des VPN-Netzes - also mit der Anzahl der VPN-Clients. Außerdem muss pro Client auf der Serverseite ein expliziter Daemon-Prozess mit separatem UDP-Port laufen, was zudem die Firewall-Konfiguration aufwändiger gestaltet. Im Folgenden wird daher der Server-Client-Modus von OpenVPN (ab Version 2) mit Hilfe von zertifikatbasierter Authentifizierung beschrieben.

Zur Signierung des Server- und der Client-Zertifikate sollte man eine eigene CA aufsetzen. Das OpenSSL-Paket, welches installiert werden muss (Linux), bringt schon die nötigen Werkzeuge dafür mit. Zum Erstellen der nötigen Zertifikate beachte das Script „OpenSSL Zertifikate selbst erstellen“.

Wichtig: führe die Befehle zum Erstellen der Zertifikate im OpenVPN-Ordner /etc/openvpn (bei SuSE Linux) aus!

Erstelle anhand der Doku also eine eigene Zertifizierungsstelle, ein Server- sowie mindestens ein Client-Zertifikat. Führe auch den Befehl zum Erstellen der PKCS12-Datei aus, vergib aber für den Server KEIN Passwort (Client Ja!), da sonst bei jedem Start des OpenVPN-Daemons dieses abgefragt wird. Ist soweit alles fertig, kann der Server konfiguriert werden.

Anmerkung: natürlich gibt es auch das Perl-Script CA.pl. Da ich jedoch davon ausgehen muss, dass nicht jedem dieses komfortable Script zur Verfügung steht und meine Doku zum Erstellen der Zertifikate wunderbare Hilfestellung gibt, verwende ich nicht CA.pl!

Server konfigurieren

In das OpenVPN-Verzeichnis (*/etc/openvpn* bei SuSE Linux) kann nun eine Konfigurationsdatei angelegt werden (z. B. *openvpn.conf*). Vorher sollte noch mit dem Befehl

openssl dhparam -out /etc/openvpn/dh1024.pem 1024

eine Diffie-Hellman-Parameter-Datei erstellt werden. Die Konfig-Datei sollte so aussehen:

```
# OpenVPN laeuft als Server und verwendet 192.168.10.0/24
# als VPN-Subnetz; IP des Servers: 192.168.10.1
mode server
tls-server
ifconfig 192.168.10.1 255.255.255.0

# IP-Adressbereich für die Clients von 192.168.10.2-192.168.10.10
ifconfig-pool 192.168.10.2 192.168.10.10

# Auf welcher IP-Adresse soll OpenVPN lauschen? (optional)
;local 192.168.0.1

# Welcher Port soll verwendet werden? (Standard: UDP/5000)
proto udp
port 1194

# Wir verwenden Ethernet-Tunnel - also das tap-Device.
# Fuer IP-Tunnel ist das tun-Device zu verwenden und ausserdem
# die server-Option statt der o.g. mode-Option.
dev tap

# Welche Schluessel/Zertifikate sollen verwendet werden?
# In welchem Verzeichnis liegen die Dateien?
cd /etc/openvpn

# Wir verwenden alles in einer Datei: CA-Cert, eigenes Cert, priv. Key
pkcs12 server.p12

# Diffie-Hellman-Parameter.
# Diese Datei kann wie folgt erstellt werden:
# openssl dhparam -out /etc/openvpn/dh1024.pem 1024
dh dh1024.pem

# Ein gemeinsamer Schluessel kann ausserdem verwendet.
# So koennen nur die Clients eine Verbindung aufbauen,
# die auch das Geheimnis kennen.
;tls-auth preshared.key 0

# Welche CRL-Datei wird verwendet?
# Die CRL ist eine von der CA unterschriebene Liste
# mit den Zertifikaten, die sich nicht anmelden dürfen.
;crl-verify crl.pem
```

 # Pruefe, ob noch eine Verbindung mit dem Client besteht.
 # Jede 10 Sekunden einen Ping, falls nach 60 Sekunden
 # keine Antwort ankommt, wird ein Verbindungsabbruch angenommen.
keepalive 10 60

 # Erhalte Verbindung falls Client-IP-Adressen
 # waehrend der Verbindung sich aendern sollten.
 # (z.B. bei Dial-Up-Reconnect)
float

 # Standardmäßig können die VPN-Clients untereinander nicht kommunizieren.
 # Um das zu bewerkstelligen ist folgender Parameter notwendig:
client-to-client

 # Kompression für den Tunnel
comp-lzo

 # Ermoeglicht mehrere gleichzeitige Verbindungen mit
 # dem gleichen Zertifikat. (nicht empfohlen)
;duplicate-cn

 # Abgeben der Prozess-Rechte nach dem Start:
*# * user Nutzer nach Rechteabgabe (mit moeglichst wenig Rechten)*
*# * group Gruppe analog*
*# * persist-key Lese die Schluessel nicht beim Neustart durch*
 # SIGUSR1 bzw. --ping-restart ein
*# * persist-tun Analog fuer den Zugriff auf das TUN/TAP-Device*
user nobody
group nobody #bei debian nogroup
persist-key
persist-tun

 # Log-Level:
 # 0 keine Ausgabe bis auf kritische Fehler
 # 4 empfohlen fuer Standardbetrieb
 # 5 und 6 zum Debuggen
 # 9 maximal
verb 4

 # VPN-Clients ins Netz 192.168.0.0 routen (unsicher!)
;push "route 192.168.0.0 255.255.255.0"
 # Daemon-Mode: keine Ausgabe auf das Terminal (stdout) sondern ins Syslog
;daemon # Sollte erst nach fertiger Konfiguration aktiviert werden

Mit **openvpn –config /etc/openvpn/[server-config-datei]** wird der Server gestartet. Bei SuSE kann alternativ **rcopenvpn start** verwendet werden (Endung der Config-Datei zwingend *.conf!). Bei Debian muss in der ***/etc/init.d/openvpn*** eine Variable ***NAME*** mit dem Namen der Konfig-Datei gesetzt werden, z. B. ***NAME=openvpn*** für die Datei ***/etc/openvpn/openvpn.conf***

Client konfigurieren

Wie eingangs erwähnt richten wir hier den Client-Rechner unter Windows ein. Lade dir also OpenVPN für Windows herunter (http://openvpn.se/). Es wird davon ausgegangen, dass OpenVPN in das Verzeichnis C:\Programme\OpenVPN installiert wurde.

Wechsle in den Ordner C:\Programme\OpenVPN\config und lege hier eine neue OVPN-Datei an, z. B. *openvpn_client.ovpn*

Öffne diese Datei mit einem beliebigen Editor (Notepad) und füge folgende Zeilen ein:

```
# OpenVPN laeuft als Client.
client

# Wir verwenden Ethernet-Tunnel - also das tap-Device.
dev tap

# Angabe des Server-Verbindungs-Daten.
# Bei mehreren remote-Eintraegen wird ein Load-Balancing versucht.
;remote my.server.local 5000  # Namen sind moeglich
remote myhome.dyndns.org 1194  # IP-Adressen auch

# Welche Schluessel/Zertifikate sollen verwendet werden?
# In welchem Verzeichnis liegen die Dateien?
# Es wird C:\Programme\OpenVPN\certs angenommen
cd C:/Programme/OpenVPN/certs

# Wir verwenden wie beim Server alles Zertifikate in einer Datei:
pkcs12 client1.p12

# Ein gemeinsamer Schluessel kann ausserdem verwendet.
# So koennen nur die Clients eine Verbindung aufbauen,
# die auch das Geheimnis kennen.
# (Ein kleiner Schutz gegen DoS-Attacken)
# ACHTUNG: Letzter Parameter muss auf Client-Seite == 1 sein!
;tls-auth preshared.key 1

# Versuche den Hostnamen des Servers aufzuloesen auch nachdem
# keine Verbindung zum Internet bestand.
# Sinnvoll bei Clients, die keine permanente Internet-Anbindung haben.
resolv-retry infinite

# Klienten brauchen keinen Port binden.
nobind
```

 # Pruefe, ob noch eine Verbindung mit dem Server besteht.
 # Jede 10 Sekunden einen Ping, falls nach 60 Sekunden
 # keine Antwort ankommt, wird ein Verbindungsabbruch angenommen.
keepalive 10 60

 # Kompression fuer den Tunnel
comp-lzo

 # Abgeben der Prozess-Rechte nach dem Start:
*# * user Nutzer nach Rechteabgabe (mit moeglichst wenig Rechten)*
*# * group Gruppe analog*
*# * persist-key Lese die Schluessel nicht beim Neustart durch*
 # SIGUSR1 bzw. --ping-restart ein
*# * persist-tun Analog fuer den Zugriff auf das TUN/TAP-Device*
user nobody
group nobody # bei Debian statt nobody nogroup
persist-key
persist-tun

 # Log-Level:
#
 # 0 keine Ausgabe bis auf kritische Fehler
 # 4 empfohlen fuer Standardbetrieb
 # 5 und 6 zum Debuggen
 # 9 maximal
verb 4

 # Daemon-Mode: keine Ausgabe auf das Terminal (stdout) sondern ins Syslog
;daemon # Sollte erst nach fertiger Konfiguration aktiviert werden

 # Lege nach dem Verbindungsaufbau automatisch eine Route zum lokalen Netz
 # hinter dem Server durch den Tunnel an.
 # Beispiel: Subnetz 192.168.2.0/24
;route 192.168.2.0 255.255.255.0

 # Default-Route ueber VPN
;route remote_host 255.255.255.255 net_gateway
;route 0.0.0.0 0.0.0.0 vpn_gateway

Nun kann über die *Eingabeaufforderung* oder mit dem OpenVPN-GUI die Verbindung zum Server hergestellt werden. Es wird dabei nach dem Passwort gefragt, welches beim Erstellen des Client-Zertifikats vergeben wurde.

Welche Probleme können auftauchen?
Wie immer: eine Menge. Ein Fehler bei mir war u. A. der, dass mein Linux-PC auf „Lokale Zeit“ eingestellt war und die Zertifikate erst nach einer Stunde gültig wurden (GMT + 1).
Stelle auch sicher, dass deine Firewall nicht den Verbindungsaufbau behindert, öffne ggf. den Port 5000!
Sollte deinem TAP-Device keine IP zugewiesen werden, dann muss evtl. noch der „DHCP-Client“-Dienst in Win2000/XP gestartet werden. Das soll's gewesen sein!